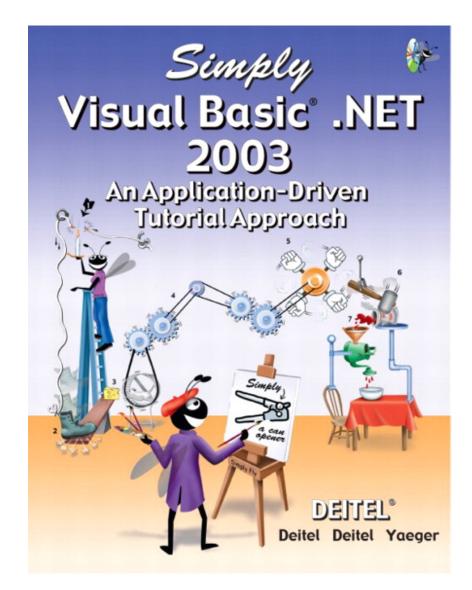
# Instructor's Manual for Simply Visual Basic® .NET 2003



Deitel Deitel Yaeger



### 🥖 O N T E N T S

	Preface	iv
1	Graphing Application	1
2	Welcome Application	4
3	Welcome Application	8
4	Designing the Inventory Application	25
5	Completing the Inventory Application	<b>3</b> 2
6	Enhancing the Inventory Application	40
7	Wage Calculator Application	50
8	Dental Payment Application	61
9	Car Payment Calculator Application	84
10	Class Average Application	<b>9</b> 4
11	Interest Calculator Application	1 <b>0</b> 5
12	Security Panel Application	116
13	Enhancing the Wage Calculator Application	132
14	Shipping Time Application	144
15	Fund Raiser Application	153
16	Craps Game Application	162
17	Flag Quiz Application	173
18	Sales Data Application	187
19	Microwave Oven Application	201

### iii Tutorial

20	Shipping Hub Application	<b>2</b> 26
21	"Cat and Mouse" Painter Application	<b>2</b> 47
22	Typing Application	259
23	Screen Scraping Application	<b>2</b> 76
24	Ticket Information Application	<b>2</b> 89
25	ATM Application	303
26	CheckWriter Application	<b>3</b> 25
27	Phone Book Application	<b>3</b> 43
28	Bookstore Application: Web Applications	<b>3</b> 59
29	Bookstore Application: Client Tier	<b>3</b> 62
30	Bookstore Application: Information Tier	<b>3</b> 70
31	Bookstore Application: Middle Tier	<b>3</b> 74
<b>3</b> 2	Enhanced Car Payment Calculator Application	<b>3</b> 87

## Preface

Thank you for considering and/or adopting our text *Simply Visual Basic .NET*. If you have not read the preface to *Simply Visual Basic .NET*, we strongly encourage you to do so. The preface contains a careful walkthrough of book's key features. We have worked hard to produce a textbook and ancillaries that we hope you and your students will find valuable.

The following ancillary resources are available:

- Simply Visual Basic .NET's program examples are included on a CD-ROM in the back of the book. The examples help instructors prepare lectures faster and aid students in their study of the Visual Basic .NET. The examples are also available for download at www.deitel.com. When extracting the source code from the ZIP file, you must use a ZIP-file reader such as WinZip (www.winzip.com) or PKZIP (www.pkware.com) that understands directories. The file should be extracted into a separate directory such as simplyvbl\_solutions.
- This Simply Visual Basic .NET Instructor's Manual contains answers to the exercises in Simply Visual Basic .NET. The Instructor's Manual CD also contains programming exercise solutions. The programs are separated into directories by tutorial and by project name.
- Companion Web site at www.prenhall.com/deitel.
- *Powerpoint Slide Show* which contains the source code for each program and key discussion points for the examples. Instructors can edit these slides for their own use in classroom discussions.
- *Test Item File* which contains hundreds of multiple-choice questions. Instructors can use these questions to create exams.

We would sincerely appreciate your comments, criticisms and corrections. Please send them to:

#### deitel@deitel.com

We will respond immediately. Please watch our Deitel & Associates, Inc. Web site and our Prentice Hall Web site for book and product updates:

#### www.deitel.com www.prenhall.com/deitel

We would like to thank the extraordinary team of publishing professionals at Prentice Hall who made *Simply Visual Basic .NET* and its ancillaries possible. Our Computer Science editor, Petra Recter, worked closely with us to ensure the timely availability and professional quality of these ancillaries.

Harvey M. Deitel Paul J. Deitel Cheryl Yaeger



### TUTORIAL

## **Graphing Application**

Introducing Computers, the Internet and Visual Basic .NET Solutions

$\bigcirc$	Instructor's Manual Exercise Solutions Tutorial 1		
-	MULTIPLE-CHOICE	1.1 The World Wide Web was developed	
	QUESTIONS	<ul><li>a) by ARPA</li><li>c) before the Internet</li></ul>	<ul><li>b) at CERN by Tim Berners-Lee</li><li>d) as a replacement for the Internet</li></ul>
		<b>1.2</b> Microsoft's initiative integrat development.	es the Internet and the Web into software
		<ul><li>a) .NET</li><li>c) Windows</li></ul>	b) BASIC d) W3C
		1.3 TextBoxes, Buttons and RadioButtons are	e examples of
		a) platforms	b) high-level languages
		c) IDEs	d) controls
		<b>1.4</b> is an example of primary memory	Drv.
		a) TCP	b) RAM
		c) ALU	d) CD-ROM
		<b>1.5</b> Visual Basic .NET is an example of a(n) _ statements accomplish more substantial tasks.	language, in which single program
		a) machine	b) intermediate-level
$\frown$		c) high-level	d) assembly
$\bigcirc$		<b>1.6</b> Which protocol is primarily intended to cro	eate a "network of networks?"
		a) TCP	b) IP
		c) OOP	d) FCL
		<b>1.7</b> A major benefit of programmunderstandable and better organized than techniques.	
		a) object-oriented	b) centralized
		c) procedural	d) HTML
		<b>1.8</b> .NET's collection of prepackaged classes a	nd methods is called the
		a) NCL	b) WCL
		c) FCL	d) PPCM
		<b>1.9</b> The information-carrying capacity of comr	nunications lines is called
		a) networking	b) secondary storage
		c) traffic	d) bandwidth
		<b>1.10</b> Which of these programming languages v	vas specifically created for .NET?
		a) C#	b) C++
		c) BASIC	d) Visual Basic
		<b>Answers:</b> 1.1) b. 1.2) a. 1.3) d. 1.4) b. 1.5) c.	1.6) b. 1.7) a. 1.8) c. 1.9) d. 1.10) a.
-	EXERCISES	<b>1.11</b> Categorize each of the following items as	
$\bigcirc$		a) CPU	b) Compiler
<u> </u>		c) Input unit	d) A word-processor program

e) A Visual Basic .NET program

Answers: a) hardware. b) software. c) hardware. d) software. e) software.

**1.12** Translator programs, such as assemblers and compilers, convert programs from one language (referred to as the source language) to another language (referred to as the target language). Determine which of the following statements are *true* and which are *false*:

- a) A compiler translates high-level-language programs into target-language programs.
- b) An assembler translates source-language programs into machine-language programs.
- c) A compiler translates source-language programs into target-language programs.
- d) High-level languages are generally machine dependent.
- e) A machine-language program requires translation before it can be run on a computer.

**Answers:** a) True. b) True. c) True. d) False. High-level languages are generally machine independent. e) False. A machine language program is native to that specific machine and can be run without translation.

**1.13** Computers can be thought of as being divided into six units.

- a) Which unit can be thought of as "the boss" of the other units?
- b) Which unit is the high-capacity "warehouse" and retains information even when the computer is powered off?
- c) Which unit might determine whether two items stored in memory are identical?
- d) Which unit obtains information from devices like the keyboard and mouse?

Answers: a) CPU. b) Secondary storage unit. c) ALU. d) Input unit.

**1.14** Expand each of the following acronyms:

a) W3C	b) TCP/IP
c) OOP	d) FCL
a) HTMI	

e) HTML

**Answers:** a) World Wide Web Consortium. b) Transmission Control Protocol/Internet Protocol. c) Object-oriented programming. d) Framework Class Library. e) HyperText Markup Language.

**1.15** What are the advantages to using object-oriented programming techniques?

**Answer:** Programs that use object-oriented programming techniques are easier to understand, correct and modify. The key advantage with using object-oriented programming is that it tends to produce software that is more understandable because it is better organized and has fewer maintenance requirements than software produced with earlier methodologies. OOP helps the programmer build applications faster by reusing existing software components. OOP also helps programmers create new software components that can be reused on future software-development projects.





## **Welcome Application**

Introducing the Visual Studio<sup>®</sup> .NET IDE Solutions

$\bigcirc$	Instructor's Manual Exercise Solutions Tutorial 2		
-	MULTIPLE-CHOICE QUESTIONS	<ul><li>2.1 The integrated development written in .NET programming languages such</li><li>a) Solution Explorer</li></ul>	environment is used for creating applications as Visual Basic.NET. b) Gates
		c) Visual Studio .NET	d) Microsoft
		2.2 The.vb filename extension indicates a	
		a) Visual Basic file	b) dynamic help file
		c) help file	d) very big file
		2.3 The pictures on toolbar Buttons are called	ed
		a) prototypes	b) icons
		c) tool tips	d) tabs
		2.4 The allows programmers to n	nodify controls visually, without writing code.
		a) <b>Properties</b> window	b) Solution Explorer
		c) menu bar	d) Toolbox
		<b>2.5</b> The hides the <b>Toolbox</b> when <b>Box</b> 's area.	the mouse pointer is moved outside the Tool-
		a) component-selection feature	b) Auto Hide feature
$\bigcirc$		c) pinned command	d) minimize command
$\bigcirc$		<b>2.6</b> A appears when the mouse for a few seconds.	pointer is positioned over an IDE toolbar icon
		a) drop-down list	b) menu
		c) tool tip	d) down arrow
		2.7 The Visual Studio .NET IDE provides	
		a) help documentation	b) a toolbar
		c) windows for accessing project files	d) All of the above.
		<b>2.8</b> The contains a list of helpful nity.	links, such as Get Started and Online Commu-
		a) Solution Explorer window	b) Properties window
		c) Start Page	d) Toolbox link
		2.9 The Properties window contains	
		a) the component object box	b) a Solution Explorer
		c) menus	d) a menu bar
		2.10 A can be enhanced by addir	g reusable components such as Buttons.
		a) control	b) Form
		c) tab	d) property
		2.11 For Web browsing, Visual Studio .NET	includes
		a) Web View	b) Excel
		c) a <b>Web</b> tab	d) Internet Explorer
$\bigcap$		<b>2.12</b> An application's GUI can include	
$\bigcirc$		a) toolbars	b) icons
		c) menus	d) All of the above.

2.13 The \_\_\_\_\_ does not contain a pin icon.

a) Dynamic Help window	b) Solution Explorer window
------------------------	-----------------------------

c) Toolbox window d) active tab

**2.14** When clicked, \_\_\_\_\_\_ in the **Solution Explorer** window will expand nodes and \_\_\_\_\_\_ will collapse nodes.

a) minus boxes; plus boxesb) plus boxes; minus boxesc) up arrows; down arrowsd) left arrows; right arrows

**2.15** Form \_\_\_\_\_\_ specify attributes such as size and position.

- a) nodes b) inputs
  - c) properties d) title bars

**Answers:** 2.1) c. 2.2) a. 2.3) b. 2.4) a. 2.5) b. 2.6) c. 2.7) d. 2.8) c. 2.9) a. 2.10) b. 2.11) d. 2.12) d. 2.13) d. 2.14) b. 2.15) c.

**EXERCISES** 2.16 (*Closing and Opening the Start Page*) In this exercise, you will learn how to close and reopen the Start Page. To accomplish this task, perform the following steps:

- a) Close Visual Studio .NET if it is open by selecting File > Exit or by clicking its close box.
- b) Start Visual Studio .NET.
- c) Close the Start Page by clicking its close box (Fig. 2.30).

💓 Mi	crosoft	Deve	lopmen	t Environ	ment [d	lesign] - Start Page			_ 🗆 ×
Eile	Edit	⊻iew	Tools	<u>W</u> indow	Help				
1	-	<b>2</b>	8	X 🗈	e k	) + Ci + 🗐 + 🖳	►	• 🐸	»» •
Sta	t Page								4 Þ 🔀
									4
	Projec	:t <u>s</u>	On	line Reso	urces	M <u>y</u> Profile			
Read	У								



d) Select Help > Show Start Page to display the Start Page.

**2.17** (*Enabling Auto Hide for the Solution Explorer Window*) In this exercise, you will learn how to use the Solution Explorer window's Auto Hide feature by performing the following steps:

- a) Open the Start Page.
- b) In the Get Started page (displayed by default), click the Open Project Button to display the Open Project dialog. You can skip to step e) if the Welcome application is already open.
- c) In the **Open Project** dialog, navigate to C:\SimplyVB\Welcome and click **Open**.
- d) In the Open Project dialog, select Welcome.sln and click Open.
- e) Position the mouse pointer on the vertical pin icon in the **Solution Explorer** window's title bar. After a few seconds, a tool tip appears displaying the words **Auto Hide** (Fig. 2.31).

7



Figure 2.31 Enabling Auto Hide.

f) Click the vertical pin icon. This action causes a Solution Explorer tab to appear on the right side of the IDE. The vertical pin icon changes to a horizontal pin icon (Fig. 2.32). Auto Hide has now been enabled for the Solution Explorer window.

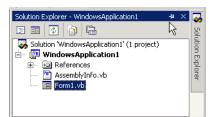


Figure 2.32 Solution Explorer window with Auto Hide enabled.

- g) Position the mouse pointer outside the Solution Explorer window to hide the window.
- h) Position the mouse pointer on the Solution Explorer tab to view the Solution Explorer window.

**2.18** (Sorting Properties Alphabetically in the Properties Window) In this exercise, you will learn how to sort the **Properties** window's properties alphabetically by performing the following steps:

- a) Open the **Welcome** application by performing steps a) through d) of Exercise 2.17. If the **Welcome** application is already open, you can skip this step.
- b) Locate the Properties window. If it is not visible, select View > Properties Window to display the Properties window.
- c) To sort properties alphabetically, click the **Properties** window's alphabetic icon (Fig. 2.33). The properties now display in alphabetic order.

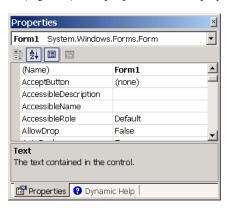


Figure 2.33 Sorting properties alphabetically.



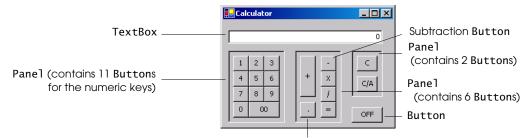


## **Welcome Application**

Introduction to Visual Programming Solutions 9

	Instructor's Manual Exercise Solutions Tutorial 3		
-	MULTIPLE-CHOICE	3.1 Property determines the Fo	orm's background color.
	QUESTIONS	a) BackColor	b) BackgroundColor
		c) RGB	d) Color
		<b>3.2</b> To save all the solution's files, select	
		,	b) File > Save
		c) File > Save All	d) File > Save As
		<b>3.3</b> When the ellipsis Button to the rig is displayed.	ght of the Font property value is clicked, the
		a) Font Property dialog	b) New Font dialog
		c) Font Settings dialog	d) <b>Font</b> dialog
		<b>3.4</b> PictureBox property cor PictureBox.	ntains a preview of the image displayed in the
		a) Picture	b) ImageName
		c) Image	d) PictureName
		<b>3.5</b> The tab allows you to creat	e your own color.
		a) Custom	b) Web
		c) System	d) User
)		3.6 The PictureBox class has namespace _	
		a) System.Windows.Forms	
		c) System.Form.Font	d) System.Form.Control
		3.7 A Label control displays the text speci	ified by property
		a) Caption	b) Data
		c) Text	d) Name
		<b>3.8</b> In mode, the application is	executing.
		a) start	b) run
		c) break	d) design
		<b>3.9</b> The command prevents prolocation of the Form's controls.	ogrammers from accidentally altering the size and
		a) Lock Controls	b) Anchor Controls
		c) Lock	d) Bind Controls
		<b>3.10</b> Pixels are	
		a) picture elements	b) controls in the <b>Toolbox</b>
		c) a set of fonts	d) a set of colors on the <b>Web</b> tab
		<b>Answers:</b> 3.1) a. 3.2) c. 3.3) d. 3.4) c. 3.5)	a. 3.6) a. 3.7) c. 3.8) b. 3.9) a. 3.10) a.
-	EXERCISES	will use the visual programming technic	to create the GUI shown in each exercise. You ques presented in this tutorial to create a vari- only GUIs, your applications will not be fully

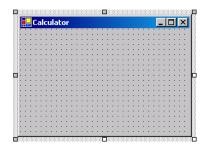
ety of GUIs. Because you are creating only GUIs, your applications will not be fully operational. For example, the Calculator GUI in Exercise 3.11 will not behave like a calculator when its Buttons are clicked. You will learn how to make your applications fully operational in later tutorials. Create each application as a separate project. 3.11 (Calculator GUI) Create the GUI for the calculator shown in Fig. 3.33.



Decimal point Button



- a) *Creating a new project.* Create a new Windows Application named Calculator.
- b) *Renaming the Form file.* Name the Form file Calculator.vb.
- c) Manipulating the Form's properties. Change the Size property of the Form to 272, 192. Change the Text property of the Form to Calculator. Change the Font property to Tahoma.



d) Adding a TextBox to the Form. Add a TextBox control by double clicking it in the Toolbox. A TextBox control is used to enter input into applications. Set the TextBox's Text property in the Properties window to 0. Change the Size property to 240, 21. Set the TextAlign property to Right; this right aligns text displayed in the TextBox. Finally, set the TextBox's Location property to 8, 16.

•	0	-	C	al	C	u	a	to	or																		-		<u> </u>	]	2	×
	į,	٠														Ċ															,c	<b>)</b> (
8	1																													0	2	]
į	1																														C	Ĵ
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		1	1	1	1	1	1	1	1	1	1	1	1		l
						1	1							1			1	1			1			1	1							l,
	•			•	•	•	•	•			•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•		•	•		
	·	•	•	·	•	•	•	÷	•	•	·	·	•	•	÷	÷	•	•	·	•	•	÷	•	•	•	÷	•	•	·	·	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	÷		÷	÷						÷	÷																	÷	÷			
	·	÷	÷	·	•	•	•	·	·	÷	·	·	•	•	·	·	•	·	·	•	•	·	•	•	•	·	•	÷	·	·	•	•
	1		1	1		1	1	•		1	1		1	1	•		1	1	1	1	1	•		1	1	•	1	1	1			1
	1	1	1	1	1	1	÷.	1	1	1	1	1	i.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	l
	÷			•	•	•	•	•			•	•	•		•		•	•	•	•		•	•	•	•	•	•		•	•	1	1

e) Adding the first Panel to the Form. Panel controls are used to group other controls. Double click the Panel icon ( Panel ) in the Toolbox to add a Panel to the Form. Change the Panel's BorderStyle property to Fixed3D to make the inside of the Panel appear recessed. Change the Size property to 88, 112. Finally, set the Location property to 8, 48. This Panel contains the calculator's numeric keys.

	-				0																		-		÷	ų	÷
• •		·	÷	·	•	·	•	• •	·	•	·	·	·	·	·	·	·	÷	•	·	·	÷	·	·	·	•	•
																										0	1
1.1	1.15	$\sim 2$	at-	10	2.5	ð. 5	5.5	a t	e.	•			•			•	•	•	•	•	•	•		•	•		1
	-	_	2	22					1	•			•			•	÷	•	•	•	÷	•			÷		
								- 8	·	·		•	·	·	•	·	·	÷	•	·	·	÷	•	·	·		•
								. 8	٠	•			·			•	÷	•	•	÷	÷	•		÷	•		
								. 8	٠	•							•	•	•	•	•	•		•	•		
								. 8	٠	÷			•				•	•	•	•	÷	•		•	÷		
								- 8	•	•		•	·	•	•	·	·	•	•	·	·	•	•	·	•		
								. 3		•			•				•	•	•	•	•	•		•	•		
									ŀ	÷								•	•		•	•			•		
								- 8	•	÷		•	·	÷	•	÷	·	÷	•	·	·	•	•	·	÷		•
								• 8	٠	•			•			•	•	•	•	•	•	•		•	•		
								. 8	٠	•							•	•	•	•	•	•		•	•		
								- 8	•	•			•	•		÷	÷	•	•	•	÷	•		÷	÷		
								• 8	٠	•		•	•			•	•	•	•	•	·	•		•	•		
								. 8	٠				•							•				•			

- f) Adding the second Panel to the Form. Click the Form. Double click the Panel icon in the Toolbox to add another Panel to the Form. Change the Panel's BorderStyle property to Fixed3D. Change the Size property to 72, 112. Finally, set the Location property to 112, 48. This Panel contains the calculator's operator keys.
- g) Adding the third (and last) Panel to the Form. Click the Form. Double click the Panel icon in the Toolbox to add another Panel to the Form. Change the Panel's BorderStyle property to Fixed3D. Change the Size property to 48, 72. Finally, set the Location property to 200, 48. This Panel contains the calculator's C (clear) and C/A (clear all) keys.

•	•	•	•	•	•	ta	•	•		-		•			•	•		• •	•	•	÷	•			
•	•	•	•	•	1	•	•	•	•		• •	•		•	•	•	•	•		1	•	•		•	
																								0	I
																			~	×.	. A	A 40	<b>.</b>		
_	_	-	-	_	-	-	_	-	-		_	_	_	-	-	-	-	-	ų,	1		8	10		2
											Ŀ								4						
											Ŀ							- 1	1						
	•			•	·			·			ŀ		•			÷	•	- 1	1			•	•		1
											Ŀ						÷	1	L É	1H				•	
											Ŀз					•	•	1	- 13				•	•	
	·	÷	•	÷	·	•	•	·	•		Ŀ		•	•	•	÷	·	- 1				•	·	• •	
	•			•	•			•			Ŀ		•	•		•	•	1	1	1		•	•	•	
			1	1	•		1				F 1		•			1	1	1	1			•	•	•	1
•	•	•	•	•	•		•	•			F -	•	•			•	•	• 1	Ľ	8		ΒE	18		
	•			•	•			•	•		Ŀ		•	•		•	•	• 1		1					
	•				•			•			Ŀ		•			•	•	1		•	•			• •	
											E -						•	- 1			•	•		• •	

h) Adding Buttons to the Form. There are 20 Buttons on the calculator. To add a Button to a Panel, double click the Button control ( ■ Button ) in the Toolbox. Then add the Button to the Panel by dragging and dropping it on the Panel. Change the Text property of each Button to the calculator key it represents. The value you enter in the Text property will appear on the face of the Button. Finally, resize the Buttons, using their Size properties. Each Button labelled 0–9, x, /, -, = and . should have a size of 24, 24. The 00 and OFF Buttons have size 48, 24. The + Button is sized 24, 64. The C (clear) and C/A (clear all) Buttons are sized 32, 24.



i) Saving the project. Select File > Save All to save your changes.

3.12 (Alarm Clock GUI) Create the GUI for the alarm clock in Fig. 3.34.

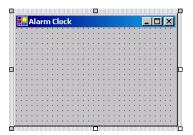
		📙 Alarm 🛙	Clock			
		HOUR	MINUTE	SECOND	ON OFF	
RadioButton -				-AM/PM	1	Buttons
GroupBox -		00:00:	DO	C PM	Timer	
	C		si	NOOZE	]	— Label
	_					

Figure 3.34 Alarm Clock GUI.

a) *Creating a new project.* Create a new Windows Application named AlarmClock.

b) *Renaming the Form file.* Name the Form file AlarmClock.vb.

c) *Manipulating the Form's properties.* Change the Size property of the Form to 256, 176. Change the Text property of the Form to Alarm Clock. Change the Font property to Tahoma.



d) Adding Buttons to the Form. Add six Buttons to the Form. Change the Text property of each Button to the appropriate text. Change the Size properties of the Hour, Minute and Second Buttons to 56, 23. The ON and OFF Buttons get size 40, 23. The Timer Button gets size 48, 32. Align the Buttons as shown in Fig. 3.34.

I		•		A	la	I	n	1	1	DI	R	2														_	.		]	2	×
			н	0	U	R		l	Ν	II	VI	JI	ΓE			S	E	C	D	VE	)			0	N			0	DF	F	:
	1	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	;	÷	÷	÷	÷	÷	÷	
		ì	1	ŝ	i	÷	÷	ŝ	÷	1	ŝ	ŝ	÷	÷	ŝ	÷	÷	÷	ŝ	÷	1	÷	ĵ,	1	1	÷	1	1	1	1	1
		·	·	·	·	·	·	·	·	·	·	·	·	·	·	•	·	·	·	·	·	·	·				C	]		1	
		÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	H		Ti	im	e	,		-
		1	÷	ŝ	ĵ,	÷	÷	ŝ	ĵ,	÷	÷	ŝ	÷	÷	ŝ	Ĵ	÷	÷	ŝ	ĵ,	÷	÷	ĵ,								
		÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷				L	j.			
		;	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	÷	ļ,	1	1	÷	÷	÷	1	÷	:
		1	÷	÷	÷.	÷	÷	÷.	÷	÷	÷	÷.	÷	÷	÷	÷.	÷	÷	÷.	÷	÷	÷	ĵ,	1	÷	÷	÷.	÷	1	÷.	:
		•	÷	÷			÷	÷			÷	÷			÷			÷	÷			÷			•			•			
		1	1	1			1	1		1	1	1		1	1	1		1	1		1	1			1		1		1	1	

e) Adding a Label to the Form. Add a Label to the Form. Change the Text property to Snooze. Set its Size to 248, 23. Set the Label's TextAlign property to Middle-Center. Finally, to draw a border around the edge of the Snooze Label, change the BorderStyle property of the Snooze Label to FixedSingle.

•				-	1	-	-	-	-		_					_	_	_		_	_	_	-				-
Н	0	U	R		L	Μ	11	NI	JI	ΓE		L	S	E١	C	0	VE	)	L		0	N		L	C	DFI	ē
	÷	÷	÷	Ĵ	÷.	:	÷	Ĵ	Ĵ	÷	÷	÷.	÷	÷	Ĵ	÷	÷	:	÷	÷	÷	÷	÷	ï			
						·				·				•							·						
	1	•	•	•	1	•	•	•	1	•	•	•	•	•	•		•	•		1	•	•		1		• •	
	2	1	1	1	1	:	1	1	1	:	1	1	1	1	1	1		:	1	1	1	÷		-	•		đ
														•									т	im	her		L
	1	•	•	•	1	1	•	•		1			1	•	•	1		•	1	1	•		1		.01		L
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1		-					I.
	÷																										
		Ξ.															÷	÷									

f) Adding a GroupBox to the Form. GroupBoxes are like Panels, except that GroupBoxes can display a title. To add a GroupBox to the Form, double click the GroupBox control ( GroupBox ) in the Toolbox. Change the Text property to AM/PM, and set the Size property to 72, 72. To place the GroupBox in the correct location on the Form, set the Location property to 104, 38.

 L		ON	OFF
	AM/PM	Ti	mer

g) Adding AM/PM RadioButtons to the GroupBox. Add two RadioButtons to the Form by dragging the RadioButton control ( RadioButton ) in the Toolbox and dropping it onto the GroupBox twice. Change the Text property of one RadioButton to AM and the other to PM. Then place the RadioButtons as shown in Fig. 3.34 by setting the Location of the AM RadioButton to 16, 16 and that of the PM RadioButton to 16, 40. Set their Size properties to 48, 24.

H	IOL	IR		P	411	NL	JTE	SECOND ON OFF
			1		÷	1	: :	· · · · · · · · · · · · · · · · · · ·
• • •	•	• •	÷	• •	·	÷	• •	
	1.1			: :		÷.	::	
								C AM
• •	•	• •	•	• •	·			Timer
		• •		• •	•		• •	
							• •	: C PM :   ·
	1.1		1	: :		1	11	

h) Adding the time Label to the Form. Add a Label to the Form and change its Text property to 00:00:00. Change the BorderStyle property to Fixed3D and the Back-Color to Black. Set the Size property to 64, 23. Use the Font property to make the time bold. Change the ForeColor to Silver (located in the Web tab) to make the time stand out against the black background. Set TextAlign to MiddleCenter to center the text in the Label. Position the Label as shown in Fig. 3.34.

HOUR MINUTE SECOND ON OFF	🛃 Alarm Cloc	k		_	
C AM	HOUR MIN	IUTE	SECOND	ON	OFF
			C AM	Ti	mer
SNOOZE		SN	JOOZE		

i) Saving the project. Select File > Save All to save your changes.

3.13 (Microwave Oven GUI) Create the GUI for the microwave oven shown in Fig. 3.35.

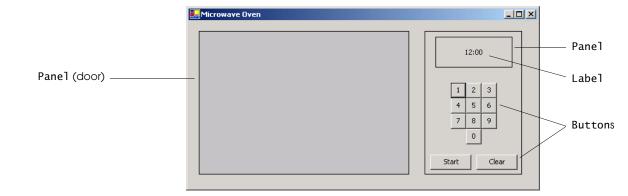
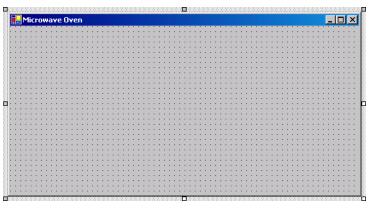


Figure 3.35 Microwave Oven GUI.

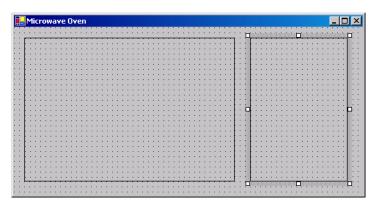
- a) Creating a new project. Create a new Windows Application named Microwave.
- b) *Renaming the Form file.* Name the Form file Microwave.vb.
- c) Manipulating the Form's properties. Change the Size property of the Form to 552, 288. Change the Text property of the Form to Microwave Oven. Change the Font property to Tahoma.



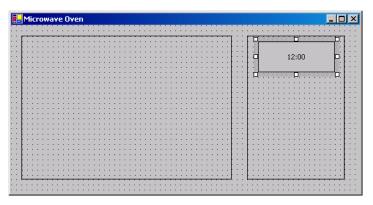
d) Adding the microwave oven door. Add a Panel to the Form by double clicking the Panel ( Panel ) in the Toolbox. Select the Panel and change the BackColor property to Silver (located in the Web tab) in the Properties window. Then change the Size to 328, 224. Next, change the BorderStyle property to FixedSingle.

]	Μ	lic	cr	o	w	a	v	e	0	v	e	n																																														I	_		I.	>	
n.		A			A			A												े	-1	 									Aa	Aa	Sa	Sa	Aa		80			a.	-			-	÷	÷	÷	÷	÷	÷	1	÷	÷										
÷		-		-	-	-					-			-	-	-		-	-	-	-	 -	-	-	-	-	-	-			-	-	-				-			÷	1							•	•	۰.	1	۰.	۰.	•	•		11					•	
ł		•	•			•	•			•	•				•													•	•	·	•	•	·	÷	•	•	·	·	•	•					•	•		•	•	•	1	•	•	•	•	•	1.1					•	
ł							•			•																		•	•	•	٠.	•	•	٠.	•	•	•	•	•	1		•			•	•		•	•	•	1		•	•		•	1.1					•	
ł																												•	•	•	•	•	•	•	•	•	•	•	•	1						•		1		•	1	1	•				11					•	
ł	1	•	•			•	•			•	•																	•	•	•	1	1	•	•	•	•	•	•	•	•		1	1	1		1	1		1	1	1	1	1	1	2							1	
ł	1			1	1			1	1																				•	1	1	1	•	1	1	1	1	•	•	1						1	1	1	1	1	1	1		1	1	1						1	
ł		1	1	1			1			1	1						1										1	1	1	1			1	1	1	1	1	1		1	8°.							1			1						11					•	
ł	1			1	1			1			1	1											1							1	1	1	1	1	1	1	1	1	1	1						1	1	1	1	1	1			1	1							1	
t			1				1				1																	1	1	1	1		1		1	1		1	1			Ľ.	Ľ.	Ľ.	Ľ.	1	Ű.	1	1	1	1	1	1	2	2	2						2	
đ		1		1			1			1																			1												81					1	1	1			1				2								
a.		1		1			1	1	1		1	1	1															۰.	1	1	÷.,	۰.	1	÷.,	۰.	۰.	÷.,	۰.	۰.	1						1		1	1	1	0	2	1	£.,	8	2						2	
1	1	1	1	1	1	1	1	1	1	1	1	1	1															1	1	1	÷.	÷.	1	1	1	1	1	1	1			1		1	1	1	1	1	1	1	0	2	1	1	0	1						2	
1																																									٤.																1.						
1		1		1			1		1				1															2	2	1	1	1	1	1	1	1	1	1	1	l t	٦.																1.						
1																																								. 6	Ξ.										1.						1.						
1																																																									ι.						
1																																								. 1	١.																ι.						
а.																																								. 1											1						1.						
3																																								.										÷							ί.,						
4																																								.	۰.	•			·	·		•	÷	÷	•	•	÷	•			ι.					•	
1																																								•	۰.	•			•	•	•	•	•	•		•	•	•	•	•	÷ .					•	
ł		•					•			•																			•	•			•	•				•	•	•		•			•	•		•	•	•		•	•	•		•	1					•	
ł		•				•	•			•	•																		•	·	•	•	·	÷	•		÷	÷	•	•						÷		•	•	•	1	۰.	•	•	•		1.1					•	
ł		•	•	•		•	•			•	•																	•	•	•	•	•	•	•	•	•	•	•	•	•	÷	•			•	•		•	•	•		•	•	•	•	•	•					•	
s.		•					•																					•	•	•			•	•	•	•	•	•	•	- 1					•	•		•	1	•	1	1	1	•		•	1					•	
ŝł.		•	•			•	•			•																		•	•	•	•	•	÷	÷	•	•	•	÷	•	•								•		•	1	•	•	•			11					•	
ł		•	•			•	•			•	•																	•	•	•	•	•	•	•	•	•	•	•	•	•		•			•	•		•	1	•	1	1	•	•		•	1					•	
Ì	88	88		88		88	88		88		88										1						18	88											88		יכ			1			1	1	1	1	1	1	1	•		•	11					1	
2		1												1			9	1			-			1						•	۰.	÷.,	1	•	۰.		1	•			-									•	1					•	1.1					•	

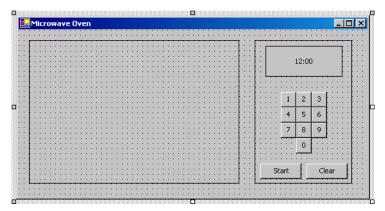
e) *Adding another Pane1*. Add another Pane1 and change its Size to 152, 224 and its BorderStyle to FixedSingle. Place the Pane1 to the right of the door Pane1, as shown in Fig. 3.35.



f) Adding the microwave oven clock. Add a Label to the right Panel by clicking the Label in the Toolbox once, then clicking once inside the right Panel. Change the Label's Text to 12:00, BorderStyle to FixedSingle and Size to 120, 48. Change TextAlign to MiddleCenter. Place the clock as shown in Fig. 3.35.

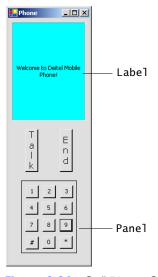


g) Adding a keypad to the microwave oven. Place a Button in the right Panel by clicking the Button control in the Toolbox once, then clicking inside the Panel. Change the Text to 1 and the Size to 24, 24. Repeat this process for nine more Buttons, changing the Text property in each to the next number in the keypad. Then add the Start and Clear Buttons, each of Size 64, 24. Do not forget to set the Text properties for each of these Buttons. Finally, arrange the Buttons as shown in Fig. 3.35. The 1 Button is located at 40, 80 and the Start Button is located at 8, 192.



h) Saving the project. Select File > Save All to save your changes.

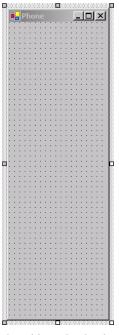
3.14 (Cell Phone GUI) Create the GUI for the cell phone shown in Fig. 3.36.



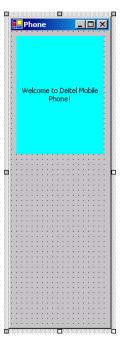


- a) *Creating a new project.* Create a new Windows Application named Phone.
- b) *Renaming the Form file.* Name the Form file Phone.vb.

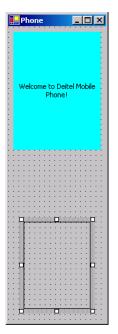
c) *Manipulating the Form's properties.* Change the Form's Text property to Phone and the Size to 160, 488. Change the Font property to Tahoma.



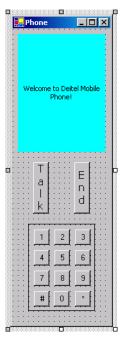
d) Adding the display Label. Add a Label to the Form. Change its BackColor to Aqua (in the Web tab palette), the Text to Welcome to Deitel Mobile Phone! and the Size to 136, 184. Change the TextAlign property to MiddleCenter. Then place the Label as shown in Fig. 3.36.



e) Adding the keypad Panel. Add a Panel to the Form. Change its BorderStyle property to FixedSingle and its Size to 104, 136.



- f) Adding the keypad Buttons. Add the keypad Buttons to the Form (12 Buttons in all). Each Button on the number pad should be of Size 24, 24 and should be placed in the Panel. Change the Text property of each Button such that numbers 0-9, the pound (#) and the star (\*) keys are represented. Then add the final two Buttons such that the Text property for one is Talk and the other is End. Change the Size of each Button to 24, 80, and notice how the small Size causes the Text to align vertically. Also change each Button's Font size to 12 points.
- g) *Placing the controls.* Arrange all the controls so that your GUI looks like Fig. 3.36.



h) Saving the project. Select File > Save All to save your changes.

3.15 (Vending Machine GUI) Create the GUI for the vending machine in Fig. 3.37.

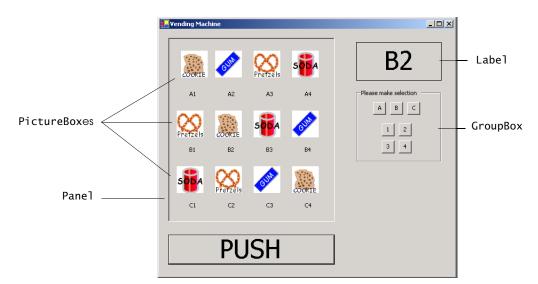
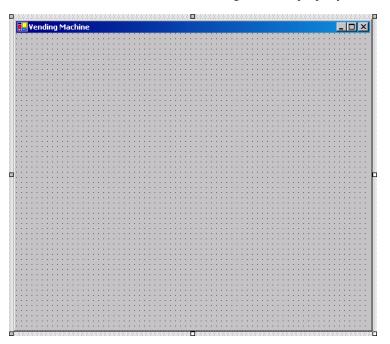
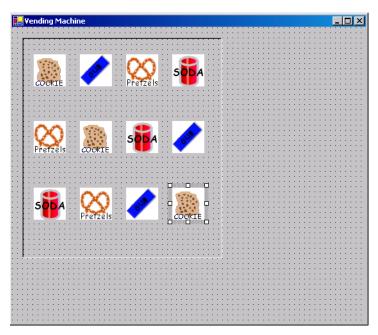


Figure 3.37 Vending Machine GUI.

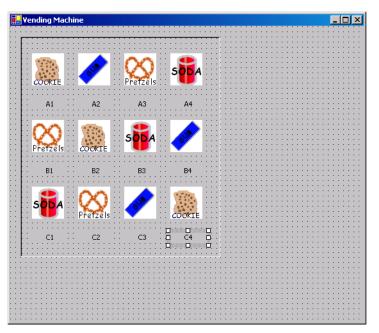
- a) *Creating a new project.* Create a new Windows Application named Vending-Machine.
- b) *Renaming the Form file*. Name the Form file VendingMachine.vb.
- c) *Manipulating the Form's properties.* Set the Text property of the Form to Vending Machine and the Size to 560, 488. Change the Font property to Tahoma.



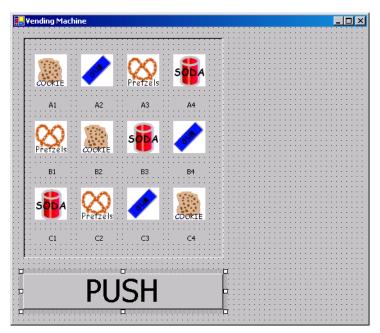
d) Adding the food selection Panel. Add a Panel to the Form, and change its Size to 312, 344 and BorderStyle to Fixed3D. Add a PictureBox to the Panel, and change its Size to 50, 50. Then set the Image property by clicking the ellipsis Button and choosing a file from the C:\Examples\TutorialO3\ExerciseImages\VendingMa-chine directory. Repeat this process for 11 more PictureBoxes.



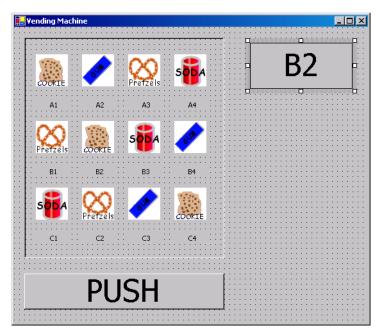
e) Adding Labels for each vending item. Add a Label under each PictureBox. Change the Text property of the Label to A1, the TextAlign property to TopCenter and the Size to 56, 16. Place the Label so that it is located as in Fig. 3.37. Repeat this process for A2 through C4 (11 Labels).



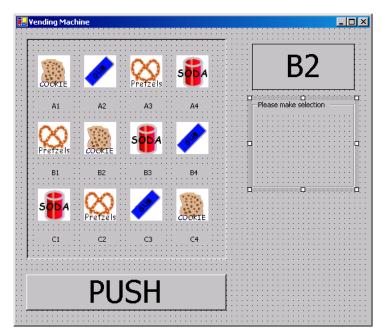
f) Creating the vending machine door (as a Button). Add a Button to the Form by dragging the Button control in the Toolbox and dropping it below the Panel. Change the Button's Text property to PUSH, its Font Size to 36 and its Size to 312, 56. Then place the Button on the Form as shown in Fig. 3.37.



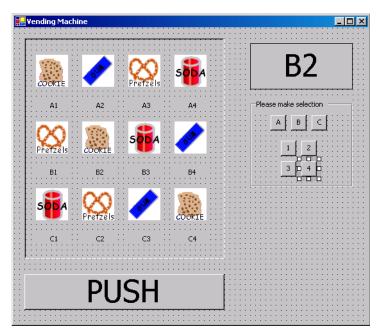
g) Adding the selection display Label. Add a Label to the Form, and change the Text property to B2, BorderStyle to FixedSingle, Font Size to 36, TextAlign to MiddleCenter and Size to 160, 72.



h) *Grouping the input Buttons.* Add a GroupBox below the Label, and change the Text property to Please make selection and the Size to 160, 136.



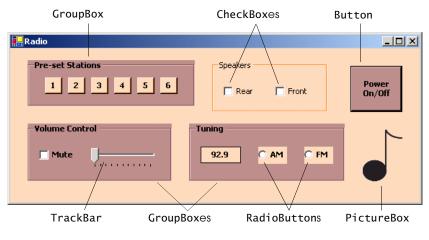
i) Adding the input Buttons. Finally, add Buttons to the GroupBox. For the seven Buttons, change the Size property to 24, 24. Then change the Text property of the Buttons such that each Button has one of the values A, B, C, 1, 2, 3 or 4, as shown in Fig. 3.37. When you are done, move the controls on the Form so that they are aligned as shown in the figure.



j) Saving the project. Select File > Save All to save your changes.

Programming Challenge 🕨

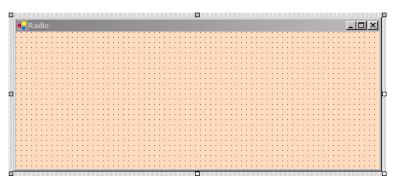
**3.16** (*Radio GUI*) Create the GUI for the radio in Fig. 3.38. [*Note:* All colors used in this exercises are from the Web palette.]



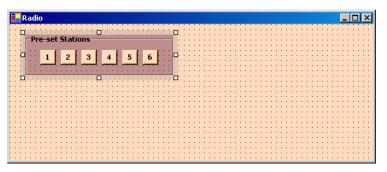


In this exercise, you will create this GUI on your own. Feel free to experiment with different control properties. For the image in the PictureBox, use the file (MusicNote.gif) found in the C:\Examples\TutorialO3\ExerciseImages\Radio directory.

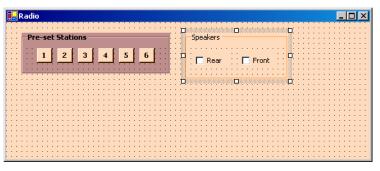
- a) Creating a new project. Create a new Windows Application named Radio.
- b) *Renaming the Form file.* Name the Form file Radio.vb.
- c) Manipulating the Form's properties. Change the Form's Text property to Radio and the Size to 576, 240. Change the Font property to Tahoma. Set BackColor to Peach-Puff.



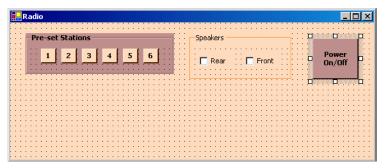
d) Adding the Pre-set Stations GroupBox and Buttons. Add a GroupBox to the Form. Set its Size to 232, 64, its Text to Pre-set Stations, its ForeColor to Black and its BackColor to RosyBrown. Change its Font to bold. Finally, set its Location to 24, 16. Add six Buttons to the GroupBox. Set each BackColor to PeachPuff and each Size to 24, 23. Change the Buttons' Text properties to 1, 2, 3, 4, 5, 6, respectively.



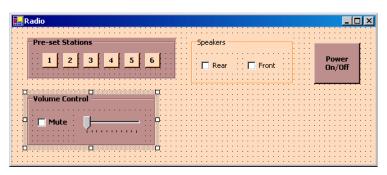
 e) Adding the Speakers GroupBox and CheckBoxes. Add a GroupBox to the Form. Set its Size to 160, 72, its Text to Speakers and its ForeColor to Black. Set its Location to 280, 16. Add two CheckBoxes to the Form. Set each CheckBox's Size to 56, 24. Set the Text properties for the CheckBoxes to Rear and Front.



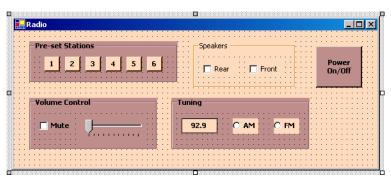
f) Adding the Power On/Off Button. Add a Button to the Form. Set its Text to Power On/Off, its BackColor to RosyBrown, its ForeColor to Black and its Size to 72, 64. Change its Font style to Bold.



g) Adding the Volume Control GroupBox, the Mute CheckBox and the Volume Track-Bar. Add a GroupBox to the Form. Set its Text to Volume Control, its BackColor to RosyBrown, its ForeColor to Black and its Size to 200, 80. Set its Font style to Bold. Add a CheckBox to the GroupBox. Set its Text to Mute and its Size to 56, 24. Add a TrackBar to the GroupBox.



h) Adding the Tuning GroupBox, the radio station Label and the AM/FM RadioButtons. Add a GroupBox to the Form. Set its Text to Tuning, its ForeColor to Black and its BackColor to RosyBrown. Set its Font style to Bold and its Size to 216, 80. Add a Label to the Form. Set its BackColor to PeachPuff, its ForeColor to Black, its BorderStyle to FixedSingle, its Font style to Bold, its TextAlign to Middle-Center and its Size to 56, 23. Set its Text to 92.9. Place the Label as shown in the figure. Add two RadioButtons to the GroupBox. Change the BackColor to Peach-Puff and change the Size to 40, 24. Set one's Text to AM and the other's Text to FM.



i) Adding the image. Add a PictureBox to the Form. Set its BackColor to Transparent, its SizeMode to StretchImage and its Size to 56, 72. Set its Image property to C:\Examples\TutorialO3\ExerciseImages\Radio\MusicNote.gif.

Radio	<u>-0×</u>
Pre-set Stations	Speakers
	Rear Front On/Off
Volume Control	
	92.9 C AM C FM D

j) Saving the project. Select File > Save All to save your changes.



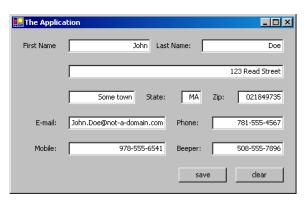


### Designing the Inventory Application

Introducing TextBoxes and Buttons Solutions

)	Instructor's Manual Exercise Solutions Tutorial 4		
-	MULTIPLE-CHOICE QUESTIONS	<ul> <li>4.1 A new Windows application is created by</li> <li>a) New &gt; Program</li> <li>c) New &gt; Project</li> </ul>	y selecting from the File menu. b) New > File d) New > Application
		<ul><li>4.2 A Label's BorderStyle property can be</li><li>a) Fixed3D</li><li>c) 3D</li></ul>	e set to b) Single d) All of the above.
		<ul><li>4.3 When creating a Label, you can specify a) alignment of the text</li><li>c) size</li></ul>	<ul><li>b) border style</li><li>d) All of the above.</li></ul>
		<ul> <li>4.4 Changing the value stored in the</li> <li>file.</li> <li>a) Name</li> <li>c) File Name</li> </ul>	<ul> <li>property will change the name of the Form</li> <li>b) File</li> <li>d) Full Path</li> </ul>
		<ul> <li>4.5 should be used to prefix all T</li> <li>a) txt</li> <li>c) Frm</li> </ul>	,
)		<ul> <li>4.6 A(n) helps the user understa</li> <li>a) Button</li> <li>c) output Label</li> </ul>	nd a control's purpose. b) descriptive Label d) title bar
		<ul> <li>4.7 A is a control in which the us</li> <li>a) Button</li> <li>c) Label</li> </ul>	ser can enter data from a keyboard. b) TextBox d) PictureBox
		<ul><li>4.8 A descriptive Label uses</li><li>a) sentence-style capitalization</li><li>c) a colon at the end of its text</li></ul>	<ul><li>b) book-title capitalization</li><li>d) Both a and c.</li></ul>
		<ul><li>4.9 You should use the font in you</li><li>a) Tahoma</li><li>c) Times</li></ul>	our Windows applications. b) MS Sans Serif d) Palatino
		<ul> <li>4.10 should be used to prefix all</li> <li>a) but</li> <li>c) Frm</li> </ul>	Button names. b) lbl d) btn
_		<b>Answers:</b> 4.1) c. 4.2) a. 4.3) d. 4.4) c. 4.5) a.	4.6) b. 4.7) b. 4.8) d. 4.9) a. 4.10) d.
	EXERCISES	GUI Design Guidelines section. A cumu control appears in Appendix C. In these that do not follow the GUI design guideli you must modify control properties so the	a summary of new GUI design tips listed in the lative list of GUI design guidelines, organized by exercises, you will find Visual Basic .NET Forms ines presented in this tutorial. For each exercise, at your end result is consistent with the guidelines applications do not provide any functionality.
		4.11 (Address Book GUI) In this exercise,	you apply the GUI design guidelines you have

learned to a graphical user interface for an address book (Fig. 4.24).





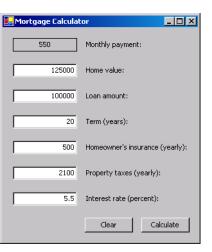
- a) *Copying the template to your working directory.* Copy the C:\Examples\TutorialO4\Exercises\AddressBook directory to your C:\Sim-plyVB directory.
- b) *Opening the application's template file.* Double click AddressBook.sln in the AddressBook directory to open the application.
- c) *Applying GUI design guidelines.* Rearrange the controls and modify properties so that the GUI conforms to the design guidelines you have learned.
- d) *Saving the project.* Select File > Save All to save your changes.

#### Answer:

🛃 Address Bo	ok	<u> </u>
First name:	John Last name:	Doe
Address:	<u></u>	123 Read Street
City:	Some town State: MA Zip	0112849735
E-mail:	John.Doe@not-a-domain.com Phone:	781-555-4567
Mobile:	978-555-6541 Beeper:	508-555-7896
	Save	Clear

- 1. Change the Form's title (Text property).
- 2. All TextBoxes should have corresponding Labels.
- 3. Labels indicating control usage should use sentence-style capitalization.
- 4. Buttons should use book-title capitalization.
- 5. Each descriptive Label text should end with a colon.

**4.12** (*Mortgage Calculator GUI*) In this exercise, you apply the GUI design guidelines you have learned to a graphical user interface for a mortgage calculator (Fig. 4.25).



**Figure 4.25** Mortgage Calculator application without GUI design guidelines applied.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial04\Exercises\MortgageCalculator directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click MortgageCalculator.sln in the MortgageCalculator directory to open the application.
- c) *Applying GUI design guidelines.* Rearrange the controls and modify properties so that the GUI conforms to the design guidelines you have learned.
- d) *Saving the project.* Select File > Save All to save your changes.

#### Answer:

🛃 Mortgage Calculator	
Home value:	125000
Loan amount:	100000
Term (years):	20
Homeowner's insurance (yearly):	500
Property taxes (yearly):	2100
Interest rate (percent):	5.5
Monthly payment:	
Calculate	Clear

- 1. Label should be placed above or to the left of the control it is describing.
- 2. Output Label's setting should be BorderStyle property Fixed3D.
- 3. Output Label initially should be blank.
- 4. Place an application's output below or to the right of the Form's input control.

**4.13** (*Password GUI*) In this exercise, you apply the GUI design guidelines you have learned to a graphical user interface for a password-protected message application (Fig. 4.26).

Password	
Enter your name:	Joe Purple
Enter your password:	*******
Re-enter your password:	******
Enter your secret message:	Click here to log in
Welcome to Simply VB .NET	
Save Log	g out Clear

Figure 4.26 Password application without GUI design guidelines applied.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial04\Exercises\Password directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click Password.sln in the Password directory to open the application.
- c) *Applying GUI design guidelines.* Rearrange the controls and modify properties so that the GUI conforms to the design guidelines you have learned.
- d) Saving the project. Select File > Save All to save your changes.

#### Answer:

Password	_ 🗆 ×
Enter your name:	Joe Purple
Enter your password:	******
Re-enter your password:	*****
Enter your secret message:	Log In
Welcome to Simply VB .NET	
Save	Log Out Clear

- 1. Keep the Label on the Buttons as short and descriptive as possible.
- 2. Arrange groups of controls approximately 2 grid units apart on a Form.
- 3. Leave approximately 2 grid units of space between the edges of the Form and controls nearest the edge. Increase the Form's width.
- 4. Buttons use book-title capitalization.

Programming Challenge 🕨

**4.14** *(Monitor Invoice GUI)* In this exercise, you apply the GUI design guidelines you have learned to a graphical user interface for an invoice application (Fig. 4.27).

Monitor Invoice				
Invoice Application				
Invoice Number	128 In	voice date	10/11/02	
Company Name: Deitel & Associates				
Address (line 1) 100 Application Drive				
Address (line 2)	Books, MA, 01276			
Type:	Quantity:	Price:	Totals:	
15"	10	150	1500	
17"	0	0	0	
19"	0	0	0	
Save		Subtotal	0	
		Tax	0	
Calculate			0	
Clear				

Invoice application without GUI design guidelines applied. **Figure 4.27** 

- a) Copying the template your working directory. Copy the to C:\Examples\TutorialO4\Exercises\MonitorInvoice directory to your C:\SimplyVB directory.
- b) Opening the application's template file. Double click MonitorInvoice.sln in the MonitorInvoice directory to open the application.
- c) Applying GUI design guidelines. Rearrange the controls and modify properties so that the GUI conforms to the design guidelines you have learned.
- d) Saving the project. Select File > Save All to save your changes.

Monitor Invoice				
Invoice Application				
Invoice number:	128	Invoice date:	10/11/02	
Company name:			Deitel & Associates	
Address (line 1):			100 Application Drive	
Address (line 2):			Books, MA 01276	
Type: 15"; 17"; 19";	Quantity:		Totals: 50 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0	
	Save	Calculate	Clear	

- 1. Use Tahoma font.
- 2. Labels indicating control usage should end with colon.
- 3. The Label and the control it describes should be aligned on the left if arranged vertically.
- 4. Label should use sentence-style capitalization.

#### **Tutorial 4**

- 5. Buttons should be placed in the top right or bottom right of a Form.
- 6. Each output Label must have a label that describes it.
- 7. Output Labels arranged vertically and used to display numbers in a calculation should have the TextAlign property set to MiddleRight.
- 8. Descriptive Labels that are in the same column vertically should be left aligned.



### TUTORIAL

### Completing the Inventory Application

Introducing Programming Solutions

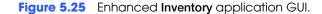
)	Instructor's Manual Exercise Solutions Tutorial 5		
-	MULTIPLE-CHOICE	5.1 A(n) represents a user	action, such as clicking a Button.
	QUESTIONS	a) statement	b) event
	QUESTIONS	c) application	d) function
		<b>5.2</b> To switch to code view, select	
		a) Code > View	b) Design > Code
		c) View > Code	d) View > File Code
		<b>5.3</b> Code that performs the functional	ity of an application
		a) normally is provided by the pro-	ogrammer
		b) can never be in the form of an	event handler
		c) always creates a graphical user	
		d) is always generated by the IDE	
		<b>5.4</b> Comments	
		a) help improve program readabi	•
		b) are preceded by the single-quo	
		c) are ignored by the compiler	d) All of the above.
		<b>5.5</b> The allows a statemen ceded by one or more whitespace char	t to continue past one line (when that character is pre- racters).
)		a) single-quote (') character	b) hyphen (-) character
		c) underscore (_) character	d) plus (+) character
		<b>5.6</b> A(n) causes an applica	tion to produce erroneous results.
		a) logic error	b) event
		c) assignment statement	d) syntax error
		<b>5.7</b> A portion of code that performs	a specific task and returns a value is known as $\boldsymbol{a}(n)$
		a) variable	b) function
		c) operand	d) identifier
		<b>5.8</b> Visual Basic .NET keywords are _	
		a) identifiers	b) reserved words
		c) case sensitive	d) properties
		<b>5.9</b> Visual Studio .NET allows you to or collapse to facilitate code editing.	organize code into, which you can expand
		a) statements	b) operators
		c) regions	d) keywords
		<b>5.10</b> An example of a whitespace chan	racter is a character.
		a) space	b) tab
		c) newline	d) All of the above.
		<b>Answers:</b> 5.1) b. 5.2) c. 5.3) a. 5.4) d.	5.5) c. 5.6) a. 5.7) b. 5.8) b. 5.9) c. 5.10) d.

**EXERCISES** 

**5.11** (*Inventory Enhancement*) Extend the **Inventory** application to include a TextBox in which the user can enter the number of shipments received in a week. Assume every ship-

ment has the same number of cartons (each of which has the same number of items). Then modify the code so that the **Inventory** application uses that value in its calculation.

Inventory Application		
Number of cartons in shipment:	0	Total:
Number of items per cartons:	0	
Number of shipments this week:	0	Calcula



- a) Copying the template application to your working directory. Copy the C:\Examples\Tutorial05\Exercises\InventoryEnhancement directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click InventoryEnhancment.sln in the InventoryEnhancement directory to open the application.
- c) *Resizing the Form*. Resize the Form you used in this tutorial by setting the Size property to 296, 144. Move the Button toward the bottom of the Form, as shown in Fig. 5.25. Its new location should be 184, 78.
- d) *Adding a Label*. Add a Label to the Form and change the Text property to Shipments this week:. Set the Location property to 16, 80. Resize the Label so that the entire text displays. Set the Label's Name property to lblShipments.
- e) Adding a TextBox. Add a TextBox to the right of the Label. Set its Text property to 0 and the Location property to 128, 80. Set the TextAlign and Size properties to the same values as for the other TextBoxes in this tutorial's example. Set the Text-Box's Name property to txtShipments.
- f) *Modifying the code.* Modify the Calculate Total Click event handler so that it multiplies the number of shipments per week with the product of the number of cartons in a shipment and the number of items in a carton.
- g) Running the application. Select Debug > Start to run your application. Enter values for the number of cartons per shipment, items per carton and shipments in the current week. Click the Calculate Button and verify that the total displayed is equal to the result when the three values entered are multiplied together. Enter a few sets of input and verify the total each time.
- h) Closing the application. Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 5.11 Solution
 2
     ' Inventorv.vb
 3
 Δ
    Public Class FrmInventory
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' handles Click event
10
       Private Sub btnCalculate_Click(ByVal sender As _
11
           System.Object, ByVal e As System.EventArgs) _
12
           Handles btnCalculate.Click
13
14
           ' multiply values input and display result in Label
15
           lblTotalResult.Text = _
16
              Val(txtCartons.Text) *
17
              Val(txtItems.Text) *
18
              Val(txtShipments.Text)
19
20
       End Sub ' btnCalculate_Click
```

# 22 End Class ' FrmInventory

**5.12** (Counter Application) Create a counter application. Your counter application will consist of a Label and Button on the Form. The Label initially displays 0, but, each time a user clicks the Button, the value in the Label is increased by 1. When incrementing the Label, you will need to write a statement such as lblTotal.Text = Val(lblTotal.Text) + 1.

	Counter
Label	17
Button	Count

21



- a) *Creating the application.* Create a new project named Counter.
- b) Changing the name of the Form file. Change the name of Form1.vb to Counter.vb.
- c) *Modifying a new Form*. Change your Form's Size property to 168, 144. Modify the Form so that the title reads **Counter**. Change the name of the Form to FrmCounter.
- d) *Changing the startup object.* Change the startup object of your application to the form you modified in *Step c*.
- e) Adding a Label. Add a Label to the Form, and place it as shown in Fig. 5.26. Make sure that the Label's Text property is set to 0 and that TextAlign property is set so that any text will appear in the middle (both horizontally and vertically) of the Label. This can be done by using the MiddleCenter TextAlign property. Also set the BorderStyle property to Fixed3D. Set the Label's Name property to lblCountTotal.
- f) Adding a Button. Add a Button to the Form so that it appears as shown in Fig. 5.26. Set the Button's Text property to contain the text Count. Set the Button's Name property to btnCount.
- g) *Creating an event handler.* Add an event handler to the **Count** Button such that the value in the Label increases by 1 each time the user clicks the **Count** Button.
- h) *Running the application*. Select Debug > Start to run your application. Click the Count Button several times and verify that the output value is incremented each time.
- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 5.12 Solution
2	' Counter.vb
3	
4	Public Class FrmCounter
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' handles Click event
0	<pre>Private Sub btnCount_Click(ByVal sender As _</pre>
1	System.Object, ByVal e As System.EventArgs) _
2	Handles btnCount.Click
3	
4	' when button is clicked add one to lblCountTotal
5	lblCountTotal.Text = Val(lblCountTotal.Text) + 1
6	
7	End Sub ' btnCount_Click

# 18 19 End Class ' FrmCounter

**5.13** (Account Information Application) Create an application that allows a user to input a name, account number and deposit amount. The user then clicks the Enter Button, which causes the name and account number to be copied and displayed in two output Labels. The deposit amount entered will be added to the deposit amount displayed in another output Label. The result is displayed in the same output Label. Every time the Enter Button is clicked, the deposit amount entered is added to the deposit amount displayed in the output Label, keeping a cumulative total. When updating the Label, you will need to write a statement such as lblDeposits.Text = Val(lblDeposits.Text) + Val(txtDepositAmount).

Account Information		
Enter information Name: John Doe Account number: 12345 Deposit amount: 10000	Enter	Account information Name: John Doe Account number: 12345 Deposit amount: 10000

Figure 5.27 Account Information GUI.

- a) *Copying the template application to your working directory.* Copy C:\Examples\Tutorial05\Exercises\AccountInformation directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click AccountInformation.sln in the AccountInformation directory to open the application.
- c) *Creating an event handler.* Add an event handler for the **Enter** Button's Click event.
- d) Coding the event handler. Code the event handler to copy information from the Name: and Account number: TextBoxes to their corresponding output Labels. Then add the value in the Deposit amount: TextBox to the Deposit amount: output Label, and display the result in the Deposit amount: output Label.
- e) *Running the application.* Select **Debug > Start** to run your application. Enter the values in Fig. 5.27 and click the **Enter** Button. Verify that the account information is displayed in the Labels on the right. Enter varying deposit amounts and click the **Enter** Button after each. Verify that the deposit amount on the right has the new values added.
- f) *Closing the application*. Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
Exercise 5.13 Solution
 1
 2
     ' AccountInformation.vb
 3
 4
    Public Class FrmAccountInformation
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' handles Click event
10
        Private Sub btnEnter_Click(ByVal sender As _
11
           System.Object, ByVal e As System.EventArgs) _
12
          Handles btnEnter.Click
```

$\bigcirc$		<pre>13 14 ' copy user input 15 lblCopiedName.Text = txtName.Text 16 lblCopiedAccountNumber.Text = Val(txtAccountNumber.Text) 17 lblBalance.Text = Val(lblBalance.Text) + _ 18 Val(txtDepositAmount.Text) 19 20 End Sub ' btnEnter_Click 21 22 End Class ' FrmAccountInformation</pre>
	What does this code do? 🕨	<b>5.14</b> After entering 10 in the txtPrice TextBox and 1.05 in the txtTax TextBox, a user clicks the Button named btnEnter. What is the result of the click, given the following code?
		<pre>Private Sub btnEnter_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ Handles btnCalculate.Click IblOutput.Text = Val(txtPrice.Text) * Val(txtTax.Text) Find Sub ' btnEnter_Click</pre>
		<b>Answer:</b> This displays the number 10.5 in a Label. (This is the amount of the sale including the tax.)
$\bigcirc$	What's wrong with this code? <b>&gt;</b>	<b>5.15</b> The following event handler should execute when the user clicks a <b>Calculate</b> Button. Identify the error(s) in its code.
		<pre>1 Private Sub btnCalculate_Click(ByVal sender As 2 System.Object, ByVal e As System.EventArgs) _ ' second line 3 Handles btnCalculate.Click 4 5 lblResult.Text = txtPrice.Text * txtTax.Text 6 End Sub ' btnCalculate_Click</pre>
		<b>Answer:</b> The first line of the event handler header is missing the line-continuation character, and the second line of the header has a comment after the line-continuation character; both are syntax errors. Also, the code should use the Val function. The corrected code should read:
		<pre>Private Sub btnCalculate_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ Handles btnCalculate.Click IblResult.Text = Val(txtPrice.Text) * Val(txtTax.Text) End Sub ' btnCalculate_Click</pre>
$\bigcirc$	Using the Debugger 🕨	<b>5.16</b> (Account Information Debugging Exercise) Copy the folder from C:\Examples\Tutorial05\Exercises\DebuggingExercise to your work folder, C:\SimplyVB, and run the Account Information application. Remove any syntax errors, so that the application runs correctly. Answer:
$\bigcirc$		Answer:

Line-continuation ———	1	' Exercise 5.16 Solution
character was missing	2	' AccountInformation.vb
Ű	3 4	Public Class FrmAccountInformation
	5	Inherits System.Windows.Forms.Form
	6	
	7	' Windows Form Designer generated code
	8	
	9	' Enter button click event
1b1Ba1ance was mispelled —	10	Private Sub btnEnter_Click(ByVal sender As _
ibibarance was mispelied	11	System.Object, ByVal e As System.EventArgs) _
	12	Handles btnEnter.Click
	13	
	14	<pre>lblBalance.Text = Val(txtDepositAmount.Text) _</pre>
	15	- Val(txtWithdrawalAmount.Text) _
	16	+ Val( <mark>lblBalance</mark> .Text)
	17	
	18 19	End Sub ' btnEnter_Click
	20	End Class ' FrmAccountInformation

Programming Challenge

**5.17** (Account Information Enhancement) Modify Exercise 5.13 so that it no longer asks for the user's name and account number, but rather asks the user for a withdrawal or deposit amount. The user can enter both a withdrawal and deposit amount at the same time. When the Enter Button is clicked, the balance is updated appropriately.

-Account Information		
Enter information Withdrawal amount: Deposit amount: 0	Account information Name: Sue Purple Account number: 12345 Balance: 0	Default name Default account number

Figure 5.28 Enhanced Account Information GUI.

- a) *Copying the template application to your working directory.* If you have not already done so, copy the C:\Examples\Tutorial05\Exercises\AccountInformation directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click AccountInformation.sln in the AccountInformation directory to open the application.
- c) *Modifying the GUI*. Modify the GUI so that it appears as in Fig. 5.28.
- d) *Setting the default values.* Set the default name and account number to the values shown in Fig. 5.28 using the **Properties** window.
- e) *Writing code to add functionality.* Update the account balance for every withdrawal (which decreases the balance) and every deposit (which increases the balance). When the balance is updated, reset the TextBoxes to zero.
- f) Running the application. Select Debug > Start to run your application. Enter various withdrawal and deposit amounts, click the Enter Button after each. Verify that after each time the Enter Button is clicked, the balance on the right of the application is updated appropriately.
- g) *Closing the application.* Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 5.17 Solution
2	' AccountInformation.vb
3	
4	Public Class FrmAccountInformation
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' event handler for Enter button
10	<pre>Private Sub btnEnter_Click(ByVal sender As _</pre>
11	System.Object, ByVal e As System.EventArgs) _
12	Handles btnEnter.Click
13	
14	<pre>lblBalance.Text = Val(txtDepositAmount.Text) _</pre>
15	- Val(txtWithdrawalAmount.Text) _
16	+ Val(lblBalance.Text)
17	
18	' reset TextBoxes
19	txtWithdrawalAmount.Text = "0"
20	<pre>txtDepositAmount.Text = "0"</pre>
21	
22	End Sub ' btnEnter_Click
23	
24	End Class ' FrmAccountInformation



# Enhancing the Inventory Application

Introducing Variables, Memory Concepts and Arithmetic Solutions

Instructor's Manual Exercise Solutions Tutorial 6		
MULTIPLE-CHOICE	6.1 Parentheses that are add as parentheses.	ded to an expression simply to make it easier to read are known
QUESTIONS	a) necessary c) embedded	<ul><li>b) redundant</li><li>d) nested</li></ul>
	6.2 The operator	r performs Integer division.
	a) \	b) +
	c) Mod	d) ^
	6.3 Every variable has a	
	a) name	<ul><li>b) value</li><li>d) All of the above.</li></ul>
	c) type	,
		ithmetic expressions must be written in form.
	<ul><li>a) straight-line</li><li>c) left-right</li></ul>	b) top-bottom d) right-left
	, c	
	<b>6.5</b> Arithmetic expressions a	
		b) from left to right s of operator precedence of precedence to the highest level of precedence
		event handlers begin with the keyword
	a) Declare	b) Dim
	c) Sub	d) Integer
	6.7 Entering a value in a Tex	xtBox raises the event.
	a) TextAltered	b) ValueChanged
	c) ValueEntered	d) TextChanged
	<b>6.8</b> The function ble.	a converts user input from a TextBox to a variable of type Dou-
	a) Convert	b) MakeDouble
	c) Val	d) WriteDouble
	6.9 Variables to store intege	r values should be declared with keyword
	a) Integer	b) Int
	c) IntVariable	d) None of the above.
	6.10 Keyword in word.	n a variable declaration indicates that the data type is the next
	a) IsA	b) Type
	c) Dim	d) As
	<b>Answers:</b> 6.1) b. 6.2) a. 6.3)	d. 6.4) a. 6.5) c. 6.6) b. 6.7) d. 6.8) c. 6.9) a. 6.10) d.

EXERCISES

**6.11** (Simple Encryption Application) This application uses a simple technique to encrypt a number. Encryption is the process of modifying data so that only those intended to receive the data can undo the changes and view the original data. The user enters the number to be encrypted via a TextBox. The application then multiplies the number by 7 and adds 5. The application displays the encrypted number in a Label as shown in Fig. 6.24.

Simple Encryption	
Enter number to encrypt:	25 Encrypt
Encrypted number: 180	

Figure 6.24 Result of completed Simple Encryption application.

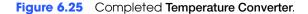
- a) *Copying the template to your working directory.* Copy the C:\Examples\TutorialO6\Exercises\SimpleEncryption directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click SimpleEncryption.sln in the SimpleEncryption directory to open the application.
- c) **Coding the Click event handler.** Encrypt the number in the Click event handler by using the preceding technique. The user input should be stored in an Integer variable (intNumber) before it is encrypted. The event handler then should display the encrypted number.
- d) *Clearing the result.* Add an event handler for the Enter number to encrypt: Text-Box's TextChanged event. This event handler should clear the Encrypted number: TextBox whenever the user enters new input.
- e) Running the application. Select Debug > Start to run your application. Enter the value 25 into the Enter number to encrypt: TextBox and click the Encrypt Button. Verify that the value 180 is displayed in the Encrypted number: output Label. Enter other values and click the Encrypt Button after each. Verify that the appropriate encrypted value is displayed each time.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 6.11 Solution
 2
    ' SimpleEncryption.vb
 3
 4
    Public Class FrmEncryption
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' handles Click event
 8
       Private Sub btnEncrypt_Click(ByVal sender As System.Object, _
 9
          ByVal e As System. EventArgs) Handles btnEncrypt. Click
10
11
          Dim intNumber As Integer
12
13
          intNumber = Val(txtInput.Text) ' obtain user input
14
15
           intNumber = intNumber * 7 + 5 ' encrypt number
16
17
          lblResult.Text = intNumber ' display encrypted number
18
       End Sub ' btnEncrypt_Click
19
20
        ' handles TextChanged event
21
       Private Sub txtInput_TextChanged(ByVal sender As _
22
           System.Object, ByVal e As System.EventArgs) _
23
          Handles txtInput.TextChanged
24
25
           lblResult.Text = ""
26
       End Sub ' txtInput_TextChanged
27
28
    End Class ' FrmEncryption
```

**6.12** (*Temperature Converter Application*) Write an application that converts a Celsius temperature, *C*, to its equivalent Fahrenheit temperature, *F*. Figure 6.25 displays the completed application. Use the following formula:



E Temperature Converter		<u> </u>
Enter a Celsius temperature:	20	Convert
Fahrenheit equivalent:	68	N



- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial06\Exercises\TemperatureConversion directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click TemperatureConversion.sln in the TemperatureConversion directory to open the application.
- c) *Coding the Click event handler.* Perform the conversion in the **Convert** Button's Click event handler. Define Integer variables to store the user-input Celsius temperature and the result of the conversion. Display the Fahrenheit equivalent of the temperature conversion.
- d) *Clearing user input.* Clear the result in the Enter a Celsius temperature: TextBox's TextChanged event.
- e) Running the application. Select Debug > Start to run your application. Enter the value 20 into the Enter a Celsius temperature: TextBox and click the Convert Button. Verify that the value 68 is displayed in the output Label. Enter other Celsius temperatures, click the Convert Button after each. Use the formula provided above to verify that the proper Fahrenheit equivalent is displayed each time.
- f) Closing the application. Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 6.12 Solution
2	' TemperatureConversion.vb
3	
4	Public Class FrmTemperatureConverter
5	Inherits System.Windows.Forms.Form
6	
7	' handles Click event
8	<pre>Private Sub btnConvert_Click(ByVal sender As System.Object, _</pre>
9	ByVal e As System.EventArgs) Handles btnConvert.Click
10	
11	' temperature variables
12	Dim intCelsius As Integer
13	Dim intFahrenheit As Integer
14	
15	intCelsius = Val(txtInput.Text) ' obtain user input
16	
17	' perform conversion
18	intFahrenheit = (9 / 5) * intCelsius + 32
19	
20	lblResult.Text = intFahrenheit
21	End Sub ' btnConvert_Click
22	
23	' handles TextChanged event
24	Private Sub txtInput_TextChanged(ByVal sender As _

25	System.Object, ByVal e As System.EventArgs) _
26	Handles txtInput.TextChanged
27	
28	lblResult.Text = ""
29	End Sub ' txtInput TextChanged
30	· – •
31	End Class ' FrmTemperatureConverter

**6.13** (Simple Calculator Application) In this exercise, you will add functionality to a simple calculator application. The calculator will allow a user to enter two numbers in the Text-Boxes. There will be four Buttons labeled +, -, / and \*. When the user clicks the Button labeled as addition, subtraction, multiplication or division, the application will perform that operation on the numbers in the TextBoxes and displays the result. The calculator also should clear the calculation result when the user enters new input. Figure 6.26 displays the completed calculator.

Ealculator		
Enter first number:	27	+ -
Enter second number:	3	* 🔲
Result:	9	N

Figure 6.26 Result of Calculator application.

- a) Copying the template to your working directory. Copy C:\Examples\Tutorial06\Exercises\SimpleCalculator directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click SimpleCalculator.sln in the SimpleCalculator directory to open the application.
- c) *Coding the addition Click event handler.* This event handler should add the two numbers and display the result.
- d) *Coding the subtraction Click event handler.* This event handler should subtract the second number from the first number and display the result.
- e) *Coding the multiplication Click event handler.* This event handler should multiply the two numbers and display the result.
- f) *Coding the division Click event handler.* This event handler should divide the first number by the second number and display the result.
- g) Clearing the result. Write event handlers for the TextBoxes' TextChanged events. Write code to clear the result Label (lblResult) after the user enters new input into either TextBox.
- h) Running the application. Select Debug > Start to run your application. Enter a first number and a second number, then verify that each of the Buttons works by clicking each, and viewing the output. Repeat this process with two new values and again verify that the proper output is displayed based on which Button is clicked.
- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
    Exercise 6.13 Solution
2
   ' SimpleCalculator.vb
3
4
  Public Class FrmCalculator
5
     Inherits System.Windows.Forms.Form
6
7
     ' handles addition Button's Click event
8
     9
       ByVal e As System. EventArgs) Handles btnAdd. Click
```

10	
11	lblResult.Text = Val(txtFirstNumber.Text) + _
12	Val(txtSecondNumber.Text)
13	End Sub ' btnAdd_Click
14	
15	' handles subtraction Button's Click event
16	<pre>Private Sub btnSubtract_Click(ByVal sender As System.Object, _</pre>
17	ByVal e As System. EventArgs) Handles btnSubtract. Click
18	
19	lblResult.Text = Val(txtFirstNumber.Text)
20	Val(txtSecondNumber.Text)
21	End Sub ' btnSubtract Click
22	-
23	' handles multiplication Button's Click event
24	<pre>Private Sub btnMultiply_Click(ByVal sender As System.Object, _</pre>
25	ByVal e As System.EventArgs) Handles btnMultiply.Click
26	
27	lblResult.Text = Val(txtFirstNumber.Text) * _
28	Val(txtSecondNumber.Text)
29	End Sub ' btnMultiply_Click
30	End Sub Bennarerpry_errek
31	' handles division Button's Click event
32	Private Sub btnDivide_Click(ByVal sender As System.Object, _
33	ByVal e As System.EventArgs) Handles btnDivide.Click
34	byvar e As system. EventArgs) handres benbrude. errek
35	<pre>lblResult.Text = Val(txtFirstNumber.Text) / _</pre>
36	Val(txtSecondNumber.Text)
37	End Sub ' btnDivide_Click
38	
39	' handles TextChanged event
40	Private Sub txtFirstNumber_TextChanged(ByVal sender _
41	As System.Object, ByVal e As System.EventArgs) _
42	Handles txtFirstNumber.TextChanged
42	Handles excertiseNumber.Textenangeu
43	<pre>lblResult.Text = ""</pre>
45	End Sub ' txtFirstNumber_TextChanged
40	End Sub ExtFIrstNumber_TextChanged
	L handles Tout Channed arout
47	' handles TextChanged event
48 49	Private Sub txtSecondNumber_TextChanged(ByVal sender _
	As System.Object, ByVal e As System.EventArgs) _
50	Handles txtSecondNumber.TextChanged
51	
52	lblResult.Text = ""
53	End Sub ' txtSecondNumber_TextChanged
54 55	
00	End Class ' FrmCalculator

```
What does this code do?
```

**6.14** This code modifies values intNumber1, intNumber2 and intResult. What are the final values of these variables?

```
1 Dim intNumber1 As Integer
2 Dim intNumber2 As Integer
3 Dim intResult As Integer
4
5 intNumber1 = 5 * (4 + 6)
6 intNumber2 = 2 ^ 2
7 intResult = intNumber1 \ intNumber2
```

Answer: intNumber1 gets 50, intNumber2 gets 4; intResult gets 12.

What's wrong with this code? **6.15** Find the error(s) in the following code, which uses variables to perform a calculation. 1 Dim intNumber1 As Integer 2 Dim intNumber2 As Integer 3 Dim intResult As Integer 4 5  $intNumber1 = (4 * 6 \land 4) / (10 Mod 4 - 2)$ intNumber2 =  $(16 \setminus 3) \wedge 2 * 6 + 1$ 6 7 intResult = intNumber1 - intNumber2 Answer: intNumber1's assignment statement contains a division by zero, which will cause a run-time error to occur. 1 Dim intNumber1 As Integer 2 Dim intNumber2 As Integer 3 Dim intResult As Integer Δ 5 ' intNumber1 =  $(4 * 6 \land 4) / (10 \mod 4 - 2)$ 6 intNumber2 =  $(16 \setminus 3) \wedge 2 * 6 + 1$ 7 intResult = intNumber1 - intNumber2 Using the Debugger 🕨 6.16 (Average Three Numbers) You have just written an application that takes three numbers as input in TextBoxes, stores the three numbers in variables and then finds the average of the numbers (note that the average is rounded to the nearest integer value). The output is displayed in a Label (Fig. 6.27, which displays the incorrect output). You soon realize, however, that the number displayed in the Label is not the average, but rather a number that does not make sense given the input. Use the debugger to help locate and remove this error. Average Three Numbers \_ 🗆 🗵 Enter first number: Calculate , 5 7 Enter second number: Enter third number: 11 Average is: 16 Figure 6.27 Average Three Numbers application for Exercise 6.16. a) Copying the template to your working directory. Copy the C:\Examples\Tutorial06\Exercises\AverageDebugging directory to your C:\SimplyVB directory. b) Opening the application's template file. Double click AverageDebugging.sln in the AverageDebugging directory to open the application. c) *Running the application*. Select **Debug > Start** to run your application. View the output to observe that the output is incorrect. d) *Closing the application.* Close the application, and view the Average.vb file in code view. e) Setting breakpoints. Set a breakpoint in the btnCalculate\_Click event handler. Run the application again, and use the debugger to help find the error(s). f) *Finding and correcting the error(s)*. Once you have found the error(s), modify the application so that it correctly calculates the average of three numbers. g) *Running the application*. Select **Debug > Start** to run your application. Enter the three values from Fig. 6.27 into the input TextBoxes provided and click the Calculate Button. Verify that the output now accurately reflects the average of these values, which is 8.

- h) Closing the application. Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

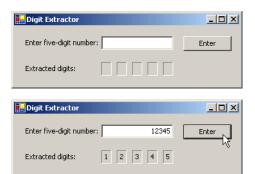
**Answer:** 

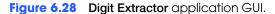
```
1
     ' Exercise 6.16 Solution
2
    ' Average.vb
3
4
    Public Class FrmAverageDebugging
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' handles Click event
 8
       Private Sub btnCalculate_Click(ByVal sender As _
9
          System.Object, ByVal e As System.EventArgs) _
10
          Handles btnCalculate.Click
11
12
           ' variables to store user inputs
13
          Dim intNumber1 As Integer
14
          Dim intNumber2 As Integer
15
          Dim intNumber3 As Integer
16
          Dim intAverage As Integer
17
18
           ' obtain user inputs
19
          intNumber1 = Val(txtFirstNumber.Text)
20
           intNumber2 = Val(txtSecondNumber.Text)
21
          intNumber3 = Val(txtThirdNumber.Text)
22
23
           ' average numbers
24
          intAverage = (intNumber1 + intNumber2 + intNumber3) / 3
25
          lblResult.Text = intAverage ' display result
26
27
       End Sub ' btnCalculate_Click
28
29
    End Class ' FrmAverageDebugging
```

**Answer:** The original code only divided the third number (intNumber3) by 3 when in fact the average ought to have been the sum of intNumber1, intNumber2 and intNumber3 divided by three. To correct the error, we included proper parentheses before intNumber1 and after intNumber3.

Programming Challenge 🕨

**6.17** (Digit Extractor Application) Write an application that allows the user to enter a fivedigit number into a TextBox. The application then separates the number into its individual digits and displays each digit in a Labe1. The application should look and behave similarly to Fig. 6.28. [*Hint:* You can use the Mod operator to extract the ones digit from a number. For instance, 12345 Mod 10 is 5. You can use integer division (\) to "peel off" digits from a number. For instance, 12345 \ 100 is 123. This allows you to treat the 3 in 12345 as a ones digit. Now you can isolate the 3 by using the Mod operator. Apply this technique to the rest of the digits.]





- a) **Creating the application.** Create a new project named DigitExtractor. Rename the Form1.vb file DigitExtractor.vb. Change the name of the Form to FrmDigitExtractor and set the startup object to FrmDigitExtractor. Add Labels, a Text-Box and a Button to the application's Form. Name the TextBox txtInput and name the Button btnEnter. Name the other controls logically based on the tips provided in earlier tutorials.
- b) Adding an event handler for btnEnter's Click event. In design view, double click btnEnter to create the btnEnter\_Click event handler. In this event handler, create five variables of type Integer. Use the Mod operator to extract each digit. Store the digits in the five variables created.
- c) Adding an event handler for txtInput's TextChanged event. In design view, double click txtInput to create the txtInput\_TextChanged event handler. In this event handler, clear the five Labels used to display each digit. This event handler clears the output whenever new input is entered.
- d) *Running the application.* Select **Debug > Start** to run your application. Enter a fivedigit number and click the **Enter** Button. Enter a new five-digit number and verify that the previous output is cleared.
- e) *Closing the application*. Close your running application by clicking its close box.
- f) *Closing the IDE*. Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 6.17 Solution
 2
    ' DigitExtractor.vb
 3
 4
    Public Class FrmDigitExtractor
 5
        Inherits System.Windows.Forms.Form
 6
 7
        ' handles Click event
 8
        Private Sub btnEnter_Click(ByVal sender As System.Object, _
 0
           ByVal e As System. EventArgs) Handles btnEnter. Click
10
11
           Dim intNumber As Integer ' five-digit number
12
13
           ' five variables for five digits
14
           Dim intFirst As Integer
15
           Dim intSecond As Integer
16
           Dim intThird As Integer
17
           Dim intFourth As Integer
18
           Dim intFifth As Integer
19
20
           intNumber = Val(txtInput.Text) ' obtain user input
21
22
           ' extract each digit
23
           intFirst = intNumber \setminus 10000
24
           intSecond = intNumber \setminus 1000 Mod 10
```

25	intThird = intNumber $\setminus$ 100 Mod 10
26	intFourth = intNumber $\setminus$ 10 Mod 10
27	intFifth = intNumber Mod 10
28	
29	' display extracted digits
30	lblFirstDigit.Text = intFirst
31	<pre>lblSecondDigit.Text = intSecond</pre>
32	lblThirdDigit.Text = intThird
33	<pre>lblFourthDigit.Text = intFourth</pre>
34	lblFifthDigit.Text = intFifth
35	End Sub ' btnEnter_Click
36	
37	' handles TextChanged event
38	<pre>Private Sub txtInput_TextChanged(ByVal sender As System.Object, _</pre>
39	ByVal e As System.EventArgs) Handles txtInput.TextChanged
40	
41	' clear Labels
42	lblFirstDigit.Text = ""
43	<pre>lblSecondDigit.Text = ""</pre>
<b>44</b>	<pre>lblThirdDigit.Text = ""</pre>
45	<pre>lblFourthDigit.Text = ""</pre>
46	lblFifthDigit.Text = ""
47	End Sub ' txtInput_TextChanged
<b>48</b>	
49	End Class ' FrmDigitExtractor





# Wage Calculator Application

Introducing Algorithms, Pseudocode and Program Control Solutions

Instructor's Manual Exercise Solutions Tutorial 7		
MULTIPLE-CHOICE	7.1 The operator returns and.	False if the left operand is larger than the right oper-
QUESTIONS	a) =	b) <
	c) <=	d) All of the above.
	<b>7.2</b> A occurs when an exe executed statement in the written prop	cuted statement does not directly follow the previously gram.
	a) transition	b) flow
	c) logical error	d) transfer of control
	<b>7.3</b> A variable or an expression th	nat is added to the Watch window is known as a
	a) watched variable	b) watched expression
	c) watch	d) watched value
	<b>7.4</b> The IfThen statement is called one action.	a statement because it selects or ignores
	a) single-selection	b) multiple-selection
	c) double-selection	d) repetition
	<b>7.5</b> The three types of control statem ment and the	nents are the sequence statement, the selection state-
	a) repeat	b) looping
	c) redo	d) repetition
	7.6 In an activity diagram, a rectangle	e with curved sides represents
	a) a complete algorithm	b) a comment
	c) an action	d) the termination of the application
	7.7 The IfThenElse selection sta	atement ends with the keywords
	a) End If Then Else	<pre>b) End If Else</pre>
	c) End Else	d) End If
	<b>7.8</b> A variable of data type Boolea	n can be assigned keyword or keyword
	a) True, False	b) 0ff, 0n
	c) True, NotTrue	d) Yes, No
	7.9 A variable whose value cannot	be changed after its initial declaration is called a
	a) Double	b) constant
	c) standard	d) Boolean
	,	s the result of adding the left and right operands to the
	a) +	b) =+
	c) +=	d) + =
	<b>Answers:</b> 7.1) d. 7.2) d. 7.3) c. 7.4) a.	7.5) d. 7.6) c. 7.7) d. 7.8) a. 7.9) b. 7.10) c.

**EXERCISES** 

**7.11** (*Currency Converter Application*) Develop an application that functions as a currency converter as shown in Fig. 7.33. Users must provide a number in the **Dollars:** TextBox and a currency name (as text) in the **Convert from Dollars to:** TextBox. Clicking the **Convert** Button will convert the specified amount into the indicated currency and display it in a

Label. Limit yourself to the following currencies as user input: Dollars, Euros, Yen and Pesos. Use the following exchange rates: 1 Dollar = 1.02 Euros, 120 Yen and 10 Pesos.

Currency Converter	
Dollars to Convert:	20.00
Convert from Dollars to:	Yen
Converted Amount:	2400.00
	Convert



- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial07\Exercises\CurrencyConverter directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click CurrencyConverter.sln in the CurrencyConverter directory to open the application.
- c) Add an event handler for the Convert Button's Click event. Double click the Convert Button to generate an empty event handler for the Button's Click event. The code for *Steps d–f* belongs in this event handler.
- d) **Obtaining the user input.** Use the Val function to convert the user input from the **Dollars:** TextBox to a Double. Assign the Double to a Decimal variable decAmount. Visual Basic .NET implicitly performs this conversion from Double to Decimal.
- e) Performing the conversion. Use an If...ElseIf...ElseIf statement to determine which currency the user entered. Assign the result of the conversion to decAmount.
- f) Displaying the result. Display the result using method String. Format with format specifier F.
- g) Running the application. Select Debug > Start to run your application. Enter a value in dollars to be converted and the name of the currency you wish to convert to. Click the **Convert** Button and, using the exchange rates above, verify that the correct output is displays.
- h) Closing the application. Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

#### **Answer:**

1	' Exercise 7.11 Solution
2	' CurrencyConverter.vb
3	
4	Public Class FrmCurrencyConverter
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form designer generated code
8	
9	' handles Click event
10	<pre>Private Sub btnConvert_Click(ByVal sender As System.Object, _</pre>
11	ByVal e As System. EventArgs) Handles btnConvert. Click
12	
13	Dim decAmount As Decimal
14	
15	decAmount = Val(txtValue.Text) ' obtain dollar amount
16	
17	' perform currency conversion
18	If $txtCurrency.Text = "Euros" Then$
19	decAmount $*= 1.02$
20	
21	<pre>ElseIf txtCurrency.Text = "Yen" Then</pre>

```
22
              decAmount *= 120
23
24
           ElseIf txtCurrency.Text = "Pesos" Then
25
              decAmount *= 10
26
           End If
27
28
           lblConvertedResult.Text = String.Format("{0:F}", decAmount)
29
        End Sub ' btnConvert_Click
30
     End Class ' FrmCurrencyConverter
31
```

7.12 (Wage Calculator Application that Performs Tax Calculations) Develop an application that calculates an employee's wages as shown in Fig. 7.34. The user should provide the hourly wage and number of hours worked per week. When the Calculate Button is clicked, the gross earnings of the user should display in the Gross earnings: TextBox. The Less FWT: TextBox should display the amount deducted for Federal taxes and the Net earnings: TextBox displays the difference between the gross earnings and the Federal tax amount. Assume overtime wages are 1.5 times the hourly wage and Federal taxes are 15% of gross earnings. The Clear Button should clear all fields.

<mark>4 W</mark> age Calcula	tor _ 🗆 🗙
Hourly wage:	12
Weekly hours:	40
Gross earnings:	\$480.00
Less FWT:	\$72.00
Net earnings:	\$408.00
Calculate	Clear

Figure 7.34 Wage Calculator GUI.

- a) **Copying the template to your working directory.** Copy the C:\Examples\Tutorial07\Exercises\ExpandedWageCalculator directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click WageCalculator.sln in the ExpandedWageCalculator directory to open the application.
- c) Modifying the Calculate Button's Click event handler. Add the code for Steps d-f to btnCalculate\_Click.
- d) *Adding a new variable*. Declare decFederalTaxes to store the amount deducted for Federal taxes.
- e) *Calculating and displaying the Federal taxes deducted.* Multiply the total earnings (decEarnings) by 0.15 (that is, 15%) to determine the amount to be removed for taxes. Assign the result to decFederalTaxes. Display this value using method String.Format with format specifier C.
- f) *Calculating and displaying the employee's net pay.* Subtract decFederalTaxes from decEarnings to calculate the employee's net earnings. Display this value using method String.Format with format specifier C.
- g) *Creating an event handler for the Clear Button.* Double click the **Clear** Button to generate an empty event handler for the Click event. This event handler should clear user input from the two TextBoxes and the results from the three Labels.
- h) Running the application. Select Debug > Start to run your application. Enter an hourly wage and the number of hours worked. Click the Calculate Button and verify that the appropriate output is displayed for gross earnings, amount taken out for federal taxes and the net earnings. Click the Clear Button and check that all fields are cleared.

- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
Answer:
```

```
1
    ' Exercise 7.12 Solution
 2
    ' WageCalculator.vb
 3
 4
    Public Class FrmWageCalculator
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form designer generated code
 8
 9
        ' handles Click event
       Private Sub btnCalculate_Click(ByVal sender As System.Object, _
10
11
          ByVal e As System. EventArgs) Handles btnCalculate. Click
12
13
           ' declare variables
14
          Dim dblHours As Double
15
          Dim decWage As Decimal
16
          Dim decEarnings As Decimal
17
          Dim decFederalTaxes As Decimal
18
          Const intHOUR_LIMIT As Integer = 40 ' declare constant
19
20
           ' assign values from user input
21
           dblHours = Val(txtHours.Text)
22
           decWage = Val(txtWage.Text)
23
24
           ' determine wage amount
25
           If dblHours <= intHOUR_LIMIT Then</pre>
26
27
              ' if under or equal to 40 hours, regular wages
28
              decEarnings = dblHours * decWage
29
           Else
30
31
              ' if over 40 hours, regular wages for first 40
32
              decEarnings = intHOUR_LIMIT * decWage
33
34
              ' time and a half for the additional hours
35
              decEarnings += _
36
                 (dblHours - intHOUR_LIMIT) * (1.5 * decWage)
37
           End If
38
39
           ' assign gross pay to the corresponding Label
40
           lblEarningsResult.Text = String.Format("{0:C}", decEarnings)
41
42
           ' assign federal taxes to the corresponding Label
43
           decFederalTaxes = decEarnings * 0.15
44
          lblFWTNumber.Text = String.Format("{0:C}", decFederalTaxes)
45
46
           ' assign net pay to the corresponding Label
47
           lblTotal.Text = String.Format("{0:C}", decEarnings - _
48
              decFederalTaxes)
49
        End Sub ' btnCalculate Click
50
51
        ' handles Clear Button's Click event
52
        Private Sub btnClear_Click(ByVal sender As System.Object, _
53
          ByVal e As System. EventArgs) Handles btnClear. Click
54
55
           ' clear each TextBox and output Label
56
           txtWage.Text = ""
           txtHours.Text = ""
57
```

58	lblEarningsResult.Text = ""	
59	<pre>lblFWTNumber.Text = ""</pre>	
60	lblTotal.Text = ""	
61	End Sub ' btnClear_Click	
62		
63	End Class ' FrmWageCalculator	

**7.13** (*Customer Charge Account Analyzer Application*) Develop an application (as shown in Fig. 7.35) that determines whether a department-store customer has exceeded the credit limit on a charge account. Each customer enters an account number (an Integer), a balance at the beginning of the month (a Decimal), the total of all items charged this month (a Decimal), the total of all credits applied to the customer's account this month (a Decimal), and the customer's allowed credit limit (a Decimal). The application should input each of these facts, calculate the new balance (= *beginning balance - credits + charges*), display the new balance and determine whether the new balance exceeds the customer's credit limit. If the customer's credit limit is exceeded, the application should display a message (in a Label at the bottom of the Form) informing the customer of this fact.

🛃 Credit Checker	-O×
Account number:	12345
Starting balance:	100.00
Total charges:	20.00
Total credits:	35.00
Credit limit:	500.00
New balance:	\$85.00
	Calculate Balance

Figure 7.35 Credit Checker GUI.

- a) *Copying the template application to your working directory.* Copy the C:\Examples\Tutorial07\Exercises\CreditChecker directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click Credit Checker.sln in the CreditChecker directory to open the application.
- c) Adding the Calculate Button's Click event handler. Double click the Calculate Balance Button to generate the empty event handler for the Click event. The code for *Steps d-g* is added to this event handler.
- d) Declaring variables. Declare an Integer variable to store the account number. Declare four Decimal variables to store the starting balance, charges, credits and credit limit. Declare a fifth Decimal variable to store the new balance in the account after the credits and charges have been applied.
- e) **Obtaining user input.** Obtain the user input from the TextBoxes' Text properties.
- f) *Calculating and displaying the new balance*. Calculate the new balance by adding the total credits to the starting balance and subtracting the charges. Assign the result to a variable. Display the result formatted as currency.
- g) *Determining if the credit limit has been exceeded.* If the new balance exceeds the specified credit limit, a message should be displayed in lblError.
- h) Handling the Account number: TextBox's TextChanged event. Double click the Account number: TextBox to generate its TextChanged event handler. This event handler should clear the other TextBoxes, the error message Label and the result Label.

- i) *Running the application.* Select **Debug > Start** to run your application. Enter an account number, your starting balance, the amount charged to your account, the amount credited to your account and your credit limit. Click the **Calculate Balance** Button and verify that the new balance displayed is correct. Enter an amount charged that exceeds your credit limit. Click the **Calculate Balance** Button and ensure that a message is displayed in the lower Label.
- j) *Closing the application.* Close your running application by clicking its close box.
- k) *Closing the IDE.* Close Visual Studio .NET by clicking its close box.

41 42 End If 43 44 End Sub ' btnCalculate_Click 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccounttNumber.TextChanged	1	' Exercise 7.13 Solution
<pre>4 Public Class FrmCreditChecker 5 Inherits System.Windows.Forms.Form 6 7 'Windows Form designer generated code 8 9 ' handles Calculate Button's Click event 9 rivate Sub btnCalculate_Click(ByVal sender As System.Object, _ 9 ByVal e As System.EventArgs) Handles btnCalculate.Click 12 13 ' declare variables 14 Dim intAccountNumber As Integer 15 Dim decTotalCharges As Decimal 16 Dim decTotalCharges As Decimal 17 Dim decTotalCharges As Decimal 18 Dim decCreditLimit As Decimal 19 Dim decTotalCharges Val(txtCountNumber.Text) 10 deCtotalCharges Val(txtTotalCharges.Text) 10 deCtotalCharges Val(txtTotalCharges.Text) 11 decTotalCharges Val(txtTotalCharges.Text) 12 decTotalCredits = Val(txtTotalCharges.Text) 13 decCreditLimit = Val(txtCreditLimit.Text) 14 decTotalCredits = Val(txtTotalCharges 15 ' calculate balance after credits and charges 16 decNewBalance = decStartBalance 17 decNewBalance = decStartBalance 18 decNewBalance = decStartBalance 19 decTotalCredits + decTotalCharges 10 ' display new balance in corresponding Label 11 blNewBalanceNumber.Text = String.Format("{0:C}", _ 11 decNewBalance) 12 ' determine if credit limit has been exceeded 13 ' display an error message 14 lblPerror.Text = "Credit Limit Exceeded!" 14 End If 15 End Sub ' btnCalculate_Click 16 ' handles TextChanged event 17 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 18 As System.Object, ByVal e As System.EventArgs) _ 18 Handles txtAccountNumber.TextChanged</pre>	2	' CreditChecker.vb
<pre>5 Inherits System.Windows.Forms.Form 6 7 'Windows Form designer generated code 8 9 'handles Calculate Button's Click event 9 Private Sub btnCalculate_Click(ByVal sender As System.Object, _ 8 ByVal e As System.EventArgs) Handles btnCalculate.Click 10 declare variables 11 Dim intAccountNumber As Integer 12 Dim decTotalCharges As Decimal 13 Dim decTotalCradits As Decimal 14 Dim decTotalCradits As Decimal 15 Dim decCreditLimit As Decimal 16 Dim decCreditLimit As Decimal 17 Dim decTotalCradits As Decimal 18 Dim decCreditLimit As Decimal 19 Dim decTotalCradits = Val(txtStartBalance.Text) 10 decTotalCradits = Val(txtCradICharges.Text) 10 decTotalCredits = Val(txtCradICharges.Text) 11 decTotalCredits = Val(txtTotalCredits.Text) 12 decTotalCredits = Val(txtTotalCredits.Text) 13 decTotalCredits + decTotalCharges 14 desplay new balance after credits and charges 15 decNewBalance = decStartBalance 16 decNewBalance &gt; decStartBalance 17 decNewBalance &gt; decCreditLimit Text) 18 decNewBalance &gt; decCreditLimit Then 19 decNewBalance &gt; decCreditLimit Then 10 lNewBalances &gt; decCreditLimit Then 11 decNewBalance &gt; decCreditLimit Then 12 detNewBalance &gt; decCreditLimit Then 13 ' if credit limit has been exceeded 14 display nerror message 16 llError.Text = "Credit Limit Exceeded!" 14 End If 15 End Sub ' btnCalculate_Click 16 'handles TextChanged event 17 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 18 As System.Object, ByVal e As System.EventArgs) _ 18 Handles txtAccountNumber_TextChanged</pre>	3	
<pre>6 7 7 7 7 7 7 7 8 7 7 7 7 7 8 7 8 7 7 7 7 7 8 7 8 7 8 7 9 7 8 7 8</pre>	4	Public Class FrmCreditChecker
<pre>' Windows Form designer generated code ' handles Calculate Button's Click event Private Sub btnCalculate_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnCalculate.Click ' declare variables ' declare variables Dim intAccountNumber As Integer Dim decTotalCradges As Decimal Dim decTotalCradges As Decimal Dim decTotalCredits As Decimal Dim decTotalCredits As Decimal intAccountNumber = Val(txtAccountNumber.Text) decStartBalance = Val(txtTotalCharges.Text) decTotalCredits = Val(txtTotalCharges.Text) decTotalCredits = Val(txtTotalCharges Text) decCreditLimit = Val(txtCreditLimit.Text) ' calculate balance after credits and charges decNewBalance = decStartBalance decTotalCredits + decTotalCharges ' determine if credit limit has been exceeded ' display new balance in corresponding Label ' determine if credit limit has been exceeded ' display an error message DibTerror.Text = "Credit Limit Exceeded!" ' detare time if credit Limit Exceeded!" Lend If End Sub ' btnCalculate_Click ' handles TextChanged event Private Sub txtAccountNumber_TextChanged(ByVal sender _ As System.Object, ByVal e As System.EventArgs) _ Handles txtAccountNumber.TextChanged</pre>	5	Inherits System.Windows.Forms.Form
<pre>% ' handles Calculate Button's Click event Private Sub btnCalculate_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnCalculate.Click / declare variables Dim intAccountNumber As Integer Dim decStartBalance As Decimal Dim decTotalCharges As Decimal Dim decTotalCredits As Decimal Dim decCreditLimit As Decimal 0 intAccountNumber = Val(txtAccountNumber.Text) decStartBalance = Val(txtStartBalance.Text) decTotalCredits = Val(txtTotalCharges.Text) decTotalCredits = Val(txtTotalCharges.Text) decTotalCredits = Val(txtTotalCredits.Text) decCreditLimit = Val(txtTordits.Text) decCreditLimit = Val(txtTorditLimit.Text) / calculate balance after credits and charges decNewBalance = decStartBalance decTotalCredits + decTotalCharges / display new balance in corresponding Label lblNewBalanceNumber.Text = String.Format("{0:C}", _ decNewBalance &gt; decCreditLimit Then / if credit limit has been exceeded / display an error message lblError.Text = "Credit Limit Exceeded!" / taiplay an error message lblError.Text = "Credit Limit Exceeded!" / handles TextChanged event / handles TextChanged event / Private Sub txtAccountNumber_TextChanged(ByVal sender _ As System.Object, ByVal e As System.EventArgs) _ Handles txtAccountNumber.TextChanged</pre>	6	
<pre>% ' handles Calculate Button's Click event Private Sub btnCalculate_Click(ByVal sender As System.Object,</pre>	7	' Windows Form designer generated code
<pre>Private Sub btnCalculate_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnCalculate.Click ' declare variables Dim intAccountNumber As Integer Dim decStartBalance As Decimal Dim decTotalCredits As Decimal Dim decTotalCredits As Decimal Dim decCreditLimit As Decimal Dim decCreditLimit As Decimal intAccountNumber = Val(txtAccountNumber.Text) decStartBalance = Val(txtStartBalance.Text) decTotalCredits = Val(txtTotalCredits.Text) decTotalCredits = Val(txtCreditLimit.Text) decCreditLimit = Val(txtCreditLimit.Text) cecCreditLimit = Val(txtCreditLimit.Text) cecCreditLimit = Val(txtCreditLimit.Text) decTotalCredits + decTotalCharges decNewBalance &gt; decCreditLimit Then cecNewBalance &gt; decCreditLimit Then cecNewBalance &gt; decCreditLimit Then cecNewBalance &gt; decCreditLimit Exceeded centewBalance &gt; decCreditLimit Exceeded cecNewBalance &gt; decCreditLimit Then cecNewBalance &gt; decCreditLimit Exceeded cectPaceNameerTextChanged(ByVal Sender _ cecNameerTextChanged ExterChanged(ByVal Sender _ cecNameerTextChanged cectPaceNameerTextChanged</pre>		
<pre>Private Sub btnCalculate_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnCalculate.Click ' declare variables Dim intAccountNumber As Integer Dim decStartBalance As Decimal Dim decTotalCredits As Decimal Dim decTotalCredits As Decimal Dim decCreditLimit As Decimal Dim decCreditLimit As Decimal intAccountNumber = Val(txtAccountNumber.Text) decStartBalance = Val(txtStartBalance.Text) decTotalCredits = Val(txtTotalCredits.Text) decTotalCredits = Val(txtCreditLimit.Text) decCreditLimit = Val(txtCreditLimit.Text) cecCreditLimit = Val(txtCreditLimit.Text) cecCreditLimit = Val(txtCreditLimit.Text) decTotalCredits + decTotalCharges decNewBalance &gt; decCreditLimit Then cecNewBalance &gt; decCreditLimit Then cecNewBalance &gt; decCreditLimit Then cecNewBalance &gt; decCreditLimit Exceeded centewBalance &gt; decCreditLimit Exceeded cecNewBalance &gt; decCreditLimit Then cecNewBalance &gt; decCreditLimit Exceeded cectPaceNameerTextChanged(ByVal Sender _ cecNameerTextChanged ExterChanged(ByVal Sender _ cecNameerTextChanged cectPaceNameerTextChanged</pre>	9	' handles Calculate Button's Click event
ByVal e As System.EventArgs) Handles btnCalculate.Click         '       declare variables         Dim intAccountNumber As Integer         Dim decTotalCharges As Decimal         Dim decTotalCharges As Decimal         Dim decTotalCredits As Decimal         Dim decCreditLimit As Decimal         Dim decCreditLimit As Decimal         Dim decCreditLimit As Decimal         Dim decCreditLimit As Decimal         destartBalance = Val(txtAccountNumber.Text)         decStartBalance = Val(txtTotalCharges.Text)         decTotalCredits = Val(txtTotalCredits.Text)         decTotalCredits = Val(txtTotalCharges.Text)         decTotalCredits = Val(txtTotalCharges.Text)         decTotalCredits = Val(txtTotalCharges.Text)         decCotalCredits = Val(txtTotalCharges.Text)         decNewBalance = decStartBalance         decTotalCredits + decTotalCharges         decNewBalance = decStartBalance         decNewBalance)         '         '         deleNewBalance > decCreditLimit_Text)         determine if credit limit has been exceeded         if credit limit has been exceeded         '       if credit limit has been exceeded         '       if credit limit has been exceeded         '       ibllError.Text = "Credit Limit Exceeded!"	10	
<pre>12 13 ' declare variables 14 Dim intAccountNumber As Integer 15 Dim decStartBalance As Decimal 16 Dim decTotalCharges As Decimal 17 Dim decTotalCredits As Decimal 18 Dim decCreditLimit As Decimal 19 Dim decNewBalance As Decimal 20 21 intAccountNumber = Val(txtAccountNumber.Text) 22 decStartBalance = Val(txtTotalCharges.Text) 23 decTotalCharges = Val(txtTotalCharges.Text) 24 decTotalCredits = Val(txtTotalCredits.Text) 25 decCreditLimit = Val(txtCreditLimit.Text) 26 27 ' calculate balance after credits and charges 28 decNewBalance = decStartBalance 29 decTotalCredits + decTotalCharges 20 ' display new balance in corresponding Label 21 blNewBalanceNumber.Text = String.Format("{0:C}", _ 33 decNewBalance &gt; decCreditLimit Then 34 ' if credit limit has been exceeded 35 ' determine if credit limit thes been exceeded 36 If decNewBalance &gt; decCreditLimit Exceeded!" 37 38 ' if credit limit has been exceeded 39 ' display an error message 39 ' blError.Text = "Credit Limit Exceeded!" 34 44 End If 45 End Sub ' btnCalculate_Click 45 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountNumber.TextChanged 40 41 41 42 As Display an externation 43 As System.Object, ByVal e As System.EventArgs) _ 44 As Display an externation 44 As Display an externation 45 As Display and externation 46 As Display and externation 47 As System.Object, ByVal e As System.EventArgs) _ 48 As Display and externation 49 As Display and externation 40 As Display and externation 41 As Display and Party As Display and Party As Display A</pre>		
<pre>13 ' declare variables 14 Dim intAccountNumber As Integer 15 Dim decStartBalance As Decimal 16 Dim decTotalCredits As Decimal 17 Dim decTotalCredits As Decimal 18 Dim decCreditLimit As Decimal 19 Dim decNewBalance As Decimal 20 21 intAccountNumber = Val(txtAccountNumber.Text) 22 decStartBalance = Val(txtTotalCharges.Text) 23 decTotalCredits = Val(txtTotalCharges.Text) 24 decTotalCredits = Val(txtTotalCredits.Text) 25 decCreditLimit = Val(txtCreditLimit.Text) 26 27 ' calculate balance after credits and charges 28 decNewBalance = decStartBalance 29 decTotalCredits + decTotalCharges 20 31 ' display new balance in corresponding Label 32 lblNewBalanceNumber.Text = String.Format("{0:C}", _ 33 decNewBalance) 34 35 ' determine if credit limit has been exceeded 36 If decNewBalance &gt; decCreditLimit Then 37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountNumber.TextChanged</pre>		,
14Dim intAccountNumber As Integer15Dim decStartBalance As Decimal16Dim decTotalCredits As Decimal17Dim decTotalCredits As Decimal18Dim decCreditLimit As Decimal19Dim decNewBalance As Decimal20intAccountNumber = Val(txtAccountNumber.Text)21intAccountNumber = Val(txtStartBalance.Text)22decStartBalance = Val(txtTotalCharges.Text)23decTotalCredits = Val(txtTotalCredits.Text)24decTotalCredits = Val(txtCreditLimit.Text)25decCreditLimit = Val(txtCreditLimit.Text)26' calculate balance after credits and charges27' calculate balance after credits and charges28decNewBalance = decStartBalance29decTotalCredits + decTotalCharges30'31' display new balance in corresponding Label32lblNewBalanceNumber.Text = String.Format("{0:C}", _33' determine if credit limit has been exceeded34' determine if credit limit has been exceeded35' determine if credit Limit Exceeded!"36' display an error message37lblFror.Text = "Credit Limit Exceeded!"38' display an error message44End If35' btnCalculate_Click45' handles TextChanged event46Y handles TextChanged event47Private Sub txtAccountNumber_TextChanged(ByVal sender _48As System.Object, ByVal e As System.EventArgs) _49Handles txtAccounttNumber.Te		' declare variables
15       Dim decStartBalance As Decimal         16       Dim decTotalCharges As Decimal         17       Dim decTotalCredits As Decimal         18       Dim decCreditLimit As Decimal         19       Dim decNewBalance As Decimal         20       intAccountNumber = Val(txtStartBalance.Text)         21       intAccountNumber = Val(txtTotalCharges.Text)         22       decStartBalance = Val(txtTotalCredits.Text)         23       decTotalCredits = Val(txtTotalCredits.Text)         24       decTotalCredits = Val(txtTotalCredits.Text)         25       decCreditLimit = Val(txtTotalCredits.Text)         26       decNewBalance after credits and charges         27       ' calculate balance after credits and charges         28       decNewBalance = decStartBalance         29       decTotalCredits + decTotalCharges         30       ' display new balance in corresponding Label         31       ' determine if credit limit has been exceeded         36       ' determine if credit limit has been exceeded         37       ' display an error message         38       ' if credit limit has been exceeded!"         39       ' btnCalculate_Click         40       Inf         41       End Sub ' btnCalculate_Click		
16       Dim decTotalCharges As Decimal         17       Dim decTotalCredits As Decimal         18       Dim decCreditLimit As Decimal         19       Dim decNewBalance As Decimal         20       intAccountNumber = Val(txtStartBalance.Text)         21       intAccountNumber = Val(txtTotalCharges.Text)         22       decStartBalance = Val(txtTotalCharges.Text)         23       decTotalCredits = Val(txtTotalCredits.Text)         24       decTotalCredits = Val(txtTotalCredits.Text)         25       decCreditLimit = Val(txtCreditLimit.Text)         26       ' calculate balance after credits and charges         28       decNewBalance = decStartBalance decTotalCredits + decTotalCharges         30       ' display new balance in corresponding Label         31       ' display new balance in corresponding Label         32       lblNewBalance)         34       -         35       ' determine if credit limit has been exceeded         36       if credit limit has been exceeded         37       ' display an error message         38       ' if credit limit has been exceeded!"         44       End Sub ' btnCalculate_Click         45       ' handles TextChanged event         46       ' handles TextChanged event <td></td> <td></td>		
17Dim decTotalCredits As Decimal18Dim decCreditLimit As Decimal19Dim decNewBalance As Decimal20intAccountNumber = Val(txtAccountNumber.Text)21intAccountNumber = Val(txtStartBalance.Text)22decTotalCharges = Val(txtTotalCharges.Text)23decTotalCredits = Val(txtTotalCredits.Text)24decCreditLimit = Val(txtCreditLimit.Text)25decCreditLimit = Val(txtCreditLimit.Text)26' calculate balance after credits and charges27' calculate balance after credits and charges28decTotalCredits + decTotalCharges29decTotalCredits + decTotalCharges30' display new balance in corresponding Label31' blNewBalanceNumber.Text = String.Format("{0:C}", _33' determine if credit limit has been exceeded34' if credit limit has been exceeded35' if credit limit has been exceeded36' display an error message37' liplerror.Text = "Credit Limit Exceeded!"48End Sub ' btnCalculate_Click44End Sub ' btnCalculate_Click45' handles TextChanged event46As System.Object, ByVal e As System.EventArgs) _49Handles txtAccountNumber.TextChanged		
18       Dim decCreditLimit As Decimal         19       Dim decNewBalance As Decimal         20       intAccountNumber = Val(txtAccountNumber.Text)         22       decStartBalance = Val(txtTotalCharges.Text)         23       decTotalCharges = Val(txtTotalCharges.Text)         24       decTotalCredits = Val(txtTotalCharges.Text)         25       decCreditLimit = Val(txtCreditLimit.Text)         26       'calculate balance after credits and charges         27       'calculate balance in corresponding Label         28       decTotalCredits + decTotalCharges         29       decTotalCredits + decTotalCharges         30       'display new balance in corresponding Label         31       'display new balance in corresponding Label         32       lblNewBalanceNumber.Text = String.Format("{0:C}", _         33       'determine if credit limit has been exceeded         34       'determine if credit limit has been exceeded         35       'determine if credit Limit Exceeded!"         36       'if credit limit has been exceeded         37       'if credit limit has been exceeded         38       'if credit limit Exceeded!"         44       End Sub ' btnCalculate_Click         45       'handles TextChanged event         46       <		-
<pre>Dim decNewBalance As Decimal intAccountNumber = Val(txtAccountNumber.Text) decStartBalance = Val(txtStartBalance.Text) decTotalCharges = Val(txtTotalCharges.Text) decTotalCredits = Val(txtTotalCredits.Text) decCreditLimit = Val(txtCreditLimit.Text) calculate balance after credits and charges decNewBalance = decStartBalance decTotalCredits + decTotalCharges ' calculate balance in corresponding Label bblNewBalanceNumber.Text = String.Format("{0:C}", _ decNewBalance) ' determine if credit limit has been exceeded If decNewBalance &gt; decCreditLimit Then ' if credit limit has been exceeded ' display an error message bblError.Text = "Credit Limit Exceeded!" End If End Sub ' btnCalculate_Click ' handles TextChanged event Private Sub txtAccountNumber_TextChanged(ByVal sender _ As System.Object, ByVal e As System.EventArgs) _ Handles txtAccountNumber.TextChanged</pre>		
<pre>intAccountNumber = Val(txtAccountNumber.Text) decStartBalance = Val(txtStartBalance.Text) decTotalCharges = Val(txtTotalCharges.Text) decTotalCredits = Val(txtTotalCredits.Text) decCreditLimit = Val(txtCreditLimit.Text) ' calculate balance after credits and charges decNewBalance = decStartBalance decTotalCredits + decTotalCharges ' display new balance in corresponding Label lblNewBalanceNumber.Text = String.Format("{0:C}", _ decNewBalance) ' if credit limit has been exceeded ' display an error message lblError.Text = "Credit Limit Exceeded!" Lend If End Sub ' btnCalculate_Click ' handles TextChanged event Private Sub txtAccountNumber_TextChanged(ByVal sender _ As System.Object, ByVal e As System.EventArgs) _ Handles txtAccountNumber.TextChanged</pre>		
<pre>21 intAccountNumber = Val(txtAccountNumber.Text) 22 decStartBalance = Val(txtStartBalance.Text) 23 decTotalCharges = Val(txtTotalCharges.Text) 24 decTotalCredits = Val(txtTotalCredits.Text) 25 decCreditLimit = Val(txtCreditLimit.Text) 26 27 ' calculate balance after credits and charges 28 decNewBalance = decStartBalance 29 decTotalCredits + decTotalCharges 30 31 ' display new balance in corresponding Label 32 lblNewBalanceNumber.Text = String.Format("{0:C}", _ 33 decNewBalance) 34 35 ' determine if credit limit has been exceeded 36 If decNewBalance &gt; decCreditLimit Then 37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountNumber.TextChanged</pre>		
<pre>decStartBalance = Val(txtStartBalance.Text) decTotalCharges = Val(txtTotalCharges.Text) decTotalCredits = Val(txtTotalCredits.Text) decCreditLimit = Val(txtCreditLimit.Text)  ' calculate balance after credits and charges decNewBalance = decStartBalance     decTotalCredits + decTotalCharges  ' calculate balance in corresponding Label ' display new balance in corresponding Label ' display new balance in corresponding Label ' determine if credit limit has been exceeded ' determine if credit limit has been exceeded ' display an error message lblError.Text = "Credit Limit Exceeded!"  End If End Sub ' btnCalculate_Click ' handles TextChanged event Private Sub txtAccountNumber_TextChanged(ByVal sender _</pre>		intAccountNumber = Val(txtAccountNumber Text)
<pre>decTotalCharges = Val(txtTotalCharges.Text) decTotalCredits = Val(txtTotalCredits.Text) decCreditLimit = Val(txtCreditLimit.Text) ' calculate balance after credits and charges decNewBalance = decStartBalance decTotalCredits + decTotalCharges ' display new balance in corresponding Label blNewBalanceNumber.Text = String.Format("{0:C}", _ decNewBalance) ' determine if credit limit has been exceeded if decNewBalance &gt; decCreditLimit Then ' if credit limit has been exceeded ' display an error message blError.Text = "Credit Limit Exceeded!" End If End Sub ' btnCalculate_Click ' handles TextChanged event Private Sub txtAccountNumber_TextChanged(ByVal sender _ As System.Object, ByVal e As System.EventArgs) _ Handles txtAccountNumber.TextChanged</pre>		· · · · · · · · · · · · · · · · · · ·
<pre>decTotalCredits = Val(txtTotalCredits.Text) decCreditLimit = Val(txtCreditLimit.Text)  ' calculate balance after credits and charges decNewBalance = decStartBalance decTotalCredits + decTotalCharges ' detotalCredits + decTotalCharges ' display new balance in corresponding Label lblNewBalanceNumber.Text = String.Format("{0:C}", _ decNewBalance) ' determine if credit limit has been exceeded if decNewBalance &gt; decCreditLimit Then ' if credit limit has been exceeded ' display an error message lblError.Text = "Credit Limit Exceeded!" End If End Sub ' btnCalculate_Click ' handles TextChanged event Private Sub txtAccountNumber_TextChanged(ByVal sender _ As System.Object, ByVal e As System.EventArgs) _ Handles txtAccountNumber.TextChanged</pre>		
<pre>decCreditLimit = Val(txtCreditLimit.Text)  decCreditLimit = Val(txtCreditLimit.Text)  decNewBalance = decStartBalance     decTotalCredits + decTotalCharges  decTotalCredits + decTotalCharges  i display new balance in corresponding Label lblNewBalanceNumber.Text = String.Format("{0:C}", _     decNewBalance)  i determine if credit limit has been exceeded if decNewBalance &gt; decCreditLimit Then  i if credit limit has been exceeded i display an error message lblError.Text = "Credit Limit Exceeded!"  End If End If Lend Sub ' btnCalculate_Click  ' handles TextChanged event Private Sub txtAccountNumber_TextChanged(ByVal sender _     As System.Object, ByVal e As System.EventArgs) _ Handles txtAccountNumber.TextChanged</pre>		
<pre>26 27 ' calculate balance after credits and charges 28 decNewBalance = decStartBalance 29 decTotalCredits + decTotalCharges 30 31 ' display new balance in corresponding Label 32 lblNewBalanceNumber.Text = String.Format("{0:C}", _ 33 decNewBalance) 34 35 ' determine if credit limit has been exceeded 36 If decNewBalance &gt; decCreditLimit Then 37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountNumber.TextChanged</pre>		
<pre>' calculate balance after credits and charges decNewBalance = decStartBalance decTotalCredits + decTotalCharges ' display new balance in corresponding Label lblNewBalanceNumber.Text = String.Format("{0:C}", _ decNewBalance) ' determine if credit limit has been exceeded if decNewBalance &gt; decCreditLimit Then ' if credit limit has been exceeded ' display an error message lblError.Text = "Credit Limit Exceeded!" Lend If Lend Sub ' btnCalculate_Click ' handles TextChanged event Private Sub txtAccountNumber_TextChanged(ByVal sender _ As System.Object, ByVal e As System.EventArgs) _ Handles txtAccountNumber.TextChanged</pre>		
<pre>decNewBalance = decStartBalance decTotalCredits + decTotalCharges i display new balance in corresponding Label blNewBalanceNumber.Text = String.Format("{0:C}", _ decNewBalance) determine if credit limit has been exceeded if decNewBalance &gt; decCreditLimit Then i if credit limit has been exceeded d i display an error message blError.Text = "Credit Limit Exceeded!" End If End Sub ' btnCalculate_Click ' handles TextChanged event Private Sub txtAccountNumber_TextChanged(ByVal sender _ As System.Object, ByVal e As System.EventArgs) _ Handles txtAccountNumber.TextChanged</pre>		' calculate balance after credits and charges
<pre>decTotalCredits + decTotalCharges decTotalCredits + decTotalCharges display new balance in corresponding Label lblNewBalanceNumber.Text = String.Format("{0:C}", _ decNewBalance) d</pre>		
<pre>30 31 ' display new balance in corresponding Label 32 lblNewBalanceNumber.Text = String.Format("{0:C}", _ 33 decNewBalance) 34 35 ' determine if credit limit has been exceeded 36 If decNewBalance &gt; decCreditLimit Then 37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountNumber.TextChanged</pre>		
<pre>31 ' display new balance in corresponding Label 32 lblNewBalanceNumber.Text = String.Format("{0:C}", _ 33 decNewBalance) 34 35 ' determine if credit limit has been exceeded 36 If decNewBalance &gt; decCreditLimit Then 37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountTNumber.TextChanged</pre>		
<pre>32 lblNewBalanceNumber.Text = String.Format("{0:C}", _ 33 decNewBalance) 34 35 ' determine if credit limit has been exceeded 36 If decNewBalance &gt; decCreditLimit Then 37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountTNumber.TextChanged</pre>		' display new balance in corresponding Label
<pre>33 decNewBalance) 34 35 ' determine if credit limit has been exceeded 36 If decNewBalance &gt; decCreditLimit Then 37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccounttNumber.TextChanged</pre>		
<pre>34 35 ' determine if credit limit has been exceeded 36 If decNewBalance &gt; decCreditLimit Then 37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountNumber.TextChanged</pre>		
<pre>35 ' determine if credit limit has been exceeded 36 If decNewBalance &gt; decCreditLimit Then 37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountNumber.TextChanged</pre>		
<pre>36 If decNewBalance &gt; decCreditLimit Then 37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountTNumber.TextChanged</pre>		' determine if credit limit has been exceeded
<pre>37 38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountTNumber.TextChanged</pre>		
<pre>38 ' if credit limit has been exceeded 39 ' display an error message 40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccountTNumber.TextChanged</pre>		
<pre>39</pre>		' if credit limit has been exceeded
<pre>40 lblError.Text = "Credit Limit Exceeded!" 41 42 End If 43 44 End Sub ' btnCalculate_Click 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccounttNumber.TextChanged</pre>		
<pre>42 End If 43 44 End Sub ' btnCalculate_Click 45 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccounttNumber.TextChanged</pre>	40	
<ul> <li>43</li> <li>44 End Sub ' btnCalculate_Click</li> <li>45</li> <li>45</li> <li>46 ' handles TextChanged event</li> <li>47 Private Sub txtAccountNumber_TextChanged(ByVal sender _</li> <li>48 As System.Object, ByVal e As System.EventArgs) _</li> <li>49 Handles txtAccounttNumber.TextChanged</li> </ul>	41	
<pre>44 End Sub ' btnCalculate_Click 45 45 46 ' handles TextChanged event 47 Private Sub txtAccountNumber_TextChanged(ByVal sender _ 48 As System.Object, ByVal e As System.EventArgs) _ 49 Handles txtAccounttNumber.TextChanged</pre>	42	End If
<ul> <li>45</li> <li>46 ' handles TextChanged event</li> <li>47 Private Sub txtAccountNumber_TextChanged(ByVal sender _</li> <li>48 As System.Object, ByVal e As System.EventArgs) _</li> <li>49 Handles txtAccounttNumber.TextChanged</li> </ul>	43	
<ul> <li>45</li> <li>46 ' handles TextChanged event</li> <li>47 Private Sub txtAccountNumber_TextChanged(ByVal sender _</li> <li>48 As System.Object, ByVal e As System.EventArgs) _</li> <li>49 Handles txtAccounttNumber.TextChanged</li> </ul>	44	End Sub ' btnCalculate Click
<ul> <li>47 Private Sub txtAccountNumber_TextChanged(ByVal sender _</li> <li>48 As System.Object, ByVal e As System.EventArgs) _</li> <li>49 Handles txtAccounttNumber.TextChanged</li> </ul>	45	
<ul> <li>47 Private Sub txtAccountNumber_TextChanged(ByVal sender _</li> <li>48 As System.Object, ByVal e As System.EventArgs) _</li> <li>49 Handles txtAccounttNumber.TextChanged</li> </ul>	46	' handles TextChanged event
<ul> <li>48 As System.Object, ByVal e As System.EventArgs) _</li> <li>49 Handles txtAccounttNumber.TextChanged</li> </ul>	47	
49 Handles txtAccounttNumber.TextChanged	48	
	49	
	50	

51	' clear all fields when account number is changed
52	<pre>txtStartBalance.Text = ""</pre>
53	<pre>txtTotalCharges.Text = ""</pre>
54	<pre>txtTotalCredits.Text = ""</pre>
55	<pre>txtCreditLimit.Text = ""</pre>
56	lblNewBalanceNumber.Text = ""
57	lblError.Text = ""
58	
59	End Sub ' txtAccountNumber_TextChanged
60	
61	End Class ' FrmCreditChecker
61	End Class ' FrmCreditChecker

What does this code do?

**7.14** Assume that txtAge is a TextBox control and that the user has entered the value 27 into this TextBox. Determine the action performed by the following code:

```
1
    Dim intAge As Integer
 2
 3
    intAge = Val(txtAge.Text)
 4
 5
    If intAge < 0 Then</pre>
 6
        txtAge.Text = "Enter a value greater than or equal to zero."
 7
    ElseIf intAge < 13 Then</pre>
 8
        txtAge.Text = "Child"
 9
    ElseIf intAge < 20 Then</pre>
10
        txtAge.Text = "Teenager"
11
    ElseIf intAge < 30 Then</pre>
12
        txtAge.Text = "Young Adult"
13
    ElseIf intAge < 65 Then</pre>
14
        txtAge.Text = "Adult"
15
    Else
16
        txtAge.Text = "Senior Citizen"
17
    End If
```

**Answer:** This code prints text when an age is inputted into the txtAge TextBox. In this case, the statement txtAge.Text = "Young Adult" executes, because the value of intAge is below 30, but not less than 20.

What's wrong with this code? **7.15** Assume that 1b1AMPM is a Label control. Find the error(s) in the following code.

```
1
    Dim intHour As Integer
 2
3
    intHour = 14
 4
5
    If intHour < 11 Then
 6
       If intHour > 0 Then
 7
            lblAMPM.Text = "AM"
8
       End If
9
    Else
10
       lblAMPM.Text = "PM"
    ElseIf intHour > 23 Then
11
12
       lblAMPM.Text = "Time Error."
13
    End If
```

**Answer:** ElseIf cannot appear after an Else within the same If...Then...Else statement. The correct code should read:

```
1
    Dim intHour As Integer
 2
 3
    intHour = 14
 4
 5
    If intHour < 11 Then
 6
       If intHour > 0 Then
 7
            lblAMPM.Text = "AM"
 8
       End If
 9
    ElseIf intHour > 23 Then
10
        lblAMPM.Text = "Time Error."
11
    Else
12
        lblAMPM.Text = "PM"
13
    End If
```

### Using the Debugger 🕨

**7.16** (Grade Calculator Application) Copy the C:\Examples\Tutorial07\Debugger directory into your working directory. This directory contains the Grades application, which takes a number from the user and displays the corresponding letter grade. For values 90–100 it should display A; for 80–89, B, for 70–79, C, for 60–69, D and for anything lower, an F. Run the application. Enter the value 85 in the TextBox and click Calculate. Notice that the application displays D when it ought to display B. Select View > Code to enter the code editor and set as many breakpoints as you feel necessary. Select Debug > Start to use the debugger to help you find the error(s). Figure 7.36 shows the incorrect output when the value 85 is input.

🖶 Grade	- 🗆 🗵
Enter grade:	85
Letter grade:	D
Ca	lculate

Figure 7.36 Incorrect output for Grade application.

	1	' Exercise 7.16 Solution
	2	'Grades.vb
	3	
	4	Public Class FrmGrade
	5	Inherits System.Windows.Forms.Form
	6	
	7	' Windows Form designer generated code
	8	
	9	' handles Click event
	10	<pre>Private Sub btnCalculate_Click(ByVal sender As System.Object, _</pre>
	11	ByVal e As System.EventArgs) Handles btnCalculate.Click
	12	
	13	Dim intGrade As Integer
	14	
	15	<pre>intGrade = Val(txtGrade.Text)</pre>
	16	
	17	' display grade corresponding to number
	18	If intGrade >= 90 Then
	19	lblDisplay.Text = "A"
Individual IfEnd If	20	ElseIf intGrade >= 80 Then
statements replaced with one	21	lblDisplay.Text = "B"
IfElseIfElse statement	22	<pre>ElseIf intGrade &gt;= 70 Then</pre>
	23	lblDisplay.Text = "C"
	24	Elself intGrade >= 60 Then

```
25 lblDisplay.Text = "D"
26 Else
27 lblDisplay.Text = "F"
28 End If
29
30 End Sub ' btnCalculate_Click
31
32 End Class ' FrmGrade
```

### Programming Challenge

**7.17** (*Encryption Application*) A company transmits data over the telephone, but it is concerned that its phones could be tapped. All its data is transmitted as four-digit Integers. The company has asked you to write an application that encrypts its data so that it may be transmitted more securely. Encryption is the process of transforming data for security reasons. Create a Form similar to Fig. 7.37. Your program should read four-digits entered by the user and encrypt the information as follows:

- a) Replace each digit by (*the sum of that digit plus 7*) *modulo 10*. We use the term **modulo** to indicate you are to use the modulus (Mod) operator.
- b) Swap the first digit with the third, and swap the second digit with the fourth.



Figure 7.37 Encryption application.

1	' Exercise 7.17 Solution
2	' Encryption.vb
3	
4	Public Class FrmEncryption
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form designer generated code
8	
9	' handles Click event
10	<pre>Private Sub btnEncrypt_Click(ByVal sender As System.Object, _</pre>
11	ByVal e As System.EventArgs) Handles btnEncrypt.Click
12	
13	' clear previous output
14	<pre>lblEncryptedNumber1.Text = ""</pre>
15	<pre>lblEncryptedNumber2.Text = ""</pre>
16	<pre>lblEncryptedNumber3.Text = ""</pre>
17	<pre>lblEncryptedNumber4.Text = ""</pre>
18	
19	Dim intNumber1 As Integer
20	Dim intNumber2 As Integer
21	Dim intNumber3 As Integer
22	Dim intNumber4 As Integer
23	
24	' retrieve numbers from TextBoxes
25	<pre>intNumber1 = Val(txtNumber1.Text)</pre>
26	intNumber2 = Val(txtNumber2.Text)

27	intNumber3 = Val(txtNumber3.Text)
28	intNumber4 = Val(txtNumber4.Text)
29	
30	' convert to 1-digit numbers
31	If intNumber1 > 9 Then
32	<pre>intNumber1 = intNumber1 Mod 10</pre>
33	<pre>ElseIf intNumber2 &gt; 9 Then</pre>
34	<pre>intNumber2 = intNumber2 Mod 10</pre>
35	<pre>ElseIf intNumber3 &gt; 9 Then</pre>
36	<pre>intNumber3 = intNumber3 Mod 10</pre>
37	<pre>ElseIf intNumber4 &gt; 9 Then</pre>
38	<pre>intNumber4 = intNumber4 Mod 10</pre>
39	End If
40	
41	' display using the following:
42	' 1st number and third number are swapped
43	' 2nd number and 4th number are swapped
44	<pre>lblEncryptedNumber1.Text = (intNumber3 + 7) Mod 10</pre>
45	<pre>lblEncryptedNumber2.Text = (intNumber4 + 7) Mod 10</pre>
46	<pre>lblEncryptedNumber3.Text = (intNumber1 + 7) Mod 10</pre>
47	<pre>lblEncryptedNumber4.Text = (intNumber2 + 7) Mod 10</pre>
<b>48</b>	
49	End Sub ' btnEncrypt_Click
<b>50</b>	
51	End Class ' FrmEncryption





# Dental Payment Application

Introducing CheckBoxes and Message Dialogs Solutions

$\bigcirc$	Instructor's Manual Exercise Solutions Tutorial 8		
-	MULTIPLE-CHOICE	8.1 How many CheckBoxes in a GUI can be	e selected at once?
	QUESTIONS	a) 0 c) 4	<ul><li>b) 1</li><li>d) any number</li></ul>
		8.2 The text that appears alongside a Check	
		a) CheckBox label	b) CheckBox name
		c) CheckBox value	d) CheckBox data
		8.3 The first argument passed to method Me	ssageBox.Show is
		a) the text displayed in the dialog's title	e bar
		b) a constant representing the Buttons	displayed in the dialog
		c) the text displayed inside the dialog	
		d) a constant representing the icon that	appears in the dialog
		<b>8.4</b> You can specify the Button(s) and icon MessageBoxButtons and consta	to be displayed in a message dialog by using the nts.
		a) MessageIcon	b) MessageBoxImages
		c) MessageBoxPicture	d) MessageBoxIcon
		8.5 are used to create complex of	conditions.
$\bigcirc$		a) Assignment operators	b) Activity diagrams
$\bigcirc$		c) Logical operators	d) Formatting codes
		<ul> <li>8.6 Operator AndAlso</li> <li>a) performs short-circuit evaluation</li> <li>c) is a comparison operator</li> <li>d) evaluates to false if both operands as</li> </ul>	b) is not a keyword re true
		8.7 A CheckBox is selected when its Checke	d property is set to
		a) On	b) True
		c) Selected	d) Checked
		<ul> <li>8.8 The condition expression1 AndAlso exprasion1 is True and expression2</li> <li>b) expression1 is False and expression2</li> <li>c) both expression1 and expression2 are</li> <li>d) both expression1 and expression2 are</li> </ul>	is False ? is True e True
		8.9 The condition <i>expression1</i> OrElse <i>expre</i>	exion2 evaluates to False when
		a) <i>expression1</i> is True and <i>expression2</i>	
		b) <i>expression1</i> is False and <i>expression2</i>	
		<ul><li>c) both <i>expression1</i> and <i>expression2</i> are</li><li>d) both <i>expression1</i> and <i>expression2</i> are</li></ul>	
		<ul><li>8.10 The condition <i>expression1</i> Xor <i>expression1</i> a) <i>expression1</i> is True and <i>expression2</i></li></ul>	is False
$\bigcirc$		b) <i>expression1</i> is False and <i>expression2</i>	
$\bigcirc$		<ul><li>c) both <i>expression1</i> and <i>expression2</i> are</li><li>d) Both a and b.</li></ul>	eTrue
		Answers: 8.1) d. 8.2) a. 8.3) c. 8.4) d. 8.5) d	c. 8.6) a. 8.7) b. 8.8) c. 8.9) d. 8.10) d.

**EXERCISES** 8.11 (*Enhanced Dental Payment Application*) Modify the Dental Payment application from this tutorial to include additional services, as shown in Fig. 8.21. Add the proper functionality (using If...Then structures) to determine whether any of the new CheckBoxes are selected and, if so, add the price of the service to the total bill.

🖶 Dental Payment	
Dental Pa	yment Form
Patient name:	
🗖 Cleaning	\$35
🗖 Cavity Filling	\$150
🗖 X-Ray	\$85
Fluoride	\$50
🥅 Root Canal	\$225
🗖 Other	\$
	Total:
	Calculate

Figure 8.21 Enhanced Dental Payment application.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial08\Exercises\DentalPaymentEnhanced directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click DentalPaymentEnhanced.sln in the DentalPaymentEnhanced directory to open the application.
- c) Adding CheckBoxes and Labels and a TextBox. Add two CheckBoxes and two Labels to the Form. The new CheckBoxes should be labelled Fluoride and Root Canal, respectively. Add these CheckBoxes and Labels beneath the X-Ray CheckBox and its price Label. The price for a Fluoride treatment is \$50; the price for a root canal is \$225. Add a CheckBox labelled Other and a Label containing a dollar sign (\$) to the Form, as shown in Fig. 8.21. Then add a TextBox to the right of the \$ Label in which the user can enter the cost of the service performed.
- d) *Modifying the Click event handler code*. Add code to the btnCalculate\_Click event handler that determines whether the new CheckBoxes have been selected. This can be done using If...Then statements that are similar to the ones already in the event handler. Use the If...Then statements to update the bill amount.
- e) *Running the application.* Select **Debug > Start** to run your application. Test your application by checking one or more of the new services. Click the **Calculate** Button and verify that the proper total is displayed. Test the application again by checking some of the services, then checking the Other CheckBox and entering a dollar value for this service. Click the **Calculate** Button and verify that the proper total is displayed, and that it includes the price for the "other" service.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

**Answer:** 

4

6

```
1 ' Exercise 8.11 Solution
2 ' DentalPayment.vb
```

3

```
Public Class FrmDentalPayment
```

```
5 Inherits System.Windows.Forms.Form
```

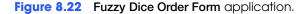
' Windows Form Designer generated code
<pre>' event handler calculates bill Private Sub btnCalculate_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnCalculate.Click</pre>
' if no CheckBox checked, display message
<pre>If (txtName.Text = "") OrElse _     (chkClean.Checked = False AndAlso _</pre>
chkXRay.Checked = False AndAlso _
chkCavity.Checked = False AndAlso _
chkFluoride.Checked = False AndAlso _
chkRootCanal.Checked = False AndAlso _ chkOther.Checked = False) Then
' display message in dialog MessageBox.Show( _ "Please enter a name and check at least one item", _
"Missing information", MessageBoxButtons.OK, _
MessageBoxIcon.Warning)
Else ' add prices
' intTotal contains amount to bill patient Dim intTotal As Integer
' if patient had a cleaning
<pre>If chkClean.Checked = True Then</pre>
intTotal += 35
End If
<pre>' if patient had cavity filled If chkCavity.Checked = True Then     intTotal += 150 End If</pre>
<pre>' if patient had x-ray taken If chkXRay.Checked = True Then intTotal += 85</pre>
End If
' if patient had Fluoride treatment
<pre>If chkFluoride.Checked = True Then     intTotal += 50</pre>
End If
' if patient had root canal
<pre>If chkRootCanal.Checked = True Then intTotal += 225</pre>
End If
' if patient had some other service performed If chkOther.Checked = True Then
If txt0therCost.Text = "" Then
MessageBox.Show("Please enter cost of service", _ "No Cost Entered", MessageBoxButtons.OK, _ MessageBoxIcon.Warning)
Else
' add cost entered

© Copyright 1992-2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

68	intTotal += Val(txtOtherCost.Text)
69	End If
70	
71	' display total
72	lblTotalResult.Text = String.Format("{0:C}", intTotal)
73	
74	End If
75	
76	End Sub ' btnCalculate_Click
77	
<b>78</b>	End Class ' FrmDentalPayment
/8	End Class FrmDentalPayment

**8.12** (*Fuzzy Dice Order Form Application*) Write an application that allows users to process orders for fuzzy dice as shown in Fig. 8.22. The application should calculate the total price of the order, including tax and shipping. TextBoxes for inputting the order number, the customer name and the shipping address are provided. Initially, these fields contain text that describes their purpose. Provide CheckBoxes for selecting the fuzzy-dice color and Text-Boxes for inputting the quantities of fuzzy dice to order. The application should also contain a Button that, when clicked, calculates the subtotals for each type of fuzzy dice ordered and the total of the entire order (including tax and shipping). Use 5% for the tax rate. Shipping charges are \$1.50 for up to 20 pairs of dice. If more than 20 pairs of dice are ordered, shipping is free.

🖶 Fuzzy Dice Ord	ler Form		<u>_   ×</u>	
	Fuzzy	Dice		
Order Number:	823			
Name:	Bob Jones			
Address:	319 Maple Street			
	Address Line 2			No Item Selected
	Anytown, ND 029	934		
				Please check item you wish to purchase
Type:	Quantity:	Price:	Totals:	
☑ White/Black	5	\$6.25	\$0.00	ОК
🗖 Red/Black	10	\$5.00	\$0.00	
🔽 Blue/Black	15	\$7.50	\$0.00	
		Subtotal:	\$0.00	
		Tax:	\$0.00	
		Shipping:	\$0.00	
		Total:	\$0.00	
			Calculate	



- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial08\Exercises\FuzzyDiceOrderForm directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click FuzzyDiceOrderForm.sln in the FuzzyDiceOrderForm directory to open the application.
- c) *Adding CheckBoxes to the Form.* Add three CheckBoxes to the Form. Label the first CheckBox White/Black, the second one Red/Black and the third Blue/Black.
- d) Adding a Click event handler and its code. Create the Click event handler for the Calculate Button. For this application, users should not be allowed to specify an item's quantity unless the item's corresponding CheckBox is checked. For the total to be calculated, the user must enter an order number, a name and a shipping address. Use logical operators to ensure that these terms are met. If they are not, display a message in a dialog.

- e) *Calculating the total cost*. Calculate the subtotal, tax, shipping and total, and display the results in their corresponding Labels.
- f) Running the application. Select Debug > Start to run your application. Test the application by providing quantities for checked items. For instance, ensure that your application is calculating 5% sales tax. If more than 20 pairs of dice are ordered, verify that shipping is free. Also, determine whether your code containing the logical operators works correctly by specifying a quantity for an item that is not checked. For instance, in Fig. 8.22, a quantity is specified for Red/Black dice, but the corresponding CheckBox is not selected. This should cause the message dialog in Fig. 8.22 to appear.
- g) *Closing the application.* Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

ByVal e As System.EventArgs) Handles btnCalculate.Click         ' display message if user does not check box         If (Val(txtWhiteBlackQuantity.Text) > 0 AndAlso _         chkWhiteBlack.Checked = False) OrElse _         (Val(txtRedBlackQuantity.Text) > 0 AndAlso _         chkRedBlack.Checked = False) OrElse _         (Val(txtRedBlackQuantity.Text) > 0 AndAlso _         chkRedBlack.Checked = False) OrElse _         (Val(txtBlueBlackQuantity.Text) > 0 AndAlso _         chkBlueBlack.Checked = False) Then         ' display message in dialog         MessageBox.Show( _         '' Please check item you wish to purchase", _         ''No Item Selected", MessageBoxButtons.OK, _         MessageBoxIcon.Exclamation)         '' display message if order number, name or address fields         ' are empty         ElseIf txtOrderNumber.Text = ''' _         OrElse txtAddressLineI.Text = ''' _         OrElse txtAddressLineI.Text = ''' _         OrElse txtCityStateZip.Text = ''' Then         ' display message in dialog         MessageBox.Show( _         ''Please fill out all information fields", _         ''Enpty Fields", MessageBoxButtons.OK, _         MessageBoxIcon.Exclamation)         ''Bise ' calculate totals         '' individual totals         ' total of white/black dice ordere	1	' Exercise 8.12 Solution
<pre>4 Public Class FrmFuzzyDiceOrderForm 5 Inherits System.Windows.Forms.Form 6 ' Windows Form Designer generated code 7 ' check validity of order before calculating totals 9 ' check validity of order before calculating totals 9 Private Sub btnCalculate_Click(ByVal sender As System.Object, 1 ByVal e As System.EventArgs) Handles btnCalculate.Click 1 display message if user does not check box 1 display message if user does not check box 1 display message if user does not check box 2 chkWhiteBlack.Checked = False) OrElse _ 2 (Val(txtWhiteBlackQuantity.Text) &gt; 0 AndAlso _ 2 (Val(txtBlueBlackQuantity.Text) = "" _ 3 (Valplay message in dialog 3 ' display message if order number, name or address fields 3 ' are empty 3 (Valplay message in dialog 3 MessageBox.Show( _ 3 "Please fill out all information fields", _ 3 " Chepty Fields", MessageBoxButtons.OK, _ 3 MessageBox.Show( _ 3 "Else txtCityStateZip.Text = "" Then 3 ' display message in dialog 4 MessageBoxLoon.Exclamation) 4 Else ' calculate totals 4 ' individual totals 4 ' individual totals 4 ' individual totals 5 ' dital of white/black dice ordered 4 Dim dewhiteBlackTotals As Decimal = _ 4 Val(txtWhiteBlackQuantity.Text) * _ 4 Val(txtWhiteBl</pre>	2	' FuzzyDiceOrderForm.vb
<pre>5 Inherits System.Windows.Forms.Form 6  7 'Windows Form Designer generated code 8 'check validity of order before calculating totals 9 rivate Sub btnCalculate_Click(ByVal sender As System.Object, 10 ByVal e As System.EventArgs) Handles btnCalculate.Click 11 ByVal e As System.EventArgs) Handles btnCalculate.Click 12 'display message if user does not check box 13 'display message if user does not check box 14 If (Val(txtRwhiteBlackQuantity.Text) &gt; 0 AndAlso _ 15 chkRwhiteBlack.Checked = False) OrElse _ 16 (Val(txtRedBlackQuantity.Text) &gt; 0 AndAlso _ 17 chkRedBlack.Checked = False) OrElse _ 18 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 19 chkBlueBlack.Checked = False) Then 10 chkBlueBlack.Checked = False) Then 11 display message in dialog 12 MessageBox.Show( _ 13 "Please check item you wish to purchase", _ 14 "No Item Selected", MessageBoxButtons.OK, _ 15 MessageBoxIcon.Exclamation) 16 'display message if order number, name or address fields 17 are empty 18 ElseIf txtOrderNumber.Text = "" _ 10 OrElse txtName.Text = "" _ 10 OrElse txtCityStateZip.Text = "" Then 13 'display message in dialog 14 MessageBox.Show( _ 15 "Please fill out all information fields", _ 16 "Empty Fields", MessageBoxButtons.OK, _ 17 MessageBoxIcon.Exclamation) 18 Else ' calculate totals 14 'individual totals 15 'individual totals 16 'individual totals 17 'individual totals 18 'individual totals 18 'individual totals 19 Orige Calculate totals As Decimal = _ 10 Val(txtWhiteBlackQuantity.Text) * _</pre>	3	
<pre>6 7 7 7 7 7 7 7 8 7 7 7 7 8 7 7 7 7 7 8 7 7 7 7 7 8 7 8 7 7 7 7 8 7 8 7 7 7 8 7 8 7 8 7 8 7 9 7 9</pre>	4	Public Class FrmFuzzyDiceOrderForm
<pre>' Windows Form Designer generated code ' check validity of order before calculating totals Private Sub btnCalculate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCalculate.Click ' display message if user does not check box ' display message if user does not check box (Val(txtWhiteBlackQuantity.Text) &gt; 0 AndAlso _</pre>		Inherits System.Windows.Forms.Form
<pre>8 9 ' check validity of order before calculating totals 9 ' check validity of order before calculating totals 9 Private Sub btnCalculate_Click(ByVal sender As System.Object, 1 ByVal e As System.EventArgs) Handles btnCalculate.Click 13 ' display message if user does not check box 14 If (Val(txtWhiteBlackQuantity.Text) &gt; 0 AndAlso _ 15 chkWhiteBlack.Checked = False) OrElse _ 16 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 17 chkRedBlack.Checked = False) OrElse _ 18 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 19 chkBlueBlack.Checked = False) OrElse _ 19 chkBlueBlack.Checked = False) OrElse _ 10 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 10 chkBlueBlack.Checked = False) Then 10 ' display message in dialog 11 ' display message in dialog 12 MessageBox.Show( _ 13 '' Please check item you wish to purchase", _ 14 '' No Item Selected", MessageBoxButtons.OK, _ 15 MessageBox.Con.Exclamation) 16 ' display message if order number, name or address fields 17 ' are empty 18 ElseIf txtOrderNumber.Text = "" _ 19 OrElse txtAddressLineI.Text = "" _ 10 OrElse txtAddressLineI.Text = "" _ 10 OrElse txtAddressLineI.Text = "" _ 11 OrElse txtCityStateZip.Text = "" Then 13 ' display message in dialog 14 ' display message in dialog 15 MessageBox.Show( _ 16 '' "Please fill out all information fields", _ 17 '' "Empty Fields", MessageBoxButtons.OK, _ 18 MessageBoxIcon.Exclamation) 19 Else ' calculate totals 10 ' individual totals 11 ' individual totals 12 ' individual totals 13 ' total of white/black dice ordered 14 Dim decWhiteBlackQuantity.Text) * _ 15 '' Val(txtWhiteBlackQuantity.Text) * _ 15 '' Val(txtWhiteBlackQuan</pre>		
<pre>9 ' check validity of order before calculating totals 9 Private Sub btnCalculate_Click(ByVal sender As System.Object, 11 ByVal e As System.EventArgs) Handles btnCalculate.Click 12 ' display message if user does not check box 14 If (Val(txtWhiteBlackQuantity.Text) &gt; 0 AndAlso _ 15 chkWhiteBlack.Checked = False) OrElse _ 16 (Val(txtRedBlack.Checked = False) OrElse _ 17 chkRedBlack.Checked = False) OrElse _ 18 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 19 chkBlueBlack.Checked = False) OrElse _ 19 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 19 chkBlueBlack.Checked = False) OrElse _ 10 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 19 chkBlueBlack.Checked = False) Then 10 ' display message in dialog 11 ' display message in dialog 12 MessageBox.Show( _ 13 '' display message if order number, name or address fields 13 ' are empty 14 OrElse txtName.Text = "" _ 15 OrElse txtName.Text = "" _ 16 OrElse txtCityStateZip.Text = "" Then 17 ' display message in dialog 18 ' are empty 19 ElseIf txtOrderNumber.Text = "" _ 10 OrElse txtCityStateZip.Text = "" Then 13 ' display message in dialog 14 ' messageBox.Show( _ 15 '' "Please fill out all information fields", _ 16 '' "Empty Fields", MessageBoxButtons.OK, _ 17 '' MessageBoxIcon.Exclamation) 18 '' total of white/black dice ordered 19 '' total of white/black dice ordered 10 m decWhiteBlackQuantity.Text) * _ 19 '' Val(txtWhiteBlackQuantity.Text) * _ 10 '' (txtRumeText = ''' _ 10 '' '' (tall totals 11 '' individual totals 12 '' '' (tall of white/black dice ordered 13 '' total of white/black dice ordered 14 Dim decWhiteBlackQuantity.Text) * _ 14 '' Val(txtWhiteBlackQuantity.Text) * _ 15 '' Val(txtWhiteBlackQuantity.Text) * _ 16 '' Val(txtWhiteBlackQuantity.Text) * _ 17 ''' (''''''''''''''''''''''''''''''''</pre>	7	' Windows Form Designer generated code
<pre>Private Sub btnCalculate_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnCalculate.Click ' display message if user does not check box If (Val(txtWhiteBlackQuantity.Text) &gt; 0 AndAlso _ chkWhiteBlack.Checked = False) OrElse _ (Val(txtRedBlackQuantity.Text) &gt; 0 AndAlso _ chkRedBlack.Checked = False) OrElse _ (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ chkRedBlack.Checked = False) Then (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ chkBlueBlack.Checked = False) Then ' display message in dialog MessageBox.Show( _ ' "Please check item you wish to purchase", _ ' No Item Selected", MessageBoxButtons.OK, _ MessageBoxIcon.Exclamation) ' display message if order number, name or address fields ' are empty ElseIf txtOrderNumber.Text = "" _ OrElse txtAddressLine1.Text = "" _ OrElse txtAddressLine1.Text = "" Then ' display message in dialog MessageBox.Show( _ ''Please fill out all information fields", _ ''Empty Fields", MessageBoxButtons.OK, _ MessageBoxIcon.Exclamation) Else ' calculate totals ' individual totals ' total of white/black dice ordered Dim decWhiteBlackQuantity.Text) * _</pre>		
ByVal e As System.EventArgs) Handles btnCalculate.Click1byVal e As System.EventArgs) Handles btnCalculate.Click1' display message if user does not check box14If (Val(txtWhiteBlackQuantity.Text) > 0 AndAlso _ chkWhiteBlack.Checked = False) OrElse _ (Val(txtRedBlackQuantity.Text) > 0 AndAlso _ chkRedBlack.Checked = False) OrElse _ (Val(txtBlueBlackQuantity.Text) > 0 AndAlso _ chkBlueBlack.Checked = False) Then16(Val(txtBlueBlackQuantity.Text) > 0 AndAlso _ chkBlueBlack.Checked = False) Then17chkBdlueBlack.Checked = False) Then18(Val(txtBlueBlackQuantity.Text) > 0 AndAlso _ chkBlueBlack.Checked = False) Then20' display message in dialog22MessageBox.Show( _ "No Item Selected", MessageBoxButtons.OK, _ MessageBoxIcon.Exclamation)23' display message if order number, name or address fields ' are empty24' display message in dialog25OrElse txtAddressLinel.Text = "" _ OrElse txtAddressLinel.Text = "" _ OrElse txtCityStateZip.Text = "" Then33' display message in dialog34' display message in dialog35MessageBox.Show( _ " "Please fill out all information fields", _ " "Empty Fields", MessageBoxButtons.OK, _ MessageBoxIcon.Exclamation)34' individual totals ' total of white/black dice ordered35/ individual totals ' total of white/black dice ordered36/ ultxtWhiteBlackQuantity.Text) * _ Val(txtWhiteBlackQuantity.Text) * _	9	' check validity of order before calculating totals
<pre>' display message if user does not check box If (Val(txtWhiteBlackQuantity.Text) &gt; 0 AndAlso _</pre>	10	<pre>Private Sub btnCalculate_Click(ByVal sender As System.Object,</pre>
<pre>' display message if user does not check box If (Val(txtWhiteBlackQuantity.Text) &gt; 0 AndAlso _</pre>		ByVal e As System.EventArgs) Handles btnCalculate.Click
<pre>14 If (Val(txtWhiteBlackQuantity.Text) &gt; 0 AndAlso _ 15 chkWhiteBlack.Checked = False) OrElse _ 16 (Val(txtRedBlackQuantity.Text) &gt; 0 AndAlso _ 17 chkRedBlack.Checked = False) OrElse _ 18 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 19 chkBlueBlack.Checked = False) Then 20 21 ' display message in dialog 22 MessageBox.Show( _ 23 "Please check item you wish to purchase", _ 24 "No Item Selected", MessageBoxButtons.OK, _ 25 MessageBoxIcon.Exclamation) 26 27 ' display message if order number, name or address fields 28 ' are empty 29 ElseIf txtOrderNumber.Text = "" _ 30 OrElse txtAddressLine1.Text = "" _ 31 OrElse txtAddressLine1.Text = "" Then 33 ' display message in dialog 34 ' display message in dialog 35 MessageBox.Show( _ 36 "Please fill out all information fields", _ 37 "Empty Fields", MessageBoxButtons.OK, _ 38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>		
<pre>15 chkWhiteBlack.Checked = False) OrElse _ 16 (Val(txtRedBlackQuantity.Text) &gt; 0 AndAlso _ 17 chkRedBlack.Checked = False) OrElse _ 18 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 19 chkBlueBlack.Checked = False) Then 20 21 ' display message in dialog 22 MessageBox.Show( _ 23          "Please check item you wish to purchase", _ 24          "No Item Selected", MessageBoxButtons.OK, _ 25          MessageBoxIcon.Exclamation) 26 27 ' display message if order number, name or address fields 28 ' are empty 29 ElseIf txtOrderNumber.Text = "" _ 30          OrElse txtAddressLine1.Text = "" _ 31          OrElse txtAddressLine1.Text = "" Then 33 34 ' display message in dialog 35         MessageBox.Show( _ 36          "Please fill out all information fields", _ 37          "Empty Fields", MessageBoxButtons.OK, _ 38          MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45</pre>		' display message if user does not check box
<pre>16 (Val(txtRedBlackQuantity.Text) &gt; 0 AndAlso _ 17 chkRedBlack.Checked = False) OrElse _ 18 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 19 chkBlueBlack.Checked = False) Then 20 21 ' display message in dialog 22 MessageBox.Show( _ 23 "Please check item you wish to purchase", _ 24 "No Item Selected", MessageBoxButtons.OK, _ 25 MessageBoxIcon.Exclamation) 26 27 ' display message if order number, name or address fields 28 ' are empty 29 ElseIf txtOrderNumber.Text = "" _ 30 OrElse txtAddressLine1.Text = "" _ 31 OrElse txtAddressLine1.Text = "" Then 33 34 ' display message in dialog 35 MessageBox.Show( _ 36 "Please fill out all information fields", _ 37 "Empty Fields", MessageBoxButtons.OK, _ 38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>		
<pre>chkRedBlack.Checked = False) OrElse _ (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ chkBlueBlack.Checked = False) Then ' display message in dialog MessageBox.Show( _ ' Please check item you wish to purchase", _ ''No Item Selected", MessageBoxButtons.OK, _ MessageBoxIcon.Exclamation) ' display message if order number, name or address fields ' are empty ElseIf txtOrderNumber.Text = "" _ OrElse txtAddressLine1.Text = "" _ OrElse txtAddressLine1.Text = "" Then ' display message in dialog MessageBox.Show( _ ''Please fill out all information fields", _ ''Empty Fields", MessageBoxButtons.OK, _ MessageBoxIcon.Exclamation) Else ' calculate totals ' individual totals ' individual totals As Decimal = _ Val(txtWhiteBlackQuantity.Text) * _</pre>		chkWhiteBlack.Checked = False) OrElse _
<pre>18 (Val(txtBlueBlackQuantity.Text) &gt; 0 AndAlso _ 19 chkBlueBlack.Checked = False) Then 20 ' display message in dialog 22 MessageBox.Show( _ 23 "Please check item you wish to purchase", _ 24 "No Item Selected", MessageBoxButtons.OK, _ 25 MessageBoxIcon.Exclamation) 26 ' 27 ' display message if order number, name or address fields 28 ' are empty 29 ElseIf txtOrderNumber.Text = "" _ 30 OrElse txtName.Text = "" _ 31 OrElse txtAddressLine1.Text = "" _ 32 OrElse txtCityStateZip.Text = "" Then 33 ' display message in dialog 34 ' display message in dialog 35 MessageBox.Show( _ 36 "Please fill out all information fields", _ 37 "Empty Fields", MessageBoxButtons.OK, _ 38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>		(Val(txtRedBlackQuantity.Text) > 0 AndAlso _
<pre>chkBlueBlack.Checked = False) Then chkBlueBlack.Checked = False) Then display message in dialog MessageBox.Show( _     "Please check item you wish to purchase", _     "No Item Selected", MessageBoxButtons.OK, _     MessageBoxIcon.Exclamation) display message if order number, name or address fields     ' are empty leseIf txtOrderNumber.Text = "" _     OrElse txtAddressLine1.Text = "" _     OrElse txtCityStateZip.Text = "" Then     ' display message in dialog     MessageBox.Show( _</pre>		chkRedBlack.Checked = False) OrElse _
<pre>20 21 ' display message in dialog 22 MessageBox.Show( _ 23 "Please check item you wish to purchase", _ 24 "No Item Selected", MessageBoxButtons.OK, _ 25 MessageBoxIcon.Exclamation) 26 27 ' display message if order number, name or address fields 28 ' are empty 29 ElseIf txtOrderNumber.Text = "" _ 30 OrElse txtName.Text = "" _ 31 OrElse txtAddressLine1.Text = "" _ 32 OrElse txtCityStateZip.Text = "" Then 33 ' display message in dialog 34 ' display message in dialog 35 MessageBox.Show( _ 36 "Please fill out all information fields", _ 37 "Empty Fields", MessageBoxButtons.OK, _ 38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _ </pre>		(Val(txtBlueBlackQuantity.Text) > 0 AndAlso _
<pre>21 ' display message in dialog 22 MessageBox.Show( _ 23 "Please check item you wish to purchase", _ 24 "No Item Selected", MessageBoxButtons.OK, _ 25 MessageBoxIcon.Exclamation) 26 27 ' display message if order number, name or address fields 28 ' are empty 29 ElseIf txtOrderNumber.Text = "" _ 30 OrElse txtName.Text = "" _ 31 OrElse txtAddressLine1.Text = "" _ 32 OrElse txtCityStateZip.Text = "" Then 33 34 ' display message in dialog 35 MessageBox.Show( _ 36 "Please fill out all information fields", _ 37 "Empty Fields", MessageBoxButtons.OK, _ 38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>	19	chkBlueBlack.Checked = False) Then
<pre>MessageBox.Show( _</pre>	20	
<pre>"Please check item you wish to purchase", _ "No Item Selected", MessageBoxButtons.OK, _ MessageBoxIcon.Exclamation) " ' display message if order number, name or address fields ' are empty ElseIf txtOrderNumber.Text = "" _ OrElse txtName.Text = "" _ OrElse txtAddressLine1.Text = "" _ OrElse txtAddressLine1.Text = "" _ OrElse txtCityStateZip.Text = "" Then ' display message in dialog MessageBox.Show( _ "Please fill out all information fields", _ "Empty Fields", MessageBoxButtons.OK, _ MessageBoxIcon.Exclamation) Else ' calculate totals ' individual totals ' total of white/black dice ordered Dim decWhiteBlackTotals As Decimal = _ Val(txtWhiteBlackQuantity.Text) * _</pre>	21	' display message in dialog
<pre>"No Item Selected", MessageBoxButtons.OK,</pre>	22	MessageBox.Show( _
<pre>MessageBoxIcon.Exclamation)  // display message if order number, name or address fields // are empty // ElseIf txtOrderNumber.Text = ""</pre>	23	"Please check item you wish to purchase", _
<pre>26 27 ' display message if order number, name or address fields 28 ' are empty 29 ElseIf txtOrderNumber.Text = "" 30 OrElse txtName.Text = "" 31 OrElse txtAddressLine1.Text = "" 32 OrElse txtCityStateZip.Text = "" Then 33 ' display message in dialog 34 ' display message in dialog 35 MessageBox.Show( "Please fill out all information fields", "Empty Fields", MessageBoxButtons.OK, 36 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = 45 Val(txtWhiteBlackQuantity.Text) *</pre>	24	<pre>"No Item Selected", MessageBoxButtons.OK, _</pre>
<pre>' display message if order number, name or address fields ' are empty ElseIf txtOrderNumber.Text = ""</pre>	25	MessageBoxIcon.Exclamation)
<pre>28 ' are empty 29 ElseIf txtOrderNumber.Text = "" _ 30 OrElse txtName.Text = "" _ 31 OrElse txtAddressLine1.Text = "" _ 32 OrElse txtCityStateZip.Text = "" Then 33 34 ' display message in dialog 35 MessageBox.Show( _ 36 "Please fill out all information fields", _ 37 "Empty Fields", MessageBoxButtons.OK, _ 38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>		
<pre>29 ElseIf txtOrderNumber.Text = ""</pre>	27	' display message if order number, name or address fields
30       OrElse txtName.Text = ""	28	'are empty
31       OrElse txtAddressLine1.Text = ""	29	<pre>ElseIf txtOrderNumber.Text = "" _</pre>
<pre>32 OrElse txtCityStateZip.Text = "" Then 33 34 ' display message in dialog 35 MessageBox.Show( _ 36 "Please fill out all information fields", _ 37 "Empty Fields", MessageBoxButtons.OK, _ 38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>	30	<pre>OrElse txtName.Text = "" _</pre>
<pre>33 34 ' display message in dialog 35 MessageBox.Show( _ 36 "Please fill out all information fields", _ 37 "Empty Fields", MessageBoxButtons.OK, _ 38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>	31	<pre>OrElse txtAddressLine1.Text = "" _</pre>
<pre>34 ' display message in dialog 35 MessageBox.Show( _ 36 "Please fill out all information fields", _ 37 "Empty Fields", MessageBoxButtons.OK, _ 38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 ' individual totals 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>	32	<pre>OrElse txtCityStateZip.Text = "" Then</pre>
<pre>MessageBox.Show(</pre>	33	
<pre>36 "Please fill out all information fields", 37 "Empty Fields", MessageBoxButtons.OK, 38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = 45 Val(txtWhiteBlackQuantity.Text) *</pre>		' display message in dialog
<pre>37     "Empty Fields", MessageBoxButtons.OK, _ 38     MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 42     ' individual totals 43     ' total of white/black dice ordered 44     Dim decWhiteBlackTotals As Decimal = _ 45     Val(txtWhiteBlackQuantity.Text) * _</pre>		MessageBox.Show( _
<pre>38 MessageBoxIcon.Exclamation) 39 40 Else ' calculate totals 41 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>		"Please fill out all information fields", _
<pre>39 40 Else ' calculate totals 41 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>		"Empty Fields", MessageBoxButtons.OK, _
<pre>40 Else ' calculate totals 41 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>		MessageBoxIcon.Exclamation)
<pre>41 42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>	39	
<pre>42 ' individual totals 43 ' total of white/black dice ordered 44 Dim decWhiteBlackTotals As Decimal = _ 45 Val(txtWhiteBlackQuantity.Text) * _</pre>	40	Else ' calculate totals
43' total of white/black dice ordered44Dim decWhiteBlackTotals As Decimal = _45Val(txtWhiteBlackQuantity.Text) * _	41	
44Dim decWhiteBlackTotals As Decimal = _45Val(txtWhiteBlackQuantity.Text) * _	42	' individual totals
45 Val(txtWhiteBlackQuantity.Text) * _		
	44	
46 lblWhiteBlackPrice.Text	45	Val(txtWhiteBlackQuantity.Text) * _
	46	lblWhiteBlackPrice.Text

47	
48	' total of red/black dice ordered
49	Dim decRedBlackTotals As Decimal = _
50	Val(txtRedBlackQuantity.Text) * _
51	lblRedBlackPrice.Text
	IDIREUBIACKPITCE.TEXT
52	
53	' total of blue/black dice ordered
54	Dim decBlueBlackTotals As Decimal = _
55	Val(txtBlueBlackQuantity.Text) * _
56	lblBlueBlackPrice.Text
57	IS IS INCOLUCIENCE
	I display assolute of displayed
58	' display totals of dice ordered
59	lblWhiteBlackTotals.Text = _
60	<pre>String.Format("{0:C}", decWhiteBlackTotals)</pre>
61	lblRedBlackTotals.Text = _
62	<pre>String.Format("{0:C}", decRedBlackTotals)</pre>
63	<pre>lblBlueBlackTotals.Text =</pre>
64	<pre>String.Format("{0:C}", decBlueBlackTotals)</pre>
65	
66	' calculate and display subtotal
67	<pre>Dim decSubtotal As Decimal = decWhiteBlackTotals + _</pre>
68	decRedBlackTotals + decBlueBlackTotals
69	
70	<pre>lblSubtotalResult.Text =</pre>
71	<pre>String.Format("{0:C}", decSubtotal)</pre>
72	
	the set of the second of the transmission
73	' calculate and display tax
74	Dim decTax As Decimal = decSubtotal * 0.05
75	
76	lblTaxResult.Text = <pre>String.Format("{0:C}", decTax)</pre>
77	
78	' shipping
79	' \$1.50 for up to 20 items
80	'free after 20 items
81	<pre>Dim intNumberOfItems As Integer = _</pre>
82	(Val(txtWhiteBlackQuantity.Text) + _
83	Val(txtRedBlackQuantity.Text) + _
84	Val(txtBlueBlackQuantity.Text))
85	
86	Dim decShippingCost As Decimal = 0.0
87	
88	L chimping is \$1 50 if loss than 20 mains and mad
	' shipping is \$1.50 if less than 20 pairs ordered
89	<pre>If intNumberOfItems &lt;= 20 Then</pre>
90	
91	<pre>decShippingCost = 1.5</pre>
92	
93	End If
94	
95	' display shipping cost
96	<pre>lblShippingResult.Text = _</pre>
97	<pre>String.Format("{0:C}", decShippingCost)</pre>
98	
99	' calculate and display total
100	<pre>Dim decTotalCharge As Decimal = decSubtotal + _</pre>
101	decTax + decShippingCost
102	
102	lhlTotalPocult Toxt
	<pre>lblTotalResult.Text = _</pre>
104	<pre>String.Format("{0:C}", decTotalCharge)</pre>
105	
106	End If
106 107	End If

108	End Sub ' btnCalculate_Click
109	
110	End Class ' FrmFuzzyDiceOrderForm

**8.13** (*Modified Fuzzy Dice Order Form Application*) Modify the Fuzzy Dice Order Form application from Exercise 8.12 to determine whether customers should receive a 7% discount off their purchase. Customers ordering more than \$500 (before tax and shipping) in fuzzy dice are eligible for this discount.

Hodified Fuzzy	Dice Order Form			
	Fuzzy	Dice		
Order Number:	2536			
Name:	Bob Jones			
Address:	318 Maple Street			
	Address Line 2			Discount Offer
	Anytown, ND 029	34		7% discount will be applied
				~r
Туре:	Quantity:	Price:	Totals:	OK
✓ White/Black	100	\$6.25	\$625.00	
🗖 Red/Black	0	\$5.00	\$0.00	
🗖 Blue/Black	0	\$7.50	\$0.00	
		Subtotal:	\$625.00	
		Tax:	\$29.06	
		Shipping:	\$0.00	
		Discount:	(\$43.75)	
		Total:	\$610.31	
			Calculate	

Figure 8.23 Modified Fuzzy Dice Order Form application.

- a) *Opening the application*. Open the application you created in Exercise 8.12.
- b) *Determining whether the total cost is over \$500.* Use an If...Then statement to determine if the amount ordered is greater than \$500.
- c) Displaying the discount and subtracting the discount from the total. If a customer orders more than \$500, display a message dialog as shown in Fig. 8.23 that informs the user that the customer is entitled to a 7% discount. The message dialog should contain an Information icon and an OK Button. Calculate 7% of the total amount, and display the discount amount in the Discount: field. Subtract this amount from the total, and update the Total: field.
- d) *Running the application*. Select **Debug > Start** to run your application. Confirm that your application calculates and displays the discount properly.
- e) *Closing the application.* Close your running application by clicking its close box.
- f) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 8.13 Solution
2	' FuzzyDiceOrderFormModified.vb
3	
4	Public Class FrmFuzzyDiceOrderFormModified
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	<pre>Private Sub btnCalculate_Click(ByVal sender As System.Object, _</pre>
10	ByVal e As System.EventArgs) Handles btnCalculate.Click

' display message if user does not check box
If (Val(txtWhiteBlackQuantity.Text) > 0 AndAlso _
chkWhiteBlack.Checked = False) OrElse
(Val(txtRedBlackQuantity.Text) > 0 AndAlso _
chkRedBlack.Checked = False) OrElse _
(Val(txtBlueBlackQuantity.Text) > 0 AndAlso _
chkBlueBlack.Checked = False) Then
' display message in dialog
MessageBox.Show( _
"Please check item you wish to purchase", _
"No Item Selected", MessageBoxButtons.OK, _
MessageBoxIcon.Exclamation)
' display message if order number, name or address fields
'are empty
<pre>ElseIf (txtOrderNumber.Text = "") _</pre>
<pre>OrElse (txtName.Text = "") _</pre>
<pre>OrElse (txtAddressLine1.Text = "") _</pre>
<pre>OrElse (txtCityStateZip.Text = "") Then</pre>
' display message in dialog
MessageBox.Show( _
"Please fill out all information fields.", $\_$
<pre>"Empty fields", MessageBoxButtons.OK, _</pre>
MessageBoxIcon.Exclamation)
Else ' calculate totals
' individual totals
' total of white/black dice ordered
Dim decWhiteBlackTotals As Decimal = _
Val(txtWhiteBlackQuantity.Text) * _
lb]WhiteBlackPrice.Text
' total of red/black dice ordered
Dim decRedBlackTotals As Decimal = _
Val(txtRedBlackQuantity.Text) * _
lblRedBlackPrice.Text
' total of blue/black dice ordered
<pre>Dim decBlueBlackTotals As Decimal = _</pre>
Val(txtBlueBlackQuantity.Text) * _
lblBlueBlackPrice.Text
' display totals for dice
lblWhiteBlackTotals.Text = _
<pre>String.Format("{0:C}", decWhiteBlackTotals)</pre>
<pre>lblRedBlackTotals.Text = _</pre>
<pre>String.Format("{0:C}", decRedBlackTotals)</pre>
<pre>lblBlueBlackTotals.Text = _</pre>
<pre>String.Format("{0:C}", decBlueBlackTotals)</pre>
' calculate and display subtotal
Dim decSubtotal As Decimal = decWhiteBlackTotals + _
decRedBlackTotals + decBlueBlackTotals
]b]Subtata]Racult Taxt
<pre>lblSubtotalResult.Text = _ String.Format("{0:C}", decSubtotal)</pre>

72	
73	' if decTotalCharge is greater than \$500
74	' display message box and give 7% discount
75	If decSubtotal > 500 Then
76	
77	MessageBox.Show( _
78	"7% discount will be applied", "Discount Offer", _
79	
	MessageBoxButtons.OK, MessageBoxIcon.Information)
30	l selles les end dienles aus de Terrille Channes distributions
31	' calculate and display new decTotalCharge with discount
32	Dim decDiscount As Decimal = decSubtotal * 0.07
33	
34	decSubtotal -= decDiscount
35	
36	' decDiscount is negative to reflect that it is
37	' being subtracted from the subtotal during display
38	lblDiscount.Text = <pre>String.Format("{0:C}", -decDiscount)</pre>
39	
70	End If
21	
72	' calculate and display tax
73	Dim decTax As Decimal = decSubtotal * 0.05
74	
<b>?</b> 5	lblTaxResult.Text = <u>String</u> .Format("{0:C}", decTax)
76	
77	' shipping
78	' \$1.50 for up to 20 items
79	' free after 20 items
00	Dim intNumberOfItems As Integer = _
01	(Val(txtWhiteBlackQuantity.Text) + _
02	Val(txtRedBlackQuantity.Text) + _
03	Val(txtBlueBlackQuantity.Text))
04	
05	Dim decShippingCost As $Decimal = 0.0$
06	bill accompringeose As beening = 0.0
07	' shipping is \$1.50 if less than 20 pairs ordered
08	If intNumberOfItems <= 20 Then
09	
10	<pre>decShippingCost = 1.5</pre>
111	
12	End If
14	' display shipping charges
15	lblShippingResult.Text = _
16	<pre>String.Format("{0:C}", decShippingCost)</pre>
17	
18	' calculate total charge
19	Dim decTotalCharge As Decimal = _
20	decSubtotal + decTax + decShippingCost
21	
22	' display total charge
23	<pre>lblTotalResult.Text = _</pre>
24	<pre>String.Format("{0:C}", decTotalCharge)</pre>
25	
26	End If
27	
28	End Sub ' btnCalculate_Click
29	
30	End Class ' FrmFuzzyDiceOrderFormModified

© Copyright 1992-2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

What does this code do? 8.14 Assume that txtName is a TextBox and that chkOther is a CheckBox next to which is a TextBox txtOther, in which the user should specify a value. What does this code segment do? 1 If (txtName.Text = "" OrElse \_ 2 (chkOther.Checked = True AndAlso \_ 3 txtOther.Text = "")) Then 4 5 MessageBox.Show("Please enter a name or value", \_ 6 "Input Error", MessageBoxButtons.OK, \_ 7 MessageBoxIcon.Exclamation) 8 9 End If **Answer:** This code displays a message dialog only if txtName.Text is empty or CheckBox chkOther is selected and its corresponding TextBox is left blank. What's wrong with this code? **8.15** Assume that txtName is a TextBox. Find the error(s) in the following code: If txtName.Text = "John Doe" Then 1 2 3 MessageBox.Show("Welcome, John!", \_ 4 MessageBoxIcon.Exclamation) 5 6 End If Answer: The call to method MessageBox. Show is missing arguments. Also, the nature of the message indicates that MessageBoxIcon.Information should be used instead of Message-BoxIcon.Exclamation. The corrected code should read: If txtName.Text = "John Doe" Then 1 2 3 MessageBox.Show("Welcome, John!", \_ 4 "Welcome", MessageBoxButtons.OK, \_ 5 MessageBoxIcon.Information) 6 7 End If

Using the Debugger

**8.16** (*Sibling Survey Application*) The Sibling Survey application displays the siblings selected by the user in a dialog. If the user checks either the **Brother(s)** or **Sister(s)** Check-Box, and the **No Siblings** CheckBox, the user is asked to verify the selection. Otherwise, the user's selection is displayed in a MessageBox. While testing this application, you noticed that it does not execute properly. Use the debugger to find and correct the logic error(s) in the code. This exercise is located in the C:\Examples\Tutorial08\Debugger\SiblingSurvey directory. Figure 8.24 shows the correct output for the application.

🖶 Sibling Survey	Invalid Input 🔀
Please select the siblings you have:	Selected combination is not possible
Erother(s)	OK
Sister(s)	
🔽 No Siblings	
Submit Survey	

Figure 8.24 Correct output for the Sibling Survey application.

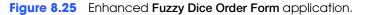
		1	' Exercise 8.16 Solution
		2	'SiblingSurvey.vb
		3	
		4	Public Class FrmSiblingSurvey
		5	Inherits System.Windows.Forms.Form
		6	L Minus I. Crudia . NET assessed as de
		7	' Visual Studio .NET generated code
		8	
		9	' display what siblings user selects
		10	Private Sub btnSubmit_Click(ByVal sender As _
		11	System.Object, ByVal e As System.EventArgs) _
		12	Handles btnSubmit.Click
		13	
		14	' check if user selects brothers or sisters
$\bigcap$		15	' and no siblings
$\bigcirc$		16	If (chkNone.Checked = True) AndAlso _
	Replaced AndAlso with OrElse	17	(chkBrother.Checked = True OrElse
		18	chkSister.Checked = True) Then
		19	cliksister.cliecked = lide) men
		20	<pre>MessageBox.Show("Selected combination is not possible", _</pre>
		21	"Invalid Input", MessageBoxButtons.OK, _
		22	MessageBoxIcon.Exclamation)
		23	
		24	' check if user selects CheckBox
		25	<pre>ElseIf chkNone.Checked = False AndAlso _</pre>
		26	chkBrother.Checked = False AndAlso _
		27	chkSister.Checked = False Then
		28	
		29	<pre>MessageBox.Show("Please check at least one CheckBox", _</pre>
		30	"Invalid Input", MessageBoxButtons.OK, _
		31	MessageBoxIcon.Exclamation)
		32	hessageboxicon. Exclamacion)
		33	' check if user has brothers and sisters
	Deployed OnElso with AndAlso	34	
	Replaced OrElse with AndAlso		ElseIf chkBrother.Checked = True AndAlso
		35	chkSister.Checked = True Then
		36	MessageBox.Show("You have brothers and sisters", _
		37	"Siblings", MessageBoxButtons.OK, _
		38	MessageBoxIcon.Information)
		39	
		40	' check if user has brothers
		41	<pre>ElseIf chkBrother.Checked = True Then</pre>
$\frown$		42	<pre>MessageBox.Show("You have at least one brother", _</pre>
()		43	"Siblings", MessageBoxButtons.OK, _
$\bigcirc$		44	MessageBoxIcon.Information)
		45	······································
		46	' check if user has sisters
		-0	

47	<pre>ElseIf chkSister.Checked = True Then</pre>
48	MessageBox.Show("You have at least one sister", _
49	"Siblings", MessageBoxButtons.OK, _
50	MessageBoxIcon.Information)
51	
52	' user has no siblings
53	Else
54	<pre>MessageBox.Show("You have no siblings", _</pre>
55	"Siblings", MessageBoxButtons.OK, _
56	MessageBoxIcon.Information)
57	
58	End If
59	
60	End Sub ' btnSubmit_Click
61	
62	End Class ' FrmSiblingSurvey

Programming Challenge

**8.17** (*Enhanced Fuzzy Dice Order Form Application*) Enhance the Fuzzy Dice Order Form application from Exercise 8.12 by replacing the Calculate Button with a Clear Button. The application should update the total cost, tax and shipping when the user changes any one of the three Quantity field's values (Fig. 8.25). The Clear Button should return all fields to their original values. [*Hint*: You will need to use the CheckBox CheckedChanged event for each CheckBox. This event is raised when the state of a CheckBox changes. Double click a CheckBox in design view to create an event handler for that CheckBox's CheckedChanged event. You also will need to assign Boolean values to the CheckBoxes' Checked properties to control their states.]

🚂 Fuzzy Dice Order Form				
	Fuzzy	Dice		
Order Number:	15233			
Name:	Bob Jones			
Address:	318 Maple Street			
	Address Line 2			
	Anytown, ND 029	934		
Type:	Quantity:	Price:	Totals:	
☑ White/Black	5	\$6.25	\$31.25	
Red/Black	3	\$5.00	\$15.00	
🔽 Blue/Black	10	\$7.50	\$75.00	
		Subtotal:	\$121.25	
		Tax:	\$6.06	
		Shipping:	\$1.50	
		Total:	\$128.81	
			Clear	



1	' Exercise 8.17 Solution
2	' FuzzyDiceOrderFormEnhanced.vb
3	
4	Public Class FrmFuzzyDiceOrderFormEnhanced
5	Inherits System.Windows.Forms.Form
6	

## **Tutorial 8**

'Windows Form Designer generated code
<pre>Private Sub txtWhiteBlackQuantity_TextChanged(ByVal sender As _</pre>
System.Object, ByVal e As System.EventArgs) Handles _
txtWhiteBlackQuantity.TextChanged
tx twill teb lackquart i ty . lex tchanged
' store quantity entered as Integer
Dim intNumberOfWhiteBlack As Integer = _
Val(txtWhiteBlackQuantity.Text)
' display message if user tries to enter a value
' without selecting CheckBox
<pre>If (intNumberOfWhiteBlack &lt;&gt; 0 _</pre>
AndAlso chkWhiteBlack.Checked = False) Then
L keen white /black quantity at 0
<pre>' keep white/black quantity at 0 tyte/white/black/oueptity/Text = 0</pre>
<pre>txtWhiteBlackQuantity.Text = 0</pre>
' display message in dialog
MessageBox.Show(
"Please check item you wish to purchase", _
"No Item Selected", MessageBoxButtons.OK, _
MessageBoxIcon.Exclamation)
hebbugebonzeonrekeralmaerony
' display message if shipping information is not supplied
ElseIf _
(txtOrderNumber.Text = "") _
<pre>OrElse (txtName.Text = "") _</pre>
<pre>OrElse (txtAddressLine1.Text = "") _</pre>
<pre>OrElse (txtCityStateZip.Text = "") Then</pre>
' display message in dialog
MessageBox.Show( _
"Please fill out all information fields.", _
"Empty Fields", MessageBoxButtons.OK, _
MessageBoxIcon.Exclamation)
' display message if negative number entered
Elself
(intNumberOfWhiteBlack < 0) Then
txtWhiteBlackQuantity.Text = 0
MessageBox.Show( _
"Please enter a positive quantity", _
"Bad Input", MessageBoxButtons.OK,
MessageBoxIcon.Exclamation)
Else ' calculate totals
' individual totals
' total of white/black dice
Dim decWhiteBlackTotals As Decimal = _
Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text
' total of red/black dice
Dim decRedBlackTotals As Decimal = _
Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text
' total of blue/black dice
Dim decBlueBlackTotals As Decimal = _
Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text

```
68
                display individual totals
69
              lblWhiteBlackTotals.Text = _
70
                 String.Format("{0:C}", decWhiteBlackTotals)
              lblRedBlackTotals.Text = _
71
                 String.Format("{0:C}", decRedBlackTotals)
72
73
              lblBlueBlackTotals.Text = _
74
                 String.Format("{0:C}", decBlueBlackTotals)
75
76
              ' subtotal, before tax and shipping
77
              Dim decSubtotal As Decimal = decWhiteBlackTotals _
78
                 + decRedBlackTotals + decBlueBlackTotals
79
80
              lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal)
81
82
              ' calculate and display tax
              Dim decTax As Decimal = decSubtotal * 0.05
83
84
85
              lblTaxResult.Text = String.Format("{0:C}", decTax)
86
87
              ' shipping
88
              ' $1.50 for up to 20 items
89
              ' free after 20 items
90
              Dim intNumberOfItems As Integer = _____
91
                 (Val(txtWhiteBlackQuantity.Text) + _
92
                 Val(txtRedBlackQuantity.Text) + _
93
                 Val(txtBlueBlackQuantity.Text))
94
95
              Dim decShippingCost As Decimal = 0.0
96
97
              ' shipping is $1.50 if under 20 items ordered
98
              If (intNumberOfItems <= 20) AndAlso _</pre>
99
                 (intNumberOfItems > 0) Then
100
101
                 decShippingCost = 1.5
102
103
              End If
104
              ' display shipping cost
105
              lblShippingResult.Text = _
106
                 String.Format("{0:C}", decShippingCost)
107
108
109
              ' calculate and display total charge
110
              Dim decTotalCharge As Decimal = decSubtotal + decTax + _
111
                 decShippingCost
112
113
              lblTotalResult.Text = String.Format("{0:C}", decTotalCharge)
114
115
           End If
116
117
        End Sub ' txtWhiteBlackQuantity_TextChanged
118
119
        Private Sub txtRedBlackQuantity_TextChanged(ByVal sender As _
120
           System.Object, ByVal e As System.EventArgs) Handles _
121
           txtRedBlackQuantity.TextChanged
122
123
           ' store quantity entered as Integer
124
           Dim intNumberOfRedBlack As Integer = _
125
              Val(txtRedBlackQuantity.Text)
126
127
           ' check validity of order before calculating totals
128
           ' and display message for invalid orders
```

```
129
130
            ' display message if user tries to enter a value
            ' without selecting CheckBox
131
           If intNumberOfRedBlack <> 0 AndAlso _
132
133
              chkRedBlack.Checked = False Then
134
135
               ' keep red/black quantity at 0
136
              txtRedBlackQuantity.Text = 0
137
              ' display message in dialog
138
139
              MessageBox.Show( _
140
                  "Please check item you wish to purchase", _
                 "No Item Selected", MessageBoxButtons.OK, _
141
142
                 MessageBoxIcon.Exclamation)
143
144
           ' display message if shipping information is not supplied
145
           ElseIf _
146
              (txtOrderNumber.Text = "") _
              OrElse (txtName.Text = "") _
147
148
              OrElse (txtAddressLine1.Text = "") _
              OrElse (txtCityStateZip.Text = "") Then
149
150
151
               ' display message in dialog
152
              MessageBox.Show( _
153
                  "Please fill out all information fields", _
                 "Empty Fields", MessageBoxButtons.OK, __
154
155
                 MessageBoxIcon.Exclamation)
156
157
            ' display message if negative number entered
158
           ElseIf _
159
              (intNumberOfRedBlack < 0) Then
160
              txtRedBlackQuantity.Text = 0
161
              MessageBox.Show( _
162
                  "Please enter a positive quantity", _
163
                 "Bad Input", MessageBoxButtons.OK, _
164
                 MessageBoxIcon.Exclamation)
165
166
           Else ' calculate totals
167
168
               ' individual totals
169
               ' total of white/black dice
170
              Dim decWhiteBlackTotals As Decimal = _
171
                 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text
172
173
              ' total of red/black dice
174
              Dim decRedBlackTotals As Decimal =
175
                 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text
176
177
               ' total of blue/black dice
178
              Dim decBlueBlackTotals As Decimal = _
                 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text
179
180
181
               ' display individual totals
              lblWhiteBlackTotals.Text = _
182
183
                  String.Format("{0:C}", decWhiteBlackTotals)
184
              lblRedBlackTotals.Text = _
185
                  String.Format("{0:C}", decRedBlackTotals)
186
              lblBlueBlackTotals.Text =
187
                 String.Format("{0:C}", decBlueBlackTotals)
188
189
               ' subtotal, before tax and shipping
```

190	Dim decSubtotal As Decimal = decWhiteBlackTotals _
191	+ decRedBlackTotals + decBlueBlackTotals
192	
193	lblSubtetelDecult Text String Formet("[0:C]" decSubtetel)
	<pre>lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal)</pre>
194	
195	' calculate and display tax
196	Dim decTax As Decimal = decSubtotal * 0.05
197	
198	lblTaxResult.Text = String.Format("{0:C}", decTax)
	1011dxxesu(1.1ex) = 5(11)(9.701)(a)((0.0)), $(0.0)$
199	
200	' shipping
201	' \$1.50 for up to 20 items
202	' free after 20 items
203	Dim intNumberOfItems As Integer = _
	· · · · · · · · · · · · · · · · · · ·
204	(Val(txtWhiteBlackQuantity.Text) + _
205	Val(txtRedBlackQuantity.Text) + _
206	Val(txtBlueBlackQuantity.Text))
207	
208	Dim decShippingCost As Decimal = 0.0
	Dhill decsirippingeose As beenilar = 0.0
209	
210	' shipping if \$1.50 if under 20 items ordered
211	<pre>If intNumberOfItems &lt;= 20 AndAlso _</pre>
212	intNumberOfItems > 0 Then
213	
214	dochinningCost 1 F
	decShippingCost = 1.5
215	
216	End If
217	
218	' display shipping cost
219	<pre>lblShippingResult.Text = _</pre>
220	
	<pre>String.Format("{0:C}", decShippingCost)</pre>
221	
222	' calculate and display total charge
223	Dim decTotalCharge As Decimal = decSubtotal + decTax + _
224	decShippingCost
225	
226	lblTotalDecult Toyt Stains Format("[0,0]" decTatal(bases)
	<pre>lblTotalResult.Text = String.Format("{0:C}", decTotalCharge)</pre>
227	
228	End If
229	
230	End Sub ' txtRedBlackQuantity_TextChanged
231	
	Derivate Cub tutDlugDlackOventity TautChanged(DuVal conder Ac
232	Private Sub txtBlueBlackQuantity_TextChanged(ByVal sender As _
233	System.Object, ByVal e As System.EventArgs) Handles _
234	txtBlueBlackQuantity.TextChanged
235	
236	' store quantity entered as Integer
237	Dim intNumberOfBlueBlack As Integer = _
238	Val(txtBlueBlackQuantity.Text)
239	
240	' check validity of order before calculating totals
241	' and display message for invalid orders
242	
242	L display massage if user trics to optom a value
	' display message if user tries to enter a value
244	' without selecting CheckBox
245	<pre>If intNumberOfBlueBlack &lt;&gt; 0 AndAlso _</pre>
246	chkBlueBlack.Checked = False Then
247	
248	koon blue/black quantity at 0
	' keep blue/black quantity at 0
249	<pre>txtBlueBlackQuantity.Text = 0</pre>
250	

252       MessageBox.Show(_         253       "Please check item you wish to purchase", _         254       "No Item Selected", MessageBoxButtons.OK, _         255       MessageBoxIcon.Exclamation)         256       ElseIf _         257       ' display message if shipping information is not supplied         258       ElseIf _         259       (txtAddressLineI.Text = "") _         260       OrElse (txtAddressLineI.Text = "") _         261       OrElse (txtAddressLineI.Text = "") _         262       OrElse (txtCityStateZip.Text = "") Then         263       ' display message in dialog         264       ' display message if negative number is entered         276       "Please fill out all information fields", _         276       "Please enter a positive quantity", _         276       "Biself _         276       "Please enter a positive quantity", _         276       "Please enter a positive quantity", _         277       "Sad Input", MessageBoxButtons.OK, _         278       MessageBoxIcon.Exclamation)         279       "Bisel factotals         270       "Please enter a positive quantity",         276       "Please enter a positive quantity",         277       "SageBoxIcon.Exclam	251	' display message in dialog
<pre>253</pre>		
<pre>254</pre>		
255MessageBoxIcon.Exclamation)266' display message if shipping information is not supplied267' display message if shipping information is not supplied268ElseIf _260OrElse (txtName.Text = "") _261OrElse (txtCityStateZip.Text = "") _262OrElse (txtCityStateZip.Text = "") _263' display message in dialog264' Please fill out all information fields", _266"Please fill out all information fields", _277"Empty Fields", MessageBoxButtons.OK, _278MessageBoxIcon.Exclamation)279' display message if negative number is entered271ElseIf _272(inNumberOfBlueBlack < 0) Then		
<pre>266 267 ' display message if shipping information is not supplied 268 Elself _ 269 (txtOrderNumber.Text = "") _ 260 OrElse (txtAddressLinel.Text = "") _ 261 OrElse (txtAddressLinel.Text = "") _ 262 OrElse (txtCityStateZip.Text = "") Then 263 264 ' display message in dialog 265 MessageBox.Show( _ 266 "Please fill out all information fields", _ 267 "Please fill out all information fields", _ 268 MessageBox.Cone.Exclamation) 269 270 ' display message if negative number is entered 271 Elself _ 272 (intNumberOfBlueBlack &lt; 0) Then 273 274 txtBlueBlackQuantity.Text = 0 275 MessageBox.Show( _ 276 "Please enter a positive quantity", _ 277 "Bad Input", MessageBoxButtons.OK, _ 278 MessageBox.Con.Exclamation) 280 Else ' calculate totals 281 282 ' individual totals 283 ' total of whiteblackdice 284 Dim decWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 Val(txtWhiteBlackQuantity.Text) * lblBlueBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtRedBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 IblWhiteBlackTotals.Text = _ 297 String.Format("(0:C)", decRedBlackTotals) 208 IblBlueBlackTotals.Text = _ 297 String.Format("(0:C)", decRedBlackTotals) 309 IblBlueBlackTotals.Text = _ 301 String.Format("(0:C)", decRedBlackTotals) 302 IblBlueBlackTotals.Text = _ 303 / subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals] 305 + decRedBlackTotals + decBlueBlackTotals 306 + decRedBlackTotals = decWhiteBlackTotals 307 IblSubtotalResult.Text = String.Format("{0:C}", decSubtotal} 308 / calculate and display tax</pre>		
<pre>257</pre>		· · · · · · · · · · · · · · · · · · ·
<pre>Elseif</pre>		
<pre>259 (txtOrderNumber.Text = "") _ 260 OrElse (txtName.Text = "") _ 261 OrElse (txtAddresslinel.Text = "") Then 262 OrElse (txtAddresslinel.Text = "") Then 263 ' 264 ' display message in dialog 265 MessageBox.Show(_ 266 " "Please fill out all information fields", _ 267 "Empty Fields", MessageBoxButtons.OK, _ 268 MessageBoxIcon.Exclamation) 270 ' display message if negative number is entered 271 ElseIf _ 272 (intNumberOfBlueBlack &lt; 0) Then 273 txtBlueBlackQuantity.Text = 0 274 txtBlueBlackQuantity.Text = 0 275 MessageBoxIcon.Exclamation) 276 "Please enter a positive quantity", _ 277 "Bad Input", MessageBoxButtons.OK, _ 278 MessageBoxCon.Exclamation) 279 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 Val(txtWhiteBlackQuantity.Text) * lblRedBlackPrice.Text 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtHedBlackQuantity.Text) * lblRedBlackPrice.Text 290 ' total of blue/black dice 291 ' total of blue/black dice 292 Dim decBueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblRedBlackPrice.Text 294 ' display individual totals 295 ' display individual totals 296 IblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decRedBlackTotals) 298 IblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 309 IblBueBlackTotals.text = _ 300 String.Format("{0:C}", decRedBlackTotals) 300 IblBlueBlackTotals.text = _ 300 String.Format("{0:C}", decRedBlackTotals) 301 JblBueBlackTotals.text = _ 302 String.Format("{0:C}", decRedBlackTotals] 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 ' calculate and display tax</pre>		
<pre>260</pre>	258	ElseIf _
<pre>261 OrElse (txtAddressLineL.Text = "") _ 262 OrElse (txtCityStateZip.Text = "") Then 263 264 ' display message in dialog 265 MessageBox.Show(_ 266 "Please fill out all information fields", _ 267 "Empty Fields", MessageBoxButtons.OK, _ 268 MessageBoxIcon.Exclamation) 269 270 ' display message if negative number is entered 271 ElseIf _ 272 (intNumberOfBlueBlack &lt; 0) Then 273 274 txtBlueBlackQuantity.Text = 0 275 MessageBox.Show(_ 276 "Please enter a positive quantity", _ 277 "Bad Input", MessageBoxUttons.OK, _ 278 MessageBox.Show(_ 276 ''Please enter a positive quantity", _ 278 MessageBoxIcon.Exclamation) 279 280 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtRedBlackTotals As Decimal = _ 286 Val(txtRedBlackTotals As Decimal = _ 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 298 Val(txtRedBlackTotals As Decimal = _ 299 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals.As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 ' display individual totals 297 String.Format("(0:C)", decCMBlackTotals) 208 lblRedBlackTotals.Text = _ 209 String.Format("(0:C)", decBlueBlackTotals) 209 ' subtotal, before tax and shipping 204 Dim decSubtotal As Decimal = decWhiteBlackTotals 209 ' subtotal, before tax and shipping 204 Dim decSubtotal As Decimal = decWhiteBlackTotals 205 ' display individual totals 206 ' subtotal, before tax and shipping 207 String.Format("(0:C)", decBlueBlackTotals 208 ' subtotal, before tax and shipping 209 ' calculate and display tax 209 ' calculate and display tax </pre>	259	<pre>(txtOrderNumber.Text = "") _</pre>
<pre>262 OrElse (txtCityStateZip.Text = "") Then 263 264 ' display message in dialog 265 MessageBox.Show(_ 266 "Please fill out all information fields", 267 "Empty Fields", MessageBoxButtons.OK, 268 MessageBox.Con.Exclamation) 269 270 ' display message if negative number is entered 271 ElseIf 272 (intNumberOfBlueBlack &lt; 0) Then 273 274 txtBlueBlackQuantity.Text = 0 275 MessageBox.Show( 276 "Please enter a positive quantity", 277 "Bad Input", MessageBoxButtons.OK, 278 MessageBox.Con.Exclamation) 279 280 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = 289 Val(txtRedBlackQuantity.Text) * lblBueBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = 293 Val(txtBueBlackTotals As Decimal = 294 Val(txtBueBlackTotals As Decimal = 295 ' display individual totals 296 lblWhiteBlackTotals.Text = 301 String.Format("{0:C}", decRedBlackTotals) 302 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 ' calculate and display tax 307 307 308 309 309 309 309 300 300 300 300 300 300</pre>	260	<pre>OrElse (txtName.Text = "") _</pre>
<pre>262 OrElse (txtCityStateZip.Text = "") Then 263 264 ' display message in dialog 265 MessageBox.Show(_ 266 "Please fill out all information fields", 267 "Empty Fields", MessageBoxButtons.OK, 268 MessageBox.Con.Exclamation) 269 270 ' display message if negative number is entered 271 ElseIf 272 (intNumberOfBlueBlack &lt; 0) Then 273 274 txtBlueBlackQuantity.Text = 0 275 MessageBox.Show( 276 "Please enter a positive quantity", 277 "Bad Input", MessageBoxButtons.OK, 278 MessageBox.Con.Exclamation) 279 280 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = 289 Val(txtRedBlackQuantity.Text) * lblBueBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = 293 Val(txtBueBlackTotals As Decimal = 294 Val(txtBueBlackTotals As Decimal = 295 ' display individual totals 296 lblWhiteBlackTotals.Text = 301 String.Format("{0:C}", decRedBlackTotals) 302 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 ' calculate and display tax 307 307 308 309 309 309 309 300 300 300 300 300 300</pre>	261	
<pre>263 264 265 266 266 267 268 266 267 268 268 269 269 270 269 270 27 26 271 272 27 27 27 27 27 27 27 27 27 27 27 27</pre>		
<pre>264</pre>		
265MessageBox.Show(_ "Please fill out all information fields", _ "Empty Fields", MessageBoxLtons.OK, _ MessageBoxLcon.Exclamation)267"Empty Fields", MessageBoxRutons.OK, _ MessageBoxLcon.Exclamation)270' display message if negative number is entered271ElseIf _ (intNumberOfBlueBlack < 0) Then		
<pre>266 "Please fill out all information fields", 267 "Empty Fields", MessageBoxButtons.OK, 268 MessageBoxIcon.Exclamation) 269 270 ' display message if negative number is entered 271 ElseIf 272 (intNumberOfBlueBlack &lt; 0) Then 273 txtBlueBlackQuantity.Text = 0 276 "Please enter a positive quantity", 277 "Bad Input", MessageBoxButtons.OK, 278 MessageBoxIcon.Exclamation) 279 280 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 285 Val(txtWhiteBlackQuantity.Text) * lblRedBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Sas Decimal = 297 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblBkdBlackTotals.Text = 297 String.Format("{0:C}", decRdBlackTotals) 308 lblBueBlackTotals.Text = 300 lblBlueBlackTotals.Text = 301 String.Format("{0:C}", decRdBlackTotals) 302 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax 309 ' calculate and display tax 309 ' calculate and display tax 301 String.Format("{0:C}", decSubtotal) 302 ' subtotal, before tax and shipping 303 ' calculate and display tax 304 Display tax 305 * calculate and display tax</pre>		
<pre>267 "Empty Fields", MessageBoxButtons.OK,</pre>		
<pre>268 MessageBoxIcon.Exclamation) 269 270 'display message if negative number is entered 271 ElseIf _ 272 (intNumberOfBlueBlack &lt; 0) Then 273 274 txtBlueBlackQuantity.Text = 0 275 MessageBox.Show( _ 276 "Please enter a positive quantity", _ 277 "Bad Input", MessageBoxButtons.OK, _ 278 MessageBoxIcon.Exclamation) 279 280 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim deckMiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 289 290 ' total of blue/black dice 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals 300 iblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals] 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 ' calculate and display tax </pre>		, _
<pre>269 270 ' display message if negative number is entered 271 ElseIf _ 272 (intNumberOfBlueBlack &lt; 0) Then 273 274 txtBlueBlackQuantity.Text = 0 275 MessageBox.Show( _ 276 "Please enter a positive quantity", _ 277 "Bad Input", MessageBoxButtons.OK, _ 278 MessageBoxIcon.Exclamation) 279 280 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackTotals As Decimal = _ 286 Val(txtWhiteBlackTotals As Decimal = _ 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblWhiteBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decRedBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 299 lblBueBlackTotals.Text = _ 290 String.Format("{0:C}", decBlueBlackTotals) 200 lblBlueBlackTotals.Text = _ 201 String.Format("{0:C}", decBlueBlackTotals) 201 lblBueBlackTotals.Text = _ 202 String.Format("{0:C}", decBlueBlackTotals) 203 ' subtotal, before tax and shipping 204 Dim decSubtotal As Decimal = decWhiteBlackTotals 205 + decRedBlackTotals + decBlueBlackTotals 206 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 207 / calculate and display tax </pre>		
<pre>270 ' display message if negative number is entered 271 Elself _ 272 (intNumberOfBlueBlack &lt; 0) Then 273 txtBlueBlackQuantity.Text = 0 275 MessageBox.Show( _ 276 "Please enter a positive quantity", _ 277 "Bad Input", MessageBoxButtons.OK, _ 278 MessageBoxIcon.Exclamation) 279 280 Else ' calculate totals 281 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 ' total of blue/black dice 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtRedBlackQuantity.Text) * lblBlueBlackPrice.Text 294 ' 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 308 lblRedBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal.before tax and shipping 303 ' subtotal.before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 306 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 ' calculate and display tax</pre>		
<pre>271 ElseIf _ 272 (intNumberOfBlueBlack &lt; 0) Then 273 274 txtBlueBlackQuantity.Text = 0 275 MessageBox.Show( _ 276 "Please enter a positive quantity", _ 277 "Bad Input", MessageBoxButtons.OK, _ 278 MessageBoxIcon.Exclamation) 279 280 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decRedBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decBlueBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 306 ' calculate and display tax </pre>		
<pre>272 (intNumberOfBlueBlack &lt; 0) Then 273 274 txtBlueBlackQuantity.Text = 0 275 MessageBox.Show(_ 276 "Please enter a positive quantity", _ 277 "Bad Input", MessageBoxButtons.OK, _ 278 MessageBoxIcon.Exclamation) 280 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 300 lblBlueBlackTotals.Text = _ 299 String.Format("{0:C}", decBlueBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		' display message if negative number is entered
<pre>272 (intNumberOfBlueBlack &lt; 0) Then 273 274 txtBlueBlackQuantity.Text = 0 275 MessageBox.Show(_ 276 "Please enter a positive quantity", _ 277 "Bad Input", MessageBoxButtons.OK, _ 278 MessageBoxIcon.Exclamation) 280 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 300 lblBlueBlackTotals.Text = _ 299 String.Format("{0:C}", decBlueBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>	271	ElseIf _
<pre>273 274 274 274 275 275 275 276 277 277 278 277 278 278 278 278 278 278</pre>		(intNumberOfBlueBlack < 0) Then
274txtBlueBlackQuantity.Text = 0275MessageBox.Show( _276"Please enter a positive quantity", _277"Bad Input", MessageBoxButtons.OK, _278MessageBoxIcon.Exclamation)279		
275MessageBox.Show( _276"Please enter a positive quantity", _277"Bad Input", MessageBoxButtons.OK, _278MessageBoxIcon.Exclamation)279280280Else ' calculate totals281' individual totals282' individual totals283' total of white/black dice284Dim decWhiteBlackTotals As Decimal = _285Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text286' total of red/black dice287' total of red/black dice288Dim decRedBlackTotals As Decimal = _289Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text290' total of blue/black dice291' total of blue/black dice292Dim decBlueBlackTotals As Decimal = _293Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text294' display individual totals295' display individual totals296IblWhiteBlackTotals.Text = _297String.Format("{0:C}", decRedBlackTotals)298IblRedBlackTotals.Text = _299String.Format("{0:C}", decBlueBlackTotals)300' subtotal, before tax and shipping301Subtotal, before tax and shipping303' subtotal As Decimal = decWhiteBlackTotals304Dim decSubtotal As Decimal = decWhiteBlackTotals305+ decRedBlackTotals + decBlueBlackTotals3061blSubtotalResult.Text = String.Format("{0:C}", decSubtotal)308' calculate and display tax <td></td> <td></td>		
<pre>276 "Please enter a positive quantity", _ 277 "Bad Input", MessageBoxButtons.OK, _ 278 MessageBoxIcon.Exclamation) 279 280 Else ' calculate totals 281 ' individual totals 281 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 280 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal As Decimal = decWhiteBlackTotals 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 309 ' calculate and display tax </pre>		
<pre>277 "Bad Input", MessageBoxButtons.OK, _ 278 MessageBoxIcon.Exclamation) 279 280 Else ' calculate totals 281 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decRedBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 ' calculate and display tax </pre>		<b>3</b>
278MessageBoxIcon.Exclamation)279280Else ' calculate totals281' individual totals282' individual totals283' total of white/black dice284Dim decWhiteBlackTotals As Decimal = _285Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text286' total of red/black dice287' total of hue/black dice288Dim decRedBlackTotals As Decimal = _289Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text290' total of blue/black dice291' total of blue/black dice293Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text294' display individual totals295' display individual totals296lblWhiteBlackTotals.Text = _297String.Format("{0:C}", decWhiteBlackTotals)298lblBlueBlackTotals.Text = _299String.Format("{0:C}", decBlueBlackTotals)300lblBlueBlackTotals.Text = _301string.Format("{0:C}", decBlueBlackTotals)302' subtotal, before tax and shipping304Dim decSubtotal As Decimal = decWhiteBlackTotals305+ decRedBlackTotals + decBlueBlackTotals306iblSubtotalResult.Text = String.Format("{0:C}", decSubtotal)308' calculate and display tax		
<pre>279 280 Else ' calculate totals 281 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decBlueBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 309 ' calculate and display tax </pre>		
<pre>280 Else ' calculate totals 281 ' individual totals 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 ' 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 ' total of blue/black dice 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 ' display individual totals 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decRedBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 ' subtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 ' calculate and display tax</pre>		
<pre>281 282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decRedBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decBlueBlackTotals) 290 lblBlueBlackTotals.Text = _ 291 string.Format("{0:C}", decBlueBlackTotals) 293 lblBlueBlackTotals.Text = _ 294 string.Format("{0:C}", decBlueBlackTotals) 304 lblBlueBlackTotals + decBlueBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 ' calculate and display tax 309 ' calculate and display tax</pre>		
<pre>282 ' individual totals 283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decBlueBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 306 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax </pre>		Else ' calculate totals
<pre>283 ' total of white/black dice 284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decBlueBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 ' calculate and display tax</pre>		
<pre>284 Dim decWhiteBlackTotals As Decimal = _ 285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 ' calculate and display tax</pre>	282	' individual totals
<pre>285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decBlueBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 ' calculate and display tax</pre>	283	' total of white/black dice
<pre>285 Val(txtWhiteBlackQuantity.Text) * lblWhiteBlackPrice.Text 286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decRedBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decBlueBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 ' calculate and display tax</pre>	284	Dim decWhiteBlackTotals As Decimal =
<pre>286 287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>	285	
<pre>287 ' total of red/black dice 288 Dim decRedBlackTotals As Decimal = _ 289 Val(txRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 101RedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>288 Dim decRedBlackTotals As Decimal =</pre>		
<pre>289 Val(txtRedBlackQuantity.Text) * lblRedBlackPrice.Text 290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 ' calculate and display tax</pre>		
<pre>290 291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>291 ' total of blue/black dice 292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		Val(txtRedBlackQuantity.lext) * IDIRedBlackPrice.lext
<pre>292 Dim decBlueBlackTotals As Decimal = _ 293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' subtotal, before tax and shipping 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>293 Val(txtBlueBlackQuantity.Text) * lblBlueBlackPrice.Text 294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>294 295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ + decRedBlackTotals + decBlueBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>295 ' display individual totals 296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>296 lblWhiteBlackTotals.Text = _ 297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 ' 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>	295	' display individual totals
<pre>297 String.Format("{0:C}", decWhiteBlackTotals) 298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ 4 decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>298 lblRedBlackTotals.Text = _ 299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ 4 decRedBlackTotals + decBlueBlackTotals 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		—
<pre>299 String.Format("{0:C}", decRedBlackTotals) 300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 303 'subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ 4 decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>300 lblBlueBlackTotals.Text = _ 301 String.Format("{0:C}", decBlueBlackTotals) 302 303 'subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ 4 decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>301 String.Format("{0:C}", decBlueBlackTotals) 302 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>302 303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>303 ' subtotal, before tax and shipping 304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>304 Dim decSubtotal As Decimal = decWhiteBlackTotals _ 305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>305 + decRedBlackTotals + decBlueBlackTotals 306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>306 307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>		
<pre>307 lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal) 308 309 ' calculate and display tax</pre>	306	
308309' calculate and display tax		
309 ' calculate and display tax		
		' calculate and display tax
310 Dim decTax As Decimal = decSubtotal * 0.05	310	
311		

312	lblTaxResult.Text = String.Format("{0:C}", decTax)
313	
314	' shipping
315	' \$1.50 for up to 20 items
316	' free after 20 items
317	Dim intNumberOfItems As Integer = _
318	Val(txtWhiteBlackQuantity.Text) + _
319	Val(txtRedBlackQuantity.Text) + _
320	Val(txtBlueBlackQuantity.Text)
321	
322	Dim decShippingCost As Decimal = 0.0
323	Dim decshippingcost As bechildi – 0.0
	Labianian is \$1.50 if under 20 items and and
324	' shipping is \$1.50 if under 20 items ordered
325	<pre>If intNumberOfItems &lt;= 20 AndAlso _</pre>
326	intNumberOfItems > 0 Then
327	
328	<pre>decShippingCost = 1.5</pre>
329	
330	End If
331	
332	' display shipping cost
333	lblShippingResult.Text = _
334	<pre>String.Format("{0:C}", decShippingCost)</pre>
335	Ser high of mac ( [ore] ) accomptingeose)
336	' calculate and display total charge
337	Dim decTotalCharge As Decimal = decSubtotal + decTax + _
	-
338	decShippingCost
339	
340	<pre>lblTotalResult.Text = String.Format("{0:C}", decTotalCharge)</pre>
341	
342	End If
343	
344	End Sub ' txtBlueBlackQuantity_TextChanged
345	
346	' clear all fields
347	<pre>Private Sub btnClear_Click(ByVal sender As System.Object, _</pre>
348	ByVal e As System.EventArgs) Handles btnClear.Click
349	
350	' set all fields to their original values
351	txtOrderNumber.Text = "0"
352	txtName.Text = "Enter name here"
353	txtAddressLine1.Text = "Address Line 1"
354	txtAddressLine2.Text = "Address Line 2"
355	<pre>txtCityStateZip.Text = "City, State, zip"</pre>
356	<pre>txtWhiteBlackQuantity.Text = "0"</pre>
357	<pre>txtRedBlackQuantity.Text = "0"</pre>
358	<pre>txtBlueBlackQuantity.Text = "0"</pre>
359	lblWhiteBlackTotals.Text = "\$0.00"
360	<pre>lblRedBlackTotals.Text = "\$0.00"</pre>
361	<pre>lblBlueBlackTotals.Text = "\$0.00"</pre>
362	<pre>lblSubtotalResult.Text = "\$0.00"</pre>
363	<pre>lblTaxResult.Text = "\$0.00"</pre>
364	lblShippingResult.Text = "\$0.00"
365	lblTotalResult.Text = "\$0.00"
366	chkWhiteBlack.Checked = False
367	chkRedBlack.Checked = False
368	chkBlueBlack.Checked = False
	CHRUTUEDTACK.CHECKEU = FAISE
369	
370	End Sub ' btnClear_Click
371	
372	Private Sub chkWhiteBlack_CheckedChanged(ByVal sender As _

373	System.Object, ByVal e As System.EventArgs) Handles _
374	chkWhiteBlack.CheckedChanged
375	en ann eos raent en en en en gea
376	louist and a show we show Trans the U.O.U.
	<pre>txtWhiteBlackQuantity.Text = "0"</pre>
377	lblWhiteBlackTotals.Text = "0"
378	
379	' individual totals
380	' total of white/black dice
381	Dim decWhiteBlackTotals As Decimal = _
382	Val(txtWhiteBlackQuantity.Text) * _
383	lblWhiteBlackPrice.Text
384	
385	' total of red/black dice
386	Dim decRedBlackTotals As Decimal = _
387	Val(txtRedBlackQuantity.Text) * _
388	lblRedBlackPrice.Text
389	Ibinedbracki i receirexe
390	' total of blue/black dice
391	Dim decBlueBlackTotals As Decimal = _
392	Val(txtBlueBlackQuantity.Text) * _
393	lblBlueBlackPrice.Text
394	
395	' display individual totals
396	lblWhiteBlackTotals.Text = _
397	<pre>String.Format("{0:C}", decWhiteBlackTotals)</pre>
398	lblRedBlackTotals.Text = _
399	<pre>String.Format("{0:C}", decRedBlackTotals)</pre>
400	lblBlueBlackTotals.Text = _
401	<pre>String.Format("{0:C}", decBlueBlackTotals)</pre>
402	
402	Laubrard before the and objection
	' subtotal, before tax and shipping
404	<pre>Dim decSubtotal As Decimal = decWhiteBlackTotals + _</pre>
405	decRedBlackTotals + decBlueBlackTotals
406	
407	<pre>lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal)</pre>
408	
409	' calculate and display tax
410	Dim decTax As Decimal = decSubtotal * 0.05
	DTIN GECTAX AS DECTINAT = GECSUDIOLAT " 0.03
411	
412	lblTaxResult.Text = String.Format("{0:C}", decTax)
413	
414	' shipping
415	' \$1.50 for up to 20 items
416	' free after 20 items
417	Dim intNumberOfItems As Integer = _
418	Val(txtWhiteBlackQuantity.Text) + _
419	Val(txtRedBlackQuantity.Text) + _
420	Val(txtBlueBlackQuantity.Text)
421	
422	Dim decShippingCost As Decimal = 0.0
423	
424	' shipping is \$1.50 if under 20 items ordered
424	
	<pre>intNumberOfItems &lt;= 20 AndAlso _ intNumberOfItems &lt;= 0 Them</pre>
426	intNumberOfItems > 0 Then
427	
428	<pre>decShippingCost = 1.5</pre>
429	
430	End If
431	
432	' display shipping cost
433	lblShippingResult.Text = _

434	<pre>String.Format("{0:C}", decShippingCost)</pre>
435	
436	' calculate and display total charge
437	Dim decTotalCharge As Decimal = decSubtotal + decTax + _
438	decShippingCost
439	
440	<pre>lblTotalResult.Text = String.Format("{0:C}", decTotalCharge)</pre>
441	End Sub L shuthitsPlack CharledChanned
442 443	End Sub ' chkWhiteBlack_CheckedChanged
443	Private Sub chkRedBlack_CheckedChanged(ByVal sender As _
445	System.Object, ByVal e As System.EventArgs) Handles _
446	chkRedBlack.CheckedChanged
447	enkkedbrack, encekedenanged
448	<pre>txtRedBlackQuantity.Text = "0"</pre>
449	lblRedBlackTotals.Text = "0"
450	
451	' individual totals
452	' total of white/black dice
453	Dim decWhiteBlackTotals As Decimal = _
454	Val(txtWhiteBlackQuantity.Text) * _
455	<pre>lblWhiteBlackPrice.Text</pre>
456	
457	' total of red/black dice
458	<pre>Dim decRedBlackTotals As Decimal = _</pre>
459	Val(txtRedBlackQuantity.Text) * _
460	lblRedBlackPrice.Text
461	
462	' total of blue/black dice
463	Dim decBlueBlackTotals As Decimal = _
464	Val(txtBlueBlackQuantity.Text) * _
465	lblBlueBlackPrice.Text
466 467	L dienlay individual totale
468	' display individual totals lblWhiteBlackTotals.Text = _
469	<pre>String.Format("{0:C}", decWhiteBlackTotals)</pre>
470	lblRedBlackTotals.Text = _
471	<pre>String.Format("{0:C}", decRedBlackTotals)</pre>
472	lblBlueBlackTotals.Text = _
473	String.Format("{0:C}", decBlueBlackTotals)
474	
475	' subtotal, before tax and shipping
476	Dim decSubtotal As Decimal = decWhiteBlackTotals + _
477	decRedBlackTotals + decBlueBlackTotals
478	
479	<pre>lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal)</pre>
480	
481	' calculate and display tax
482	Dim decTax As Decimal = decSubtotal * 0.05
483	
484	<pre>lblTaxResult.Text = String.Format("{0:C}", decTax)</pre>
485	1 shimin
486	' shipping ' \$1.50 for up to 20 items
487 488	' \$1.50 for up to 20 items ' free after 20 items
400 489	Dim intNumberOfItems As Integer = _
409	Val(txtWhiteBlackQuantity.Text) + _
490	Val(txtRedBlackQuantity.Text) + _
492	Val(txtBlueBlackQuantity.Text)
493	
494	Dim decShippingCost As $Decimal = 0.0$

495			
496	<pre>If intNumberOfItems &lt;= 20 AndAlso intNumberOfItems &gt; 0 Then</pre>		
497			
	<pre>decShippingCost = 1.5</pre>		
498			
499	End If		
500			
501	' display shipping cost		
502	lblShippingResult.Text = _		
503	<pre>String.Format("{0:C}", decShippingCost)</pre>		
504	Sering Tormat ( [0.0] ; decomposition		
505	' calculate and display total charge		
506	Dim decTotalCharge As Decimal = decSubtotal + decTax + _		
507	decShippingCost		
508			
509	lblTotalResult.Text = <pre>String.Format("{0:C}", decTotalCharge)</pre>		
510			
511	End Sub ' chkRedBlack_CheckedChanged		
	Lind Sub Clikkeublack_checkeuchangeu		
512			
513	Private Sub chkBlueBlack_CheckedChanged(ByVal sender As _		
514	System.Object, ByVal e As System.EventArgs) Handles _		
515	chkBlueBlack.CheckedChanged		
516			
517	<pre>txtBlueBlackQuantity.Text = "0"</pre>		
518	lblBlueBlackTotals.Text = "0"		
519			
	to deal to defend the end of		
520	'individual totals		
521	' total of white/black dice		
522	Dim decWhiteBlackTotals As Decimal = _		
523	Val(txtWhiteBlackQuantity.Text) * _		
524	lblWhiteBlackPrice.Text		
525			
526	' total of red/black dice		
527	Dim decRedBlackTotals As Decimal = _		
528	Val(txtRedBlackQuantity.Text) * _		
529	lblRedBlackPrice.Text		
530			
531	' total of blue/black dice		
532	Dim decBlueBlackTotals As Decimal = _		
533	Val(txtBlueBlackQuantity.Text) * _		
534	lb]B]ueB]ackPrice.Text		
535	15151debideki i recireke		
	1. Address Rest address Association		
536	' display individual totals		
537	lblWhiteBlackTotals.Text = _		
538	<pre>String.Format("{0:C}", decWhiteBlackTotals)</pre>		
539	lblRedBlackTotals.Text = _		
540	<pre>String.Format("{0:C}", decRedBlackTotals)</pre>		
541	<pre>lblBlueBlackTotals.Text = _</pre>		
542	<pre>String.Format("{0:C}", decBlueBlackTotals)</pre>		
543			
544	' subtotal, before tax and shipping		
545	Dim decSubtotal As Decimal = decWhiteBlackTotals + _		
546	decRedBlackTotals + decBlueBlackTotals		
547			
548	<pre>lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal)</pre>		
549			
550	' calculate and display tax		
551	Dim decTax As Decimal = decSubtotal $* 0.05$		
552			
553	lblTaxPacult Taxt - String Format("(0.0)" decTax)		
	<pre>lblTaxResult.Text = String.Format("{0:C}", decTax)</pre>		
554			
555	' shipping		

556	' \$1.50 for up to 20 items
557	' free after 20 items
558	Dim intNumberOfItems As Integer = _
559	Val(txtWhiteBlackQuantity.Text) +
560	Val(txtRedBlackQuantity.Text) + _
561	Val(txtBlueBlackQuantity.Text)
562	Dim decShippingCost As Decimal = 0.0
563	
564	' shipping is \$1.50 if under 20 items ordered
565	<pre>If intNumberOfItems &lt;= 20 AndAlso _</pre>
566	<pre>intNumberOfItems &gt; 0 Then</pre>
567	
<b>568</b>	<pre>decShippingCost = 1.5</pre>
569	
570	End If
571	
572	' display shipping cost
573	lblShippingResult.Text = _
574	<pre>String.Format("{0:C}", decShippingCost)</pre>
575	
576	' calculate and display total charge
577	Dim decTotalCharge As Decimal = decSubtotal + decTax + _
578	decShippingCost
579	
580	lblTotalResult.Text = <u>String</u> .Format("{0:C}", decTotalCharge)
581	
582	End Sub ' chkBlueBlack_CheckedChanged
583	
584	End Class ' FrmFuzzyDiceOrderFormEnhanced





# Car Payment Calculator Application

Introducing the Do While...Loop and Do Until...Loop Repetition Statements Solutions

)	Instructor's Manual Exercise Solutions Tutorial 9		
-	MULTIPLE-CHOICE	9.1 Thestatement exect	ites until its loop-continuation condition becomes True.
	QUESTIONS	a) Do WhileLoop	b) Do UntilLoop
	Gelenene	c) Do	d) Loop
		9.2 The statement execu	ites until its loop-continuation condition becomes False.
		a) Do WhileLoop	b) Do UntilLoop
		c) Do	d) Do While
		<b>9.3</b> A(n) loop occurs v False.	when a condition in a Do WhileLoop never becomes
		a) infinite	b) undefined
		c) nested	d) indefinite
		<b>9.4</b> A is a variable that I will execute.	helps control the number of times that a set of statements
		a) repeater	b) counter
		c) loop	d) repetition control statement
		9.5 The control allows u	sers to add and view items in a list.
		a) ListItems	b) SelectBox
		c) ListBox	d) ViewBox
)		<b>9.6</b> In a UML activity diagram, a(r flow of activity.	a) symbol joins two flows of activity into one
		a) merge	b) combine
		c) action state	d) decision
		9.7 Property returns an	object containing all the values in a ListBox.
		a) All	b) List
		c) ListItemValues	d) Items
		9.8 Method deletes all t	he values in a ListBox.
		a) Remove	b) Delete
		c) Clear	d) Del
		9.9 Items's method add	ls an item to a ListBox.
		a) Include	b) Append
		c) Add	d) Insert
		rate.	monthly payments on a loan based on a fixed interest
		a) MonPmt	b) Payment
		c)MonthlyPayment	d) Pmt
		<b>Answers:</b> 9.1) b. 9.2) a. 9.3) a. 9.4)	b. 9.5) c. 9.6) a. 9.7) d. 9.8) c. 9.9) c. 9.10) d.

**EXERCISES** 

**9.11 (Table of Powers Application)** Write an application that displays a table of numbers from 1 to an upper limit, along with each number's squared value (for example, the number n to the power 2, or  $n \land 2$ ) and cubed value (the number n to the power 3, or  $n \land 3$ ). The users should specify the upper limit, and the results should be displayed in a ListBox, as in Fig. 9.20.

Table	e of Powe	rs	_ 🗆 ×
Upper	limit:	5	Calculate
N 1 2 3 4 5	N^2 1 4 9 16	N^3 1 8 27 64	
5	25	125	

Figure 9.20 Table of Powers application's Form.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial09\Exercises\TableOfPowers directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click TableOfPowers.sln in the TableOfPowers directory to open the application.
- c) *Adding a ListBox*. Add a ListBox to the application, as shown in Fig. 9.20. Name the ListBox lstResults.
- d) Adding the Upper limit: TextBox event handler. Double click the Upper limit: Text-Box to generate an event handler for this TextBox's TextChanged event. In this event handler, clear the ListBox.
- e) Adding the Calculate Button event handler. Double click the Calculate Button to generate the empty event handler btnCalculate\_Click. Add the code specified by the remaining steps to this event handler.
- f) Clearing the ListBox. Use method Clear on the Items property to clear the ListBox from any previous data.
- g) Obtaining the upper limit supplied by the user. Assign the value entered by the user in the Upper limit: TextBox to a variable. Note that the TextBox's Name property is set to txtInput.
- h) Adding a header. Use method Add on the Items property to insert a header in the ListBox. The header should label three columns—N, N^2 and N^3. Column headings should be separated by tab characters.
- i) *Calculating the powers from 1 to the specified upper limit.* Use a Do Until...Loop to calculate the squared value and the cubed value of each number from 1 to the upper limit, inclusive. Add an item to the ListBox containing the current number being analyzed, its squared value and its cubed value.
- j) *Incrementing the counter*. Remember to increment the counter appropriately each time through the loop.
- k) Running the application. Select Debug > Start to run your application. Enter an upper limit and click the Calculate Button. Verify that the table of powers displayed contains the correct values.
- 1) *Closing the application.* Close your running application by clicking its close box.
- m) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 9.11 Solution
2	' TableOfPowers.vb
3	
4	Public Class FrmTableOfPowers
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' handles Click event
0	<pre>Private Sub btnCalculate_Click(ByVal sender As System.Object, _</pre>
1	ByVal e As System.EventArgs) Handles btnCalculate.Click
2	

```
13
           Dim intLimit As Integer = 0
                                           upper limit set by user
14
           Dim intCounter As Integer = 1 ' counter begins at 1
15
16
           ' clear ListBox
17
           lstResults.Items.Clear()
18
19
           ' retrieve user input
20
           intLimit = Val(txtInput.Text)
21
22
           ' add header
23
           lstResults.Items.Add("N" & ControlChars.Tab & "N^2" & _
24
              ControlChars.Tab & "N^3")
25
26
           ' calculate and display square and cube of 1 to intLimit
27
           Do Until intCounter > intLimit
28
29
              lstResults.Items.Add(intCounter & ControlChars.Tab & _
30
                 intCounter ^ 2 & ControlChars.Tab & intCounter ^ 3)
31
32
               ' increment counter
33
              intCounter += 1
34
           Loop
35
36
        End Sub ' btnCalculate_Click
37
38
        ' handles TextChanged event
39
        Private Sub txtInput_TextChanged(ByVal sender As System.Object, _
40
           ByVal e As System. EventArgs) Handles txtInput.TextChanged
41
42
           lstResults.Items.Clear()
43
        End Sub ' txtInput_TextChanged
44
45
     End Class ' FrmTableOfPowers
```

**9.12** (Mortgage Calculator Application) A bank offers mortgages that can be repaid in 5, 10, 15, 20, 25 or 30 years. Write an application that allows a user to enter the price of a house (the amount of the mortgage) and the annual interest rate. When the user clicks a Button, the application displays a table of the mortgage length in years together with the monthly payment, as shown in Fig. 9.21.

🔛 Mortgage Calculator		
Mortgage amount:	175000	Calculate
Annual interest rate:	5.7	.0
Mortgage Length (Years)	Monthly P	ayment
5	\$3,358.88	
10	\$1,916.60	
15	\$1,448.54	
20	\$1,223.66	
25	\$1,095.65	
30	\$1,015.70	)

Figure 9.21 Mortgage Calculator application's Form.

- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial09\Exercises\MortgageCalculator directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click MortgageCalculator.sln in the MortgageCalculator directory to open the application.
- c) Adding a ListBox to display the results. Add a ListBox as shown in Fig. 9.21. Name the ListBox lstResults.

- d) Adding a Calculate Button event handler. Double click the Calculate Button to generate the empty event handler btnCalculate\_Click. Add the code specified in the remaining steps to your event handler.
- e) *Converting the annual interest rate to the monthly interest rate.* To convert the annual interest rate from a percent value into its Double equivalent, divide the annual rate by 100. Then divide the Double annual rate by 12 to obtain the monthly rate.
- f) Clearing the ListBox. Use method Clear on the Items property to clear the ListBox from any previous data.
- g) Displaying a header. Use method Add to display a header in the ListBox. The header should be the column headers "Mortgage Length (Years)" and "Monthly Payment", separated by a tab character.
- h) Using a repetition statement. Add a Do While...Loop repetition statement to calculate six monthly payment options for the user's mortgage. Each option has a different number of years that the mortgage can last. For this exercise, use the following number of years: 5, 10, 15, 20, 25 and 30.
- i) *Converting the length of the mortgage from years to months.* Convert the number of years to months.
- j) Calculating the monthly payments for six different mortgages. Use the Pmt function to compute the monthly payments. Pass to the function the monthly interest rate, the number of months in the mortgage and the mortgage amount. Remember that the mortgage amount must be negative, as it represents an amount of money being paid out by the lender.
- k) Displaying the results. Use method Add on the Items property to display the length of the mortgage in years and the monthly payment in the ListBox. You will need to use three tab characters to ensure that the monthly payment appears in the second column.
- Running the application. Select Debug > Start to run your application. Enter a mortgage amount and annual interest rate, then click the Calculate Button. Verify that the monthly payments displayed contain the correct values.
- m) Closing the application. Close your running application by clicking its close box.
- n) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 9.12 Solution
2	' MortgageCalculator.vb
3	
4	Public Class FrmMortgageCalculator
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' handles Calculate Button's Click event
0	Private Sub btnCalculate_Click(ByVal sender As System.Object, _
1	ByVal e As System.EventArgs) Handles btnCalculate.Click
2	
3	Dim intMortgageAmount As Integer = 0 ' mortgage amount
4	Dim dblAnnualRate As Double = $0$ ' annual interest rate
5	Dim dblMonthlyRate As Double = 0 ' monthly interest rate
6	Dim decPayment As Decimal = 0 'monthly payment amount
7	Dim intYears As Integer = 5 'years in mortgage
8	Dim intMonths As Integer = 0 ' months in mortgage
9	bhill fileholichis As fileeger = 0 lilohens fil illohegage
20	' obtain user input
21	<pre>intMortgageAmount = Val(txtMortgageAmount.Text)</pre>
	dblAnnualRate = Val(txtRate.Text) / 100
22 23	ubininualitate = val(txthate.lext) / 100
24	L colculate monthly interact rate
-	' calculate monthly interest rate

```
25
           dblMonthlyRate = dblAnnualRate / 12
26
27
            ' clear previous results from ListBox
28
           lstResults.Items.Clear()
29
30
            ' add header to ListBox
31
           lstResults.Items.Add("Mortgage Length (Years)" & _
32
              ControlChars.Tab & "Monthly Payment")
33
34
           ' perform Pmt calculation and display result for
35
           ' 5, 10, 15, 20, 25 and 30 years
36
           Do While intYears <= 30
37
38
               ' convert years to months for the calculation
39
              intMonths = intYears * 12
40
41
               ' perform calculation
42
              decPayment = Convert.ToDecimal( _
43
                 Pmt(dblMonthlyRate, intMonths, -intMortgageAmount))
44
45
               ' display result
46
              lstResults.Items.Add(intYears & ControlChars.Tab & _
47
                 ControlChars.Tab & ControlChars.Tab & _
48
                 String.Format("{0:C}", decPayment))
49
50
               ' increment counter
51
              intYears += 5
52
           Loop
53
54
        End Sub ' btnCalculate_Click
55
56
     End Class ' FrmMortgageCalculator
```

**9.13** (Office Supplies Application) Create an application that allows a user to make a list of office supplies to buy, as shown in Fig. 9.22. The user should enter the supply in a TextBox and click the Buy Button to add it to the ListBox. The Clear Button removes all the items from the ListBox.

🖳 Office Supplies	
Supply:	
Pencils Pens Paper	
Buy	Clear

Figure 9.22 Office Supplies application's Form.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial09\Exercises\OfficeSupplies directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click OfficeSupplies.sln in OfficeSupplies directory to open the application.
- c) *Adding a ListBox*. Add a ListBox to the Form. Name the ListBox lstSupplies. Place and size it as shown in Fig. 9.22.
- d) Adding an event handler for the Buy Button. Double click the Buy Button to generate the event handler btnBuy\_Click. The event handler should obtain the user input

from the TextBox. The user input is then added as an item into the ListBox. After the input is added to the ListBox, clear the **Supply:** TextBox.

- e) Adding an event handler for the Clear Button. Double click the Clear Button to generate the event handler btnClear\_Click. The event handler should use the Clear method on the Items property to clear the ListBox.
- f) Running the application. Select Debug > Start to run your application. Enter several items into the Supply: TextBox and click the Buy Button after entering each item. Verify that each item is added to the ListBox. Click the Clear Button and verify that all items are removed from the ListBox.
- g) *Closing the application.* Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

Answer:

```
1
      Exercise 9.13 Solution
 2
     ' OfficeSupplies.vb
 3
 4
    Public Class FrmOfficeSupplies
 5
        Inherits System.Windows.Forms.Form
 6
 7
        ' handles Buy Button's Click event
 8
       Private Sub btnBuy_Click(ByVal sender As System.Object, _
 9
           ByVal e As System.EventArgs) Handles btnBuy.Click
10
11
           ' add supply item to ListBox, clear input TextBox
12
           lstSupplies.Items.Add(txtOfficeSupply.Text)
13
           txtOfficeSupply.Text =
14
        End Sub ' btnBuy_Click
15
16
        ' handles Clear Button's Click event
17
       Private Sub btnClear_Click(ByVal sender As System.Object, _
18
          ByVal e As System. EventArgs) Handles btnClear. Click
19
20
           lstSupplies.Items.Clear() ' clear supply items
21
        End Sub ' btnClear_Click
22
23
    End Class ' FrmOfficeSupplies
```

```
What does this code do?
```

**9.14** What is the result of the following code?

```
1
    Dim intX As Integer = 1
    Dim intMysteryValue As Integer = 1
2
3
4
    Do While intX < 6
5
6
       intMysteryValue *= intX
7
       intX += 1
8
    Loop
0
10
    lblDisplay.Text = intMysteryValue
```

**Answer:** intX = 6, intMysteryValue = 120.

What's wrong with this code?

**9.15** Find the error(s) in the following code:

a) Assume that the variable intX is declared and initialized to 1. The loop should total the numbers from 1 to 10.

```
1 Dim intTotal As Integer = 0
2
3 Do Until intX <= 10
4
5 intTotal += intX
6 intX += 1
7 Loop</pre>
```

**Answer:** This loop will never execute, as intX is already less than or equal to 10. The code should use > instead of <=. An alternative solution would be to convert the loop to a Do While...Loop.

```
1 Dim intTotal As Integer = 0
2
3 Do Until intX > 10
4
5 intTotal += intX
6 intX += 1
7 Loop
```

b) Assume that the variable intCounter is declared and initialized to 1. The loop should sum the numbers from 1 to 100.

```
1 Do While intCounter <= 100
2
3 intTotal += intCounter
4 Loop
5
6 intCounter += 1</pre>
```

**Answer:** This is an infinite loop, as intCounter will never be greater than 100. The statement that increments intCounter must be placed within the Do While...Loop statement.

```
1 Do While intCounter <= 100
2
3 intTotal += intCounter
4 intCounter += 1
5 Loop</pre>
```

c) Assume that the variable intCounter is declared and initialized to 1000. The loop should iterate from 1000 to 1.

```
1 Do While intCounter > 0
2
3 lblDisplay.Text = intCounter
4 intCounter += 1
5 Loop
```

**Answer:** The values must decrease. The value 1 should be subtracted from, rather than added to, intCounter.

```
1 Do While intCounter > 0
2
3 lblDisplay.Text = intCounter
4 intCounter -= 1
5 Loop
```

d) Assume that the variable intCounter is declared and initialized to 1. The loop should execute five times, adding the numbers 1-5 to a ListBox.

```
1 Do While intCounter < 5
2
3 lstNumbers.Items.Add(intCounter)
4 intCounter += 1
5 Loop</pre>
```

**Answer:** This loop will execute only four times. To fix the application, the loop-continuation condition should use the <= operator, rather than the < operator.

```
1 Do While intCounter <= 5
2
3 lstNumbers.Items.Add(intCounter)
4 intCounter += 1
5 Loop</pre>
```

Using the Debugger

**9.16** (Odd Numbers Application) The Odd Numbers application should display all of the odd integers between one and the number input by the user. Copy the Odd Numbers application from C:/Examples/Tutorial09/Debugger to your working directory. Run the application. Notice that, after you enter a value into the Upper limit: TextBox and click the View Button, an infinite loop occurs. Use the debugger to find and fix the error(s) in the application. Figure 9.23 displays the correct output for the application.

📙 Odd Numbe	ers	_ 🗆 🗵
Upper limit:	10	View
Odd number 1 3 5 7 9	8:	

Figure 9.23 Correct output for the Odd Numbers application.

```
' Exercise 9.16 Solution
 1
 2
    ' OddNumbers.vb
 3
 4
    Public Class FrmOddNumbers
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Forms Designer generated code
 8
 9
        ' display odd numbers from one to number input by user
10
       Private Sub btnView_Click(ByVal sender As System.Object, _
11
          ByVal e As System. EventArgs) Handles btnView. Click
12
13
          Dim intLimit As Integer = 0 ' upper limit set by user
14
          Dim intCounter As Integer = 1 ' counter begins at 1
15
16
          lstResults.Items.Clear() ' clear ListBox
17
           intLimit = Val(txtLimit.Text) ' retrieve upper limit
18
          lstResults.Items.Add("Odd numbers:") ' display header
19
20
          Do While intCounter < intLimit</pre>
21
22
              ' determine and display odd numbers
23
              If intCounter Mod 2 <> 0 Then
```

25 End If Incorrect code given to 26 students incremented - intCounter += 1 ' increment counter 27 intLimit instead of 28 Loop intCounter 29 30 End Sub ' btnView\_Click 31 32

24

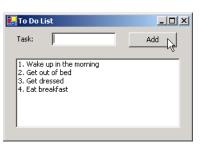
Programming Challenge

9.17 (To Do List Application) Use a ListBox as a to do list. Enter each item in a TextBox, and add it to the ListBox by clicking a Button. The item should be displayed in a numbered list as in Fig. 9.24. To do this, we introduce property Count, which returns the number of items in a ListBox's Items property. The following is a sample call to assign the number of items displayed in the lstSample ListBox to an Integer variable:

lstResults.Items.Add(intCounter)

intCount = lstSample.Items.Count

End Class ' FrmOddNumbers





1	' Exercise 9.17 Solution
2	' ToDoList.vb
3	
4	Public Class FrmToDoList
5	Inherits System.Windows.Forms.Form
6	
7	'Windows Form Designer generated code
8	
9	<pre>Private Sub btnAdd_Click(ByVal sender As System.Object, _</pre>
10	ByVal e As System.EventArgs) Handles btnAdd.Click
11	
12	Dim intItemNumber As Integer
13	
14	' set number of item
15	<pre>intItemNumber = lstOutput.Items.Count + 1</pre>
16	
17	' add input with number to ListBox
18	lstOutput.Items.Add(intItemNumber & ". " & _
19	txtInput.Text)
20	
21	' clear TextBox
22	<pre>txtInput.Text = ""</pre>
23	End Sub ' btnAdd_Click
24	
25	End Class ' FrmToDoList





# Class Average Application

Introducing the Do...Loop While and Do...Loop Until Repetition Statements Solutions

Instructor's Manual Exercise Solutions Tutorial 10			
MULTIPLE-CHOICE	<b>10.1</b> A(n) occurs when a loop-continuation condition in a DoLoop While never becomes False.		
QUESTIONS	a) infinite loop	b) counter-controlled loop	
	c) control statement	d) nested control statement	
	<b>10.2</b> Set property to True to enable a Button.		
	a) Disabled	b) Focus	
	c) Enabled	d) ButtonEnabled	
	<ul> <li>10.3 The statement executes at least once and continues executing until its loop-termination condition becomes True.</li> </ul>		
	a) Do WhileLoop	b) DoLoop Until	
	c) DoLoop While	d) Do UntilLoop	
	<b>10.4</b> The statement executes at lease continuation condition becomes False.	ast once and continues executing until its loop-	
	a) DoLoop Until	b) Do UntilLoop	
	c) Do WhileLoop	d) DoLoop While	
	<b>10.5</b> Method transfers the focus to a control.		
	a) GetFocus	b) Focus	
	c) Transfer	d) Activate	
	<b>10.6</b> A contains the sum of a series of values.		
	a) total	b) counter	
	c) condition	d) loop	
	10.7 Property of conta	ins the number of items in a ListBox.	
	a) Count,ListBox	b) ListCount, Items	
	c) ListCount, ListBox	d) Count, Items	
	<b>10.8</b> A(n) occurs when a loop exe necessary.	cutes for one more or one less iteration than is	
	a) infinite loop	b) counter-controlled loop	
	c) off-by-one error	d) nested control statement	
	<b>10.9</b> A DoLoop Until repetition statement's loop-termination condition is evaluated		
	a) only the first time the body executes	b) before the body executes	
	c) after the body executes	d) None of the above	
	<b>10.10</b> If its continuation condition is initially	False, a DoLoop While repetition statement	
	a) never executes	b) executes until the condition becomes True	
	c) executes until the condition becomes True	d) executes only once	
	<b>Answers:</b> 10.1) a. 10.2) c. 10.3) b. 10.4) d. 10.4	5) b. 10.6) a. 10.7) d. 10.8) c. 10.9) c. 10.10) d.	

**EXERCISES** 10.11 (Modified Class Average Application) Modify the Class Average application, as in Fig. 10.18, so that the Average Button is disabled until 10 grades have been entered.

Class Average		Class Average	
Grade list:	Enter grade:	Grade list:	Enter grade
		88 90 67	
	Add Grade	67 73 100	Add Grade
	Class average:	75 85	Class avera
		90 67 80	
	Average		Average

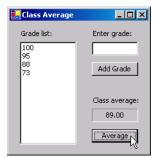


- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial10\Exercises\ModifiedClassAverage directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click ClassAverage.sln in the ModifiedClassAverage directory to open the application.
- c) *Initially disabling the Average Button.* Use the **Properties** window to modify the **Average** Button in the Form so that it is disabled when the application first executes by initially setting its Enabled property to False.
- d) *Enabling the Average Button after 10 grades have been entered.* Add code to the btnAdd\_Click event handler so that the **Average** Button becomes enabled when 10 grades have been entered.
- e) Disabling the Average Button after the calculation has been performed. Add code to the btnAverage\_Click event handler so that the Average Button is disabled once the calculation result has been displayed.
- f) Running the application. Select Debug > Start to run your application. Enter 10 grades and ensure that the Average Button is disabled until all 10 grades are entered. Verify that the Add Grade Button is disabled after 10 grades are entered. Once the Average Button is enabled, click it and verify that the average displayed is correct. The Average Button should then become disabled again, and the Add Grade Button should be enabled.
- g) *Closing the application.* Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
    ' Exercise 10.11 Solution
2
    ' ClassAverage.vb
3
4
    Public Class FrmClassAverage
5
       Inherits System.Windows.Forms.Form
 6
 7
       ' Windows Form Designer generated code
8
9
       ' handles Add Grade Button's Click event
10
      11
         ByVal e As System. EventArgs) Handles btnAdd. Click
12
13
         ' clear previous grades and calculation result
14
         If lblOutput.Text <> "" Then
15
            lblOutput.Text = ""
16
            lstGrades.Items.Clear()
17
         End If
18
```

```
19
           display grade in ListBox
20
          lstGrades.Items.Add(Val(txtInput.Text))
21
          txtInput.Clear() ' clear grade from TextBox
22
          txtInput.Focus() ' transfer focus to TextBox
23
24
          ' prohibit users from entering more than 10 grades
25
          If lstGrades.Items.Count >= 10 Then
            btnAdd.Enabled = False ' disable Add Grade Button
26
27
            btnAverage.Enabled = True ' enable Average Button
28
            btnAverage.Focus() ' transfer focus to Average Button
29
          End If
30
31
       End Sub ' btnAdd_Click
32
33
       ' handles Average Button's Click event
34
       Private Sub btnAverage_Click(ByVal sender As System.Object, _
35
          ByVal e As System. EventArgs) Handles btnAverage. Click
36
37
          ' initialization phase
38
          Dim intTotal As Integer = 0
39
          Dim intGradeCounter As Integer = 0
40
          Dim intGrade As Integer = 0
41
          Dim dblAverage As Double = 0
42
43
          ' sum grades in ListBox
44
          Do
45
             ' read grade from ListBox
46
47
            intGrade = lstGrades.Items.Item(intGradeCounter)
48
            intTotal += intGrade ' add grade to total
                                   ' increment counter
49
             intGradeCounter += 1
50
          Loop Until intGradeCounter >= 10
51
52
          53
          lblOutput.Text = String.Format("{0:F}", dblAverage)
54
          btnAverage.Enabled = False ' disable Average Button
55
56
          txtInput.Focus() ' reset focus to Enter grade: TextBox
57
       End Sub ' btnAverage_Click
58
59
    End Class ' FrmClassAverage
```

**10.12** (Class Average Application That Handles Any Number of Grades) Rewrite the Class Average application to handle any number of grades, as in Fig. 10.19. Note that, because the application does not know how many grades the user will enter, the Buttons must be enabled at all times.



**Figure 10.19** Modified **Class Average** application handling an unspecified number of grades.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial10\Exercises\UndeterminedClassAverage directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click ClassAverage.sln in the UndeterminedClassAverage directory to open the application.
- c) *Never disabling the Add Grade Button.* Remove code from the btnAdd\_Click event handler so that the Add Grade Button is not disabled after entering 10 grades.
- d) Summing the grades in the ListBox. Modify code in the btnAverage\_Click event handler so that intGradeCounter is incremented until it is equal to the number of grades entered. Use lstGrades.Items.Count to determine the number of items in the ListBox. The number returned by the Count property will be zero if there are no grades entered. Use an If...Then selection statement to avoid division by zero and display a message dialog to the user if there are no grades entered when the user clicks the Average Button.
- e) *Calculating the class average.* Modify the code in the btnAverage\_Click event handler so that dblAverage is computed by using intGradeCounter rather than the value 10.
- f) Running the application. Select Debug > Start to run your application. Enter 10 grades and click the Average Button. Verify that the average displayed is correct. Follow the same actions but this time for 15 grades, then for 5 grades. Each time, verify that the appropriate average is displayed.
- g) *Closing the application.* Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 10.12 Solution
     ' ClassAverage.vb
 2
 3
 4
    Public Class FrmClassAverage
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' handles Add Grade Button's Click event
10
       Private Sub btnAdd_Click(ByVal sender As System.Object, _
11
          ByVal e As System. EventArgs) Handles btnAdd. Click
12
13
           ' clear previous grades and calculation result
14
           If lblOutput.Text <> "" Then
              lblOutput.Text = ""
15
16
              lstGrades.Items.Clear()
17
          End If
18
19
           ' display grade in ListBox
20
          lstGrades.Items.Add(Val(txtInput.Text))
21
           txtInput.Clear() ' clear grade from TextBox
22
           txtInput.Focus() ' transfer focus to TextBox
23
       End Sub ' btnAdd_Click
24
25
        ' handles Average Button's Click event
26
       Private Sub btnAverage_Click(ByVal sender As System.Object, _
27
          ByVal e As System. EventArgs) Handles btnAverage. Click
28
29
           ' initialization phase
30
          Dim intTotal As Integer = 0
31
          Dim intGradeCounter As Integer = 0
32
          Dim intGrade As Integer = 0
33
          Dim dblAverage As Double = 0
34
```

```
35
            no grades entered
36
           If lstGrades.Items.Count = 0 Then
37
              MessageBox.Show("Please enter at least one grade", _
38
                 "Enter Grade", MessageBoxButtons.OK, __
39
                 MessageBoxIcon.Exclamation)
40
           Else
41
42
              ' sum grades in ListBox
43
              Do
44
45
                 ' read grade from ListBox
46
                 intGrade = lstGrades.Items.Item(intGradeCounter)
47
                 intTotal += intGrade
                                         ' add grade to total
48
                                        ' increment counter
                 intGradeCounter += 1
49
              Loop Until intGradeCounter >= lstGrades.Items.Count
50
51
              dblAverage = intTotal / intGradeCounter ' calculate average
52
              lblOutput.Text = String.Format("{0:F}", dblAverage)
53
              txtInput.Focus() ' reset focus to Enter grade: TextBox
54
           End If
55
        End Sub ' btnAverage_Click
56
57
     End Class ' FrmClassAverage
```

**10.13** (Arithmetic Calculator Application) Write an application that allows users to enter a series of numbers and manipulate them. The application should provide users with the option of adding or multiplying the numbers. Users should enter each number in a TextBox. After entering each number, users should click a Button and the number should be inserted in a ListBox. The GUI should behave as in Fig. 10.20.

Arithmetic Calculator		Arithmetic Calculate	or 💶 🗖
Operands list:	Enter operand:	Operands list:	Enter operand:
	11	11	12
	Enter		Enter
	Add		Add
	Multiply		Multiply
	Result:		Result:
Arithmetic Calculator		Arithmetic Calculate	
Arithmetic Calculator Operands list:	Enter operand:	Operands list:	or
Operands list:		Operands list:	
Operands list:		Operands list:	
Operands list:	Enter operand:	Operands list:	Enter operand:
Operands list:	Enter operand:	Operands list:	Enter operand:
Operands list:	Enter operand: Enter	Operands list:	Enter operand: Enter Add

Figure 10.20 Arithmetic Calculator application.

a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial10\Exercises\ArithmeticCalculator directory to your C:\SimplyVB directory.

- b) **Opening the application's template file.** Double click ArithmeticCalculator.sln in the ArithmeticCalculator directory to open the application.
- c) Add a ListBox to display the entered numbers. Add a ListBox. Place and size it as in Fig. 10.22.
- d) Creating an event handler for the Enter Button. Create the Click event handler for the Enter Button. If the result of a previous calculation is displayed, this event handler should clear the result and disable the addition and multiplication Buttons. It should then insert the current number in the Operands list: ListBox. When the ListBox contains at least two numbers, the event handler should then enable the addition and multiplication Buttons.
- e) *Summing the grades in the ListBox.* Define the Click event handler for the Add Button. This event handler should compute the sum of all of the values in the **Operands list:** ListBox and display the result in a Label lblResult.
- f) Define the Click event handler for the Multiply Button. This event handler should compute the product of all of the values in the Operands list: ListBox and display the result in the Label lblResult.
- g) Running the application. Select Debug > Start to run your application. Enter two values, then click the Add and Multiply Buttons. Verify that the results displayed are correct. Also, make sure that the Add and Multiply Buttons are not enabled until two values have been entered.
- h) Closing the application. Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
' Exercise 10.13 Solution
 1
 2
    ' ArithmeticCalculator.vb
 3
 4
    Public Class FrmArithmeticCalculator
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' handles Enter Button's Click event
10
       Private Sub btnEnter_Click(ByVal sender As System.Object, _
11
          ByVal e As System. EventArgs) Handles btnEnter. Click
12
13
           ' clear ListBox and lblResult if necessary
14
          If lblResult.Text <> "" Then
              lblResult.Text = ""
15
16
              lstNumbers.Items.Clear()
17
              btnAdd.Enabled = False ' disable operation Buttons
18
              btnMultiply.Enabled = False
19
           End If
20
21
          lstNumbers.Items.Add(txtInput.Text) ' add number to ListBox
22
23
           ' enable binary operation Buttons when
24
           ' user has entered two numbers
25
          If lstNumbers.Items.Count = 2 Then
26
              btnAdd.Enabled = True
27
              btnMultiply.Enabled = True
28
           End If
29
30
           txtInput.Clear() ' clear TextBox
31
           txtInput.Focus() ' set focus to TextBox
32
       End Sub ' btnEnter_Click
33
34
        ' handles addition Button's Click event
35
        Private Sub btnAdd_Click(ByVal sender As System.Object,
```

36	ByVal e As System.EventArgs) Handles btnAdd.Click
37	
38	' initialize total and counter
39	Dim dblTotal As Double = $0$
40	Dim intCounter As Integer = 0
41	
42	' sum numbers in ListBox
43	Do
44	dblTotal += Val(lstNumbers.Items.Item(intCounter))
45	intCounter += 1
46	<pre>Loop While intCounter &lt; lstNumbers.Items.Count</pre>
47	
<b>48</b>	lblResult.Text = dblTotal
49	End Sub ' btnAdd_Click
50	
51	' handles multiplication Button's Click event
52	<pre>Private Sub btnMultiply_Click(ByVal sender As System.Object, _</pre>
53	ByVal e As System.EventArgs) Handles btnMultiply.Click
54	
55	' initialize dblTotal and intCounter
56	Dim dblTotal As Double = $1$
57	Dim intCounter As Integer = 0
58	
59	' multiply numbers in ListBox
60	Do
61	dblTotal *= Val(lstNumbers.Items.Item(intCounter))
62	intCounter += 1
63	<pre>Loop While intCounter &lt; lstNumbers.Items.Count</pre>
64	
65	lblResult.Text = dblTotal
66	End Sub ' btnMultiply_Click
67	
68	End Class ' FrmArithmeticCalculator

```
What does this code do?
                               10.14 What is the result of the following code?
```

```
1
     Dim intY As Integer
 2
    Dim intX As Integer
 3
    Dim intMysteryValue As Integer
 4
 5
    intX = 1
 6
     intMysteryValue = 0
 7
 8
    Do
 9
        intY = intX \wedge 2
10
        lstDisplay.Items.Add(intY)
11
        intMysteryValue += 1
12
        intX += 1
13
     Loop While intX <= 10</pre>
14
15
```

```
lblResult.Text = intMysteryValue
```

Answer: The value displayed in lblResult is 11.

```
What's wrong with this code?
```

10.15 Find the error(s) in the following code. This code should add 10 to the value in intY and store it in intZ. It then should reduce the value of intY by one and repeat until intY is less than 10. The output Label lblResult should display the final value of intZ.

```
Dim intY As Integer = 10
 1
2
    Dim intZ As Integer = 2
3
4
    Do
5
       intZ = intY + 10
6
    Loop Until intY < 10
7
8
    intY -= 1
9
10
    lblResult.Text = intZ
```

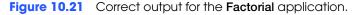
**Answer:** This code will loop infinitely because the statement that decrements intY (line 8) is not inside the repetition statement. Correct the code as follows:

```
1
   Dim intY As Integer = 10
2
   Dim intZ As Integer = 2
3
4
   Do
5
      intZ = intY + 10
6
      intY -= 1
7
   Loop Until intY < 10
8
9
   lblResult.Text = intZ
```

Using the Debugger

**10.16** (*Factorial Application*) The Factorial application calculates the factorial of an integer input by the user. The factorial of an integer is the product of the integers from one to that number. For example, the factorial of 3 is 6  $(1 \times 2 \times 3)$ . While testing the application you noticed that it does not execute correctly. Use the debugger to find and fix the logic error(s) in the application. Figure 10.21 displays the correct output for the Factorial application.

Eactorial	<u>- 0 ×</u>
Enter number:	3
Factorial:	6
	Calculate



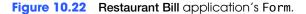
1	' Exercise 10.16 Solution
2	' Factorial.vb
3	
4	Public Class FrmFactorial
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' calculate factorial of user input number
10	<pre>Private Sub btnCalculate_Click(ByVal sender As System.Object, _</pre>
11	ByVal e As System.EventArgs) Handles btnCalculate.Click
12	
13	Dim intInput As Integer ' user input
14	Dim intFactorial As Integer = 1 ' holds factorial
15	

	16	intInput = Val(txtFactorial.Text) ' get user input
	17	
	18	' loop until intInput equals zero
	19	Do
	20	intFactorial *= intInput ' calculate factorial
	21	<pre>intInput -= 1 ' decrement counter</pre>
Replaced Until with While	22	Loop While intInput > 1 ' test condition
	23	
	24	lblResult.Text = intFactorial ' display factorial
	25	End Sub ' btnCalculate_Click
	26	
	27	End Class ' FrmFactorial

Programming Challenge

**10.17** (*Restaurant Bill Application*) Develop an application that calculates a restaurant bill. The user should be able to enter the item ordered, the quantity of the item ordered and the price per item. When the user clicks the Add Item Button, your application should display the number ordered, the item ordered and the price per unit in three ListBoxes as shown in Fig. 10.22. When the user clicks the Total Bill Button, the application should calculate the total cost. For each entry in the ListBox, multiply the cost of each item by the number of items ordered.

Restaurant	Bill	_ <b>_</b> ×
Quantity:		Add Item
Menu item:		Total Bill
Price:		
Quantity:	Menu item:	Price:
2	Lunch special Apple pie slice	6.99 1.25
	Total cost:	\$16.48



1	' Exercise 10.17 Solution
2	'RestaurantBill.vb
3	
4	Public Class FrmRestaurantBill
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' handles Add Item Button's Click event
10	<pre>Private Sub btnAddItem_Click(ByVal sender As System.Object, _</pre>
11	ByVal e As System.EventArgs) Handles btnAddItem.Click
12	
13	' display user input in ListBoxes
14	<pre>lstQuantity.Items.Add(Val(txtQuantity.Text))</pre>
15	<pre>lstItem.Items.Add(txtItem.Text)</pre>
16	<pre>lstPrice.Items.Add(Val(txtPrice.Text))</pre>
17	
18	' clear TextBoxes
19	<pre>txtItem.Clear()</pre>

20	txtQuantity.Clear()
21	txtPrice.Clear()
22	
23	<pre>txtQuantity.Focus() ' set the focus to Quantity: TextBox</pre>
24	End Sub ' btnAddItem_Click
25	
26	' handles Total Bill Button's Click event
27	<pre>Private Sub btnTotal_Click(ByVal sender As System.Object, _</pre>
28	ByVal e As System.EventArgs) Handles btnTotal.Click
29	
30	Dim intCounter As Integer = 0
31	Dim decCost As Decimal = $0$
32	
33	' calculate bill
34	Do
35	<pre>decCost += lstPrice.Items.Item(intCounter) * _</pre>
36	<pre>lstQuantity.Items.Item(intCounter)</pre>
37	intCounter += 1
38	<pre>Loop While intCounter &lt; lstPrice.Items.Count()</pre>
39	
40	' display result
41	lblTotalCost.Text = String.Format("{0:C}", decCost)
42	End Sub ' btnTotal_Click
43	
44	End Class ' FrmRestaurantBill





### Interest Calculator Application

Introducing the For...Next Repetition Statement Solutions

)	Instructor's Manual Exercises Solutions Tutorial 11		
-	MULTIPLE-CHOICE	11.1 "Hello" has data type	
	QUESTIONS	a) String	b) StringLiteral
	QUEUNONU	c) Character	d) StringText
		<b>11.2</b> A provides the ability to enter control.	er or display multiple lines of text in the same
		a) TextBox	b) NumericUpDown
		c) MultilineTextBox	d) multiline NumericUpDown
		11.3 The NumericUpDown control allows you to	o specify
		a) a maximum value the user can select	b) a minimum value the user can select
		<ul><li>c) an increment for the values presented to the user</li></ul>	<ul><li>d) All of the above.</li></ul>
		11.4 is optional in a ForNext head	der when the control variable's increment is 1.
		a) Keyword To	b) The initial value of the control variable
		c) Keyword Step	d) The final value of the control variable
		11.5 Setting TextBox property ScrollBars to	creates a vertical scrollbar.
		a) True	b) Vertical
		c) Up	d) Both
)		<b>11.6</b> is used to determine whether	a ForNext loop continues to iterate.
		a) The initial value of the control variable	-
		c) Keyword Step	d) The control variable
		<b>11.7</b> In a ForNext loop, the control variable	is incremented (or decremented) .
		a) after the body of the loop executes	b) when keyword To is reached
		c) while the loop-continuation condition is False	d) while the body of the loop executes
		<b>11.8</b> Setting a NumericUpDown control's cannot enter invalid values in the control.	property to True ensures that the user
		a) Increment	b) ScrollBars
		c) ReadOnly	d) InValid
		11.9 The and proper NumericUpDown control.	ties limit the values users can select in the
		a) Maximum,Minimum	b) Top, Bottom
		c) High, Low	d) Max,Min
		<b>11.10</b> ForNext header can be u numbers between 1 and 10.	sed to vary the control variable over the odd
		a) For intI = 1 To 10 Step 1	b) For intI = 1 To 10 Step 2
		c) For intI = $1 \text{ To } 10 \text{ Step } -1$	d) For intI = 1 To 10 Step -2
		<b>Answers:</b> 11.1) a. 11.2) a. 11.3) d. 11.4) c. 11.8	5) b or d. 11.6) d. 11.7) a. 11.8) c. 11.9) a.
		11.10) b.	
)_			
-			

**EXERCISES** 11.11 (*Present Value Calculator Application*) A bank wants to show its customers how much they would need to invest to achieve a specified financial goal (future value) in 5, 10,

15, 20, 25 or 30 years. Users must provide their financial goal (the amount of money desired after the specified number of years has elapsed), an interest rate and the length of the investment in years. Create an application that calculates and displays the principal (initial amount to invest) needed to achieve the user's financial goal. Your application should allow the user to invest money for 5, 10, 15, 20, 25 or 30 years. For example, if a customer wants to reach the financial goal of \$15,000 over a period of 5 years when the interest rate is 6.6%, the customer would need to invest \$10,896.96 as shown in Fig. 11.16.

Present Value Calculator	Present Value Calculator	_ <b>_</b> X
Future value:Calculate	Future value: 15000	Calculate
Interest rate:	Interest rate: 6.6	
Years: 5	Years: 20	
Annual amount needed:	Annual amount needed: Year Amount on Deposit Needed 5 \$10,896.96 10 \$7,916.24 15 \$5,750.67 20 \$4,177.80	×



- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial11\Exercises\PresentValue directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click PresentValue.sln in the PresentValue directory to open the application.
- c) Adding the NumericUpDown control. Place and size the NumericUpDown so that it follows the GUI Design Guidelines. Set the NumericUpDown control's Name property to updYear. Set the NumericUpDown control to allow only multiples of five for the number of years. Also, allow the user to select only a duration that is in the specified range of values.
- d) Adding a multiline TextBox. Add a TextBox to the Form below the NumericUpDown control. Change the size to 272, 88, and position the TextBox on the Form so that it follows the GUI Design Guidelines. Then set that TextBox to display multiple lines and a vertical scrollbar. Also ensure that the user cannot modify the text in the TextBox.
- e) Adding a Click event handler and adding code. Add a Click event handler for the Calculate Button. Once in code view, add code to the application such that, when the Calculate Button is clicked, the multiline TextBox displays the necessary principal for each five-year interval. Use the following version of the present-value calculation formula:

 $p = a / (1 + r)^{n}$ 

where

*p* is the amount needed to achieve the future value*r* is the annual interest rate (for example, .05 is equivalent to 5%)*n* is the number of years*a* is the future-value amount.

- f) Running the application. Select Debug > Start to run your application. Enter amounts for the future value, interest rate and number of years. Click the Calculate Button and verify that the year intervals and the amount on deposit needed for each is correct. Test the application again, this time entering 30 for the number of years. Verify that the vertical scrollbar appears to display all of the output.
- g) *Closing the application.* Close your running application by clicking its close box.
- h) *Closing the IDE*. Close Visual Studio .NET by clicking its close box.

' Exercise 11.11 Solution 1 2 ' PresentValue.vb 3 4 Public Class FrmPresentValue 5 Inherits System.Windows.Forms.Form 6 7 ' Windows Form Designer generated code 8 9 ' handles Calculate Button's Click event 10 Private Sub btnCalculate\_Click(ByVal sender As System.Object, \_ 11 ByVal e As System. EventArgs) Handles btnCalculate. Click 12 13 ' clear txtResult from previous results 14 txtResult.Clear() 15 16 ' declare variables Dim decFutureValue As Decimal 17 18 Dim dblRate As Double 19 Dim intYears As Integer 20 Dim decPresentValue As Decimal 21 22 ' retrieve values from user input 23 decFutureValue = Val(txtFutureValue.Text) 24 dblRate = Val(txtRate.Text) 25 intYears = updYear.Text 26 ' set initial output line 27 28 txtResult.Text = "Year" & ControlChars.Tab & \_ 29 "Amount on Deposit Needed" & ControlChars.CrLf 30 31 ' calculate principal and display result 32 Dim intCounter As Integer 33 34 For intCounter = 5 To intYears Step 5 35 decPresentValue = \_ 36 decFutureValue / ((1 + (dblRate / 100)) ^ intCounter) 37 38 ' append result to txtResult's text property 39 txtResult.Text &= intCounter & ControlChars.Tab & \_ 40 String.Format("{0:C}", decPresentValue) & \_ ControlChars.CrLf 41 42 43 Next 44 45 End Sub ' btnCalculate\_Click 46 47 End Class ' FrmPresentValue

**11.12** (*Compound Interest: Comparing Rates Application*) Write an application that calculates the amount of money in an account after 10 years for interest rate amounts of 5%–10%. For this application, users must provide the initial principal.

📙 Compari	ng Rates
Principal:	1000 Calculate
Result:	
Rate(%)	Amount after 10 years
5	\$1,628.89
6	\$1,790.85
7	\$1,967.15
8	\$2,158.92
9	\$2,367.36
110	\$2,593.74

Figure 11.17 Comparing Rates GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial11\Exercises\ComparingRates directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click ComparingRates.sln in the ComparingRates directory to open the application.
- c) Adding a multiline TextBox. Add a TextBox to the Form below the Result: Label control. Change the size to 256, 104, and position the TextBox on the Form so that it follows the GUI Design Guidelines (Fig. 11.17). Then set that TextBox to display multiple lines. Also ensure that the user cannot modify the text in the TextBox.
- d) Adding a Click event handler and adding code. Add a Click event handler for the Calculate Button. Once in code view, add code to the application such that, when the Calculate Button is clicked, the multiline TextBox displays the amount in the account after 10 years for interest rates of 5, 6, 7, 8, 9 and 10 percent. Use the following version of the interest-calculation formula:

 $a = p (1 + r)^{n}$ 

where

*p* is the original amount invested (the principal)

- *r* is the annual interest rate (for example, .05 is equivalent to 5%)
- *n* is the number of years

*a* is the investment's value at the end of the *n*th year.

- e) **Running the application.** Select **Debug > Start** to run your application. Enter the principal amount for an account and click the **Calculate** Button. Verify that the correct amounts after 10 years are then displayed, based on interest rate amounts of 5%-10%.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 11.12 Solution
 2
    ' ComparingRates.vb
 3
 4
    Public Class FrmComparingRates
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' invoke when Calculate Button is pressed
10
       Private Sub btnCalculate_Click(ByVal sender As System.Object, _
11
          ByVal e As System. EventArgs) Handles btnCalculate. Click
12
13
           ' declare local variables
14
          Dim strOutput As String
15
          Dim intRateCounter As Integer
16
          Dim decPrincipal As Decimal = Val(txtPrincipal.Text)
          Dim decAmount As Decimal = 0
17
```

```
18
19
            ' set output header
           strOutput = "Rate(%)" & ControlChars.Tab & ControlChars.Tab _
20
21
              & "Amount after 10 years" & ControlChars.CrLf
22
23
            ' calculate amount for each rate and append to string
24
           For intRateCounter = 5 \text{ To } 10
25
              decAmount =
26
                  decPrincipal * ((1 + intRateCounter / 100) ^ 10)
27
28
              strOutput &= (intRateCounter & ControlChars.Tab & _
29
                  ControlChars.Tab & String.Format("{0:C}", decAmount) _
30
                 & ControlChars.CrLf)
31
           Next
32
33
           txtResult.Text = strOutput ' display result
34
        End Sub ' btnCalculate_Click
35
36
     End Class ' FrmComparingRates
```

**11.13** (Validating Input to the Interest Calculator Application) Enhance the Interest Calculator application with error checking. Test for whether the user has entered valid values for the principal and interest rate. If the user enters an invalid value, display a message in the multiline TextBox. Figure 11.18 demonstrates the application handling an invalid input.

📙 Interest Calo	ulator		_ 🗆 X
Principal:		100	Calculate
Interest rate:		-5	
Years:		1 🕂	
Yearly account	balance:		
The informatio range of value	n input was not s.	within th	e correct 🔺
			7

Figure 11.18 Interest Calculator application with error checking.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial11\Exercises\InterestCalculatorEnhancement directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click InterestCalculator.sln in the InterestCalculatorEnhancement directory to open the application.
- c) Adding a Click event handler and adding code. Add a Click event handler for the Calculate Button. Once in code view, modify the code to validate the input. The principal should be a positive amount greater than 0. Also, the interest rate should be greater than 0, but less than 100.
- d) *Displaying the error message.* Display the text "The information was not within the correct range of values." in txtResult if the values are not valid.
- e) *Running the application.* Select **Debug > Start** to run your application. Enter invalid data for the principal and interest rate. The invalid data can include negative numbers and letters. Verify that entering invalid data and clicking the **Calculate** Button results in the error message displayed in Fig. 11.18.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

Exercise 11.13 Solution 1 2 ' InterestCalculator.vb 3 4 Public Class FrmInterestCalculator 5 Inherits System.Windows.Forms.Form 6 7 ' Windows Form Designer generated code 8 9 ' handles Calculate Button's Click event 10 Private Sub btnCalculate\_Click(ByVal sender As System.Object, \_ 11 ByVal e As System. EventArgs) Handles btnCalculate. Click 12 13 ' declare variables to store user input 14 Dim decPrincipal As Decimal 15 Dim dblRate As Double 16 Dim intYear As Integer 17 Dim decAmount As Decimal ' store each calculation 18 Dim strOutput As String ' store output 19 20 ' retrieve user input decPrincipal = Val(txtPrincipal.Text) 21 22 dblRate = Val(txtRate.Text) 23 intYear = updYear.Text 24 25 If decPrincipal > 0 AndAlso dblRate > 0 26 AndAlso dblRate < 100 Then 27 28 ' set output header 29 strOutput = "Year" & ControlChars.Tab \_ 30 & "Amount on Deposit" & ControlChars.CrLf 31 32 ' calculate amount after each year and append to string 33 For intYear = 1 To intYear 34 35 decAmount = decPrincipal \* \_ 36  $(1 + db]Rate / 100) \wedge intYear$ 37 strOutput &= (intYear & ControlChars.Tab & \_ 38 String.Format("{0:C}", decAmount) & \_ 39 ControlChars.CrLf) 40 41 Next 42 43 Else 44 strOutput = "The information input was not within the" \_ 45 & " correct range of values." 46 End If 47 48 txtResult.Text = strOutput ' display result 49 End Sub ' btnCalculate\_Click 50 51 End Class ' FrmInterestCalculator

What does this code do?

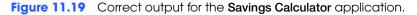
**11.14** What is the value of intResult after the following code executes? Assume that intPower, intI, intResult and intNumber are all declared as Integers.

1 intPower = 5
2 intNumber = 10
3 intResult = intNumber

$\bigcirc$		<pre>4 5 For intI = 1 To (intPower - 1) 6 intResult *= intNumber 7 Next</pre>		
		<b>Answer:</b> This code segment raises intNumber to the intPower power. In this case, intRe-sult gets 10 <sup>5</sup> (100000).		
What's wror	ng with this code? 🕨	<ul><li>11.15 Assume that the variable intCounter is declared as an Integer for both a and b. Identify and correct the error(s) in each of the following:</li><li>a) This statement should display in a ListBox all numbers from 100 to 1 in decreasing order.</li></ul>		
		<pre>1 For intCounter = 100 To 1 2 lstDisplay.Items.Add(intCounter) 3 Next</pre>		
		<b>Answer:</b> The code needs Step -1 at the end of the ForNext header.		
		<pre>1 For intCounter = 100 To 1 Step -1 2 lstDisplay.Items.Add(intCounter) 3 Next</pre>		
		b) The following code should display in a ListBox the odd Integers from 19 to 1 in decreasing order.		
$\supset$		<pre>1 For intCounter = 19 To 1 By -1 2 lstDisplay.Add(intCounter) 3 Next</pre>		
		<b>Answer:</b> By -1 should be Step -2 and lstDisplay.Add(intCounter) should be lstDisplay.Items.Add(intCounter).		
		<pre>1 For intCounter = 19 To 1 Step -2 2 lstDisplay.Items.Add(intCounter) 3 Next</pre>		
Usi	ing the Debugger 🕨	<b>11.16</b> (Savings Calculator Application) The Savings Calculator application calculates the		

amount that the user will have on deposit after one year. The application calculates the amount that the user will have on deposit after one year. The application gets the initial amount on deposit from the user, and assumes that the user will add \$100 dollars to the account every month for the entire year. No interest is added to the account. While testing the application, you noticed that the amount calculated by the application was incorrect. Use the debugger to locate and correct any logic error(s). Figure 11.19 displays the correct output for this application.

Savings Calculator	
Starting amount:	100
Amount after one year:	1300
	Calculate



	1	' Exercise 11.16 Solution
	2	' Savings.vb
	3	
	4	Public Class FrmSavings
	5	Inherits System.Windows.Forms.Form
	6	
	7 8	' Windows Form Designer generated code
	0 9	L colculate emputer in account often and year
		' calculate amount in account after one year
	10	<pre>Private Sub btnCalculate_Click(ByVal sender As System.Object, _</pre>
	11	ByVal e As System.EventArgs) Handles btnCalculate.Click
	12	
	13	Dim intTotal As Integer = 0 ' amount on deposit
	14	Dim intCounter As Integer = 1 ' counter starts at 1
	15	
	16	intTotal = Val(txtStartAmount.Text) ' get amount on deposit
	17	
Incorrect code given to	18	' add \$100 a month for one year
students looped 13 times	19	——— For intCounter = 1 To 12
(0-12), instead of 12 (1-12)	20	<pre>intTotal += 100 ' add money</pre>
	21	Next
	22	
	23	lblTotal.Text = intTotal ' display total after 12 months
	24	End Sub ' btnCalculate Click
	25	
	26	End Class ' FrmSavings

Programming Challenge

**11.17** (*Pay Raise Calculator Application*) Develop an application that computes the amount of money an employee makes each year over a user-specified number of years. The employee receives an hourly wage and a pay raise once every year. The user specifies the hourly wage and the amount of the raise (in percentages per year) in the application.

📙 Pay Ra	aise Calculator			×
Starting	wage per hour:	10	Calculate	]
Amount	of raise (in %):	3		
Years:		3 +		
Yearly a	mount earned:			
Year 1 2 3	Amount \$20,800.00 \$21,424.00 \$22,066.72			
	\$22,000.72			

Figure 11.20 Pay Raise application's GUI.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial11\Exercises\PayRaise directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click PayRaise.sln in the PayRaise directory to open the application.
- c) Adding controls to the Form. Add two NumericUpDown controls to the Form. The first NumericUpDown control should be provided to allow the user to specify the pay raise percentage. The user should only be able to specify percentages in the range of 3–8 percent. Create the second NumericUpDown control for users to select the number of years in the range 1–50. Then add a multiline TextBox control to the application. Ensure that the user cannot modify the text in the NumericUpDown and TextBox con-

trols. Resize and move the controls you created so that they follow the GUI Design Guidelines as in Fig. 11.20.

- d) Adding a Click event handler and adding code. Add a Click event handler for the Calculate Button. Once in code view, add code to use the For...Next statement to compute the yearly salary amounts, based on the yearly pay raise.
- e) *Running the application.* Select **Debug > Start** to run your application. Enter a starting wage per hour, the size of the yearly raise and the number of years worked. Click the **Calculate** Button and verify that the correct amount after each year is displayed in the **Yearly amount earned:** TextBox.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 11.17 Solution
 2
    ' PayRaise.vb
 3
 4
    Public Class FrmPayRaise
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Window Form Designer generated code
 8
9
        ' invoke when btnCalculate Button is clicked
10
        Private Sub btnCalculate_Click(ByVal sender As _
11
          System.Object, ByVal e As System.EventArgs) _
12
          Handles btnCalculate.Click
13
14
          Dim intYears As Integer = updYears.Text
15
          Dim intCounter As Integer
16
          Dim decWage As Decimal = Val(txtWage.Text)
17
          Dim intCurrentYear As Integer = 0
18
          Dim decTotal As Decimal = 0
19
20
           ' create headers to display in txtResult
21
           txtResult.Text = "Year" & ControlChars.Tab & _
22
              "Amount" & ControlChars.CrLf
23
24
           ' calculate first year's total
25
          decTotal += (decWage * 40 * 52)
26
27
           ' display wages per year with raise
28
          For intCounter = 1 To intYears Step 1
29
30
              ' determine if raise should be applied
31
             If intCounter <> 1 Then
32
33
                 ' calculate total with raise amount
34
                 decTotal *= 1 + ((updRaise.Text)/100)
35
             End If
36
37
              intCurrentYear += 1 ' increment intYear count
38
39
              ' append amounts to string displayed in
40
              ' txtResult TextBox
41
              txtResult.Text &= (intCurrentYear & _
42
                 ControlChars.Tab & _
43
                 String.Format("{0:C}", decTotal) & _
44
                 ControlChars.CrLf)
45
          Next
46
```

47 End Sub ' btnCalculate\_Click
48
49 End Class ' FrmPayRaise

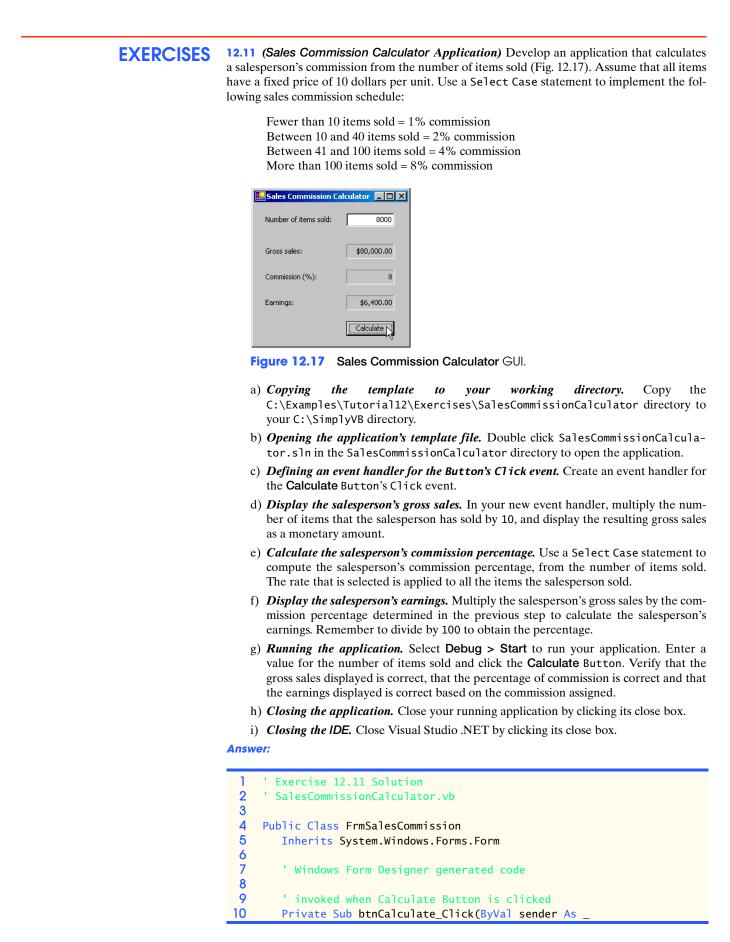




## Security Panel Application

Introducing the Select Case Multiple-Selection Statement Solutions

$\bigcirc$	Instructor's Manual Exercise Solutions Tutorial 12		
-	MULTIPLE-CHOICE	-	ds signify the end of a Select Case statement.
	QUESTIONS	a) End Case c) End Select Case	b) End Select d) Case End
		12.2 The expression	returns the current system time and date.
		a) Date.DateTime c) Date.Now	<ul><li>b) Date.SystemDateTime</li><li>d) Date.SystemTimeDate</li></ul>
		12.3 You can hide infor	mation entered into a TextBox by setting that TextBox's haracter; that character will be displayed for every character
		a) PrivateChar	b) Mask
		c) MaskingChar	d) PasswordChar
$\bigcirc$		<ul><li>b) Having a Case state the value to the righ</li><li>c) Preceding a Case sta statement</li></ul>	se statements in the same Select Case statement ment in which the value to the left of a To keyword is larger than
$\bigcirc$		-	used to specify a range in a Case statement.
		a) Also c) To	b) Between d) From
		a) A comma	ultiple values tested in a Case statement.
		c) Keyword Also	<ul><li>b) An underscore</li><li>d) A semicolon</li></ul>
			,
		a) Append	inserts a value in a ListBox. b) Items.Insert
		c) Insert	d) Items.Add
			of the To keyword in a Case statement is larger than the value on
		a) a syntax error occur	e statement never executes
		<b>12.9</b> The expression follow	ing the keywords Select Case is called a(n)
		a) guard condition	b) controlling expression
		c) selection expression	d) case expression
		<b>12.10</b> To prevent a user fr False.	om modifying text in a TextBox, set its property to
		a) Enabled	b) Text
$\bigcirc$		c) TextChange	d) Editable
$\sim$		Answers: 12.1) b. 12.2) c. 1	2.3) d. 12.4) c. 12.5) c. 12.6) a. 12.7) d. 12.8) c. 12.9) b. 12.10) a.



```
TT
           System.Object, ByVal e As System.EventArgs) _
12
           Handles btnCalculate.Click
13
14
           lblGrossSalesResult.Text = String.Format("{0:C}", _
15
              Val(txtItemsSold.Text) * 10)
16
17
           Dim intItemsSold As Integer = Val(txtItemsSold.Text)
18
19
           ' Select Case used to determine commission percentage
20
           Select Case intItemsSold
21
22
              Case Is < 10
23
                 lblCommissionPercentageResult.Text = 1
24
25
              Case 10 To 40
26
                 lblCommissionPercentageResult.Text = 2
27
28
              Case 41 To 100
29
                 lblCommissionPercentageResult.Text = 4
30
31
              Case Is > 100
32
                 lblCommissionPercentageResult.Text = 8
33
34
           End Select
35
36
           ' calculate the earnings
37
           lblEarningsResult.Text = String.Format("{0:C}", _
38
              lblGrossSalesResult.Text * _
39
              (lblCommissionPercentageResult.Text / 100))
40
41
        End Sub ' btnCalculate_Click
42
43
     End Class ' FrmSalesCommission
```

**12.12** (Cash Register Application) Use the numeric keypad from the Security Panel application to build a Cash Register application (Fig. 12.18). In addition to numbers, the cash register should include a decimal point Button. Apart from this numeric operation, there should be Enter, Delete, Clear and Total Buttons. Sales tax should be calculated on the amount purchased. Use a Select Case statement to compute sales tax. Add the tax amount to the subtotal to calculate the total. Display the tax and total for the user. Use the following sales-tax percentages, which are based on the amount of money spent:

Amounts under 100 = 5% (.05) sales tax Amounts between 100 and 500 = 7.5% (.075) sales tax Amounts above 500 = 10% (.10) sales tax

🖶 Cash Registe	r _ 🗆 🗙	📙 Cash Register	
\$12.34		\$	
1 2	3 Enter	1 2 3	3 Enter
4 5	6 Total	4 5 (	5 Total
7 8	9 Delete	7 8 9	9 Delete
0	. Clear	0	. Clear
Subtotal:	\$0.00	Subtotal:	\$12.34
Tax:	\$0.00	Tax:	\$0.62
Total:	\$0.00	Total:	\$12.96



- a) Copying the template to your working directory. Copy the C:\Examples\Tutoriall2\Exercises\CashRegister directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click CashRegister.sln in the CashRegister directory to open the application.
- c) Define event handlers for the numeric Buttons and decimal point in the keypad. Create event handlers for each of these Button's Click events. Have each event handler concatenate the proper value to the TextBox at the top of the Form.
- d) **Define an event handler for the Enter Button's Click event.** Create an event handler for this Button's Click event. Have this event handler add the current amount to the subtotal and display the new subtotal.
- e) **Define an event handler for the Total Button's Click event.** Create an event handler for this Button's Click event. Have this event handler use the subtotal to compute the tax amount.
- f) **Define an event handler for the Clear Button's Click event.** Create an event handler for this Button's Click event. Have this event handler clear the user input and display the value \$0.00 for the subtotal, sales tax and total.
- g) **Define an event handler for the Delete Button's Click event.** Create an event handler for this Button's Click event. Have this event handler clear only the data in the TextBox.
- h) Running the application. Select Debug > Start to run your application. Use the keypad to enter various dollar amounts, clicking the Enter Button after each. After several amounts have been entered, click the Total Button and verify that the appropriate sales tax and total are displayed. Enter several values again and click the Delete Button to clear the current input. Click the Clear Button to clear all the output values.
- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

2 'CashRegister.vb	
3	
4 Public Class FrmCashRegister	
5 Inherits System.Windows.Forms.Form	
6	
7 'Windows Form Designer generated code	
8	
9 ' invoked when btnOne is clicked	
<pre>10 Private Sub btnOne_Click(ByVal sender As _</pre>	
<pre>System.Object, ByVal e As System.EventArgs) _</pre>	

12	Handles btnOne.Click
13	hundres benone errek
14	<pre>txtCurrentPrice.Text &amp;= "1"</pre>
15	End Sub ' btnOne_Click
16	
17	' invoked when btnTwo is clicked
18	<pre>Private Sub btnTwo_Click(ByVal sender As _</pre>
19	System.Object, ByVal e As System.EventArgs) _
20	Handles btnTwo.Click
21	Handles belliwo.errek
22	txtCurrentPrice.Text &= "2"
23	End Sub ' btnTwo_Click
24	
25	' invoked when btnThree is clicked
26	<pre>Private Sub btnThree_Click(ByVal sender As _</pre>
27	System.Object, ByVal e As System.EventArgs) _
28	Handles btnThree.Click
	Hanules Duningee.Click
29	
30	<pre>txtCurrentPrice.Text &amp;= "3"</pre>
31	End Sub ' btnThree_Click
32	
33	' invoked when btnFour is clicked
34	Private Sub btnFour_Click(ByVal sender As _
35	System.Object, ByVal e As System.EventArgs)
36	Handles btnFour.Click
37	
38	<pre>txtCurrentPrice.Text &amp;= "4"</pre>
39	End Sub ' btnFour_Click
40	
41	' invoked when btnFive is clicked
42	Private Sub btnFive_Click(ByVal sender As _
43	
	System.Object, ByVal e As System.EventArgs) _
44	Handles btnFive.Click
45	
46	<pre>txtCurrentPrice.Text &amp;= "5"</pre>
47	End Sub ' btnFive_Click
48	
49	' invoked when btnSix is clicked
50	Private Sub btnSix_Click(ByVal sender As _
51	System.Object, ByVal e As System.EventArgs)
52	Handles btnSix.Click
53	
54	<pre>txtCurrentPrice.Text &amp;= "6"</pre>
55	End Sub ' btnSix_Click
56	
57	' invoked when btnSeven is clicked
58	Private Sub btnSeven_Click(ByVal sender As _
59	System.Object, ByVal e As System.EventArgs)
60	Handles btnSeven.Click
61	
62	<pre>txtCurrentPrice.Text &amp;= "7"</pre>
63	End Sub ' btnSeven_Click
64	
65	' invoked when btnEight is clicked
66	<pre>Private Sub btnEight_Click(ByVal sender As _</pre>
67	System.Object, ByVal e As System.EventArgs) _
68	Handles btnEight.Click
	Hanules Duiligit. CITCK
69	
70	txtCurrentPrice.Text &= "8"
71	End Sub ' btnEight_Click
72	

73

74	<pre>Private Sub btnNine_Click(ByVal sender As _</pre>
75	System.Object, ByVal e As System.EventArgs) _
76	
	Handles btnNine.Click
77	
78	<pre>txtCurrentPrice.Text &amp;= "9"</pre>
79	End Sub ' btnNine_Click
80	
81	' invoked when btnZero is clicked
82	Private Sub btnZero_Click(ByVal sender As _
83	
	System.Object, ByVal e As System.EventArgs) _
84	Handles btnZero.Click
85	
86	txtCurrentPrice.Text &= "0"
87	End Sub ' btnZero_Click
88	
89	' invoked when btnPoint is clicked
90	Private Sub btnPoint_Click(ByVal sender As _
91	System.Object, ByVal e As System.EventArgs) _
92	Handles btnPoint.Click
93	
94	txtCurrentPrice.Text &= "."
95	End Sub ' btnPoint_Click
96	
97	' invoked when btnEnter is clicked
98	
	Private Sub btnEnter_Click(ByVal sender As _
99	System.Object, ByVal e As System.EventArgs) _
100	Handles btnEnter.Click
101	
102	' variable to store new value
103	Dim decAmount As Decimal
104	
105	' store value in txtCurrentPrice to decAmount
106	
	<pre>decAmount = Val(txtCurrentPrice.Text)</pre>
107	
108	' add input amount to dblTotal and clear TextBox
109	lblSubTotalValue.Text = String.Format("{0:C}", _
110	lblSubTotalValue.Text + decAmount)
111	
112	<pre>txtCurrentPrice.Clear() ' clear the TextBox</pre>
113	End Sub ' btnEnter_Click
114	
	1. An other distribution for an experimental distribution.
115	' invoked when btnTotal is clicked
116	<pre>Private Sub btnTotal_Click(ByVal sender As _</pre>
117	System.Object, ByVal e As System.EventArgs) _
118	Handles btnTotal.Click
119	
120	Dim dblTaxRate As Double
121	
122	' determines tax rate based on subtotal
123	Select Case lblSubTotalValue.Text
124	
125	Case Is $< 100$
126	dblTaxRate = 0.05
127	
128	Case 100 To 500
129	db]TaxRate = 0.075
130	
	C T
131	Case Is $> 500$

invoked when btnNine is clicked

dblTaxRate = 0.1

132

133

134	End Select
135	
136	' display subtotal, tax and total
137	<pre>lblSubTotalValue.Text = String.Format("{0:C}", _</pre>
138	lblSubTotalValue.Text)
139	
140	lblTaxValue.Text = String.Format("{0:C}", _
141	lblSubTotalValue.Text * dblTaxRate)
142	
143	lblTotalValue.Text = <pre>String.Format("{0:C}", _</pre>
144	(lblSubTotalValue.Text + _
145	lblSubTotalValue.Text * dblTaxRate))
146	End Sub ' btnTotal_Click
147	
148	' invoked when btnClear is clicked
149	Private Sub btnClear_Click(ByVal sender As _
150	System.Object, ByVal e As System.EventArgs) _
151	Handles btnClear.Click
152	
153	' clear txtCurrentPrice and set remaining Labels to \$0.00
154	txtCurrentPrice.Clear()
155	lblSubTotalValue.Text = "\$0.00" lblTaxValue.Text = "\$0.00"
156 157	b lb l Total Value. Text = $$0.00$
157	
159	End Sub ' btnClear_Click
160	' invoked when btnDelete is clicked
161	Private Sub btnDelete Click(ByVal sender As
162	System.Object, ByVal e As System.EventArgs) _
163	Handles btnDelete.Click
164	
165	<pre>txtCurrentPrice.Clear() ' clear the TextBox</pre>
166	End Sub ' btnDelete Click
167	
	End Class ' FrmCashRegister

**12.13** (*Income Tax Calculator Application*) Create an application that computes the amount of income tax that a person must pay, depending upon that person's salary. Your application should perform as shown in Fig. 12.19. Use the following income ranges and corresponding tax rates:

Under \$20,000 = 2% income tax \$20,000 - \$50,000 = 5% income tax \$50,001 - \$75,000 = 10% income tax \$75,001 - \$100,000 = 15% income tax Over \$100,000 = 20% income tax

Income Tax Calculator	- D ×	🛃 Income Tax C	alculator	- D ×
Yearly salary:	50500	Yearly salary:		
Income Tax:		Income Tax:	\$	5,050.00
Calcu	late		Calcu	late

Figure 12.19 Income Tax Calculator GUI.

a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial12\Exercises\IncomeTaxCalculator directory to your C:\SimplyVB directory.

- b) *Opening the application's template file.* Double click IncomeTaxCalculator.sln in the IncomeTaxCalculator directory to open the application.
- c) **Define an event handler for the Calculate Button's Click event.** Use the designer to create an event handler for this Button's Click event. Have this event handler use a Select Case statement to determine the user's income-tax percentage. For simplicity, this value should then be multiplied by the user's salary and displayed in the output Label.
- d) **Running the application.** Select **Debug > Start** to run your application. Enter a yearly salary and click the **Calculate** Button. Verify that the appropriate income tax is displayed, based on the ranges listed in the exercise description.
- e) *Closing the application.* Close your running application by clicking its close box.
- f) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 12.13 Solution
2	' IncomeTax.vb
3	
4	Public Class FrmIncomeTax
5 6	Inherits System.Windows.Forms.Form
7 8	' Windows Form Designer generated code
9	' invoked when Calculate Button is clicked
10	Private Sub btnCalculate_Click(ByVal sender As _
11	System.Object, ByVal e As System.EventArgs) _
12	Handles btnCalculate.Click
13	
14	Dim dblPercent As Double
15	Dim intSalary As Integer = Val(txtSalary.Text)
16	
17	' determine income percentage
18	Select Case intSalary
9	·
20	<b>Case Is &lt; 20000</b>
21	dblPercent = $0.02$
22 23	
23	Case 20000 To 50000
24	dblPercent = $0.05$
25	
26	Case 50001 To 75000
27	dblPercent = 0.1
28	
29	Case 75001 To 100000
30	dblPercent = 0.15
31	
32	<b>Case Is &gt;</b> 100000
33	dblPercent = 0.2
34	
35	End Select
36	
37	' display result in currency format
38	lblResult.Text = <pre>String.Format("{0:C}", _</pre>
39	intSalary * dblPercent)
40	
41	<pre>txtSalary.Clear() ' clear the TextBox</pre>
42	End Sub ' btnCalculate_Click
43	
44	End Class ' FrmIncomeTax

What does this code do?

**12.14** What is output by the following code? Assume that btnDonation is a Button, txt-Donation is a TextBox and lblMessage is an output Label.

```
1
    Private Sub btnDonation_Click(ByVal sender As
 2
       System.Object, ByVal e As System.EventArgs) _
 3
       Handles btnDonation.Click
 4
 5
       Select Case Val(txtDonationAmount.Text)
 6
 7
          Case 0
 8
             lblMessage.Text = "Please consider donating to our cause."
 9
10
          Case 1 To 100
11
             lblMessage.Text = "Thank you for your donation."
12
13
          Case Is > 100
14
              lblMessage.Text = "Thank you very much for your donation!"
15
16
          Case Else
17
             lblMessage.Text = "Please enter a valid amount."
18
19
       End Select
20
21
    End Sub
```

**Answer:** The output Label 1blMessage displays "Please consider donating to our cause" if the user inputs 0 for a donation amount, "Thank you for your donation" if the user enters a dollar amount between 1 and 100, "Thank you very much for your donation" if the user enters a value greater than 100 dollars and "Please enter a valid amount" if the user enters invalid data.

What's wrong with this code?

**12.15** This Select Case statement should determine whether an Integer is even or odd. Find the error(s) in the following code:

```
1 Select Case intValue Mod 2
2
3 Case 0
4 IblOutput.Text = "Odd Integer"
5
6 Case 1
7 IblOutput.Text = "Even Integer"
8
9 End Select
```

Answer: Line 4 and line 7 should be swapped.

```
1 Select Case intValue Mod 2
2
3 Case 0
4 IblOutput.Text = "Even Integer"
5
6 Case 1
7 IblOutput.Text = "Odd Integer"
8
9 End Select
```

Using the Debugger 12.16 (Discount Calculator Application) The Discount Calculator application determines the discount the user will receive, based on how much money the user spends. A 15% discount is received for purchases over \$200, a 10% discount is received for purchases between \$150-\$199, a 5% discount is received for purchases between \$100-\$149 and a 2% discount is received for purchases between \$50-\$99. While testing your application, you notice that the application is not calculating the discount properly for some values. Use the debugger to find and fix the logic error(s) in the application. Figure 12.20 displays the correct output for the application. Discount Calculator - U X Discoun × 250 **i**) Your discount is: 15% Enter amount spent: View Discount OK Figure 12.20 Correct output for the Discount Calculator application. Answer: 1 Exercise 12.16 Solution 2 ' Discount.vb 3 4 Public Class FrmDiscountCalculator 5 Inherits System.Windows.Forms.Form 6 7 ' Windows Form Designer generated code 8 9 ' display user's discount 10 Private Sub btnView\_Click(ByVal sender As System.Object, \_ 11 ByVal e As System. EventArgs) Handles btnView. Click 12 13 Dim intTotal As Integer ' amount spent 14 Dim strOutput As String ' displays discount 15 Removed superfluous 16 intTotal = Val(txtAmount.Text) ' get user's total Case Is > 200 statement that 17 prevented correct Case Is > 200 statement from 18 Select Case intTotal executing 19 20 Case 50 To 99 ' user spent between \$50-99 21 strOutput = "Your discount is: 2%" Incorrect code given to 22 students displayed a 5% 23 Case 100 To 149 ' user spent between \$100-149 discount for values between 24 strOutput = "Your discount is: 5%" 100 and 150 inclusive 25 26 Case 150 To 199 ' user spent between \$150-199 27 strOutput = "Your discount is: 10%" 28 Case Is >= 200 ' user spent over \$200 29 30 strOutput = "Your discount is: 15%" 31 32 Case Else ' user spent less than \$50 33 strOutput = "No discount" 34 35 End Select 36 37 ' display discount to user 38 MessageBox.Show(strOutput, "Discount", \_ 39 MessageBoxButtons.OK, MessageBoxIcon.Information) 40 41 End Sub ' btnView\_Click

42 43 End Class ' FrmDiscountCalculator

Programming Challenge

**12.17** (Enhanced Cash Register Application) Modify the Cash Register application (Exercise 12.12) to include the operations addition, subtraction and multiplication. Remove the Enter Button, and replace it with the addition (+), subtraction (-) and multiplication (\*) Buttons. These Buttons should take the value displayed in the Subtotal: field and the value displayed in the upper Label and perform the operation of the clicked Button. The result should be displayed in the Subtotal: field. Figure 12.21 displays the enhanced Cash Register application GUI.

Cash Register	<u> </u>	Cash Register	
\$12.34		\$	
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	Total Delete Clear	1 2 3 4 5 6 7 8 5 + 0 -	Total Delete Clear
Subtotal:	\$0.00	Subtotal:	(\$12.34)
Tax:	\$0.00	Tax:	\$0.00
Total:	\$0.00	Total:	\$0.00
🖳 Cash Register		Cash Register	
\$10.05		\$	
1 2 3	Total	1 2 3	Total

1 2 3 4 5 6 7 8 5 9 + 0 . - *	1 2 4 5 7 8 + 0 5 -	3 6 9 Clear *
Subtotal: (\$12.34)	Subtotal:	(\$2.29)
Tax: \$0.00	Tax:	\$0.00
Total: \$0.00	Total:	\$0.00

Figure 12.21 Enhanced Cash Register GUI.

1	' Exercise 12.17 Solution
2	' CashRegister.vb
3	
4	Public Class FrmCashRegister
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' invoked when btnOne is clicked
10	Private Sub btnOne_Click(ByVal sender As _
11	System.Object, ByVal e As System.EventArgs) _
12	Handles btnOne.Click
13	

14	txtCurrentPrice.Text &= "1"
15	End Sub ' btnOne_Click
16	
17	' invoked when btnTwo is clicked
18	Private Sub btnTwo_Click(ByVal sender As _
19	System.Object, ByVal e As System.EventArgs) _
20	Handles btnTwo.Click
20	Handles DLITWO.CITCK
22	<pre>txtCurrentPrice.Text &amp;= "2"</pre>
23	End Sub ' btnTwo_Click
24	
25	' invoked when btnThree is clicked
26	<pre>Private Sub btnThree_Click(ByVal sender As _</pre>
27	System.Object, ByVal e As System.EventArgs) _
28	Handles btnThree.Click
29	
30	<pre>txtCurrentPrice.Text &amp;= "3"</pre>
31	End Sub ' btnThree_Click
32	Ella Sab Dell'Ill'ee_ellek
33	' invoked when btnFour is clicked
34	<pre>Private Sub btnFour_Click(ByVal sender As _</pre>
35	System.Object, ByVal e As System.EventArgs) _
36	Handles btnFour.Click
37	
38	<pre>txtCurrentPrice.Text &amp;= "4"</pre>
39	End Sub ' btnFour Click
40	-
41	' invoked when btnFive is clicked
42	Private Sub btnFive_Click(ByVal sender As _
43	System.Object, ByVal e As System.EventArgs) _
43	
	Handles btnFive.Click
45	
46	<pre>txtCurrentPrice.Text &amp;= "5"</pre>
47	End Sub ' btnFive_Click
48	
49	' invoked when btnSix is clicked
50	Private Sub btnSix_Click(ByVal sender As _
51	System.Object, ByVal e As System.EventArgs) _
52	Handles btnSix.Click
53	
54	<pre>txtCurrentPrice.Text &amp;= "6"</pre>
55	End Sub ' btnSix_Click
56	End bub Schotk_errek
57	' invoked when btnSeven is clicked
58	
	Private Sub btnSeven_Click(ByVal sender As _
59	System.Object, ByVal e As System.EventArgs) _
60	Handles btnSeven.Click
61	
62	txtCurrentPrice.Text &= "7"
63	End Sub ' btnSeven_Click
64	
65	' invoked when btnEight is clicked
66	<pre>Private Sub btnEight_Click(ByVal sender As _</pre>
67	System.Object, ByVal e As System.EventArgs) _
68	Handles btnEight.Click
69	handles benergie.errek
	tytCunnontDrico Toyt 9 101
70	txtCurrentPrice.Text &= "8"
71	End Sub ' btnEight_Click
72	
73	' invoked when btnNine is clicked
74	<pre>Private Sub btnNine_Click(ByVal sender As _</pre>

75	System.Object, ByVal e As System.EventArgs) _
76	Handles btnNine.Click
	nanutes beinghie.ertek
77	
78	txtCurrentPrice.Text &= "9"
79	End Sub ' btnNine Click
80	
	I developed only a 7 and developed
81	' invoked when btnZero is clicked
82	<pre>Private Sub btnZero_Click(ByVal sender As _</pre>
83	System.Object, ByVal e As System.EventArgs) _
84	Handles btnZero.Click
85	
86	txtCurrentPrice.Text &= "O"
87	End Sub ' btnZero_Click
88	
89	' invoked when btnPoint is clicked
90	
	<pre>Private Sub btnPoint_Click(ByVal sender As _</pre>
91	System.Object, ByVal e As System.EventArgs) _
92	Handles btnPoint.Click
93	
94	<pre>txtCurrentPrice.Text &amp;= "."</pre>
95	End Sub ' btnPoint_Click
96	
97	' invoked when btnTotal is clicked
98	<pre>Private Sub btnTotal_Click(ByVal sender As _</pre>
99	
	System.Object, ByVal e As System.EventArgs) _
100	Handles btnTotal.Click
101	
102	Dim dblTaxRate As Double
103	
	I determine the mate beaution subtate?
104	' determines tax rate based on subtotal
105	Select Case lblSubTotalValue.Text
106	
107	Case Is < 100
108	db]TaxRate = 0.05
	ubildx rate = 0.05
109	
110	Case 100 To 500
111	dbTaxRate = 0.075
112	
113	Case Is $> 500$
114	dblTaxRate = 0.1
115	
116	End Select
117	
118	' dicplay subtatal tax and total in Labols
	' display subtotal, tax and total in Labels
119	lblSubTotalValue.Text = String.Format("{0:C}", _
120	lblSubTotalValue.Text)
121	
122	<pre>lblTaxValue.Text = String.Format("{0:C}", _</pre>
123	
	lblSubTotalValue.Text * dblTaxRate)
124	
125	lblTotalValue.Text = <pre>String.Format("{0:C}", _</pre>
126	lblSubTotalValue.Text + (lblSubTotalValue.Text * _
127	dblTaxRate))
128	End Sub ' btnTotal_Click
129	
130	' invoked when btnClear is clicked
131	<pre>Private Sub btnClear_Click(ByVal sender As _</pre>
132	System.Object, ByVal e As System.EventArgs) _
133	Handles btnClear.Click
134	
135	' clear all Labels and set total and tax to \$0.00

136txtCurrentPrice.Clear()1371b1SubTotalValue.Text = "\$0.00"1381b1TaxValue.Text = "\$0.00"	
138 lblTaxValue.Text = "\$0.00"	
<pre>138 lblTaxValue.Text = "\$0.00"</pre>	
<pre>139 lblTotalValue.Text = "\$0.00"</pre>	
141	
142 ' invoked when btnDelete is clicked	
<pre>143 Private Sub btnDelete_Click(ByVal sender As _</pre>	
144 System.Object, ByVal e As System.EventArgs) _	
145 Handles btnDelete.Click	
146	
147 txtCurrentPrice.Clear() ' clear the TextBox	
149	
150 ' invoked when plus Button is pressed	
<pre>151 Private Sub btnPlus_Click(ByVal sender As _</pre>	
152 System.Object, ByVal e As System.EventArgs) _	
153 Handles btnPlus.Click	
154	
155 Dim decAmount As Decimal	
156	
<pre>158 decAmount = Val(txtCurrentPrice.Text)</pre>	
159	
160 ' add input amount to dblTotal and clear Label	
<pre>161 lblSubTotalValue.Text = String.Format("{0:C}", _</pre>	
<pre>162 lblSubTotalValue.Text + decAmount)</pre>	
163	
164 txtCurrentPrice.Clear() ' clear the TextBox	
165 End Sub ' btnPlus_Click	
166	
167 ' invoked when minus Button clicked	
169 System.Object, ByVal e As System.EventArgs) _	
170 Handles btnMinus.Click	
171	
172 Dim decAmount As Decimal	
173	
174 ' store txtCurrentPrice to subtract from lblSubTotal\	/alue
<pre>175 decAmount = Val(txtCurrentPrice.Text)</pre>	
176	
177 ' subtract decAmount from lblSubTotalValue	
<pre>178 lblSubTotalValue.Text = String.Format("{0:C}", _</pre>	
179 IblSubTotalValue.Text - decAmount)	
-	
181 txtCurrentPrice.Clear() ' clear txtCurrentPrice	
182 End Sub ' btnMinus_Click	
183	
184 ' invoked when multiply Button is clicked	
<pre>185 Private Sub btnMultiply_Click(ByVal sender As _</pre>	
186 System.Object, ByVal e As System.EventArgs) _	
187 Handles btnMultiply.Click	
188	
189 Dim decAmount As Decimal ' store price entered	
190	
140	
191 ' store amount to multiply	
<pre>191 ' store amount to multiply 192 decAmount = Val(txtCurrentPrice.Text)</pre>	
<pre>191 ' store amount to multiply 192 decAmount = Val(txtCurrentPrice.Text) 193</pre>	
<pre>191 ' store amount to multiply 192 decAmount = Val(txtCurrentPrice.Text) 193 194 ' multiply subtotal amount with decAmount</pre>	
<pre>191 ' store amount to multiply 192 decAmount = Val(txtCurrentPrice.Text) 193</pre>	

197		
198	<pre>txtCurrentPrice.Clear() ' clear the TextBox</pre>	
199	End Sub ' btnMultiply_Click	
200		
201	End Class ' FrmCashRegister	





# Enhancing the Wage Calculator Application

Introducing Function Procedures and Sub Procedures Solutions

)	Instructor's Manual Exercise Solutions Tutorial 13			
	MULTIPLE-CHOICE	<b>13.1</b> A procedure defined with keywor	d Sub	
	QUESTIONS	<ul><li>a) must specify a return type</li><li>c) returns a value</li></ul>	<ul><li>b) does not accept parameters</li><li>d) does not return a value</li></ul>	
			applications from small, manageable pieces is known	
		as a) divide and conquer c) click and mortar	<ul><li>b) returning a value</li><li>d) a building-block algorithm</li></ul>	
		<ul><li>13.3 What is the difference between Sub and Function procedures?</li><li>a) Sub procedures return values, Function procedures do not.</li><li>b) Function procedures return values, Sub procedures do not.</li><li>c) Sub procedures accept parameters, Function procedures do not.</li><li>d) Function procedures accept parameters, Sub procedures do not.</li></ul>		
)		<ul> <li>13.4 What occurs after a procedure call is made?</li> <li>a) Control is given to the called procedure. After the procedure is run, the application continues execution at the point where the procedure call was made.</li> <li>b) Control is given to the called procedure. After the procedure is run, the application continues execution with the statement after the called procedure's definition.</li> <li>c) The statement before the procedure call is executed.</li> <li>d) The application terminates.</li> </ul>		
		<ul><li>13.5 Functions can return</li><li>a) zero</li><li>c) one or more</li></ul>	value(s). b) exactly one d) any number of	
		<ul><li>13.6 Which of the following must be tr</li><li>a) The number of arguments in the ters in the procedure header.</li></ul>	, <b>.</b>	
		<b>13.7</b> Which of the following statemen Function procedure?	nts correctly returns the variable intValue from a	
		<ul><li>a) Return Dim intValue</li><li>c) intValue Return</li></ul>	b) Return intValue As Integer d) Return intValue	
		<b>13.8</b> The Button executes the next statement in the application. If the next ment to execute contains a procedure call, the called procedure executes in its entirety		
		<ul><li>a) Step Into</li><li>c) Step Over</li></ul>	<ul><li>b) Step Out</li><li>d) Steps</li></ul>	
			luding the keyword Sub or Function, the procedure ion procedure return type) is known as the procedure	
)		a) body	b) title	

**13.10** Method \_\_\_\_\_\_ of class Math calculates the square root of the value passed as an argument.

a) SquareRoot	b) Root
c) Sqrt	d) Square

Answers: 13.1) d. 13.2) a. 13.3) b. 13.4) a. 13.5) b. 13.6) c. 13.7) d. 13.8) c. 13.9) d. 13.10) c.

**EXERCISES** 

**13.11** (*Temperature Converter Application*) Write an application that performs various temperature conversions (Fig. 13.28). The application should be capable of performing two types of conversions: degrees Fahrenheit to degrees Celsius and degrees Celsius to degrees Fahrenheit.

Temperature Converter 💶 🗙	E Temperature Converter I I I
Degrees: 17	Degrees: 46
17 degrees Celsius is equal to 62.60 degrees Fahrenheit.	46 degrees Fahrenheit is equal to 7.78 degrees Celsius.
Convert To Fahrenheit	Convert To Fahrenheit Convert To Celsius

Figure 13.28 Temperature Converter GUI.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial13\Exercises\TemperatureConversion directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click TemperatureConversion.sln in the TemperatureConversion directory to open the application.
- c) *Convert Fahrenheit to Celsius*. To convert degrees Fahrenheit to degrees Celsius, use this formula:

dblCelsius = (5 / 9) \* (dblFahrenheit - 32)

d) *Convert Celsius to Fahrenheit*. To convert degrees Celsius to degrees Fahrenheit, use this formula:

dblFahrenheit = (9 / 5) \* dblCelsius + 32

- e) Adding event handlers to your application. Double click each Button to add the proper event handlers to your application. These event handlers will call procedures (that you will define in the next step) to convert the degrees entered to either Fahrenheit or Celsius. Each event handler will display the result in the application's output Label.
- f) Adding Function procedures to your application. Create Function procedures to perform each conversion, using the formulas above. The user should provide the temperature to convert.
- g) Formatting the temperature output. To format the temperature information, use the String.Format method. Use F as the formatting code to limit the temperature to two decimal places.
- h) Running the application. Select Debug > Start to run your application. Enter a temperature value. Click the Convert to Fahrenheit Button and verify that correct output is displayed based on the formula given. Click the Convert to Celsius Button and again verify that the output is correct.
- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

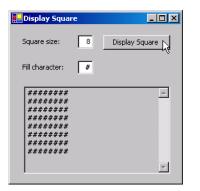
**Answer:** 

- 2 ' TemperatureConversion.vb

3

4	Public Class FrmTemperatureConverter
5	Inherits System.Windows.Forms.Form
6	
7	L Mindows Form Deciman accounted and
	' Windows Form Designer generated code
8	
9	' converts degrees to Fahrenheit
10	<pre>Private Sub btnConvertFahrenheit_Click(ByVal sender As _</pre>
11	System.Object, ByVal e As System.EventArgs) _
12	Handles btnConvertFahrenheit.Click
	nancies binconvertramenneit.citck
13	
14	Dim dblDegree As Double = Val(txtDegrees.Text)
15	
16	lblOutput.Text = dblDegree & _
17	" degrees Celsius is equal to " & _
18	<pre>ControlChars.CrLf &amp; String.Format("{0:F}", _</pre>
19	ConvertToFahrenheit(dblDegree)) & _
20	" degrees Fahrenheit."
21	
22	End Sub ' btnConvertFahrenheit_Click
23	
24	' converts degrees to Celsius
25	Private Sub btnConvertCelsius_Click(ByVal sender As _
26	System.Object, ByVal e As System.EventArgs) _
27	Handles btnConvertCelsius.Click
28	
29	
	<pre>Dim dblDegree As Double = Val(txtDegrees.Text)</pre>
30	
31	lblOutput.Text = dblDegree & _
32	" degrees Fahrenheit is equal to " & _
33	<pre>ControlChars.CrLf &amp; String.Format("{0:F}", _</pre>
34	ConvertToCelsius(dblDegree)) & _
35	
	" degrees Celsius."
36	
37	End Sub ' btnConvertCelsius_Click
38	
39	' convert degree to Fahrenheit
40	Function ConvertToFahrenheit(ByVal dblDegree As Double) As Double
	function convertion an ennerc(byvar ub begree As bouble) As bouble
41	
42	Return (9 / 5) * dblDegree + 32
43	End Function ' ConvertToFahrenheit
44	
45	' convert degree to Celsius
46	Function ConvertToCelsius(ByVal dblDegree As Double) As Double
	runceron converciocersius(byvar ubibegree AS bouble) AS DOUBLE
47	
<b>48</b>	Return (5 / 9) * (dblDegree - 32)
49	End Function ' ConvertToCelsius
50	
51	End Class ' FrmTemperatureConverter

**13.12** (*Display Square Application*) Write an application that displays a solid square composed of a character input by the user (Fig. 13.29). The user also should input the size.





- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial13\Exercises\DisplaySquare directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click DisplaySquare.sln in the DisplaySquare directory to open the application.
- c) Adding a Sub procedure. Write a Sub procedure DisplaySquare to display the solid square. The size should be specified by the Integer parameter intSize. The character that fills the square should be specified by the String parameter strFillCharacter. You should use a For...Next statement nested within another For...Next statement to create the square. The outer For...Next specifies what row is currently being displayed. The inner For...Next appends all the characters that form the row to a display String.
- d) Adding an event handler for your Button's Click event. Double click the Display Square Button to create the event handler. Program the event handler to call procedure DisplaySquare.
- e) **Displaying the output**. Use the multiline TextBox provided to display the square. For example, if intSize is 8 and strFillCharacter is #, the application should look similar to Fig. 13.29.
- f) Running the application. Select Debug > Start to run your application. Enter a size for the square (the length of each side) and a fill character. Click the Display Square Button. A square should be displayed of the size you specified, using the character you specified.
- g) *Closing the application.* Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 13.12 Solution
 2
    ' DisplaySquare.vb
 3
 4
    Public Class FrmDisplaySquare
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' display square in TextBox
10
       Sub DisplaySquare(ByVal intSize As Integer, _
11
           ByVal strFillCharacter As String)
12
13
           ' declare loop variables
14
          Dim intRow As Integer
                                    ' number of rows counter
15
          Dim intColumn As Integer ' number of columns counter
16
          Dim strOutput As String ' output String
17
18
           ' loop until intRow reaches value of first argument (intSize)
```

```
19
           For intRow = 1 To intSize
20
21
              ' loop until intColumn reaches value of intSize
22
              For intColumn = 1 To intSize
23
                 strOutput &= strFillCharacter
24
              Next
25
26
              strOutput &= ControlChars.CrLf ' add line to output
27
           Next
28
29
           txtOutput.Text = strOutput ' display square in output area
30
        End Sub ' DisplaySquare
31
32
        ' handles Display Square Button's Click event
33
        Private Sub btnDisplaySquare_Click(ByVal sender As _
34
           System.Object, ByVal e As System.EventArgs) _
35
           Handles btnDisplaySquare.Click
36
37
           ' if valid input is entered
38
           If txtSideSize.Text <> "" AndAlso _
39
              txtFillCharacter.Text <> "" Then
40
41
              DisplaySquare(Val(txtSideSize.Text), _
42
                 txtFillCharacter.Text)
43
           Flse
44
              MessageBox.Show("Square size and fill character needed", _
45
                 "Incorrect Input", MessageBoxButtons.OK, __
46
                 MessageBoxIcon.Exclamation)
47
           End If
48
49
        End Sub ' btnDisplaySquare_Click
50
51
     End Class ' FrmDisplaySquare
```

**13.13** (*Miles Per Gallon Application*) Drivers often want to know the miles per gallon their cars get so they can estimate gasoline costs. Develop an application that allows the user to input the numbers of miles driven and the number of gallons used for a tank of gas.

Miles Per Gallon			
Miles driven:	304.5		
Gallons used:	13.1		
Miles per gallon:	23.24		
Calculate MPG			

Figure 13.30 Miles Per Gallon application.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial13\Exercises\MilesPerGallon directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click MilesPerGallon.sln in the MilesPerGallon directory to open the application.
- c) *Calculating the miles per gallon*. Write a Function procedure MilesPerGallon that takes the number of miles driven and gallons used (entered by the user), calculates the amount of miles per gallon and returns the miles per gallon for a tankful of gas.

- d) **Displaying the result**. Create a Click event handler for the **Calculate MPG** Button that invokes the Function procedure MilesPerGallon and displays the result returned from the procedure as in Fig. 13.30.
- e) *Running the application.* Select **Debug > Start** to run your application. Enter a value for the number of miles driven and the amount of gallons used. Click the **Calculate MPG** Button and verify that the correct output is displayed.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

**Answer:** 

```
1
     ' Exercise 13.13 Solution
    ' MilesPerGallon.vb
 2
 3
 4
    Public Class FrmMilesPerGallon
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 0
        ' calculate and return miles per gallon
10
       Function MilesPerGallon( _
11
           ByVal dblMilesDriven As Double,
12
          ByVal dblGallonsUsed As Double) As Double
13
14
          Return dblMilesDriven / dblGallonsUsed
15
       End Function ' MilesPerGallon
16
17
        ' handles CalculateMPG Button's Click event
       Private Sub btnCalculateMPG_Click(ByVal sender As _
18
19
           System.Object, ByVal e As System.EventArgs) _
20
          Handles btnCalculateMPG.Click
21
22
           ' display miles per gallon
23
          lblOutputValue.Text = String.Format("{0:F}", _
24
             MilesPerGallon(Val(txtMilesDriven.Text), _
25
             Val(txtGallonsUsed.Text)))
26
27
       End Sub ' btnCalculateMPG_Click
28
29
    End Class ' FrmMilesPerGallon
```

```
What does this code do?
```

**13.14** What does the following code do? Assume this procedure is invoked by using Mystery(70, 80).

```
1
     Sub Mystery(ByVal intNumber1 As Integer, ByVal _
 2
        intNumber2 As Integer)
 3
 4
        Dim intX As Integer
 5
        Dim dblY As Double
 6
 7
        intX = intNumber1 + intNumber2
 8
        dblY = intX / 2
 9
10
        If dblY <= 60 Then</pre>
11
           lblResult.Text = "<= 60 "</pre>
12
        Flse
13
           lblResult.Text = "Result is " & dblY
14
        End If
```

15

16 End Sub ' Mystery

**Answer:** This code calculates the average of two numbers. The user is informed of the average, but only if the average is above 60. In this case the two numbers entered are 70 and 80, which results in an average of 75, which will be displayed.

What's wrong with this code?

13.15 Find the error(s) in the following code, which should take an Integer value as an argument and return the value of that argument multiplied by two.

1 Function TimesTwo(ByVal intNumber As Integer) As Integer
2
3 Dim intResult As Integer
4
5 intResult = intNumber \* 2
6 End Function ' TimesTwo

**Answer:** A Function procedure should Return a value. The default return value for a Function procedure with an Integer return type is 0. This Function procedure will always return 0. Corrected code:

```
1 Function TimesTwo(ByVal intNumber As Integer) As Integer
2
3 Dim intResult As Integer
4
5 intResult = intNumber * 2
6
7 Return intResult
8 End Function ' TimesTwo
```

Using the Debugger

**13.16** (Gas Pump Application) The Gas Pump application calculates the cost of gas at a local gas station. This gas station charges \$1.41 per gallon for **Regular** grade gas, \$1.47 per gallon for **Special** grade gas and \$1.57 per gallon for **Super+** grade gas. The user enters the number of gallons to purchase and clicks the desired grade. The application calls a Sub procedure to compute the total cost from the number of gallons entered and the selected grade. While testing your application, you noticed that one of your totals was incorrect, given the input.

🚛 Gas Pump			<u>_   X</u>
Number of gallons:	14	Total:	\$20.58
Regular	Special		Super+

Figure 13.31 Gas Pump application executing correctly.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutoriall3\Debugger\GasPumpIncorrect directory to your C:\SimplyVB directory.
- b) Opening the application's template file. Double click GasPump.sln in the GasPump-Incorrect directory to open the application.
- c) *Running the application*. Select **Debug > Start** to run your application. Determine which total is incorrect.

- d) **Setting a breakpoint**. Set a breakpoint at the beginning of the event handler that is providing incorrect output. For instance, if the **Regular** Button is providing incorrect output when clicked, add a breakpoint at the beginning of that Button's Click event handler. Use the debugger to help find any logic error(s) in the application.
- *Modifying the application*. Once you have located the error(s), modify the application so that it behaves correctly.
- f) Running the application. Select Debug > Start to run your application. Enter a number of gallons and click the Regular, Special and Super+ Buttons. After each Button is clicked, verify that the total displayed is correct based on the prices given in this exercise's description.
- g) *Closing the application.* Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

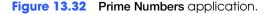
```
1
     ' Exercise 13.16 Solution
 2
    ' GasPump.vb
 3
 4
    Public Class FrmGasPump
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
       Dim intGallons As Integer = 0 ' number of gallons
10
11
        ' handles Regular Button's Click event
12
       Private Sub btnRegular_Click(ByVal sender As System.Object, _
13
          ByVal e As System. EventArgs) Handles btnRegular. Click
14
15
          intGallons = Val(txtNumberGallons.Text)
16
17
           ' call procedure to determine total
18
           ' first argument is Button's Text
19
          Total(btnRegular.Text, intGallons)
20
       End Sub ' btnRegular_Click
21
22
        ' handles Special Button's Click event
23
        Private Sub btnSpecial_Click(ByVal sender As System.Object, _
24
          ByVal e As System. EventArgs) Handles btnSpecial. Click
25
26
          intGallons = Val(txtNumberGallons.Text)
27
28
           ' call procedure to determine total
29
           ' first argument is Button's Text
30
          Total(btnSpecial.Text, intGallons)
31
       End Sub ' btnSpecial_Click
32
33
        ' handles Super Button's Click event
34
       Private Sub btnSuper_Click(ByVal sender As System.Object, _
35
          ByVal e As System. EventArgs) Handles btnSuper. Click
36
37
           intGallons = Val(txtNumberGallons.Text)
38
39
           ' call procedure to determine total
40
           ' first argument is Button's Text
41
          Total(btnSuper.Text, intGallons)
42
       End Sub ' btnSuper_Click
43
44
        ' calculate total cost of gas
45
        Sub Total(ByVal strGrade As String, ByVal intGallons As Integer)
```

46 47 ' determine grade selected and output total 48 Select Case strGrade 49 50 Case "Regular" 51 lblTotalResult.Text = \_ 52 String.Format("{0:C}", 1.41 \* intGallons) 53 54 Case "Special" 55 lblTotalResult.Text = String.Format("{0:C}", 1.47 \* intGallons) 56 Code provided to student had 57 value 1.91 in place of 1.47 58 Case "Super+" 59 lblTotalResult.Text = \_ 60 String.Format("{0:C}", 1.57 \* intGallons) 61 62 End Select 63 End Sub ' Total 64 65 66 End Class ' FrmGasPump

Programming Challenge

**13.17** (*Prime Numbers Application*) An Integer greater than 1 is said to be prime if it is divisible by only 1 and itself. For example, 2, 3, 5 and 7 are prime numbers, but 4, 6, 8 and 9 are not. Write an application that takes two numbers (representing a lower bound and an upper bound) and determines all of the prime numbers within the specified bounds, inclusive.

Prime Numbers	Prime Numbers
Lower bound: 2	Lower bound: 2
Upper bound: 200	Upper bound: 200
Prime numbers:	Prime numbers:          181         191         193         197         199
Calculate Primes	Calculate Primes



- a) *Creating the application*. Create an application named PrimeNumbers and have its GUI appear as shown in Fig. 13.32. Add an event handler for the **Calculate Primes** Button's Click event.
- b) *Checking for prime numbers*. Write a Function procedure Prime that returns True if a number is prime, False otherwise.
- c) *Limiting user input*. Allow users to enter a lower bound (intLower) and an upper bound (intUpper). Prevent the user from entering bounds less than or equal to 1, or an upper bound that is smaller than the lower bound.
- d) **Displaying the prime numbers.** Call Function procedure Prime from your event handler to determine which numbers between the lower and upper bounds are prime. Then have the event handler display the prime numbers in a multiline, scrollable TextBox, as in Fig. 13.32.
- e) Running the application. Select Debug > Start to run your application. Enter a lower bound and an upper bound that is smaller than the lower bound. Click the Calculate Primes Button. You should receive an error message. Enter negative bounds

and click the **Calculate Primes** Button. Again, you should receive an error message. Enter valid bounds and click the **Calculate Primes** Button. This time, the primes within that range should be displayed.

- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
    ' Exercise 13.17 Solution
 2
    ' PrimeNumbers.vb
 3
 4
    Public Class FrmPrimeNumbers
 5
        Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 0
        ' determine if number is prime
10
       Function Prime(ByVal intNumber As Integer) As Boolean
11
12
          Dim intCount As Integer ' declare counter
13
14
           ' set square root of intNumber as limit
15
          Dim intLimit As Integer = Math.Sqrt(intNumber)
16
17
           ' loop until intCount reaches square root of intNumber
18
           For intCount = 2 To intLimit
19
20
              If intNumber Mod intCount = 0 Then
21
                 Return False ' number is not prime
22
              End If
23
24
          Next
25
26
           Return True ' number is prime
27
       End Function ' Prime
28
29
        ' handles Calculate Primes Button's Click event
30
       Private Sub btnCalculatePrimes_Click(ByVal sender As _
31
           System.Object, ByVal e As System.EventArgs) _
32
          Handles btnCalculatePrimes.Click
33
34
           ' declare variables
35
          Dim intLowerBound As Integer = Val(txtLowerBound.Text)
36
          Dim intUpperBound As Integer = Val(txtUpperBound.Text)
37
          Dim intCounter As Integer
38
          Dim strOutput As String
39
40
          If intLowerBound <= 1 OrElse intUpperBound <= 1 Then</pre>
41
              MessageBox.Show("Bounds must be greater than 1", _
42
                 "Invalid Bounds", MessageBoxButtons.OK, _
43
                 MessageBoxIcon.Exclamation)
44
           ElseIf intUpperBound < intLowerBound Then</pre>
45
              MessageBox.Show("Upper bound cannot be less than " & _
46
                 "lower bound", "Invalid Bounds", MessageBoxButtons.OK, _
47
                 MessageBoxIcon.Exclamation)
48
           Else
49
50
              ' loop from lower bound to upper bound
51
              For intCounter = intLowerBound To intUpperBound
52
53
                 ' if prime number, display in TextBox
```

54	<pre>If Prime(intCounter) = True Then</pre>
55	<pre>strOutput &amp;= (intCounter &amp; ControlChars.CrLf)</pre>
56	End If
57	
58	Next
59	
60	End If
61	
62	<pre>txtPrimeNumbers.Text = strOutput</pre>
63	End Sub ' btnCalculatePrimes_Click
64	
65	End Class ' FrmPrimeNumbers

© Copyright 1992-2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.





# Shipping Time Application

Using Dates and Timers Solutions

Exercises Solutions Tutorial 14		
MULTIPLE-CHOICE	14.1 The allows you to store	and manipulate date information easily.
QUESTIONS	a) Date structure	b) DatePicker control
QUEUNUNU	c) GroupBox control	d) Now property
	14.2 You can to a Date varia	able.
	a) add hours	b) add days
	c) subtract hours	d) All of the above.
	<b>14.3</b> To subtract one day from Date va to dtmDay.	ariable dtmDay's value, assign the value returned by
	a) dtmDay.AddHours(-24)	<pre>b) dtmDay.SubtractDays(1)</pre>
	<pre>c) dtmDay.AddDays(-1)</pre>	d) Both a and c.
	<b>14.4</b> The time 3:45 and 35 seconds in according to the format string	the afternoon would be formatted as 03:45:35 PM
	a) "hh:mm:ss"	b) "hh:mm:ss tt"
	c) "hh:mm:ss am:pm"	d) "h:m:s tt"
	<b>14.5</b> A(n) event occurs before the Form is displayed.	
	a) LoadForm	b) InitializeForm
	c) Load	d) FormLoad
	<b>14.6</b> Timer property Interval sets the	rate at which Tick events occur in
	a) nanoseconds	b) microseconds
	c) milliseconds	d) seconds
	14.7 To set Date dtmNow's time five hou	urs earlier, use
		urs(5) b) dtmNow = dtmNow.AddHours(-5)
	c) dtmNow = dtmNow.AddHours(5)	) d) dtmNow.AddHours(-5)
	<b>14.8</b> A(n) is a container.	
	a) GroupBox	b) Form
	c) Timer	d) Both a and b.
	<b>14.9</b> A Date variable stores hour values in the range	
	a) 1 to 12	b) 0 to 12
	c) 0 to 24	d) 0 to 23
	<b>14.10</b> A DateTimePicker's display the date.	property specifies the format string with which to
	a) CustomFormat	b) FormatString
	c) Format	d) Text

**EXERCISES** 

**14.11** (*World Clock Application*) Create an application that displays the current time in Los Angeles, Atlanta, London and Tokyo. Use a Timer to update the clock every second. Assume that your local time is the time in Atlanta. Atlanta is three hours later than Los Angeles. London is five hours later than Atlanta. Tokyo is eight hours later than London. The application should look similar to Fig. 14.20.

🔜 World Clock	_ <b>_</b> ×
Los Angeles:	12/11/2002 2:14:17 PM
Atlanta:	12/11/2002 5:14:17 PM
London:	12/11/2002 10:14:17 PM
Tokyo:	12/12/2002 6:14:17 AM



- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial14\Exercises\WorldClock directory to your C:\SimplyVB directroy.
- b) Opening the application's template file. Double click WorldClock.sln in the WorldClock directory to open the application.
- c) Adding a Timer to the Form. Add a Timer control to the World Clock application. Set the Timer control's name property to tmrClock.
- d) Adding a Tick event handler for tmrClock. Add a Tick event handler for Timer tmrClock. The event handler should calculate and display the current times for Los Angeles, Atlanta, London and Tokyo. Use the Date variable's ToShortDateString and ToLongTimeString methods to create the display text.
- e) **Running the application.** Select **Debug > Start** to run your application. Look at the clock on your machine to verify that the time for Los Angeles is three hours earlier, the time in Atlanta is the same as what your clock says, the time in London is five hours later, and the time in Tokyo is 13 hours later (eight hours later than London).
- f) Closing the application. Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 14.11 Solution
2	' WorldClock.vb
3	
4	Public Class FrmWorldClock
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' update times
10	<pre>Private Sub tmrClock_Tick(ByVal sender As System.Object, _</pre>
11	ByVal e As System.EventArgs) Handles tmrClock.Tick
12	
13	' retrieve current time
14	Dim dtmNow As Date = Date.Now
15	
16	' display Los Angeles time
17	<pre>lblLATime.Text = dtmNow.AddHours(-3).ToShortDateString &amp; _</pre>
18	<pre>" &amp; dtmNow.AddHours(-3).ToLongTimeString</pre>
19	
20	' display Atlanta time
21	lblAtlantaTime.Text = dtmNow.ToShortDateString & _
22	" " & dtmNow.ToLongTimeString
23	
24	' display London time
25	<pre>lblLondonTime.Text = dtmNow.AddHours(5).ToShortDateString &amp; _</pre>
26	" " & dtmNow.AddHours(5).ToLongTimeString
27	
<b>28</b>	' display Tokyo time
29	<pre>lblTokyoTime.Text = dtmNow.AddHours(13).ToShortDateString &amp; _</pre>

30	" " & dtmNow.AddHours(13).ToLongTimeString
31	
32	End Sub ' tmrClock_Tick
33	
34	End Class ' FrmWorldClock

**14.12** (*Shipping Time Application Enhancement*) During the winter, a distribution center in Denver, Colorado needs to receive seafood shipments to supply the local ski resorts. Enhance the Shipping Time application by adding Denver, Colorado as another shipping destination. Denver is two time zones west of Portland, meaning time is two hours earlier than Portland, Maine. Because there are no direct flights to Denver, shipments from Portland will take 8 hours.

🖶 ShippingTime		<u>_   ×</u>	
	Current time is:	02:46:14 PM	
Drop Off			
Enter drop-	Enter drop-off time: 19:46 PM		
Express Shipping to Las Vegas			
Delivery time:	Saturday, July 19, 200	3 at 3:00 AM	
Express Shipping to Denver			
Delivery time:	Saturday, July 19, 200	3 at 9:00 AM	



- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial14\Exercises\ShippingTimeEnhanced directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click ShippingTime.sln in the ShippingTimeEnhanced directory to open the application.
- c) *Inserting a GroupBox*. Resize the Form to fit the **Express Shipping to Denver** GroupBox as shown in Fig. 14.21. Add a GroupBox to the Form. Change the Text property of the GroupBox to indicate that it will contain the delivery time in Denver. Resize and move the GroupBox so that it resembles the GUI shown in Fig. 14.21.
- d) *Inserting Labels*. In the GroupBox you just created, add an output Label to display the delivery time for a seafood shipment to Denver and a corresponding descriptive Label.
- e) *Inserting code to the DisplayDeliveryTime procedure*. Add code to Display-DeliveryTime procedure to compute and display the delivery time in Denver.
- f) Running the application. Select Debug > Start to run your application. Select various drop off times and ensure the delivery times are correct for both Las Vegas and Denver.
- g) Closing the application. Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 14.12 Solution
2	' ShippingTime.vb
3	
4	Public Class FrmShippingTime
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	

9	' update current time every second
10	<pre>Private Sub tmrClock_Tick(ByVal sender As System.Object, _</pre>
11	ByVal e As System.EventArgs) Handles tmrClock.Tick
12	byvar e no bystem Eventra gby handres em crock. Trek
	I water comment at a
13	' print current time
14	lblCurrentTime.Text = String.Format("{0:hh:mm:ss tt}", _
15	Date.Now)
16	
17	End Sub ' tmrClock_Tick
18	
19	' initialize DateTimePicker status when Form loads
20	Private Sub FrmShippingTime_Load(ByVal sender As _
21	System.Object, ByVal e As System.EventArgs) Handles _
22	MyBase.Load
23	
24	<pre>Dim dtmCurrentTime As Date = Date.Now ' store current time</pre>
25	
26	' set range of possible drop-off times
27	dtpDropOff.MinDate = New Date(dtmCurrentTime.Year, _
28	dtmCurrentTime.Month, dtmCurrentTime.Day, 0, 0, 0)
29	demear energine in a concernent metabay, 0, 0, 0)
30	dtpDropOff.MaxDate = dtpDropOff.MinDate.AddDays(1)
31	
32	' display the delivery time
33	DisplayDeliveryTime()
34	
35	End Sub ' FrmShippingTime_Load
36	
37	' update ship time on change of drop-off time
38	<pre>Private Sub dtpDropOff_ValueChanged(ByVal sender As _</pre>
39	System.Object, ByVal e As System.EventArgs) Handles _
40	dtpDropOff.ValueChanged
41	a cpor opor i ri a racenarigea
42	' display the delivery time
43	DisplayDeliveryTime()
43	DisplayDeriveryTime()
45	End Sub ' dtpDropOff_ValueChanged
46	
47	' calculates and displays the delivery time
48	Sub DisplayDeliveryTime() As Date
49	
50	' print initial delivery time
51	<pre>Dim dtmDelivery As Date = DepartureTime()</pre>
52	
53	' add 3 hours to departure and display result
54	<pre>dtmDelivery = dtmDelivery.AddHours(3)</pre>
55	<pre>lblLasVegasTime.Text = dtmDelivery.ToLongDateString _</pre>
56	& " at " & dtmDelivery.ToShortTimeString
57	
58	' add 6 hours to departure and display result
59	
	dtmDelivery = dtmDelivery.AddHours(6)
60	<pre>lblDenverTime.Text = dtmDelivery.ToLongDateString _</pre>
61	& " at " & dtmDelivery.ToShortTimeString
62	
63	End Sub ' DisplayDeliveryTime
64	
65	' returns flight departure time for selected drop-off time
66	Function DepartureTime() As Date
67	
68	<pre>Dim dtmCurrentDate As Date = Date.Now ' store current date</pre>
69	Dim dtmDepartureTime As Date 'store departure time

/0	
71	' determine which flight the shipment takes
72	Select Case dtpDropOff.Value.Hour
73	
74	' seafood will be on the noon flight
75	Case 0 To 10
76	dtmDepartureTime = New Date(dtmCurrentDate.Year, _
77	<pre>dtmCurrentDate.Month, dtmCurrentDate.Day, 12, 0, 0)</pre>
78	
79	' seafood will be on tomorrow's noon flight
80	Case 23
81	dtmCurrentDate = dtmCurrentDate.AddDays(1)
82	dtmDepartureTime = New Date(dtmCurrentDate.Year, _
83	dtmCurrentDate.Month, dtmCurrentDate.Day, 12, 0, 0)
84	
85	' seafood will be on midnight flight
86	Case Else
87	dtmCurrentDate = dtmCurrentDate.AddDays(1)
88	dtmDepartureTime = New Date(dtmCurrentDate.Year, _
89	dtmCurrentDate.Month, dtmCurrentDate.Day, 0, 0, 0)
90	
91	End Select
92	
93	Return dtmDepartureTime ' return the flight's departure time
94	End Function ' DepartureTime
95	
96	End Class ' FrmShippingTime

14.13 (Alarm Application) Create an application that allows the user to set an alarm clock. The application should allow the user to set the exact time of the alarm by using a DateTime-Picker. While the alarm is set, the user should not be able to modify the DateTimePicker. If the alarm is set and the current time matches or exceeds the time in the DateTimePicker, play the computer's "beep" sound. (Your computer must have the necessary hardware for sound.) The user should be able to cancel an alarm by using a **Reset** Button. This Button is disabled when the application starts.

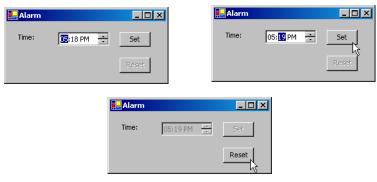


Figure 14.22 Alarm GUI.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial14\Exercises\AlarmClock directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click AlarmClock.sln in the AlarmClock directory to open the application.
- c) *Inserting a DateTimePicker*. Add a DateTimePicker control to the Form. Set the DateTimePicker to display only the time, as is shown in Fig. 14.22. Set the DateTimePicker control's Size property to 80, 20, and move the control so that it appears as it does in Fig. 14.22.

- d) Coding the Set Button's Click event handler. Add a Click event handler for the Set Button. This event handler should disable the Set Button and the DateTime-Picker and enable the Reset Button.
- e) Coding the Timer's Tick event handler. Define the Tick event handler for the Timer. A Tick event should occur every 1000 milliseconds (one second). If the alarm is set and the current time matches or exceeds the time in the DateTimePicker, play the computer's "beep" sound by calling the Beep function. To call the Beep function, type Beep() on its own line in your code.
- f) Coding the Reset Button's Click event handler. Define the Click event handler for the Reset Button. When the Reset Button is clicked, the GUI should be set back to its original state.
- g) **Running the application.** Select **Debug > Start** to run your application. Use the DateTimePicker and the **Set** Button to set a time for the alarm to go off. Wait for that time to verify that the alarm will make beeping sounds. Click the **Reset** Button to set a new time for the alarm to go off.
- h) *Closing the application.* Close your running application by clicking its close box.
- i) *Closing the IDE.* Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 14.13 Solution
 2
     ' AlarmClock.vb
 3
 4
    Public Class FrmAlarmClock
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' set time for alarm to go off
10
       Private Sub btnSetTime_Click(ByVal sender As System.Object, _
11
          ByVal e As System. EventArgs) Handles btnSetTime. Click
12
13
           ' disable user input
14
           btnSetTime.Enabled = False
15
           dtpAlarmTime.Enabled = False
16
17
           btnReset.Enabled = True ' enable reset
18
       End Sub ' btnSetTime_Click
19
20
        ' timer ticks once every minute
21
       Private Sub tmrTimerAlarm_Tick(ByVal sender As System.Object, _
22
          ByVal e As System. EventArgs) Handles tmrTimerAlarm. Tick
23
24
           ' sound the alarm
25
          If dtpAlarmTime.Value.Hour = Date.Now.Hour AndAlso _
26
              dtpAlarmTime.Value.Minute = Date.Now.Minute AndAlso _
27
              btnReset.Enabled = True Then
28
29
              ' call the Beep function
30
              Beep()
31
           End If
32
33
       End Sub ' tmrTimerAlarm_Tick
34
35
        ' return to initial state
36
       Private Sub btnReset_Click(ByVal sender As System.Object, _
37
           ByVal e As System. EventArgs) Handles btnReset. Click
38
39
           ' return all GUI controls to initial state
40
           btnSetTime.Enabled = True
41
          btnReset.Enabled = False
```

$\supset$	<pre>42 dtpAlarmTime.Enabled = True 43 End Sub ' btnReset_Click 44 45 End Class ' FrmAlarmClock</pre>		
	What does this code do? 🕨	14.14 This code creates a Date variable. What date does this variable contain?	
		Dim dtmTime As Date = New Date(2003, 1, 2, 3, 4, 5) Answer: This variable contains the date January 2, 2003 at 3:04:05 A.M.	
	What's wrong with this code?	<b>14.15</b> The following lines of code are supposed to create a Date variable and increment its hour value by two. Find the error(s) in the code.	
		<pre>Dim dtmNow As Date = Date.Now dtmNow.AddHours(2)</pre>	
		<b>Answer:</b> Method AddHours does not actually increment the Date variable, but instead returns a new Date variable with the updated value. Thus, the preceding code will not successfully add two hours to dtmNow. Correct the code as follows:	
$\bigcirc$		<pre>Dim dtmNow As Date = Date.Now dtmNow = dtmNow.AddHours(2)</pre>	
	Programming Challenge	<b>14.16</b> ( <i>Parking Garage Fee Calculator</i> ) Create an application that computes the fee for parking a car in a parking garage (Fig. 14.23). The user should provide the Time In: and Time Out: values by using DateTimePickers. The application should calculate the cost of parking in the garage for the specified amount of time. Assume that parking costs three dollars an hour. When calculating the total time spent in the garage, you can ignore the seconds value, but treat the minutes value as a fraction of an hour (1 minute is 1/60 of an hour). For simplicity, assume that no overnight parking is allowed, so each car leaves the garage on the same day in which it arrives.	
		Fee Calculator   Time in:   5:20:31 PM   Time out:   8:20:31 PM   Fee:   \$9.00     Calculate	
		<ul> <li>Figure 14.23 Parking Garage Fee Calculator GUI.</li> <li>a) Copying the template to your working directory. Copy the C:\Examples\Tutorial14\Exercises\ParkingGarageFeeCalculator directory to your C:\SimplyVB directory.</li> <li>b) Opening the application's template file. Double click ParkingGarageFeeCalculator directory to open the application</li> </ul>	

- c) **Inserting the DateTimePicker controls.** Add two DateTimePicker controls to the Form. Set the DateTimePickers so that they show the time only. Set the Size property of each DateTimePicker control to 80, 20, and move the DateTimePickers so that they are positioned as in Fig. 14.23.
- d) *Writing the Function procedure Fee.* Define a Function procedure Fee that accepts four Integers as parameters—the hour value of the Time In:, the hour value of the Time Out:, the minute value of the Time In: and the minute value of the Time Out:. Using this information, procedure Fee should calculate the fee for parking in the garage. The Function procedure should then return this value as a Decimal.

- e) *Coding the Calculate Button's Click event handler*. Add the Click event handler for the **Calculate** Button. This event handler should call Fee to obtain the amount due. It should then display the amount (formatted as currency) in a Label.
- f) Running the application. Select Debug > Start to run your application. Use the DateTimePickers' up and down arrows to select a time the car was placed in the garage and the time the car was taken out of the garage. Click the Calculate Button and verify that the correct fee is displayed.
- g) *Closing the application.* Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 14.16 Solution
 2
    ' ParkingGarageFeeCalculator.vb
 3
 4
    Public Class FrmParkingGarageFeeCalculator
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' calculates cost of parking in garage
10
       Function Fee(ByVal intTimeOutHour As Integer, _
11
            ByVal intTimeInHour As Integer, ByVal intTimeOutMinute
12
            As Integer, ByVal intTimeInMinute As Integer) As Decimal
13
14
           ' determines number of elapsed hours
15
          Dim intHours As Integer = intTimeOutHour - intTimeInHour
16
17
           ' determines number of elapsed minutes
18
          Dim intMinutes As Integer = intTimeOutMinute - intTimeInMinute
19
20
          If intMinutes < 0 Then</pre>
21
              intHours -= 1
22
              intMinutes += 60
23
          End If
24
25
           Return (intHours + (intMinutes / 60)) * 3
26
       End Function ' Fee
27
28
        ' called when Calculate Button is clicked
29
       Private Sub btnCalculate_Click(ByVal sender As System.Object, _
30
          ByVal e As System. EventArgs) Handles btnCalculate. Click
31
32
          Dim intFee As Decimal = 0
33
34
           ' calls procedure Fee
35
          intFee = Fee(dtpTimeOut.Value.Hour, _
              dtpTimeIn.Value.Hour, dtpTimeOut.Value.Minute, _
36
37
              dtpTimeIn.Value.Minute)
38
39
           ' output fee as currency
40
           lblFeeResult.Text = String.Format("{0:C}", intFee)
41
       End Sub ' btnCalculate_Click
42
43
    End Class ' FrmParkingGarageFeeCalculator
```





# **Fund Raiser Application**

Introducing Scope, Pass-by-Reference and Option Strict Solutions

)	Instructor's Manual Exercise Solutions Tutorial 15		
-	MULTIPLE-CHOICE QUESTIONS	<ul> <li>15.1 In the Property Pages dialog,</li> <li>a) Build</li> <li>c) General</li> </ul>	must be selected to access Option Strict. b) <b>Designer Defaults</b> d) <b>Imports</b>
		<ul><li>15.2 When Option Strict is set to 0n, v</li><li>a) are passed by value</li><li>c) might need to be converted explice</li><li>d) are used only within the block in v</li></ul>	b) are passed by reference citly to a different type to avoid errors
		<ul><li>15.3 A variable declared inside a class, b</li><li>a) local variable</li><li>c) instance variable</li></ul>	ut outside a procedure, is called a(n) b) hidden variable d) constant variable
		,	ls in class to convert from one data type
		a) ChangeTo c) ConvertTo	b) Convert d) ChangeType
		<b>15.5</b> When Option Strict is in an error.	, the conversion attempt intX = dblPercent results
)		a) On c) Off	b) True d) False
		<ul><li>15.6 Keyword indicates pass-</li><li>a) ByReference</li><li>c) Ref</li></ul>	by-reference. b) ByRef d) Reference
		<b>15.7</b> With, changes made to a parameter variable's value do not affect the of the variable in the calling procedure.	
		<ul><li>a) Option Strict</li><li>c) pass-by-reference</li></ul>	<ul><li>b) pass-by-value</li><li>d) None of the above.</li></ul>
		<ul> <li>15.8 Instance variables</li> <li>a) are members of class</li> <li>c) can be accessed by a procedure in</li> <li>d) All of the above.</li> </ul>	b) are prefixed by m_ the same class
		<ul><li><b>15.9</b> Assigning a "smaller" type to a "lar, a) narrowing</li></ul>	b) shortening
		<ul><li>c) widening</li><li>15.10 A value of type Boolean can be im</li></ul>	· ·
		a) Integer c) Object	b) String d) Double
		<b>Answers:</b> 15.1) a. 15.2) c. 15.3) c. 15.4) b	o. 15.5) a. 15.6) b. 15.7) b. 15.8) d. 15.9) c. 15.10) c.

**EXERCISES** 

**15.11** (*Task List Application*) Create an application that allows users to add items to a daily task list. The application should also display the number of tasks to be performed. Use method Convert.ToString to display the number of tasks in a Label. The application should look like the GUI in Fig. 15.29.

🔚 Task List	
Task:	Add Task
Task list: Call client Meeting at 4:00	Number of tasks: 2

Figure 15.29 Task List application's GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial15\Exercises\TaskList directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click TaskList.sln in the TaskList directory to open the application.
- c) **Setting Option Strict to On.** Use the directions provided in the box, *Enabling Option Strict*, to set Option Strict to On.
- d) Adding the Add Task Button's Click event handler. Double click the Add Task Button to generate the empty event handler btnAdd\_Click. This event handler should display the user input in the ListBox and clear the user input from the Text-Box. The event handler should also update the Label that displays the number of tasks. Use method Convert.ToString to display the number of tasks in the Label. Finally, the event handler should transfer the focus to the TextBox.
- e) *Running the application.* Select **Debug > Start** to run your application. Enter several tasks, click the **Add Task** Button after each. Verify that each task is added to the **Task list:** ListBox, and that the number of tasks is incremented with each new task.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) *Closing the IDE*. Close Visual Studio .NET by clicking its close box.

```
' Exercise 15.11 Solution
 1
    ' TaskList.vb
 2
 3
 4
    Public Class FrmTaskList
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 0
        ' handles Add Task Button's Click event
10
       Private Sub btnAdd_Click(ByVal sender As System.Object, _
11
          ByVal e As System. EventArgs) Handles btnAdd. Click
12
13
          lstTasks.Items.Add(txtTask.Text) ' insert task into ListBox
14
15
           txtTask.Clear() ' clear TextBox of user input
16
17
           ' convert Integer to String to display number of tasks
18
           lblOutput.Text = Convert.ToString(lstTasks.Items.Count)
19
20
           txtTask.Focus() ' transfer the focus to the TextBox
21
       End Sub ' btnAdd_Click
22
23
    End Class ' FrmTaskList
```

**15.12** (*Quiz Average Application*) Develop an application that computes a student's average quiz score for all of the quiz scores entered. The application should look like the GUI in Fig. 15.30. Use method Convert.ToInt32 to convert the user input to an Integer. Use

instance variables with module scope to keep track of the sum of all the quiz scores entered and the number of quiz scores entered.

🖳 Quiz Average			<u> </u>
Quiz score:	90	Submit	Score
Number taken:	4	Average:	89

<b>Figure 15.30</b>	Quiz Average	application's GUI.
---------------------	--------------	--------------------

- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial15\Exercises\QuizAverage directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click QuizAverage.sln in the QuizAverage directory to open the application.
- c) Setting Option Strict to On. Use the directions provided in the box, Enabling Option Strict, to set Option Strict to On.
- d) Adding instance variables. Add two instance variables -m\_intTotalScore, which keeps track of the sum of all the quiz scores entered, and m\_intTaken, which keeps track of the number of quiz scores entered.
- e) Adding the Grade Quiz Button's event handler. Double click the Submit Score Button to generate the empty event handler btnCalculate\_Click. The code required in *Steps f-k* should be placed in this event handler.
- f) Obtaining user input. Use method Convert.ToInt32 to convert the user input from the TextBox to an Integer.
- g) **Updating the number of quiz scores entered.** Increment the number of quiz scores entered.
- h) *Updating the sum of all the quiz scores entered.* Add the current quiz score to the current total to update the sum of all the quiz scores entered.
- i) *Calculating the average score.* Divide the sum of all the quiz scores entered by the number of quiz scores entered to calculate the average score.
- j) Displaying the average score. Use method Convert.ToString to display the average quiz grade in the Average: field.
- k) *Displaying the number of quizzes taken*. Use method Convert.ToString to display the number of quiz scores entered in the **Number taken**: field.
- Running the application. Select Debug > Start to run your application. Enter several quiz scores, clicking the Submit Score Button after each. With each new score, verify that the Number taken: field is incremented and that the average is updated correctly.
- m) Closing the application. Close your running application by clicking its close box.
- n) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 15.12 Solution
2	' QuizAverage.vb
3	
4	Public Class FrmQuizAverage
5	Inherits System.Windows.Forms.Form
6	
7	' instance variables store total score and number quizzes taken
8	Dim m_intTotalScore As Integer = 0
9	Dim m_intTaken As Integer = 0
10	
11	' Windows Form Designer generated code
12	
13	' handles Submit Score Button's Click event
14	<pre>Private Sub btnCalculate_Click(ByVal sender As System.Object, _</pre>

```
15
           ByVal e As System.EventArgs) Handles btnCalculate.Click
16
17
           Dim intScore As Integer
18
           Dim intAverage As Integer
19
20
           ' obtain and convert user input
21
           intScore = Convert.ToInt32(Val(txtScore.Text))
22
23
           ' update number of quizzes taken
24
           m_intTaken += 1
25
26
           ' update total score
27
           m_intTotalScore += intScore
28
29
           ' calculate average score
30
           intAverage = m_intTotalScore \ m_intTaken
31
32
           ' display average score
33
           lblAverage.Text = Convert.ToString(intAverage)
34
35
           ' display number of quizzes taken
36
           lblTaken.Text = Convert.ToString(m_intTaken)
37
38
        End Sub ' btnCalculate_Click
39
40
     End Class ' FrmQuizAverage
```

**15.13** (*Maximum Application*) Modify the Maximum application from Chapter 13 (Fig. 15.31) to use keyword ByRef to pass a fourth argument to procedure Maximum by reference. Also, use methods from class Convert to perform any necessary type conversions.

🔜 Maximum	
Enter first value:	3.5
Enter second value:	8.9
Enter third value:	8.3
Maximum:	8.9
	Maximum

Figure 15.31 Maximum application's GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial15\Exercises\Maximum directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click Maximum.sln in the Maximum directory to open the application.
- c) *Setting Option Strict to On.* Use the directions provided in the box, *Enabling Option Strict*, to set Option Strict to On.
- d) Adding a local variable. Add local variable dblMaximum of type Double to event handler btnMaximum\_Click. The code required in *Steps d-f* should be placed in this event handler. Variable dblMaximum will store the result of procedure Maximum.
- e) **Passing four arguments to procedure Maximum.** Use method Convert.ToDouble to convert the user input from the TextBoxes to Doubles. Pass these three values as the first three arguments to procedure Maximum. Pass local variable dblMaximum as the fourth argument to procedure Maximum.
- f) *Displaying the maximum value.* Use method Convert.ToString to display local variable dblMaximum in the Maximum: field.

- g) *Changing procedure Maximum to a Sub procedure.* Change procedure Maximum to a Sub procedure. Make sure that Sub procedure Maximum no longer returns a value and does not specify a return type. The modifications required in *Steps g-h* should be performed on this Sub procedure.
- h) Adding a fourth parameter to procedure Maximum. Add a fourth parameter dblFinalMaximum of type Double to Maximum's procedure header. Use keyword ByRef to specify that this argument will be passed by reference. Remove the declaration of variable dblFinalMaximum from the body of procedure Maximum.
- i) *Running the application.* Select **Debug > Start** to run your application. Enter three different values into the input fields and click the **Maximum** Button. Verify that the largest value is displayed in the **Maximum:** field.
- j) *Closing the application.* Close your running application by clicking its close box.
- k) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 15.13 Solution
2	' Maximum.vb
3	
4	Public Class FrmMaximum
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' obtain values in each TextBox, call procedure Maximum
10	<pre>Private Sub btnMaximum_Click(ByVal sender As System.Object, _</pre>
11	ByVal e As System.EventArgs) Handles btnMaximum.Click
12	
13	Dim dblMaximum As Double
14	
15	<pre>Maximum(Convert.ToDouble(Val(txtFirst.Text)), _</pre>
16	Convert.ToDouble(Val(txtSecond.Text)), _
17	Convert.ToDouble(Val(txtThird.Text)), dblMaximum)
18	
19	<pre>lblOutput.Text = Convert.ToString(dblMaximum)</pre>
20	End Sub ' btnMaximum_Click
21	
22	' find maximum of three parameter values
23	Sub Maximum(ByVal dblOne As Double, ByVal dblTwo _
24	As Double, ByVal dblThree As Double, _
25	ByRef dblFinalMaximum As Double)
26	
27	Dim dblTemporaryMaximum As Double
28	
29	dblTemporaryMaximum = Math.Max(dblOne, dblTwo)
30	dblFinalMaximum = Math.Max(dblTemporaryMaximum, dblThree)
31	End Sub ' Maximum
32	
33	End Class ' FrmMaximum

What does this code do?

15.14 What is displayed in Label lblDisplay when the following code is executed?

1	Public Class FrmScopeTest
2	Inherits System.Windows.Forms.Form
3	
4	Dim intValue2 As Integer = 5
5	
6	<pre>Private Sub btnEnter_Click(ByVal sender As System.Object, _</pre>

```
ByVal e As System.EventArgs) Handles btnEnter.Click
 8
 9
          Dim intValue1 As Integer = 10
10
          Dim intValue2 As Integer = 3
11
12
          Test(intValue1)
13
           lblDisplay.Text = Convert.ToString(intValue1)
14
       End Sub ' btnEnter_Click
15
16
       Sub Test(ByRef intValue1 As Integer)
17
           intValue1 *= intValue2
18
       End Sub ' Test
19
20
    End Class ' FrmScopeTest
```

Answer: Label lblDisplay displays the value of variable intValue1 (50). When the code invokes Sub procedure Test, it passes intValue1 pass-by-reference. Any changes made to intValue1 in Sub procedure Test are reflected in btnEnter\_Click's local variable intValue1. When Sub procedure Test multiplies intValue1 by intValue2, intValue2 is the class instance variable, whose value is 5. Procedure Test does not have access to btnEnter\_Click's local variable intValue2.

What's wrong with this code?

**15.15** Find the error(s) in the following code (the procedure should assign the value 14 to variable intResult).

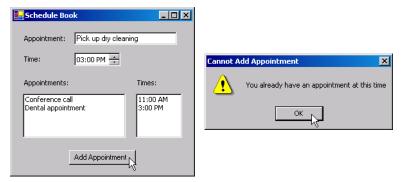
```
1 Sub Sum()
2 Dim strNumber As String = "4"
3 Dim intNumber As Integer = 10
4 Dim intResult As Integer
5
6 intResult = strNumber + intNumber
7 End Sub ' Sum
```

Answer: The code must explicitly convert strNumber to Integer:

```
1 Sub Sum()
2 Dim strNumber As String = "4"
3 Dim intNumber As Integer = 10
4 Dim intResult As Integer
5
6 intResult = Convert.ToInt32(strNumber) + intNumber
7 End Sub ' Sum
```

Programming Challenge

**15.16** (Schedule Book Application) Develop an application that allows a user to enter a schedule of appointments and their respective times. Create the Form in Fig. 15.32 and name the application Schedule Book. Add a Function procedure called TimeTaken that returns a Boolean value. Each time a user enters a new appointment, Function procedure TimeTaken determines if the user has scheduled more than one appointment at the same time. If TimeTaken returns True, the user will be notified via a message dialog. Otherwise, the appointment should be added to the ListBoxes. Set Option Strict to On and use methods from class Convert as necessary.





```
' Exercise 15.16 Solution
 1
 2
    ' ScheduleBook.vb
 3
 4
    Public Class FrmScheduleBook
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form designer generated code
 8
9
        ' handles Add Appointment Button's Click event
10
       Private Sub btnAdd_Click(ByVal sender As System.Object, _
11
          ByVal e As System. EventArgs) Handles btnAdd. Click
12
13
           ' appointment scheduled for given time
          Dim blnTimeTaken As Boolean = TimeTaken()
14
15
16
           ' display message if appointment conflict
17
          If blnTimeTaken = True Then
18
19
             MessageBox.Show("You already have an appointment " & _
20
                  "at this time", "Cannot Add Appointment", _
21
                 MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
22
           ' otherwise add appointment and time to ListBoxes
23
24
          Else
25
              lstAppointments.Items.Add(txtAppointment.Text)
26
             lstTimes.Items.Add(dtpTime.Value.ToShortTimeString())
27
          End If
28
29
           ' clear user input from TextBoxes
30
           txtAppointment.Clear()
31
       End Sub ' btnAdd_Click
32
33
        ' determines if an appointment already exists at specified time
34
        Function TimeTaken() As Boolean
35
36
           ' determine number of appointments
37
          Dim intItems As Integer = lstTimes.Items.Count()
38
39
           ' determines if items are in time ListBox
40
          If intItems <> 0 Then
41
42
             Dim intCounter As Integer
43
44
              ' search ListBox to determine if an appointment
45
              ' has been made for that time
46
              For intCounter = 0 To intItems - 1
```

' compare times in ListBox with user entry
<pre>If Convert.ToString(lstTimes.Items.Item( _</pre>
intCounter)) = dtpTime.Value.ToShortTimeString()
Return True
End If
Next
End If
Return False
End Function ' TimeTaken
End Class ' FrmScheduleBook





# **Craps Game Application**

Introducing Random-Number Generation Solutions

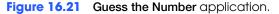
$\bigcirc$	Instructor's Manual				
$\bigcirc$	Exercise Solutions Tutorial 16				
	MULTIPLE-CHOICE	16.1 A Random object can generate pseudorar	ndom numbers of type		
	QUESTIONS	a) Integer	b) Single		
	QUESTIONS	c) Double	d) Both a and c.		
		<b>16.2</b> A is a group of related classe	s in the Framework Class Library.		
		a) classpace	b) directory		
		c) namespace	d) library		
		<b>16.3</b> Object variable names should be prefixed	d with		
		a) var	b) obj		
		c) ran	d) ojt		
		16.4 The Next method of class Random can be	called using		
		a) one argument	b) no arguments		
		c) two arguments	d) All of the above.		
		<b>16.5</b> The statement assigns intVa 20.	lue a random number in the range from 5 to		
			<pre>) b) intValue = objRandom.Next(4, 20) ) d) intValue = objRandom.Next(4, 21)</pre>		
$\bigcirc$		16.6 The method specifies the file	from which an image is loaded.		
$\bigcirc$		a) Next in class Random	b) FromFile in class Image		
		c) GetCurrentDirectory in class Directory	d) None of the above.		
		16.7 The System. IO namespace contains clas	ses and methods to		
		a) access files and directories	b) display graphics in an application		
		c) insert multimedia into an application	d) All of the above.		
		<b>16.8</b> The values returned by the m dom numbers.	ethod of class Random are actually pseudo-ran-		
		a) NextRandom	b) PseudoRandom		
		c) Next	d) Pseudo		
		16.9 When creating random numbers, the se	cond argument passed to the Next method is		
		a) equal to the maximum value you wish	to be generated		
		b) equal to one more than the maximum	-		
		c) equal to one less than the maximum value you wish to be generated			
		d) equal to the minimum value you wish	to be generated		
		<b>16.10</b> A(n) is a group of related, r			
		a) namespace	b) variable		
		c) enumeration	d) None of the above.		
		<b>Answers:</b> 16.1) d. 16.2) c. 16.3) b. 16.4) d. 16	.5) a. 16.6) b. 16.7) a. 16.8) c. 16.9) b. 16.10) c.		
$\bigcirc$					
$\checkmark$		1611 (Guass the Number Application) Do	valor on application that concretes a random		

**EXERCISES** 

**16.11** (*Guess the Number Application*) Develop an application that generates a random number and prompts the user to guess the number (Fig. 16.21). When the user clicks the **New Game** Button, the application chooses a number in the range 1 to 100 at random. The user

enters guesses into the **Guess:** TextBox and clicks the **Enter** Button. If the guess is correct, the game ends, and the user can start a new game. If the guess is not correct, the application should indicate if the guess is higher or lower than the correct number.

Guess the Number	🛃 Guess the Number	- 🗆 🗵
I have a number between 1 and 100. Can you guess my number?	I have a number between 1 and 100. Can you guess my number?	
Guess: 26 Enter	Guess: En	ter
Result: New Game	Result: Correct! New (	Same



- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial16\Exercises\GuessNumber directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click GuessNumber.sln in the GuessNumber directory to open the application (Fig. 16.21).
- c) *Creating a Random object.* Create two instance variables. The first variable should store a Random object and the second variable should store a random-generated number.
- d) Adding a Click event handler for the Enter Button. Add a Click event handler for the Enter Button that retrieves the value entered by the user and compares that value to the random-generated number. If the guess is correct, display Correct! in the output Labe1. Then disable the Enter Button, and enable the New Game Button. If the user's guess is higher than the correct answer, display Too high... in the output Labe1. If the user's guess is lower than the correct answer, display Too low... in the output Labe1.
- e) Adding a Click event handler for the New Game Button. Add a Click event handler for the New Game Button that generates a new random number for the instance variable. The event handler should then disable the New Game Button, enable the Enter Button and clear the Result: TextBox.
- f) *Running the application.* Select **Debug > Start** to run your application. Enter guesses (clicking the **Enter** Button after each) until you have successfully determined the answer. Click the **New Game** Button and test the application again.
- g) *Closing the application*. Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 16.11 Solution
 2
     ' GuessNumber.vb
 3
 4
    Public Class FrmGuessNumber
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
       Dim m_objRandom As Random = New Random
10
       Dim m_intNumber As Integer = m_objRandom.Next(1, 101)
11
        ' handles Enter button click event
12
13
       Private Sub btnEnter_Click(ByVal sender As System.Object, _
14
          ByVal e As System.EventArgs) Handles btnEnter.Click
15
16
           ' check answer
17
           If Val(txtGuessNumber.Text) = m_intNumber Then
18
              lblOutput.Text = "Correct!"
19
              btnEnter.Enabled = False
```

```
20
              btnNewGame.Enabled = True
21
           ElseIf Val(txtGuessNumber.Text) > m_intNumber Then
22
              lblOutput.Text = "Too high....'
23
           Else
24
              lblOutput.Text = "Too low..."
25
           End If
26
27
           txtGuessNumber.Clear()
28
           txtGuessNumber.Focus()
29
        End Sub ' btnEnter_Click
30
31
        ' restart game with new number
32
        Private Sub btnNewGame_Click(ByVal sender As System.Object, _
33
           ByVal e As System. EventArgs) Handles btnNewGame. Click
34
35
           ' generate new random number
36
           m_intNumber = m_objRandom.Next(1, 101)
37
           btnEnter.Enabled = True
38
           btnNewGame.Enabled = False
           lblOutput.Text = "" ' clear result
39
40
        End Sub ' btnNewGame_Click
41
42
     End Class ' FrmGuessNumber
```

**16.12** (*Dice Simulator Application*) Develop an application that simulates rolling two sixsided dice. Your application should have a **Roll Button** that, when clicked, displays two dice images corresponding to random numbers. It should also display the number of times each face has appeared. Your application should appear similar to Fig. 16.22.

Dice Sin	nulator		📙 Dice Sim	ulator	<u>-0×</u>
Side 1:	0		Side 1:	18	
Side 2:	0		Side 2:	17	
Side 3:	0		Side 3:	20	
Side 4:	0		Side 4:	21	
Side 5:	0		Side 5:	12	<b>—</b>
Side 6:	0	Roll	Side 6:	16	Roll



- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial16\Exercises\DiceSimulator directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click DiceSimulator.sln in the DiceSimulator directory to open the application.
- c) **Displaying the die image.** Create a Sub procedure named DisplayDie that takes a PictureBox control as an argument. This method should generate a random number to simulate a die roll. Then display the die image in the corresponding PictureBox control on the Form. The die image should correspond to the random number that was generated. To set the image, refer to the code presented in Fig. 16.20.
- d) Adding a Click event handler for the Roll Button. Add a Click event handler for the Roll Button. Call method DisplayDie in this event handler to display the images for both dice.
- e) **Displaying the frequency.** Add a Sub procedure called DisplayFrequency that uses a Select Case statement to update the number of times each face has appeared. Create an enumeration for the dice faces which will be used in the Select Case statement.

- f) Running the application. Select Debug > Start to run your application. Click the Roll Button several times. Each time, two die faces should be displayed at random. Verify after each roll that the appropriate face values on the left are incremented.
- g) *Closing the application.* Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
Answer:
```

```
1
    ' Exercise 16.12 Solution
 2
    ' DiceSimulator.vb
 3
 4
    Imports System.IO
 5
 6
    Public Class FrmDiceSimulator
 7
       Inherits System.Windows.Forms.Form
 8
 9
        ' Windows Form Designer generated code
10
11
        ' die face constants
12
       Enum FaceNames
13
          ONE = 1
14
          TWO = 2
15
          THREE = 3
16
          FOUR = 4
17
          FIVE = 5
18
          SIX = 6
19
       End Enum
20
21
        ' declare Random object reference
22
       Dim m_objRandomNumber As Random = New Random
23
24
        ' display results of roll
25
       Private Sub btnRoll_Click(ByVal sender As System.Object, _
26
          ByVal e As System. EventArgs) Handles btnRoll.Click
27
28
           ' method randomly assigns a face to each die
29
          DisplayDie(picDie1)
30
          DisplayDie(picDie2)
31
       End Sub ' btnRoll_Click
32
33
        ' get a random die image
34
       Sub DisplayDie(ByVal picDie As PictureBox)
35
36
           ' generate random integer in range 1 to 6
37
          Dim intFace As Integer = m_objRandomNumber.Next(1, 7)
38
39
           ' load corresponding image
40
           picDie.Image = Image.FromFile( _
41
              Directory.GetCurrentDirectory & "/Images/die" & _
42
              intFace & ".png")
43
44
           ' method displays rolls frequency
45
          DisplayFrequency(intFace)
46
       End Sub ' DisplayDie
47
48
        ' display the rolls frequency
49
       Sub DisplayFrequency(ByVal intFace As Integer)
50
           ' increment and display frequency values
51
52
           Select Case intFace
53
```

54	Case FaceNames.ONE
55	<pre>lblOutput1.Text = _</pre>
56	<pre>Convert.ToString(Convert.ToInt32(lblOutput1.Text) + 1)</pre>
57	
<b>58</b>	Case FaceNames.TWO
59	<pre>lblOutput2.Text = _</pre>
60	<pre>Convert.ToString(Convert.ToInt32(lblOutput2.Text) + 1)</pre>
61	
62	Case FaceNames.THREE
63	<pre>lblOutput3.Text = _</pre>
64	<pre>Convert.ToString(Convert.ToInt32(lblOutput3.Text) + 1)</pre>
<b>65</b>	
66	Case FaceNames.FOUR
67	<pre>lblOutput4.Text = _</pre>
68	<pre>Convert.ToString(Convert.ToInt32(lblOutput4.Text) + 1)</pre>
69	
70	Case FaceNames.FIVE
71	<pre>lblOutput5.Text = _</pre>
72	<pre>Convert.ToString(Convert.ToInt32(lblOutput5.Text) + 1)</pre>
73	
74	Case FaceNames.SIX
75	<pre>lblOutput6.Text = _</pre>
76	<pre>Convert.ToString(Convert.ToInt32(lblOutput6.Text) + 1)</pre>
77	
78	End Select
79	
80	End Sub ' DisplayFrequency
81	
82	End Class ' FrmDiceSimulator

**16.13** (*Lottery Numbers Picker Application*) A lottery commission offers four different lottery games to play: Three number, Four number, Five number and Five number + 1 lotteries. Each game has independent numbers. Develop an application that randomly picks numbers for all four games and displays the generated numbers in a GUI (Fig. 16.23). The games are played as follows

- Three-number lotteries require players to choose three numbers in the range of 0–9.
- Four-number lotteries require players to choose four numbers, in the range of 0–9.
- Five-number lotteries require players to choose five numbers in the range of 1–39.
- Five-number + 1 lotteries require players to choose five numbers in the range of 1–49 and an additional number in the range of 1–42.

🛃 Lottery Picker		
Three number lottery:	07 05 05	
Four number lottery:	09 07 09 00	
Five number lottery:	21 07 38 07 15	
Five number + 1 lottery:	24 40 26 01 18 34	
	Generate	

Figure 16.23 Lottery Picker application.

a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial16\Exercises\LotteryPicker directory to your C:\SimplyVB directory.

- b) *Opening the application's template file.* Double click LotteryPicker.sln in the LotteryPicker directory to open the application.
- c) *Generating random numbers*. Create a Function procedure that will generate the random numbers for all four games.
- d) *Drawing numbers for the games.* Add code into your application to generate numbers for all four games. To make the applications simple, allow repetition of numbers.
- e) *Running the application.* Select **Debug > Start** to run your application. Click the **Generate** Button multiple times. Make sure the values displayed are within the ranges described in the exercise description.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 16.13 Solution
 2
    ' LotteryPicker.vb
 3
 4
    Public Class FrmLotteryPicker
 5
        Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
       Public m_objRandom As Random = New Random
10
11
        ' display random lottery numbers
12
       Private Sub btnGenerate_Click(ByVal sender As _
13
           System.Object, ByVal e As System.EventArgs) _
14
          Handles btnGenerate.Click
15
16
           ' generate three numbers
           lblOutput3.Text = Generate(0, 10) & " " & _
17
18
              Generate(0, 10) & " " & Generate(0, 10)
19
20
           ' generate four numbers
21
          lblOutput4.Text = Generate(0, 10) & " " & _
              Generate(0, 10) & " " & Generate(0, 10) & " "
22
23
              & Generate(0, 10)
24
25
           ' generate five numbers
26
          lblOutput5.Text = Generate(1, 40) & " " & _
27
              Generate(1, 40) & " " & Generate(1, 40) &
              " " & Generate(1, 40) & " " & Generate(1, 40)
28
29
30
           ' generate five plus one numbers
           lblOutput5Plus1.Text = Generate(1, 50) & " " &
31
              Generate(1, 50) & " " & Generate(1, 50) & " " & _
32
33
              Generate(1, 50) & " " & Generate(1, 50)
34
35
           lblOutputExtra1.Text = Generate(1, 43)
36
37
       End Sub ' btnGenerate Click
38
39
        ' generate random numbers
40
        Function Generate(ByVal intLow As Integer, _
41
          ByVal intHigh As Integer) As String
42
43
          Return String.Format("{0:D2}", _
44
              m_objRandom.Next(intLow, intHigh))
45
46
       End Function ' Generate
```

$\bigcirc$		47 48 End Class ' FrmLotteryPicker Answer:	
	What does this code do?	16.14 What does the following code do?	
		<pre>1 Sub PickRandomNumbers() 2 3 Dim intNumber1 As Integer 4 Dim dblNumber As Double 5 Dim intNumber2 As Integer 6 Dim objRandom As Random = New Random 7 8 intNumber1 = objRandom.Next() 9 dblNumber = 5 * objRandom.NextDouble() 10 intNumber2 = objRandom.Next(1, 10) 11 lblInteger1.Text = Convert.ToString(intNumber1) 12 lblDouble1.Text = Convert.ToString(dblNumber) 13 lblInteger2.Text = Convert.ToString(intNumber2) 14 End Sub ' PickRandomNumbers</pre>	
		<b>Answer:</b> intNumber1 gets a positive integer (between 0 and Int32.MaxValue), dblNumber gets a floating-point number between 0 and 5 (not including 5) and intNumber2 gets an integer between 1 and 10 (not including 10).	
$\bigcirc$	What's wrong with this code? 🕨	<b>16.15</b> This Sub procedure should assign a random Decimal number (in the range 0 to Int32.MaxValue) to Decimal decNumber. (Assume that Option Strict is On.) Find the error(s) in the following code.	
		<pre>1 Sub RandomDecimal() 2 3 Dim decNumber As Decimal 4 Dim objRandom As Random = New Random 5 6 decNumber = objRandom.Next() 7 lblDisplay.Text = Convert.ToString(decNumber) 8 9 End Sub ' RandomDecimal</pre>	
		<b>Answers:</b> Random objects can produce Integers and Doubles only; this will yield only an Integer random number. [There is no way to have the Random class generate a Decimal random value, so the solution is a bit contrived. The closest approximation is to use NextDouble.]	
$\bigcirc$		<pre>1 Sub RandomDecimal() 2 3 Dim decNumber As Decimal 4 Dim objRandom As Random = New Random 5 6 decNumber = Int32.MaxValue * 7 Convert.ToDecimal(objRandom.NextDouble()) 8 lblDisplay.Text = Convert.ToString(decNumber) 9 10 End Sub ' RandomDecimal</pre>	

170

Programming Challenge

**16.16** (*Multiplication Teacher Application*) Develop an application that helps children learn multiplication. Use random-number generation to produce two positive one-digit integers that display in a question, such as "How much is 6 times 7?" The student should type the answer into a TextBox. If the answer is correct, then the application randomly displays one of three messages: **Very Good!**, **Excellent!** or **Great Job!** in a Label and displays the next question. If the student is wrong, the Label displays the message **No. Please try again**.

Multiplication Teacher	Multiplication Teacher	< I
How much is 0 times 3?	How much is 5 times 8?	
Submit Answer	Submit Answer	
Result:	Result: Excellent!	



- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial16\Exercises\MultiplicationTeacher directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click Multiplication-Teacher.sln in the MultiplicationTeacher directory to open the application.
- c) *Generating the questions.* Add a method into your application (Fig. 16.24) to generate each new question.
- d) *Determining whether the right answer was entered*. Add code into your application to call the method created in the previous step. After this method has been called, determine whether the student answered the question correctly, and display the appropriate message.
- e) **Displaying a random message.** Add a procedure GenerateOutput that displays a random message congratulating the student for answering correctly. This method should be called if the student answered the question correctly.
- f) Running the application. Select Debug > Start to run your application. Enter several correct answers and at least one incorrect answer. Verify that No. Please try again is displayed when you are incorrect, and one of the other responses is displayed at random when you are correct.
- g) Closing the application. Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 16.16 Solution
 2
     ' MultiplicationTeacher.vb
 3
 4
    Public Class FrmMultiplicationTeacher
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 0
        ' create new random object
10
       Dim m_objRandomObject As Random = New Random
11
12
        ' random numbers for questions
13
       Dim m_intRandomNumber1 As Integer
14
       Dim m_intRandomNumber2 As Integer
15
       Dim m_strQuestion As String ' String for the question being asked
16
       Dim m_intCorrectAnswer As Integer ' correct answer to question
17
18
        ' handles load event for FrmMultiplicationTeacher
19
       Private Sub FrmMultiplicationTeacher_Load(ByVal sender As _
20
           System.Object, ByVal e As System.EventArgs)
```

```
21
           Handles MyBase.Load
22
23
           GenerateQuestion() ' generate a question
24
        End Sub ' FrmMultiplicationTeacher_Load
25
26
         ' handles click event for btnSubmit Button
27
        Private Sub btnSubmit_Click(ByVal sender As _
28
           System.Object, ByVal e As System.EventArgs) _
29
           Handles btnSubmit.Click
30
31
           ' retrieve user's answer
32
           Dim intAnswer As Integer = Convert.ToInt32(Val(txtAnswer.Text))
33
34
           txtAnswer.Clear() ' clear the TextBox
35
36
           ' check if user answer is correct
37
           If intAnswer = m_intCorrectAnswer Then
38
              GenerateOutput() ' display correct message
39
              GenerateQuestion() ' create another question
40
41
           Else ' answer was wrong, try again
42
              lblResponse.Text = "No. Please try again."
43
           End If
44
45
        End Sub ' btnSubmit_Click
46
47
        ' generates a new question
48
        Sub GenerateQuestion()
49
50
           ' create two random numbers
51
           m_intRandomNumber1 = m_objRandomObject.Next(0, 10)
52
           m_intRandomNumber2 = m_objRandomObject.Next(0, 10)
53
54
           ' record the correct answer
55
           m_intCorrectAnswer = m_intRandomNumber1 * m_intRandomNumber2
56
57
           ' construct the question
58
           m_strQuestion = "How much is " & _
              m_intRandomNumber1.ToString() & " times " & _
59
60
              m_intRandomNumber2.ToString() & "?"
61
62
            ' display the question
63
           lblQuestion.Text = m_strQuestion
64
        End Sub ' GenerateQuestion
65
66
         ' generates correct message
67
        Sub GenerateOutput()
68
           Dim intNumber As Integer = m_objRandomObject.Next(0, 3)
69
70
            ' show random message
71
           Select Case intNumber
72
73
              Case 0
                 lblResponse.Text = "Very Good!"
74
75
76
              Case 1
77
                 lblResponse.Text = "Excellent!"
78
79
              Case 2
                 lblResponse.Text = "Great Job!"
80
81
```

_	9
_	_
	~

1	82 83	End Select
	84 85	End Sub ' GenerateOutput
	86	End Class ' FrmMultiplicationTeacher





# Flag Quiz Application

Introducing One-Dimensional Arrays and ComboBoxes Solutions

)	Instructor's Manual Exercise Solutions Tutorial 17			
-	MULTIPLE-CHOICE	17.1 Arrays can be declared to hold values of		
	QUESTIONS	a) type Double	b) type Integer	
		c) type String	d) any data type	
		<b>17.2</b> The elements of an array are related by the fact that they have the same na		
		a) constant value	b) subscript	
		c) type	d) value	
		17.3 Method returns the largest in	idex in the array.	
		a) GetUpperBound	b) GetUpperLimit	
		c) GetLargestIndex	d) GetUpperSubscript	
		17.4 The first element in every array is the		
		a) subscript	b) zeroth element	
		c) length of the array	d) smallest value in the array	
		17.5 Arrays		
		a) are controls	b) always have one dimension	
		c) keep data in sorted order at all times	d) are objects	
\		<b>17.6</b> The initializer list can		
/		a) be used to determine the size of the array		
		b) contain a comma-separated list of initial values for the array elements		
		c) be empty	d) All of the above.	
		17.7 Which method call sorts array strWords	_	
		a) Array.Sort(strWords)	<ul><li>b) strWords.SortArray()</li><li>d) Sant(strWords)</li></ul>	
		c) Array.Sort(strWords, 1)	d) Sort(strWords)	
		17.8 The ComboBox control combines a TextB		
		<ul><li>a) DateTimePicker</li><li>c) NumericUpDown</li></ul>	b) ListBox d) Label	
		<b>17.9</b> To search for a period (.) in a String ca	,	
		a) String.Search(strTest, ".")	b) String.IndexOf(strTest, ".")	
		<pre>c) strTest.IndexOf(".")</pre>	d) strTest.Search(".")	
		17.10 Property contains the size o	,	
		a) Elements	b) ArraySize	
		c) Length	d) Size	
		<b>Answers:</b> 17.1) d. 17.2) c. 17.3) a. 17.4) b. 17.	5) d. 17.6) d. 17.7) a. 17.8) b. 17.9) c. 17.10) c.	
_				

EXERCISES

**17.11** (Enhanced Flag Quiz Application) Enhance the Flag Quiz application by counting the number of questions that were answered correctly (Fig. 17.31). After all the questions have been answered, display a message in a Label that describes how well the user performed. The following table shows which messages to display:

Number of correct answers	Message
5	Excellent!



🔛 Flag Quiz		_ 🗆 ×
Flag	Select country:	Submit
	United States	Next Flag
	Correct! Done!	Excellent



- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial17\Exercises\FlagQuiz2 directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click FlagQuiz.sln in the FlagQuiz2 directory to open the application.
- c) Adding a variable to count the number of correct answers. Add an instance variable m\_intNumberCorrect, and initialize it to 0. You will use this variable to count the number of correct answers submitted by the user.
- d) *Counting the correct answers.* Increment m\_intNumberCorrect in the **Submit** Button's event handler whenever the submitted answer is correct.
- e) **Displaying the message.** Write a procedure DisplayMessage that displays a message in lblScore depending on the value of m\_intNumberCorrect. Call this procedure from the **Submit** Button's event handler when the quiz is completed.
- f) Running the application. Select Debug > Start to run your application. The finished application should behave as in Fig. 17.31. Run the application a few times, enter a different number of correct answers each time to verify that the correct feedback is displayed.
- g) Closing the application. Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 17.11 Solution
 2
    ' FlagQuiz.vb
 3
 4
    Public Class FrmFlagQuiz
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' String array stores country names
 8
       Dim m_strOptions As String() = New String() { _
 9
           "Russia", "China", "United States", "Italy",
10
           "Australia", "South Africa", "Brazil", "Spain"}
11
12
        ' Boolean array tracks displayed flags
13
       Dim m_blnUsed As Boolean() = _
14
          New Boolean(m_strOptions.GetUpperBound(0)) {}
15
       Dim m_intCount As Integer = 1 ' number of flags shown
16
17
                                       ' current flag's country
       Dim m_strCountry As String
18
19
       Dim m_intNumberCorrect As Integer = 0
20
21
        ' Windows Form Designer generated code
22
23
        ' handles Flag Quiz Form's Load event
```

24

25

26 27

28 29

30

31 32

33

34 35

36

37 38

39 40 41

42

43 44

**45** 

46

47

48 49

50

51 52

53

54

55 56

57

58 59

60

61 62

63

64 65

66

67

68

69 70

71

72 73

74

75 76

77

78 79

80

81 82

83

84

Private Sub FrmFlagQuiz\_Load(ByVal sender As System.Object, \_ ByVal e As System. EventArgs) Handles MyBase. Load Array.Sort(m\_strOptions) ' alphabetize country names ' display country names in ComboBox cboOptions.DataSource = m\_strOptions DisplayFlag() ' display first flag in PictureBox End Sub ' FrmFlagQuiz\_Load ' return full path name of image file as a String Function BuildPathName() As String ' begin with country name Dim strOutput As String = m\_strCountry ' locate space character if there is one Dim intSpace As Integer = strOutput.IndexOf(" ") ' remove space from country name if there is one If intSpace > 0 Then strOutput = strOutput.Remove(intSpace, 1) End If strOutput = strOutput.ToLower() ' make characters lowercase strOutput &= ".png" ' add file extension ' add path name strOutput = strOutput.Insert(0, \_ System.Environment.CurrentDirectory & "\images\") Return strOutput ' return full path name End Function ' BuildPathName ' return an unused random number Function GetUniqueRandomNumber() As Integer Dim objRandom As Random = New Random() Dim intRandom As Integer ' generate random numbers until unused flag is found Do intRandom = objRandom.Next(0, m\_blnUsed.Length) Loop Until m\_blnUsed(intRandom) = False ' indicate that flag has been used m\_blnUsed(intRandom) = True Return intRandom ' return index for new flag End Function ' GetUniqueRandomNumber ' display random flag in PictureBox Sub DisplayFlag() ' unique index ensures that a flag is used no more than once Dim intRandom As Integer = GetUniqueRandomNumber() ' retrieve country name from array m\_strOptions m\_strCountry = m\_strOptions(intRandom)

85	' get image's full path name
86	<pre>Dim strPath As String = BuildPathName()</pre>
87	picFlag.Image = Image.FromFile(strPath) ' display image
88	End Sub ' DisplayFlag
89	
90	' handles Submit Button's Click event
91	<pre>Private Sub btnSubmit_Click(ByVal sender As System.Object, _</pre>
92	ByVal e As System.EventArgs) Handles btnSubmit.Click
93	
94	' retrieve answer from ComboBox
95	Dim strResponse As String = $\_$
96	Convert.ToString(cboOptions.SelectedValue)
97	
98	' verify answer
99	If strResponse = m_strCountry Then
100	lblFeedback.Text = "Correct!"
101	<pre>m_intNumberCorrect += 1 ' update correct answers counter</pre>
101	Else
102	
103	<pre>lblFeedback.Text = "Sorry, incorrect." End If</pre>
104	
	Linform week if with it over
106 107	' inform user if quiz is over
	<pre>If m_intCount &gt;= 5 Then ' quiz is over The function of th</pre>
108	lblFeedback.Text &= " Done!"
109	btnNext.Enabled = False
110	btnSubmit.Enabled = False
111	cboOptions.Enabled = False
112	
113	DisplayScore()
114	Else 'quiz is not over
115	<pre>btnSubmit.Enabled = False</pre>
116	<pre>btnNext.Enabled = True</pre>
117	End If
118	
119	End Sub ' btnSubmit_Click
120	
121	' handles Next Flag Button's Click event
122	<pre>Private Sub btnNext_Click(ByVal sender As System.Object, _</pre>
123	ByVal e As System.EventArgs) Handles btnNext.Click
124	
125	DisplayFlag() ' display next flag
126	<pre>lblFeedback.Text = "" ' clear output</pre>
127	
128	' change selected country to first in ComboBox
129	<pre>cboOptions.SelectedIndex = 0</pre>
130	
131	<pre>m_intCount += 1 ' update number of flags shown</pre>
132	
133	<pre>btnSubmit.Enabled = True</pre>
134	<pre>btnNext.Enabled = False</pre>
135	End Sub ' btnNext_Click
136	
137	' displays message about number of correct answers
138	Sub DisplayScore()
139	
140	Select Case m_intNumberCorrect
141	
142	Case 5
143	<pre>lblScore.Text = "Excellent"</pre>
144	
145	Case 4

146	<pre>lblScore.Text = "Very good"</pre>
147	
148	Case 3
149	<pre>lblScore.Text = "Good"</pre>
150	
151	Case 2
152	<pre>lblScore.Text = "Poor"</pre>
153	
154	Case Else
155	<pre>lblScore.Text = "Fail"</pre>
156	
157	End Select
158	
159	End Sub ' DisplayScore
160	
161	End Class ' FrmFlagQuiz
151 152 153 154 155 156 157 158 159 160	<pre>lblScore.Text = "Poor" Case Else lblScore.Text = "Fail" End Select End Sub ' DisplayScore</pre>

**17.12** (*Salary Survey Application*) Use a one-dimensional array to solve the following problem: A company pays its salespeople on a commission basis. The salespeople receive \$200 per week, plus 9% of their gross sales for that week. For example, a salesperson who grosses \$5000 in sales in a week receives \$200 plus 9% of \$5000, a total of \$650. Write an application (using an array of counters) that determines how many of the salespeople earned salaries in each of the following ranges (assuming that each salesperson's salary is truncated to an integer amount): \$200–\$299, \$300–\$399, \$400–\$499, \$500–\$599, \$600–\$699, \$700–\$799, \$800–\$899, \$900–\$999 and over \$999.

Allow the user to enter the sales for each employee in a TextBox. The user should click the **Calculate** Button to calculate that salesperson's salary. When the user is done entering this information, clicking the **Show Totals** Button should display how many of the salespeople earned salaries in each of the above ranges. The finished application should behave like Fig. 17.32.

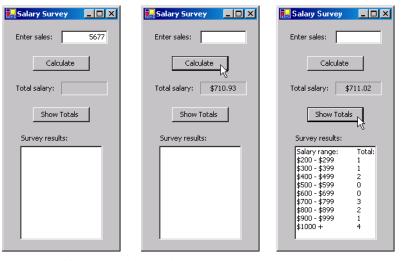


Figure 17.32 Salary Survey GUI.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial17\Exercises\SalarySurvey directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click SalarySurvey.sln in the SalarySurvey directory to open the application.
- c) *Creating an array of salary ranges.* Create a String array, and initialize it to contain the salary ranges (the Strings displayed in the ListBox's first column).
- d) *Create an array that represents the number of salaries in each range.* Create an empty Decimal array to store the number of employees who earn salaries in each range.

- e) Creating an event handler for the Calculate Button. Write event handler btnCalculate\_Click. Obtain the user input from the Enter sales: TextBox. Calculate the commission due to the employee and add that amount to the base salary. Increment the element in array decSalaries that corresponds to the employee's salary range. This event handler should also display the employee's salary: Label.
- f) Writing an event handler for the Show Totals Button. Create event handler btnShowTotals\_Click to display the salary distribution in the ListBox. Use a For...Next statement to display the range (an element in strSalaryRanges) and the number of employees whose salary falls in that range (an element in decSalaries).
- g) Running the application. Select Debug > Start to run your application. Enter several sales amounts using the Calculate Button. Click the Show Totals Button and verify that the proper amounts are displayed for each salary range, based on the salaries calculate from your input.
- h) Closing the application. Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 17.12 Solution
 2
     ' SalarySurvey.vb
 3
 4
     Public Class FrmSalarySurvey
 5
        Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' salary ranges
10
        Dim m_strSalaryRanges As String() = New String() { _
           "$200 - $299", "$300 - $399", "$400 - $499", _
"$500 - $599", "$600 - $699", "$700 - $799", _
11
12
           "$800 - $899", "$900 - $999", "$1000 + "}
13
14
15
        ' number of employees in each salary range
16
        Dim m_intSalaries As Integer() = New Integer( _
17
           m_strSalaryRanges.GetUpperBound(0)) {}
18
19
        ' handles Calculate Button's Click event
20
        Private Sub btnCalculate_Click(ByVal sender As System.Object, _
21
           ByVal e As System. EventArgs) Handles btnCalculate. Click
22
23
           ' obtain total sales
24
           Dim decSales As Decimal = Convert.ToDecimal( _
25
              Val(txtInputSales.Text))
26
27
           ' employee's base salary
28
           Dim decTotalSalary As Decimal = 200
29
30
           ' add commission to total salary
31
           decTotalSalary += Convert.ToDecimal(decSales * 0.09)
32
33
           ' display salary in a Label
34
           lblTotalSalary.Text = String.Format("{0:C}", decTotalSalary)
35
36
           ' increment the correct counter in array intSalaries
37
           Select Case decTotalSalary
38
39
              Case Is < 300
40
                  m_intSalaries(0) += 1
41
```

42

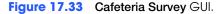
Case Is < 400

43  $m_intSalaries(1) += 1$ 44 45 Case Is < 50046  $m_intSalaries(2) += 1$ 47 48 Case Is < 60049  $m_intSalaries(3) += 1$ 50 51 Case Is < 70052  $m_intSalaries(4) += 1$ 53 54 Case Is < 80055  $m_intSalaries(5) += 1$ 56 57 Case Is < 90058  $m_intSalaries(6) += 1$ 59 60 Case Is < 100061  $m_intSalaries(7) += 1$ 62 63 Case Is >= 1000 64  $m_intSalaries(8) += 1$ 65 66 End Select 67 68 txtInputSales.Clear() ' clear TextBox 69 End Sub ' btnCalculate\_Click 70 71 ' handles click event for btnShowTotals Button 72 Private Sub btnShowTotals\_Click(ByVal sender As System.Object, \_ 73 ByVal e As System. EventArgs) Handles btnShowTotals. Click 74 75 Dim intIndex As Integer = 0 ' counter 76 77 ' clear all items in the ListBox 78 lstSalaryTotals.Items.Clear() 79 80 ' add header to ListBox 81 lstSalaryTotals.Items.Add("Salary range:" & \_ 82 ControlChars.Tab & "Total:") 83 84 ' displays total for each salary range 85 For intIndex = 0 To m\_strSalaryRanges.GetUpperBound(0) 86 lstSalaryTotals.Items.Add(m\_strSalaryRanges(intIndex) & \_ 87 ControlChars.Tab & m\_intSalaries(intIndex)) 88 Next 89 90 End Sub ' btnShowTotals\_Click 91

92 End Class ' FrmSalarySurvey

**17.13** (*Cafeteria Survey Application*) Twenty students were asked to rate, on the scale from 1 to 10, the quality of the food in the student cafeteria, with 1 being "awful" and 10 being "excellent." Allow the user input to be entered using a ComboBox. Place the 20 responses in an Integer array, and determine the frequency of each rating. Display the frequencies as a histogram in a multiline, scrollable TextBox. Figure 17.33 demonstrates the completed application.

Cafeteria Survey	🛃 Cafeteria Survey 📃 🗆 🗙	Cafeteria Survey
Rate cafateria food:	Rate cafateria food: 5	Rate cafateria food: 5
Submit Rating	Submit Rating	Submit Rating
Survey results:	Survey results:	Survey results:
<u> </u>	<u> </u>	Rating Frequency
		2 *
		4 **** 5 ****
v	<b>T</b>	5 <sup>+++</sup> 7 ***
	·	,



- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial17\Exercises\CafeteriaSurvey directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click CafeteriaSurvey.sln in the CafeteriaSurvey directory.
- c) *Creating an array of the possible ratings.* Create an array of 10 consecutive integers, called m\_intChoices to contain the integers in the range 1–10, inclusive.
- d) Adding a ComboBox. Add a ComboBox to the GUI as in Fig. 17.33. The ComboBox will display the possible ratings. Set property DropDownStyle to DropDownList.
- e) *Displaying the possible ratings when the application starts.* Write the event handler for the Load event so that the DataSource of the ComboBox is set to intChoices when the application starts.
- f) *Creating an array to store the responses.* Create an Integer array of length 11 named m\_intResponses. This will be used to store the number of responses in each of the 10 categories (element 0 will not be used).
- g) *Counting the number of responses.* Create an Integer variable named m\_intResponseCounter to keep track of how many responses have been input.
- h) Storing the responses. Write the event handler btnSubmit\_Click to increment m\_intResponseCounter. Store the response in array m\_intResponses. Call procedure DisplayHistogram to display the results.
- i) **Creating procedure DisplayHistogram.** Add a header to the TextBox. Use nested For...Next loops to display the ratings in the first column. The second column uses asterisks to indicate how many students surveyed submitted the corresponding rating.
- j) Running the application. Select Debug > Start to run your application. Enter 20 responses using the Submit Rating Button. Verify that the resulting histogram displays the responses entered.
- k) Closing the application. Close your running application by clicking its close box.
- 1) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
Exercise 17.13 Solution
 1
 2
     ' CafeteriaSurvey.vb
 3
 Δ
    Public Class FrmCafeteriaSurvey
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' possible answers
10
       Dim m_intChoices As Integer() = { _
11
           1, 2, 3, 4, 5, 6, 7, 8, 9, 10
12
```

' counter for number of responses	
Dim m_intResponseCounter As Integer = 0	
<pre>' array to keep track of all responses Dim m_intResponses As Integer() = New Integer(11) {}</pre>	
<pre>' handles Submit Rating Button's Click event Private Sub btnSubmit_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnSubmit.Click</pre>	
<pre>' retrieve user input Dim intResponse As Integer = _ Convert.ToInt32(cboInput.SelectedItem)</pre>	
<pre>If m_intResponseCounter &lt; 20 Then     m_intResponses(intResponse) += 1 End If</pre>	
$m_intResponseCounter += 1$	
<pre>If m_intResponseCounter = 20 Then</pre>	
' disable btnSubmit Button so no more ' responses can be entered btnSubmit.Enabled = False	
DisplayHistogram() End If	
End Sub ' btnSubmit_Click	
' handles Cafeteria Survey Form's Load event Private Sub FrmCafeteriaSurvey_Load(ByVal sender As _ System.Object, ByVal e As System.EventArgs) Handles MyBase.Loa	d
<pre>cboInput.DataSource = m_intChoices End Sub ' FrmCafeteriaSurvey_Load</pre>	
' displays histogram Sub DisplayHistogram()	
' construct output String with the frequencies as a histogram Dim strOutput As String = "Rating" & ControlChars.Tab & _ "Frequency" & ControlChars.CrLf	
Dim intRatings As Integer Dim intCounter As Integer	
' add entry to TextBox for each rating For intRatings = 1 To 10	
<pre>strOutput &amp;= (intRatings.ToString &amp; ControlChars.Tab)</pre>	
<pre>' display asterisk for each user who gave this rating For intCounter = 1 To m_intResponses(intRatings)     strOutput &amp;= "*" Next</pre>	
<pre>strOutput &amp;= ControlChars.CrLf Next</pre>	

74 txtOutput.Text = strOutput display results in TextBox 75 End Sub ' DisplayHistogram 76 77 End Class ' FrmHistogram What does this code do? 17.14 This procedure declares intNumbers as its parameter. What does it return? Function Mystery(ByVal intNumbers As Integer()) As Integer() 1 2 3 Dim intI As Integer 4 Dim intLength As Integer = intNumbers.Length - 1 5 Dim intTempArray As Integer() = \_ 6 New Integer(intLength) {} 7 8 For intI = intLength To 0 Step -19 intTempArray(intLength - intI) = intNumbers(intI) 10 Next 11 12 **Return** intTempArray 13 End Function ' Mystery Answer: This procedure takes the contents of parameter intNumbers and reverses the order of its contents, returning the reversed array to the caller. What's wrong with this code? 17.15 The code that follows uses a For...Next loop to sum the elements in an array. Find the error(s) in the following code: 1 Sub SumArray() 2 3 Dim intSum As Integer 4 Dim intCounter As Integer 5 Dim intNumbers As Integer() = \_ 6 New Integer() {1, 2, 3, 4, 5, 6, 7, 8} 7 8 For intCounter = 0 To intNumbers.Length 9 intSum += intNumbers(intCounter) 10 Next 11 12 End Sub ' SumArray Answer: Array intNumbers does have intNumbers.Length number of elements, but the indices are zero through intNumbers.Length - 1. The For...Next loop increments intCounter beyond the highest index in the array which results in a run-time error. The correct code is as follows: 1 Sub SumArray() 2 Dim intSum As Integer 3 Dim intCounter As Integer 4 Dim intNumbers As Integer() = \_ 5 New Integer() {1, 2, 3, 4, 5, 6, 7, 8} 6 7 For intCounter = 0 To intNumbers.GetUpperBound(0) 8 intSum += intNumbers(intCounter) 9 Next 10 End Sub ' SumArray 11

Programming Challenge

**17.16** (*Road Sign Test Application*) Write an application that will test the user's knowledge of road signs. Your application should display a random sign image and ask the user to select the sign name from a ComboBox. This application should look like Fig. 17.34. *Hint*: The application is similar to the **Flag Quiz** application. You can find the images in C:\Examples\Tutorial17\Exercises\images. Remember to set Option Strict to On.



Figure 17.34 Road Sign Test GUI.

```
1
       Exercise 17.16 Solution
2
     ' RoadSignTest.vb
 3
4
    Public Class FrmRoadSignTest
5
        Inherits System.Windows.Forms.Form
 6
 7
        ' String array stores sign names
8
        Dim m_strOptions As String() = New String() { _
9
           "Do Not Enter", "Narrow bridge", "No bicycles", _
           "No left turn", "No Pedestrians", "No U-turn", _
10
           "Road Narrows", "Stop", "Stop sign ahead", _
"Traffic signals ahead", "Winding road ahead", _
11
12
13
           "Yield"}
14
15
        ' Boolean array tracks displayed signs
16
        Dim m_blnUsed As Boolean() = _
17
           New Boolean(m_strOptions.GetUpperBound(0) {}
18
19
        Dim m_intCount As Integer = 1 ' number of signs shown
20
        Dim m_intCorrectAnswer As Integer ' index of current sign
21
22
        ' Windows Form Designer generated code
23
24
        ' handles Road Sign Test Form's Load event
25
        Private Sub FrmRoadSignTest_Load(ByVal sender As System.Object, _
26
           ByVal e As System. EventArgs) Handles MyBase. Load
27
28
           Array.Sort(m_strOptions) ' alphabetize sign names
29
30
           ' display sign names in ComboBox
31
           cboOptions.DataSource = m_strOptions
32
33
           DisplaySign() ' display first sign in PictureBox
34
        End Sub ' FrmRoadSignTest_Load
35
36
        ' return full path name of image file as a String
37
        Function BuildPathName() As String
38
39
           ' return full path name
40
           Return System.Environment.CurrentDirectory & _
41
              "\images\sign" & m_intCorrectAnswer & ".png"
42
        End Function ' BuildPathName
43
```

44	' return an unused random number
45	Function GetUniqueRandomNumber() As Integer
46	Dim objRandom As Random = New Random()
47	Dim intRandom As Integer
48	
49	' generate random numbers until unused sign is found
50	Do
51	<pre>intRandom = objRandom.Next(0, m_blnUsed.Length)</pre>
52	Loop Until m_blnUsed(intRandom) = False
53	
54	' indicate that sign has been used
55	<pre>m_blnUsed(intRandom) = True</pre>
56	
57	Return intRandom ' return index for new sign
58	End Function ' GetUniqueRandomNumber
59	
60	' display random sign in PictureBox
61	Sub DisplaySign()
62	
63	' unique index ensures that a sign is used no more than once
64	m_intCorrectAnswer = GetUniqueRandomNumber()
65	
66	' get image's full path name
67	Dim strPath As String = BuildPathName()
68	
69	<pre>picSign.Image = Image.FromFile(strPath) ' display image</pre>
70	End Sub ' DisplaySign
71	
72	' handles Submit Button's Click event
73	Private Sub btnSubmit_Click(ByVal sender As System.Object, _
74	ByVal e As System.EventArgs) Handles btnSubmit.Click
75	byvar e As system. EventArgs) handres bensubinte. errek
76	' retrieve answer from ComboBox
70	
	Dim strResponse As String =
78	Convert.ToString(cboOptions.SelectedValue)
79	
80	'verify answer
81	<pre>If strResponse = m_strOptions(m_intCorrectAnswer) Then</pre>
82	<pre>lblFeedback.Text = "Correct!"</pre>
83	Else
84	<pre>lblFeedback.Text = "Incorrect."</pre>
85	End If
86	
87	' inform user if test is over
88	<pre>If m_intCount &gt;= 5 Then ' test is over</pre>
89	<pre>lblFeedback.Text &amp;= " Done!"</pre>
90	<pre>btnNext.Enabled = False</pre>
91	<pre>btnSubmit.Enabled = False</pre>
92	cboOptions.Enabled = False
93	Else 'test is not over
94	btnSubmit.Enabled = False
95	btnNext.Enabled = True
96	End If
97	
98	End Sub ' btnSubmit_Click
90 99	
100	' handles Next Sign Button's Click event
100	
	Private Sub btnNext_Click(ByVal sender As System.Object,
102	ByVal e As System.EventArgs) Handles btnNext.Click
103	
104	DisplaySign() ' display next sign

105	lblFeedback.Text = ""
106	
107	' change selected sign to first in ComboBox
108	<pre>cboOptions.SelectedIndex = 0</pre>
109	
110	<pre>m_intCount += 1 ' update number of signs shown</pre>
111	
112	<pre>btnSubmit.Enabled = True</pre>
113	<pre>btnNext.Enabled = False</pre>
114	End Sub ' btnNext_Click
115	
116	End Class ' FrmRoadSignTest





# **Sales Data Application**

Introducing Two-Dimensional Arrays, RadioButtons and the MSChart Control Solutions

)	Instructor's Manual Exercise Solutions Tutorial 18		
-	MULTIPLE-CHOICE	18.1 RadioButton controls should be prefixe	d with
	QUESTIONS	a) rad c) btn	b) rbn d) radbtn
		<b>18.2</b> A two-dimensional array in which each called a array.	row contains the same number of columns is
		a) data c) tabular	<ul><li>b) rectangular</li><li>d) All of the above.</li></ul>
		,	
		<ul><li><b>18.3</b> In an <i>m</i>-by-<i>n</i> array, the <i>m</i> stands for</li><li>a) the number of columns in the array</li></ul>	b) the total number of array elements
		c) the number of rows in the array	d) the number of elements in each row
		<b>18.4</b> The statement assigns an a dimensional Integer array intArray.	array of three columns and five rows to two-
		a) intArray = New Integer(5, 3)	<pre>b) intArray = New Integer(4, 2)</pre>
		<pre>c) intArray = New Integer(4, 3)</pre>	d) intArray = New Integer $(5, 2)$
		<b>18.5</b> To change the MSChart graph's title size <b>erties</b> dialog.	e, use the tab of the MSChart <b>Prop</b> -
		a) Size	b) Title
)		c) Fonts	d) Font Size
		<b>18.6</b> Use a to group RadioButton	ns on the Form.
		a) GroupBox control	b) ComboBox control
		c) ListBox control	d) None of the above.
		<b>18.7</b> Use the tab of the MSCha include a border around your chart.	rt Properties dialog to access properties to
		a) Border	b) Backdrop
		c) BorderStyle	d) Background
		<b>18.8</b> A point of height is approximately equa	l to
		a) 1/72"	b) 1"
		c) 1/4"	d) 1/36"
		<b>18.9</b> The <b>Chart Type</b> GroupBox of the MSCha tab.	rt control's <b>Properties</b> dialog is located in the
		a) Chart	b) Graph
		c) Series	d) ChartType
		<b>18.10</b> The <b>Fonts</b> tab of the MSChart contro	's Properties dialog allows you to change the
		a) <b>Style</b>	b) Width
		c) Color	d) All of the above.
		<b>Answers:</b> 18.1) a. 18.2) b. 18.3) c. 18.4) b. 18	6.5) c. 18.6) a. 18.7) b. 18.8) a. 18.9) a. 18.10) d.

**EXERCISES** 

**18.11** (*Stock Price Application*) It is often useful to track a company's stock price over time by using a line graph. You will learn to create a line graph by using an MSChart control in this exercise. Create an application that allows users to enter values for a company's stock

price at the end of six consecutive quarters and graph that data, using an MSChart control (Fig. 18.23).

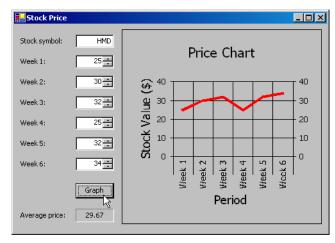


Figure 18.23 Stock Price application GUI.

- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial18\Exercises\StockPrice directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click StockPrice.sln in the StockPrice directory to open the application.
- c) Changing the graph type and color. To change the graph type and color, click the chart in Design View and select the ActiveX -Properties hyperlink from the Properties window to display the Properties dialog. In the Chart tab's Chart Type Group-Box, select Line (if it is not already selected), and make sure the RadioButton is set to 2D. Click the Series Color tab, and, in the Edge/Line GroupBox, change the color to red (if this color is not already selected).
- d) *Inserting code in the Graph Button's Click event handler*. Double click the Graph Button in **Design** view to generate the Graph Button Click event handler. Insert an If...Then statement that verifies that the user entered a stock name. If the value in the TextBox is the empty string, the application should display a message dialog.
- e) **Checking the remaining NumericUpDowns.** For the chart to function properly, the user must enter values in all NumericUpDowns. Add If...ElseIf...Else statements to verify that there is a value greater than zero in each NumericUpDown (using the Value property), and display an error message in a MessageBox if any values are missing (contain the zero).
- f) Calculating the average. Calculate the average of all of the values from the NumericUpDowns, and output the result in the output Label lblAverage.
- g) **Inserting data into an array.** Create a 6-by-2 array from the data that will display the week number on the *x*-axis and prices on the *y*-axis. (The week number should be in the first column, and the stock price should be in the second column.) Use the NumericUpDown's Text property to retrieve the value stored in the control as a String.
- h) **Displaying the chart**. Assign the array to the ChartData property to display the data in the array. Remember to set Label lblMessage's Visible property to False and set the MSChart control's Visible property to True. A control's **Visible** property determines if the control is displayed on the Form (True) or hidden (False).
- i) **Running the application.** Select **Debug > Start** to run your application. Enter the name of a stock, and the stock's price for each week. Click the **Graph** Button. Verify that the average displayed is correct, and that the graph displayed shows the proper data. Also, make sure your graph's labels appear as in Fig. 18.23.
- j) *Closing the application.* Close your running application by clicking its close box.
- k) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 18.11 Solution
 2
     ' StockPrice.vb
 3
 4
    Public Class FrmStockPrice
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        Private Sub btnGraph_Click(ByVal sender As
10
          System.Object, ByVal e As System.EventArgs) _
11
          Handles btnGraph.Click
12
13
           ' declaring two-dimensional array
14
          Dim strPrice(,) As String = New String(5, 1) {}
15
          Dim dblAverage As Double = 0 ' average stock price
16
          Dim intCounter As Integer = 0 ' missing stock price counter
17
           ' if stock name is missing
18
19
          If txtStockName.Text = "" Then
20
              MessageBox.Show("Stock name is required", _
21
                 "Stock Name Missing", MessageBoxButtons.OK, _
22
                 MessageBoxIcon.Error)
23
24
           ' if stock prices were missing
25
          ElseIf updFirst.Value = 0 OrElse updSecond.Value = 0
26
              OrElse updThird.Value = 0 OrElse updFourth.Value = 0
27
              OrElse updFifth.Value = 0 OrElse updSixth.Value = 0 Then
28
29
              MessageBox.Show("Please fill in all weekly values", _
30
                 "Missing Price", MessageBoxButtons.OK, _
31
                 MessageBoxIcon.Exclamation)
32
          Else
33
              lblMessage.Visible = False
34
              chStock.Visible = True
35
36
              ' calculate average
37
              dblAverage = (updFirst.Value + updSecond.Value + _
38
                 updThird.Value + updFourth.Value + _
39
                 updFifth.Value + updSixth.Value) / 6
40
41
              ' output average price
42
              lblAverage.Text = String.Format("{0:F}", dblAverage)
43
44
              ' put all data in a two-dimensional array
45
              strPrice = New String(,) {{"Week 1", txtFirst.Text}, _
46
                 {"Week 2", txtSecond.Text}, _
47
                 {"Week 3", txtThird.Text}, _
48
                 {"Week 4", txtFourth.Text}, _
49
                 {"Week 5", txtFifth.Text}, _
50
                 {"Week 6", txtSixth.Text}}
51
52
              ' bind data to chart
53
              chStock.ChartData = strPrice
54
55
          End If
56
57
        End Sub ' btnGraph_Click
```

## 59 End Class ' FrmStockPrice

58

**18.12** *(Enhanced Lottery Picker)* A lottery commission offers four different lottery games to play: three-number, four-number, five-number and five-number + 1 lotteries. In Tutorial 16, your Lottery Picker application could select duplicate numbers for each lottery. In this exercise, you enhance the Lottery Picker to prevent duplicate numbers for the five-number and five-number + 1 lotteries (Fig. 18.24). According to this new requirement the games are now played as follows:

- Three-number lotteries require players to choose three numbers in the range of 0–9.
- Four-number lotteries require players to choose four numbers, in the range of 0–9.
- Five-number lotteries require players to choose five unique numbers in the range of 1–39.
- Five-number + 1 lotteries require players to choose five unique numbers in the range of 1–49 and an additional unique number in the range of 1–42.

🛃 Lottery Picker	_ 🗆 ×
Three number lottery:	08 08 05
Four number lottery:	09 03 01 01
Five number lottery:	23 18 05 01 24
Five number + 1 lottery:	16 08 03 39 22 41
	Generate

Figure 18.24 Enhanced Lottery Numbers Picker application.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial18\Exercises\EnhancedLotteryPicker directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click EnhancedLotteryPicker.sln in the EnhancedLotteryPicker directory to open the application.
- c) **Declaring a two-dimensional array to maintain unique random numbers.** Declare an instance variable m\_blnNumbers that stores a 2-by-50 Boolean array. You will use this array later in this exercise to test whether a lottery number has already been chosen.
- d) *Initializing the array*. Each time the user clicks the **Generate** Button, the application should initialize the array by declaring its rows and setting the initial values. Write a ClearArray procedure that uses a For...Next statement to assign each value in the m\_blnNumbers array to False.
- e) Modifying the Generate Function procedure. You will modify the Generate Function procedure to use the Boolean array to pick unique random numbers. Begin by writing a statement that generates a random number and assigns its value to an Integer variable intNumber.
- f) **Determining whether the random number has already been selected**. Use an If...Then statement to determine whether the maximum lottery number is less than 40. (This happens when the upper limit on the random number equals 40.) In this case, you will examine the first row of the array. To maintain unique numbers, you will set the value of the element in that row whose index equals the random number to True (indicating that it has been picked). For example, if the random number 34 has been picked, blnNumbers(0)(34) would contain the value True. To test whether a number has been picked, use a Do While...Loop statement inside the If...Then statement to access that element of the array. If the array element's value is True, use the body of the loop to assign a new random number to intNumber. If the value in the array is False, use the condition in the Do While...Loop header to ignore the

body of the loop. Just outside the Do While...Loop, include a statement that modifies the array to indicate that the number has now been picked.

- g) **Completing the application**. Use a second If...Then statement to determine whether the maximum lottery number is greater than 40. In this case, you will examine the second row of the array. Repeat the process in the previous step. Remember to return the value stored in intNumber at the end of the Generate Function procedure.
- h) *Running the application.* Select **Debug > Start** to run your application. Click the **Generate** Button and verify that the values displayed fall into the ranges specified in the exercise's description.
- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
    ' Exercise 18.12 Solution
2
    ' LotteryPicker.vb
 3
 4
    Public Class FrmLotteryPicker
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
8
9
       Dim m_objRandom As Random = New Random ' create Random object
10
11
        ' declare new array to store chosen lottery numbers
12
       Dim m_blnNumbers As Boolean(,) = New Boolean(1, 49) {}
13
14
        ' display random lottery numbers
15
       Private Sub btnGenerate_Click(ByVal sender As _
          System.Object, ByVal e As System.EventArgs) _
16
17
          Handles btnGenerate.Click
18
19
          ClearArray() ' clear array values
20
21
           ' generate three numbers
22
          lblOutput3.Text = Generate(0, 10) & " " & _
23
             Generate(0, 10) & " " & Generate(0, 10)
24
25
           ' generate four numbers
26
          lblOutput4.Text = Generate(0, 10) & " " & _
27
             Generate(0, 10) & " " & Generate(0, 10) & " " _
28
             & Generate(0, 10)
29
30
           ' generate five numbers
31
          lblOutput5.Text = Generate(1, 40) & " " & _
32
             Generate(1, 40) & " " & Generate(1, 40) &
33
              " " & Generate(1, 40) & " " & Generate(1, 40)
34
35
           ' generate five plus one numbers
36
          lblOutput5Plus1.Text = Generate(1, 50) & " " &
             Generate(1, 50) & " " & Generate(1, 50) & " " & _
37
38
             Generate(1, 50) & " " & Generate(1, 50)
39
40
          lblOutputExtra1.Text = Generate(1, 43) ' generate extra numbers
41
       End Sub ' btnGenerate_Click
42
43
        ' generate random numbers
44
       Function Generate(ByVal intLow As Integer, _
45
          ByVal intHigh As Integer) As String
```

```
46
47
           ' generate random number with given boundaries
           Dim intNumber As Integer = _
48
49
              m_objRandom.Next(intLow, intHigh)
50
51
           ' use first row for five-number lottery
52
           If intHigh = 40 Then
53
54
              ' select new random number
              Do While m_blnNumbers(0, intNumber) = True
55
56
                 intNumber = m_objRandom.Next(intLow, intHigh)
57
              Loop
58
59
              m_blnNumbers(0, intNumber) = True ' mark number as used
60
           End If
61
62
           ' use second row for five-number + 1 lottery
63
           If intHigh > 40 Then
64
65
              ' select another random number that is not used
66
              Do While m_blnNumbers(1, intNumber) = True
67
                 intNumber = m_objRandom.Next(intLow, intHigh)
68
              Loop
69
70
              m_blnNumbers(1, intNumber) = True ' mark number as used
71
           End If
72
73
           Return String.Format("{0:D2}", intNumber)
74
        End Function ' Generate
75
76
        ' assigns all values in array to False
77
        Sub ClearArray()
78
79
           Dim intI As Integer = 0
80
81
           ' assign all row 0 cells False
82
           For intI = 0 To 39
83
              m_blnNumbers(0, intI) = False
84
           Next
85
86
           ' assign all row 1 cells False
87
           For intI = 0 To 49
88
              m_blnNumbers(1, intI) = False
89
           Next
90
91
        End Sub ' ClearArray
92
93
     End Class ' FrmLotteryPicker
```

**18.13** (Student Grades Application) A teacher needs an application that computes each student's grade average (on a scale of 0 to 100 points) and the class average for ten students. The application should add a student's name and test average (separated by a tab character) to a ListBox and calculate the class grade average each time the user clicks the Submit Grades Button (Fig. 18.25). The Submit Grades Button should be disabled after ten students' grades have been entered.

🛃 Student Grades				
Input Grade			98.33	
Student name:	Mike	Mike 8	39.33	
Test 1:	89			
Test 2:	85			
Test 3:	94			
	Submit Grades	Class aver	age:	93.83

Figure 18.25 Student Grades application.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial18\Exercises\StudentGrades directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click StudentGrades.sln in the StudentGrades directory to open the application.
- c) *Declare instance variables.* Declare an Integer counter and a 10-by-2 String array as instance variables.
- d) Coding the Submit Grades Button's Click event handler. Double click the Submit Grades Button to generate its Click event handler. Write code in the event handler to retrieve input from the TextBoxes. Then store the student's name in the first column of the two-dimensional String array and the student's test average in the second column of the array. Use a Function procedure to calculate the student's test average.
- e) **Computing the class average**. Add the student's name and the student's test average (separated by a tab character) to the ListBox. Then calculate and display the class average, using a Function procedure. [*Hint*: You should use the two-dimensional String array and the Integer counter to calculate the class average.]
- f) *Completing the event handler*. Increment the counter by one after calculating the class average. If ten students' grades have been entered, disable the **Submit Grades** Button.
- g) Running the application. Select Debug > Start to run your application. Enter 10 students and their grades. Make sure that all averages are correct, and that the Submit Grades Button is disabled after 10 students' grades have been entered.
- h) *Closing the application.* Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 18.13 Solution
 2
     ' StudentGrades.vb
 3
 4
    Public Class FrmStudentGrades
 5
        Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' array of students and test averages
10
        Dim m_strStudents As String(,) = New String(9, 1) {}
11
12
        ' counter for number of grades in array
13
        Dim m_intCounter As Integer = 0
14
15
        ' handles click event for btnSubmit Button
16
        Private Sub btnSubmit_Click(ByVal sender As System.Object, _
17
           ByVal e As System. EventArgs) Handles btnSubmit. Click
18
```

```
19
            extract user input
20
           Dim dblTest1 As Double = Val(txtTest1.Text)
21
           Dim dblTest2 As Double = Val(txtTest2.Text)
22
           Dim dblTest3 As Double = Val(txtTest3.Text)
23
           ' add student name and test average
24
25
           ' to the array of students
26
           m_strStudents(m_intCounter, 0) = txtName.Text
27
           m_strStudents(m_intCounter, 1) = _
28
              String.Format("{0:F}", _
29
                 StudentAverage(dblTest1, dblTest2, dblTest3))
30
31
           ' display student name and average in ListBox
32
           lstNames.Items.Add(m_strStudents(m_intCounter, 0) &
33
              ControlChars.Tab & m_strStudents(m_intCounter, 1))
34
35
           ' display class average
           lblClassOutput.Text = _
36
37
              String.Format("{0:F}", ClassAverage())
38
39
           ' increment grade counter
40
           m_intCounter += 1
41
42
           ' maximum of ten students
43
           If m_intCounter = 10 Then
ΔΔ
              btnSubmit.Enabled = False
45
           End If
46
47
        End Sub ' btnSubmit Click
48
49
        ' returns student average
50
        Function StudentAverage(ByVal intTest1 As Double, _
51
           ByVal intTest2 As Double, ByVal intTest3 As Double) _
52
           As Double
53
54
           ' return the average of 3 test scores
55
           Return ((intTest1 + intTest2 + intTest3) / 3)
56
        End Function ' StudentAverage
57
58
        ' returns class average
59
        Function ClassAverage() As Double
60
           Dim intCounter As Integer = 0 ' counter variable
           Dim dblTotal As Double = 0.0 ' total test score
61
62
63
           ' iterate through array of students
64
           For intCounter = 0 To m_intCounter
65
              dblTotal += _
66
                 Convert.ToDouble(m_strStudents(intCounter, 1))
67
           Next
68
69
           ' return the class average
70
           Return (dblTotal / (m_intCounter + 1))
71
        End Function ' ClassAverage
72
73
     End Class ' FrmStudentGrades
```

What does this code do?

**18.14** What is returned by the following code? Assume that GetStockPrices is a Function procedure that returns a 2-by-31 array, with the first row containing the stock price at the

beginning of the day and the last row containing the stock price at the end of the day, for each day of the month.

```
1
    Function Mystery() As Integer()
 2
       Dim intPrices As Integer(,) = New Integer(1, 30) {}
 3
 4
       intPrices = GetStockPrices()
 5
 6
       Dim intResult As Integer() = New Integer(30) {}
 7
       Dim intI As Integer
 8
 9
       For intI = 0 To 30
10
           intResult(intI) = intPrices(0, intI) - intPrices(1, intI)
11
       Next
12
13
       Return intResult
14
    End Function ' Mystery
```

**Answer:** The Function procedure returns a one-dimensional array containing the daily stock price change for each day of the month.

What's wrong with this code?

**18.15** Find the error(s) in the following code. The TwoDArrays procedure should create a two-dimensional array and initialize all its values to one.

```
1
    Sub TwoDArrays()
2
        Dim intArray As Integer(,)
 3
 4
        intArray = New Integer(3, 3) {}
 5
 6
        Dim intI As Integer
 7
 8
        ' assign 1 to all cell values
9
        For intI = 0 To 3
10
           intArray(intI, intI) = 1
11
        Next
12
13
    End Sub ' TwoDArrays
```

**Answer:** To assign each element in a two-dimensional array two nested For...Next loops should be used. The corrected code is as follows:

```
1
    Sub TwoDArrays()
 2
       Dim intArray As Integer(,)
 3
        intArray = New Integer(3, 3) {}
        Dim intI As Integer
 4
 5
       Dim intJ As Integer
 6
 7
        ' assign 1 to all cell values
 8
        For intI = 0 To 3
 9
10
           For int J = 0 To 3
11
              intArray(intI, intJ) = 1
12
           Next
13
14
        Next
15
16
    End Sub ' TwoDArrays
```

Programming Challenge

**18.16** (*Enhanced Student Grades Application*) Modify the application in Exercise 18.13 to include a bar graph that displays each student's grade and the class average. The chart should display only after the user enters the grades for ten students (Fig. 18.26). Until then, a Label should display the text, "Enter ten grades to display the chart." Student names and the text Class Average should display on the x-axis, and grades should display on the y-axis. Also write code that ensures that the user has entered values in each TextBox when the Submit Grades Button is clicked.

📙 Student Grades		_ 🗆 🗵
Input Grade		
Student name:		
Test 1:		
Test 2:		
Test 3:		
	Submit Grades Class average:	
	Enter ten grades to display chart.	

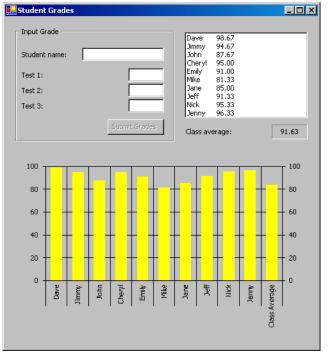


Figure 18.26 Enhanced Student Grades application.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial18\Exercises\EnhancedStudentGrades directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click EnhancedStudent-Grades.sln in the EnhancedStudentGrades directory to open the application. If you have not completed Exercise 18.13, follow the steps in Exercise 18.13 before proceeding to the next step. If you have completed Exercise 18.13, copy the code you wrote into this application.
- c) Completing the application design. Place a Label on the Form that reads "Enter ten grades to display the chart." Then insert an MSChart control at the bottom of the Form. Set its Visible property to False to hide the graph initially. A control's Visible property determines if the control is displayed on the Form (True) or hidden (False). Open the MSChart control's Properties dialog, set the Chart Type to a 2D Bar/PictoGraph and set the Series Interior Color property (under the Series Color tab) to yellow.
- d) *Inserting code to check input*. Write code that determines whether the user has entered values in each TextBox. If the student name is missing, display a MessageBox indicating that the user must enter a student name. If any grades are missing, display a MessageBox indicating that three grades are required.
- e) *Enhancing the user interface.* Write code to clear each TextBox after the user clicks the **Submit Grades** Button. Then insert code to set the focus of the application to the **Student Name:** TextBox.
- f) Displaying the chart. Write a procedure that displays each of the ten students' test averages and the class average in the MSChart control. Recall that student names and the text Class Average should display on the x-axis, and grades should display on the y-axis. Finally, set the chart's Visible property to True, and set the Visible property of the Label you added in Step b to False.
- g) Running the application. Select Debug > Start to run your application. Enter 10 sets of grades. Verify that the resulting chart displays the proper data, and that the chart is formatted as in Fig. 18.26.
- h) Closing the application. Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 18.16 Solution
2	' StudentGrades.vb
3	
4	Public Class FrmStudentGrades
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' array of students and test averages
10	<pre>Dim m_strStudents As String(,) = New String(10, 1) {}</pre>
11	
12	' counter for number of grades in array
13	Dim m_intCounter As Integer = 0
14	
15	' handles click event for btnSubmit Button
16	<pre>Private Sub btnSubmit_Click(ByVal sender As System.Object, _</pre>
17	ByVal e As System.EventArgs) Handles btnSubmit.Click
18	
19	' if student name is missing
20	<pre>If txtName.Text = "" Then</pre>
21	<pre>MessageBox.Show("Student Name is required", _</pre>
22	"Student Name Missing", MessageBoxButtons.OK, _
23	MessageBoxIcon.Exclamation)
24	

```
25
                if a test grade is missing
           ElseIf txtTest1.Text = ""
26
              OrElse txtTest2.Text = ""
27
              OrElse txtTest3.Text = "" Then
28
29
30
              MessageBox.Show("Test grade is missing", _
31
                 "Grade Missing", MessageBoxButtons.OK, _
32
                 MessageBoxIcon.Exclamation)
33
34
              ' if all the conditions are satisfied
35
              ' then perform calculations and display chart
36
           Else
37
38
              ' extract user input
39
              Dim dblTest1 As Double = Val(txtTest1.Text)
40
              Dim dblTest2 As Double = Val(txtTest2.Text)
41
              Dim dblTest3 As Double = Val(txtTest3.Text)
42
43
               ' add student name and test average
44
              ' to the array of students
45
              m_strStudents(m_intCounter, 0) = txtName.Text
46
              m_strStudents(m_intCounter, 1) = _
47
                 String.Format("{0:F}", _
48
                    StudentAverage(dblTest1, dblTest2, dblTest3))
49
50
              ' display student name and average in ListBox
51
              lstNames.Items.Add(m_strStudents(m_intCounter, 0) & _
52
                 ControlChars.Tab & m_strStudents(m_intCounter, 1))
53
54
               ' display class average
55
              lblClassOutput.Text = _
56
                 String.Format("{0:F}", ClassAverage())
57
58
              ' increment grade counter
59
              m_intCounter += 1
60
61
               ' maximum of ten students
62
              If m_intCounter = 10 Then
63
                 btnSubmit.Enabled = False
64
                 ShowChart()
65
              End If
66
67
              ' clear TextBoxes and set focus
68
              txtName.Clear()
69
              txtTest1.Clear()
70
              txtTest2.Clear()
71
              txtTest3.Clear()
72
              txtName.Focus()
73
           End If
74
75
        End Sub ' btnSubmit_Click
76
77
        ' returns student average
78
        Function StudentAverage(ByVal intTest1 As Double, _
79
           ByVal intTest2 As Double, ByVal intTest3 As Double) _
80
           As Double
81
82
           Return ((intTest1 + intTest2 + intTest3) / 3)
83
        End Function ' StudentAverage
84
85
        ' returns class average
```

86	Function ClassAverage() As Double
87	Dim intCounter As Integer = 0 ' counter variable
88	Dim dblTotal As Double = $0.0$ ' store total test scores
89	
90	' iterate through array of students
91	<pre>For intCounter = 0 To m_intCounter</pre>
92	dblTotal += _
93	<pre>Convert.ToDouble(m_strStudents(intCounter, 1))</pre>
94	Next
<b>95</b>	
96	' return average test score for the class
97	Return (dblTotal / (m_intCounter + 1))
<b>98</b>	End Function ' ClassAverage
99	
100	' places data in Chart and displays it
101	Sub ShowChart()
102	lblChart.Visible = False ' hide the Label
103	chGrades.Visible = True ' display the grade chart
104	<pre>m_strStudents(10, 0) = "Class Average"</pre>
105	<pre>m_strStudents(10, 1) = String.Format("{0:F}", ClassAverage())</pre>
106	chGrades.ChartData = m_strStudents
107	End Sub ' ShowChart
108	
109	End Class ' FrmStudentGrades





## Microwave Oven Application

Building Your Own Classes and Objects Solutions Instructor's Manual **Exercise Solutions Tutorial 19 MULTIPLE-CHOICE 19.1** A Button appears flat if its \_\_\_\_\_ property is set to Flat. a) BorderStyle b) FlatStyle QUESTIONS c) Style d) BackStyle **19.2** Keyword \_\_\_\_\_\_ introduces a class definition. a) NewClass b) ClassDef c) VBClass d) Class 19.3 Keyword \_\_\_\_\_\_ is used to create an object. a) CreateObject b) Instantiate c) Create d) New **19.4** String characters are of data type \_\_\_\_\_ a) Char b) StringCharacter c) Character d) strCharacter **19.5** The \_\_\_\_\_\_ is used to retrieve the value of an instance variable. a) Get accessor of a property b) Retrieve method of a class c) Client method of a class d) Set accessor of a property 19.6 When you enter the header for a constructor in Visual Studio .NET then press Enter, the keywords \_\_\_\_\_\_ are created for you. a) End Public Class b) End Procedure c) End Sub d) End **19.7** An important difference between constructors and other methods is that \_\_\_\_\_ a) constructors cannot specify a return data type b) constructors cannot specify any parameters c) other methods are implemented as Sub procedures d) constructors can assign values to instance variables 19.8 A class can yield many \_\_\_\_\_, just as a primitive data type can yield many variables. b) objects a) names c) values d) types 19.9 The Set accessor enables you to \_\_\_\_\_ b) modify data a) provide range checking d) All of the above. c) provide data validation 19.10 Instance variables declared Private are not accessible \_\_\_\_ a) outside the class b) by other methods of the same class c) by other members of the same class d) inside the same class Answers: 19.1) b. 19.2) d. 19.3) d. 19.4) a. 19.5) a. 19.6) c. 19.7) a. 19.8) b. 19.9) d. 19.10) a.

EXERCISES

**19.11** (*Triangle Creator Application*) Create an application that allows the user to enter the lengths for the three sides of a triangle as Integers. The application should then determine whether the triangle is a right triangle (two sides of the triangle form a 90 degree angle), an equilateral triangle (all sides of equal length) or neither. The application's GUI is completed for you (Fig. 19.49). You must create a class to represent a triangle object and define the event handler for the **Create** Button.

Triangle Creator	Triangle Creator
Side1: 3 Create	Side1: 3 Create
Side2: 4	Side2: 3
Side3: 5	Side3: 3
You created a right triangle!	You created an equilateral triangle!
🔚 Triangle Creator	
Side1: 3 Create	
Side2: 3	
Side3: 5	
51065.	

Figure 19.49 Triangle Creator application with all possible outputs

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial19\Exercises\Triangle directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click Triangle.sln in the Triangle directory to open the application.
- c) *Creating the Triangle class.* Add a class to the project, and name it Triangle. This is where you will define the properties of the Triangle class.
- d) Defining the necessary properties. Define a constructor that will take the lengths of the three sides of the triangle as arguments. Create three properties that enable clients to access and modify the lengths of the three sides. If the user enters a negative value, that side should be assigned the value zero.
- e) Adding additional features. Create two more properties in the Triangle class: One determines whether the sides form a right triangle, the other an equilateral triangle. These properties are considered read-only, because you would naturally define only the Get accessor. There is no simple Set accessor that can make a triangle a right triangle or an equilateral triangle without first modifying the lengths of the triangle's sides. To create a read-only property (where the Set accessor is omitted), precede keyword Property with the keyword ReadOnly.
- f) Adding code to event handler. Now that you have created your Triangle class, you can use it to create objects in your application. Double click the Create Button in Design View to generate the event handler. Create new variables to store the three lengths from the TextBoxes; then, use those values to create a new Triangle object.
- g) *Displaying the result*. Use an If...ElseIf statement to determine if the triangle is a right triangle, an equilateral triangle or neither. Display the result in a Label.
- h) Running the application. Select Debug > Start to run your application. Create various inputs until you have create an equilateral triangle, a right triangle and a triangle that is neither right nor equilateral. Verify that the proper output is displayed for each.
- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1 'Exercise 19.11 Solution
2 'Triangle.vb
3 'Represent a triangle
4
5 Public Class Triangle
6
7 'declare three private side values
```

```
204
```

```
8
       Private m_intSide1 As Integer
9
       Private m_intSide2 As Integer
10
       Private m_intSide3 As Integer
11
12
        ' Triangle constructor (side1, side2 and side3)
13
       Public Sub New(ByVal side1Value As Integer, _
14
          ByVal side2Value As Integer, _
15
          ByVal side3Value As Integer)
16
17
          Side1 = side1Value
18
          Side2 = side2Value
19
          Side3 = side3Value
       End Sub ' New
20
21
        ' get and set Side1
22
23
       Public Property Side1() As Integer
24
25
           ' return m_intSide1 value
26
          Get
27
             Return m_intSide1 ' return length of side1
28
           End Get ' end of Get accessor
29
30
           ' set side value
31
          Set(ByVal Value As Integer)
32
33
              ' make sure value is non-negative
34
             If Value > 0 Then
35
                 m_intSide1 = Value
36
             Else
37
                 m_intSide1 = 0 ' set to zero
             End If
38
39
40
           End Set ' end of Set accessor
41
42
       End Property ' Side1
43
44
        ' get and set Side2
45
       Public Property Side2() As Integer
46
47
           ' return m_intSide2 value
48
          Get
49
             Return m_intSide2 ' return length of side2
          End Get ' end of Get accessor
50
51
52
           ' set m_intSide2 value
53
          Set(ByVal Value As Integer)
54
55
              ' check if value is negative
56
             If Value > 0 Then
57
                 m_intSide2 = Value ' set value
58
             Else
59
                 m_intSide2 = 0
60
             End If
61
62
           End Set ' end of Set accessor
63
64
       End Property ' Side2
65
66
        ' get and set Side3
67
       Public Property Side3() As Integer
68
```

```
69
             return m_intSide3 value
70
           Get
71
              Return m_intSide3 ' return length of side3
72
           End Get ' end of Get accessor
73
74
            ' set m_intSide3
75
           Set(ByVal Value As Integer)
76
77
               ' make sure value is nonnegative
78
              If Value > 0 Then
79
                 m_intSide3 = Value ' set value
80
              Flse
81
                 m_intSide3 = 0 ' set to zero
82
              End If
83
84
           End Set ' end of Set accessor
85
86
        End Property ' Side3
87
88
         ' test if triangle is equilateral
89
        Public ReadOnly Property Equilateral() As Boolean
90
91
           ' check sides, return True or False
92
           Get
93
94
               ' test if sides are equal
95
              If m_intSide1 = m_intSide2 _
96
                 AndAlso m_intSide1 = m_intSide3 Then
97
98
                  Return True ' indicate that sides are equal
99
              Else
100
                  Return False ' indicate triangle is not equal sided
101
              End If
102
103
           End Get ' end of Get accessor
104
105
        End Property ' Equilateral
106
107
        ' check if sides create right triangle
108
        Public ReadOnly Property RightTriangle() As Boolean
109
110
            ' check sides, return True or False
111
           Get
112
113
               ' check length
114
              If (m_intSide1 ^ 2) + _
115
                  (m_intSide2 \land 2) = _
116
                  (m_intSide3 ^ 2) Then
117
118
                  Return True ' it is a right triangle
119
120
                  ' check another length combination
121
              ElseIf (m_intSide1 ^ 2) + _
122
                  (m_intSide3 \land 2) = _
123
                  (m_intSide2 ^ 2) Then
124
125
                  Return True ' it is a right triangle
126
127
                  ' check last length combination
128
              ElseIf (m_intSide2 ^ 2) + _
129
                  (m_intSide3 \land 2) =
```

130	(m_intSide1 ^ 2) Then
131	
132	Return True ' it is a right triangle
133	
134	Else
135	<b>Return False '</b> it is not a right triangle
136	
137	End If
138	
139	End Get ' end of Get accessor
140	
141	End Property ' RightTriangle
142	
143	End Class ' Triangle

1	' Exercise 19.11 Solution
2	' TriangleCreator.vb
3	
4	Public Class FrmTriangleCreator
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' create and test triangle properties
10	<pre>Private Sub btnCreate_Click(ByVal sender As System.Object, _</pre>
11	ByVal e As System.EventArgs) Handles btnCreate.Click
12	
13	Dim objTriangle As Triangle ' create Triangle reference
14	
15	' values for three sides
16	<pre>Dim intSide1 As Integer = Convert.ToInt32(Val(txtSide1.Text))</pre>
17	Dim intSide2 As Integer = Convert.ToInt32(Val(txtSide2.Text))
18 19	<pre>Dim intSide3 As Integer = Convert.ToInt32(Val(txtSide3.Text))</pre>
20	
20	lblDisplay.Text = "" ' clear display Label
22	' create triangle object
23	objTriangle = New Triangle(intSide1, intSide2, intSide3)
23	objirrangre = New Trangre(Incsider, Incsidez, Incsides)
25	' test for right triangle
26	If (objTriangle.RightTriangle = True) Then
27	lblDisplay.Text = "You created a right triangle!"
28	is is is pray next = not created a right triangle.
29	' test for equilateral triangle
30	ElseIf (objTriangle.Equilateral = True) Then
31	<pre>lblDisplay.Text = "You created an equilateral triangle!"</pre>
32	Else
33	
34	' triangle is neither right nor equilateral
35	<pre>lblDisplay.Text = ("You created a triangle that is " &amp; _</pre>
36	"neither right nor equilateral!")
37	End If
38	
39	End Sub ' btnCreate_Click
40	
41	End Class ' FrmTriangleCreator

**19.12** (*Modified Microwave Oven Application*) Modify the tutorial's Microwave Oven application to include an additional digit, which would represent the hour. Allow the user to enter up to 9 hours, 59 minutes, and 59 seconds (Fig. 19.50).

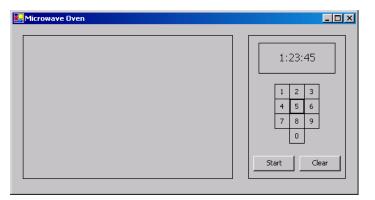


Figure 19.50 Microwave Oven application's GUI.

- a) **Copying the template to your working directory.** Copy the C:\Examples\Tutorial19\Exercises\MicrowaveOven2 directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click MicrowaveOven2.sln in the MicrowaveOven2 directory to open the application.
- c) Adding the hour variable. To allow cooking time that includes the hour digit, you will need to modify the Time class. Define a new Private instance variable to represent the hour. Change the Time constructor to take in as its first argument (now Time should have three arguments) the hour amount. You will also have to modify the **Start** Button event handler and the DisplayTime method to include an hour variable.
- d) *Adding the Hour property*. Use the Minute and Second properties as your template to create the property for the hour. Remember, we are allowing an additional digit to represent the hour (hour < 10).
- e) *Changing the padding amount*. Change the calls to the PadLeft method to be consistent with the new time format.
- f) Extracting the hour. Add a call to the Substring method so that hour gets the first digit in the m\_strTime String. Also, change the calls to the Substring method for minute and second so that they extract the proper digits from the m\_strTime String.
- g) Accessing the first five digits. Change the If...Then statement from the Display-Time method to take and display the first five digits entered by the user.
- h) *Edit the Timer object*. Edit the tmrClock\_Tick event handler to provide changes to hours and its corresponding minutes and seconds.
- i) *Displaying the time*. Edit the Format String so that the display Label includes the hour.
- j) Running the application. Select Debug > Start to run your application. Enter various times and verify that the application counts down properly. Enter an amount of time that is 10 hours or longer, and verify that the application handles invalid input correctly.
- k) *Closing the application.* Close your running application by clicking its close box.
- 1) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1 'Exercise 19.12 Solution
2 'Time.vb
3 'Represents time in 24-hour format and contains properties
4
5 Public Class Time
6
```

```
declare Integers for hour, minute and second
7
 8
       Private m_intHour As Integer
9
       Private m_intMinute As Integer
10
       Private m_intSecond As Integer
11
12
        ' Time constructor (hour, minute and second supplied)
13
       Public Sub New(ByVal hourValue As Integer, _
14
          ByVal minuteValue As Integer, _
15
          ByVal secondValue As Integer)
16
17
                               ' invokes Hour set accessor
          Hour = hourValue
18
          Minute = minuteValue ' invokes Minute set accessor
          Second = secondValue ' invokes Second set accessor
19
20
       End Sub ' New
21
22
       Public Property Hour() As Integer
23
24
           ' return value
25
          Get
26
             Return m_intHour
27
          End Get ' Get accessor
28
29
           ' set m_intHour value
30
          Set(ByVal Value As Integer)
31
              ' if hour value is valid
32
33
             If (Value < 10) Then
34
                 m_intHour = Value
35
             Else
36
                 m_intHour = 0 ' set invalid input to 0
37
             End If
38
39
           End Set ' Set accessor
40
       End Property ' Hour
41
42
43
        ' property Minute
44
       Public Property Minute() As Integer
45
46
           ' return m_intMinute value
47
          Get
48
             Return m_intMinute
49
          End Get ' end of Get accessor
50
51
           ' set m_intMinute value
52
          Set(ByVal Value As Integer)
53
54
              ' if minute value entered is valid
55
             If (Value < 60) Then
56
                 m_intMinute = Value
57
             Else
58
                 m_intMinute = 0 ' set invalid input to 0
59
              End If
60
61
           End Set ' end of Set accessor
62
63
       End Property ' Minute
64
65
        ' property Second
66
       Public Property Second() As Integer
67
```

```
68
             return m_intSecond value
69
           Get
70
              Return m_intSecond
71
           End Get ' Get accessor
72
73
            ' set m_intSecond value
74
           Set(ByVal Value As Integer)
75
76
               ' if second value entered are invalid
77
              If (Value < 60) Then
78
                 m intSecond = Value
79
              Flse
80
                 m_intSecond = 0 ' set invalid input to 0
81
              End If
82
83
           End Set ' Set accessor
84
85
        End Property ' Second
86
87
     End Class ' Time
```

```
1
     ' Exercise 19.12 Solution
 2
     ' MicrowaveOven.vb
 3
 4
    Public Class FrmMicrowaveOven
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' contains time entered as a String
 8
       Private m_strTime As String = ""
 9
10
        ' holds the time
11
       Private m_objTime As Time
12
13
        ' Windows Form Designer generated code
14
15
        ' event handler appends 1 to time string
        Private Sub btnOne_Click(ByVal sender As System.Object, _
16
17
          ByVal e As System. EventArgs) Handles btnOne. Click
18
19
          Beep()
                              ' sound beep
20
          m_strTime &= "1" ' append digit to time input
21
                             ' display time input properly
          DisplayTime()
22
        End Sub ' btnOne_Click
23
24
        ' event handler appends 2 to time string
25
        Private Sub btnTwo_Click(ByVal sender As System.Object, _
26
          ByVal e As System. EventArgs) Handles btnTwo.Click
27
28
          Beep()
                              ' sound beep
29
                              ' append digit to time input
          m_strTime &= "2"
                            ' display time input properly
30
          DisplayTime()
31
       End Sub ' btnTwo_Click
32
33
        ' event handler appends 3 to time string
34
        Private Sub btnThree_Click(ByVal sender As System.Object, _
35
          ByVal e As System. EventArgs) Handles btnThree. Click
36
37
          Beep()
                              ' sound beep
38
          m_strTime &= "3"
                              ' append digit to time input
39
                              ' display time input properly
          DisplayTime()
```

```
40
        End Sub ' btnThree_Click
41
42
        ' event handler appends 4 to time string
        Private Sub btnFour_Click(ByVal sender As System.Object, _
43
44
           ByVal e As System. EventArgs) Handles btnFour. Click
45
46
                              ' sound beep
           Beep()
47
           m_strTime &= "4"
                             ' append digit to time input
                            ' display time input properly
48
           DisplayTime()
49
        End Sub ' btnFour_Click
50
51
        ' event handler appends 5 to time string
52
        Private Sub btnFive_Click(ByVal sender As System.Object, _
53
           ByVal e As System. EventArgs) Handles btnFive. Click
54
55
           Beep()
                              ' sound beep
           m_strTime &= "5" ' append digit to time input
56
           DisplayTime() ' display time input properly
57
58
        End Sub ' btnFive_Click
59
60
        ' event handler appends 6 to time string
        Private Sub btnSix_Click(ByVal sender As System.Object, _
61
62
           ByVal e As System. EventArgs) Handles btnSix. Click
63
64
                              ' sound beep
           Beep()
           m_strTime &= "6" ' append digit to time input
65
           DisplayTime() ' display time input properly
66
67
        End Sub ' btnSix_Click
68
69
        ' event handler appends 7 to time string
70
        Private Sub btnSeven_Click(ByVal sender As System.Object, _
71
           ByVal e As System. EventArgs) Handles btnSeven. Click
72
73
                              ' sound beep
           Beep()
74
           m_strTime &= "7" ' append digit to time input
                            ' display time input properly
75
           DisplayTime()
76
        End Sub ' btnSeven_Click
77
78
        ' event handler appends 8 to time string
        Private Sub btnEight_Click(ByVal sender As System.Object, _
79
80
           ByVal e As System. EventArgs) Handles btnEight. Click
81
82
                              ' sound beep
           Beep()
83
           m_strTime &= "8"
                             ' append digit to time input
                            ' display time input properly
84
           DisplayTime()
        End Sub ' btnEight_Click
85
86
87
        ' event handler appends 9 to time string
88
        Private Sub btnNine_Click(ByVal sender As System.Object, _
89
           ByVal e As System. EventArgs) Handles btnNine. Click
90
91
           Beep()
                              ' sound beep
92
           m_strTime &= "9"
                             ' append digit to time input
           DisplayTime() ' display time input properly
93
94
        End Sub ' btnNine_Click
95
96
        ' event handler appends 0 to time string
97
        Private Sub btnZero_Click(ByVal sender As System.Object, _
98
           ByVal e As System. EventArgs) Handles btnZero. Click
99
                              ' sound beep
100
           Beep()
```

© Copyright 1992-2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```
101
           m_strTime &= "0'
                                append digit to time input
102
                               ' display time input properly
           DisplayTime()
103
        End Sub ' btnZero_Click
104
105
         ' event handler starts the microwave oven's cooking process
        Private Sub btnStart_Click(ByVal sender As System.Object, _
106
107
           ByVal e As System. EventArgs) Handles btnStart. Click
108
109
           Dim intHour As Integer
110
           Dim intSecond As Integer
111
           Dim intMinute As Integer
112
113
           ' pad extra spaces in m_strTime with zero
114
           m_strTime = m_strTime.PadLeft(5, Convert.ToChar("0"))
115
116
           ' extract seconds, minutes and hours
117
           intSecond = Convert.ToInt32(m_strTime.Substring(3))
118
           intMinute = Convert.ToInt32(m_strTime.Substring(1, 2))
119
           intHour = Convert.ToInt32(m_strTime.Substring(0, 1))
120
121
           ' create Time object to contain time entered by user
122
           m_objTime = New Time(intHour, intMinute, intSecond)
123
124
            ' display the time
125
           lblDisplay.Text = String.Format("{0:D1}:{1:D2}:{2:D2}", _
126
              m_objTime.Hour, m_objTime.Minute, m_objTime.Second)
127
128
           m_strTime = "" ' clear m_strTime for future input
129
           tmrClock.Enabled = True
                                              ' begin timer
130
           pnlWindow.BackColor = Color.Yellow ' turn "light" on
131
        End Sub ' btnStart_Click
132
133
        ' event handler to clear input
134
        Private Sub btnClear_Click(ByVal sender As System.Object, _
135
           ByVal e As System. EventArgs) Handles btnClear. Click
136
137
           ' reset each method or variable to its initial setting
138
           lblDisplay.Text = "Microwave Oven"
139
           m_strTime = ""
140
           m_{objTime} = New Time(0, 0, 0)
141
           tmrClock.Enabled = False
                                               ' turn timer off
142
           pnlWindow.BackColor = pnlWindow.DefaultBackColor
143
144
           btnStart.Enabled = True ' enable Start Button
145
        End Sub ' btnClear_Click
146
147
        ' method to display formatted time in timer window
148
        Private Sub DisplayTime()
149
150
           Dim intSecond As Integer
151
           Dim intMinute As Integer
152
           Dim intHour As Integer
153
           Dim strDisplay As String ' the display String
154
155
           ' disallow extra input if number of hours > 9
156
           If m_strTime.Length > 5 Then
157
158
              ' take only the first 5 digits
159
              m_strTime = m_strTime.Substring(0, 5)
160
           Fnd Tf
161
```

162	' pad the empty spaces of the display String with "O"
163	strDisplay = m_strTime.PadLeft(5, Convert.ToChar("0"))
164	
165	' extract seconds and minutes and hours
166	intSecond = Convert.ToInt32(strDisplay.Substring(3))
167	<pre>intMinute = Convert.ToInt32(strDisplay.Substring(1, 2))</pre>
168	<pre>intHour = Convert.ToInt32(strDisplay.Substring(0, 1))</pre>
169	
170	' display number of hours, minutes, and seconds
171	<pre>lblDisplay.Text = String.Format("{0:D1}:{1:D2}:{2:D2}", _</pre>
172	intHour, intMinute, intSecond)
173	End Sub ' DisplayTime
174	
175	' event handler displays new time each second
176	Private Sub tmrClock_Tick(ByVal sender As System.Object, _
177	ByVal e As System.EventArgs) Handles tmrClock.Tick
178	
179	' perform countdown, subtract one second
180	If m_objTime.Second > 0 Then
181	m_objTime.Second -= 1
182	ElseIf m_objTime.Minute > 0 Then
183	<pre>m_objTime.Minute -= 1 ' subtract one minute</pre>
184	m_objTime.Second = 59 ' reset seconds for new minute
185	ElseIf $m_{obj}$ Time.Hour > 0 Then
186	<pre>m_objTime.Hour -= 1 ' subtract one hour</pre>
187	m_objTime.Minute = 59 ' reset minutes for new hour
188	<pre>m_objTime.Second = 59 ' reset seconds for new minute</pre>
189	Else 'no more seconds
190	tmrClock.Enabled = False ' stop the timer
191	Beep() 'sound beep
192	lblDisplay.Text = "Done!" 'display "Done" message
193	pnlWindow.BackColor = pnlWindow.DefaultBackColor
194	piriwindow.backcoror = piriwindow.beraurebackcoror
194	Detum
196	Return End If
190	
197	Low Country Man Data Alan
	' refresh the display time
199	<pre>lblDisplay.Text = String.Format("{0:D1}:{1:D2}:{2:D2}", _</pre>
200	<pre>m_objTime.Hour, m_objTime.Minute, m_objTime.Second)</pre>
201	End Sub ' tmrTimer_Tick
202	
203	End Class ' FrmMicrowaveOven

**19.13** (Account Information Application) The local bank wants you to create an application that will allow them to view their clients' information. The interface is created for you; you need to implement the class (Fig. 19.51). Once the application is completed, the bank manager should be able to click the Next or Previous Button to run through each client's information. The information is stored in four arrays containing first names, last names, account numbers and account balances.

📙 Account Informa	ation _ 🗆 🗙
First name:	
Last name:	
Account number:	
Balance:	
Previous	Next

Figure 19.51 Account Information application GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial19\Exercises\AccountInformation directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click AccountInformation.sln in the AccountInformation directory to open the application.
- c) **Determining variables for the class.** Examine the code from AccountInformation.vb, including all the properties that the Client object uses to retrieve the information.
- d) *Creating the Client class.* Create a new class, and call it Client. Add this class to the project. Define four Private instance variables to represent each property value, to ensure that each Client object contains all the required information about each client. Use those variables to define a constructor.
- e) *Defining each property*. Each Private variable should have a corresponding property, allowing the user to set or get each Private variable's value.
- f) Adding more information. In the FrmAccountInformation\_Load event handler, add two more accounts. Include names, account numbers, and balances for each corresponding array.
- g) Running the application. Select Debug > Start to run your application. Enter information for multiple accounts. Click the Previous and Next Buttons to ensure that each account's information is stored properly.
- h) Closing the application. Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 19.13 Solution
2	' Client.vb
3	' represent client balance information
4	
5	Public Class Client
6	
7	Private m_strFirstName As String ' first name
8	Private m_strLastName As String ' last name
9	Private m_intAccount As Integer ' account number
10	Private m_intbalance As Decimal ' account balance
11	
12	' Client constructor, first and last names, account number
13	' and account balance supplied
14	Public Sub New(ByVal strFirstName As String, _
15	ByVal strLastName As String, ByVal intAccount As Integer, _
16	ByVal decBalance As Decimal)
17	
18	First = strFirstName
19	Last = strLastName
20	Account = intAccount
21	Balance = decBalance

22 End Sub ' New 23 24 ' property First 25 Public Property First() As String 26 27 ' return m\_strFirstName 28 Get 29 Return m\_strFirstName ' return first name 30 End Get ' end of Get accessor 31 32 ' set first name 33 Set(ByVal Value As String) 34 m\_strFirstName = Value 35 End Set ' end of Set accessor 36 37 End Property ' First 38 39 ' property Last 40 Public Property Last() As String 41 ' return m\_strLastName 42 43 Get 44 Return m\_strLastName ' return last name 45 End Get ' end of Get accessor 46 47 ' set last name Set(ByVal Value As String) 48 **49** m\_strLastName = Value 50 End Set ' end of Set accessor 51 52 End Property ' Last 53 54 ' account number 55 Public Property Account() As Integer 56 57 ' return m\_intAccount 58 Get 59 Return m\_intAccount ' return account number End Get ' end of Get accessor 60 61 62 ' set account number 63 Set(ByVal intAccountValue As Integer) 64 65 ' account number can not be negative 66 If intAccountValue > -1 Then 67 m\_intAccount = intAccountValue 68 Else 69 m\_intAccount = 0 ' default to zero 70 End If 71 72 End Set ' end of Set accessor 73 74 End Property ' Account 75 76 ' account balance 77 Public Property Balance() As Decimal 78 79 ' return m\_intbalance 80 Get 81 Return m\_intbalance ' return account balance 82 End Get ' end of Get accessor

```
83
84
           ' set the account balance
85
           Set(ByVal Value As Decimal)
86
              m_intbalance = Value
87
           End Set ' end of Set accessor
88
89
        End Property ' Balance
90
91
     End Class ' Client
```

1	' Exercise 19.13 Solution
2	' AccountInformation.vb
3	
4	Public Class FrmAccountInformation
5	Inherits System.Windows.Forms.Form
6	
7	Private m_objName(9) As Client ' Client object
8	Private m_intPosition As Integer = 0 ' current account
	reverse in increase in a sinteger = 0 Current account
9	
10	'Windows Form Designer generated code
11	
12	' create array of Client objects
13	Private Sub FrmAccountInformation_Load(ByVal sender As _
14	
	System.Object, ByVal e As System.EventArgs) _
15	Handles MyBase.Load
16	
17	Dim intCount As Integer ' counter variable
18	
19	' array of first names
20	Dim strFirstName() As String = _
21	Dim Stiffistivane() As Stiffig = _
	<pre>New String() {"John", "Sarah", "Jack", "Adam", "Diane", _</pre>
22	"David", "Kristin", "Jennifer", "", ""}
23	
24	' array of last names
25	Dim strLastName() As String = _
26	<pre>New String() {"Blue", "White", "Red", "Brown", _</pre>
27	"Yellow", "Black", "Green", "Orange", "", ""}
28	renow, black, dreen, blange, , j
29	'array of account numbers
30	<pre>Dim intAccount() As Integer = _</pre>
31	New Integer() {1234652, 1234666, 1234678, 1234681, _
32	1234690, 1234770, 1234787, 1234887, 0, 0}
33	
34	' array of account balances
35	Dim decBalance() As Decimal = _
36	<pre>New Decimal() {Convert.ToDecimal(1000.78), _</pre>
37	<pre>Convert.ToDecimal(2056.24), Convert.ToDecimal(978.65), _</pre>
38	Convert.ToDecimal(990.0), Convert.ToDecimal(432.75), _
39	Convert.ToDecimal(780.78), Convert.ToDecimal(4590.63), _
40	Convert.ToDecimal(7910.11), 0, 0}
41	
42	' loop and create 10 Client objects
43	<pre>For intCount = 0 To m_objName.GetUpperBound(0)</pre>
44	
45	' create new object and store into Client array
46	<pre>m_objName(intCount) = New Client(strFirstName(intCount), _</pre>
47	<pre>strLastName(intCount), intAccount(intCount), _</pre>
48	decBalance(intCount))
49	Next
50	next.
50	

© Copyright 1992-2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```
51
        End Sub ' FrmAccountInformation_Load
52
53
        ' display next object
54
        Private Sub btnNext_Click(ByVal sender As System.Object, _
55
           ByVal e As System. EventArgs) Handles btnNext. Click
56
57
           m_intPosition += 1 ' increment position
58
59
           ' if position is last (top) object
60
           If m_intPosition > m_objName.GetUpperBound(0) Then
              m_intPosition = 0 ' set to first position in array
61
62
              DisplayInformation() ' display information
63
           Else
64
              DisplayInformation()
65
           End If
66
67
        End Sub ' btnNext_Click
68
69
        ' display previous object
70
        Private Sub btnPrevious_Click(ByVal sender As System.Object, _
71
           ByVal e As System. EventArgs) Handles btnPrevious. Click
72
73
           m_intPosition -= 1 ' decrement position
74
75
           ' if position is last (bottom) object
76
           If m_intPosition < 0 Then</pre>
77
78
              ' set to last position in array
79
              m_intPosition = m_objName.GetUpperBound(0)
80
              DisplayInformation()
81
           Else
82
              DisplayInformation() ' display information
83
           End If
84
85
        End Sub ' btnPrevious_Click
86
87
        ' display information
88
        Private Sub DisplayInformation()
89
90
           ' use m_intPosition as index for each object
91
           txtFirst.Text = m_objName(m_intPosition).First
92
           txtLast.Text = m_objName(m_intPosition).Last
93
           txtAccount.Text = _
94
              Convert.ToString(m_objName(m_intPosition).Account)
95
96
           ' format as currency
97
           txtBalance.Text = _
98
              String.Format("{0:C}", m_objName(m_intPosition).Balance)
99
100
        End Sub ' DisplayInformation
101
102 End Class ' FrmAccountInformation
```

What does this code do?

**19.14** What does the following code do? The first code listing contains the definition of class Shape. Each Shape object represents a closed shape with a number of sides. The second code listing contains a method (Mystery) created by a client of class Shape. What does this method do?

```
1
    Public Class Shape
 2
 3
        Private m_intSides As Integer
 4
 5
        ' constructor with number of sides
 6
        Public Sub New(ByVal intSides As Integer)
 7
           Side = intSides
 8
        End Sub ' New
 9
10
        ' set and get side value
11
        Public Property Side() As Integer
12
13
           ' return m intSides
14
           Get
15
              Return m_intSides
16
           End Get ' end of Get accessor
17
18
           ' set m_intSides
19
           Set(ByVal Value As Integer)
20
21
              If Value > 0 Then
22
                 m_intSides = Value
23
              Else
24
                 m_intSides = 0
25
              End If
26
27
           End Set ' end of Set accessor
28
29
        End Property ' Side
30
31
    End Class ' Shape
```

```
1
    Public Function Mystery(ByVal objShape As Shape) As String
2
       Dim strShape As String
3
 4
        ' determine case with objShape.Side
 5
       Select Case objShape.Side
 6
 7
          Case Is < 3
 8
              strShape = "Not a Shape"
9
10
          Case 3
11
              strShape = "Triangle"
12
13
          Case 4
14
              strShape = "Square"
15
16
          Case Else
17
             strShape = "Polygon"
18
19
       End Select
20
21
       Return strShape
22
    End Function ' Mystery
```

**Answer:** The Shape class defines a shape with a given number of sides. Method Mystery determines the shape of it's Shape and returns the name of the shape. Method Mystery takes a Shape object as an argument.

		2	

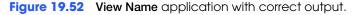
8

What's wrong with this code?	<b>19.15</b> Find the error(s) in the following code. The following method should create a new Shape object with intNumberSides sides. Assume the Shape class from Exercise 19.14.				
	<pre>1 Private Sub ManipulateShape(ByVal intNumberSides As Integer) 2 Dim objShape As Shape = New Shape(3) 3 4 Shape.m_intSides = intNumberSides 5 End Sub ' ManipulateShape</pre>				
	<b>Answer:</b> The method should create a Shape object with intNumberSides sides—not a Shape object with 3 sides. Also, a Private variable (m_intSides) cannot be accessed from outside the class. The correct code is as follows:				
	<pre>Private Sub ManipulateShape(ByVal intNumberSides As Integer) 2</pre>				
	<pre>3 Dim objShape As Shape = New Shape(intNumberSides) 4</pre>				
	5 'or 6				
	<pre>7 'Dim objShape As Shape 8 'objShape.Side = intNumberSides</pre>				
	9 End Sub ' ManipulateShape				

Using the Debugger

**19.16** (View Name Application) The View Name application allows the user to enter the user's first and last name. When the user clicks the View Name Button, a MessageBox that displays the user's first and last name appears. The application creates an instance of Class Name. This class uses its property definitions to set the first-name and last-name instance variables. Copy the Names directory from C:\Examples\Tutorial19\Exercises\Debugger to your Debugger directory. Open and run the application. While testing your application, you noticed that the MessageBox did not display the correct output. Use the debugger to find the logic error(s) in the application. The application with the correct output is displayed in Fig. 19.52.

🔛 View Name		-	
First name:	Jessy	Name	First name: Jessy Last name: Brown
	View Name		OK



```
Exercise 19.16 Solution
 1
     ۰.
 2
     'Name.vb
 3
     ' Name class definition
 4
 5
    Public Class Name
 6
 7
        Private m_strFirstName As String
 8
        Private m_strLastName As String
 9
10
        ' Name constructor, first and last names supplied
11
        Public Sub New(ByVal strFirstName As String, _
12
           ByVal strLastName As String)
13
```

	14	First = strFirstName
	15	Last = strLastName
	16	End Sub ' New
	17	
	18	' property First
	19	Public Property First() As String
	20	
	21	' return first name
	22	Get
	23	Return m strFirstName
	24	End Get
	25	
	26	' set first name
	27	Set(ByVal Value As String)
	28	
Original line assigned	29	m_strFirstName = Value
m_strFirstName to Value	30	End Set
	31	
	32	End Property ' First
	33	
	34	' property Last
	35	Public Property Last() As String
	36	
	37	' return last name
	38	Get
	39	Return m strLastName
	40	End Get
	41	
	42	' set last name
	43	Set(ByVal Value As String)
Original line assigned data	44	
tom strFirstName, rather	45	m strLastName = Value
than to m strLastName	46	End Set
_	47	
	48	End Property ' Second
	49	
	50	End Class ' Name

#### ' Exercise 19.16 Solution 1 ' ViewName.vb 2 3 4 Public Class FrmViewName 5 Inherits System.Windows.Forms.Form 6 7 Private m\_objName As Name ' Name object 8 9 ' Windows Form Designer generated code 10 Private Sub btnView\_Click(ByVal sender As System.Object, \_ 11 12 ByVal e As System.EventArgs) Handles btnView.Click 13 14 Dim strOutput As String ' holds first name and last name 15 16 ' create new Name 17 m\_objName = New Name(txtFirst.Text, txtLast.Text) 18 19 ' assign user's name to strOutput 20 strOutput = "First name: " & m\_objName.First & \_ 21 ControlChars.CrLf & "Last name: " & m\_objName.Last 22

23	'output name
24	<pre>MessageBox.Show(strOutput, "Name", _</pre>
25	MessageBoxButtons.OK, MessageBoxIcon.Information)
26 27	End Sub ' btnView_Click
28	End Class ' FrmViewName

Programming Challenge

**19.17** (*DVD Burner Application*) Create an application that simulates a DVD burner. Users create a DVD with their choice of title and bonus materials. The GUI is provided for you (Fig. 19.53). You will create a class (DVDObject) to represent the DVD object and another class (Bonus) to represent bonus materials for a DVD object.

🔜 D¥D Burner	
DVD Information	Bonus Materials
Movie title:	Description:
Minutes:	Minutes:
	Description:
Create	Minutes:
Information	Description:
	Minutes:

Figure 19.53 DVD Burner application's GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial19\Exercises\DVDBurner directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click DVDBurner.sln in the DVD-Burner directory to open the application.
- c) *Creating the bonus-material object.* Create a class, and name it Bonus. The class's objects will each represent one bonus-material item on the DVD. Each Bonus object should have a name (description) and a length (in minutes). Use this tutorial's Time class as your guide in creating the properties for the name and length of each bonus material.
- d) *Creating the DVD class.* Create a class, and name it DVD0bject. This class contains the movie title and the length of the movie. The class should also include an array of three Bonus items.
- e) *Creating the necessary variables.* Before you define the **Create** Button's event handler, create an DVD0bject class instance variable. Inside the **Create** Button's event handler, create the necessary variables to store the information from the TextBoxes on the GUI. Also, this is where you need to create the array of Bonus objects to store the bonus materials.
- f) *Adding bonus-material information*. Add the description and length of each bonus item to the Bonus array you created from the previous step.
- g) *Creating a DVD object*. Use information about the movie, its title, length and the array of bonus materials to make your DVD object.
- h) *Displaying the output*. The Information Button's Click event is already defined for you. Locate the event handler, add a String containing the complete information on the DVD object that you created earlier and display this String to a MessageBox.
- Running the application. Select Debug > Start to run your application. Enter information for several DVDs. After information is entered for each, click the Create Button. Then, click the Information Button and verify that the information being displayed is correct for your newly created DVD.
- j) *Closing the application.* Close your running application by clicking its close box.
- k) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 19.17 Solution
2	'Bonus.vb
3	' represent bonus items on a DVD
4	
5	Public Class Bonus
	Public Class Bollus
6	
7	' name of bonus material
8	Private m_strName As String
9	
10	' length of the bonus material
11	Private m_intItemLength As Integer
12	
13	' Bonus constructor, name and item length
14	Public Sub New(ByVal nameValue As String, _
15	ByVal lengthValue As Integer)
16	
17	Name = nameValue
18	
19	ItemLength = lengthValue End Sub ' New
	End Sub New
20	
21	' set or get name of bonus material
22	Public Property Name() As String
23	
24	' return m_strName
25	Get
26	Return m_strName ' return name
27	End Get ' end of Get accessor
28	
29	' set description name
30	Set(ByVal Value As String)
31	Secury value As Stillig)
32	Life decomination is present them 20 phonestering
	' if description is greater than 20 characters
33	If Value.Length $> 20$ Then
34	
35	' take first 20 characters
36	Value = Value.Substring(0, 20)
37	m_strName = Value
38	Else
39	
40	' set name
41	m strName = Value
42	End If
43	
44	End Set ' end of Set accessor
45	
40	End Property ' Name
	End Property Name
47	
48	' set or get amount of items
49	Public Property ItemLength() As Integer
50	
51	' return m_intItemLength
52	Get
53	Return m_intItemLength ' return length
54	End Get ' end of Get accessor
55	
56	' set minute value
57	Set(ByVal Value As Integer)
	Sec(byvar value AS Integer)
58	the state of the s
59	' make sure minute is non-negative

© Copyright 1992-2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```
If Value > 0 Then
60
61
                 m_intItemLength = Value
62
              Else
63
                 m_intItemLength = 0
64
              End If
65
66
           End Set ' end of Set accessor
67
68
        End Property ' ItemLength
69
70
    End Class ' Bonus
```

```
1
    ' Exercise 19.17 Solution
 2
    ' DVDObject.vb
 3
    ' represent items on a DVD
 4
 5
   Public Class DVDObject
 6
       Private m_strMovieTitle As String
                                               ' name of movie
 7
       Private m_objBonusMaterial() As Bonus ' array of Bonus objects
 8
       Private m_intMovieLength As Integer
                                              ' length of movie
 9
10
        ' DVDObject constructor
11
       Public Sub New(ByVal nameValue As String, _
12
          ByVal bonusValue() As Bonus, _
13
          ByVal movieLengthValue As Integer)
14
15
           ' call property to set values
16
          MovieTitle = nameValue
17
          MovieLength = movieLengthValue
18
19
           ' assign bonusValue array to m_objBonusMaterial
20
          m_objBonusMaterial = bonusValue
21
22
       End Sub ' New
23
24
        ' set or get movie title
25
       Public Property MovieTitle() As String
26
27
           ' return m_strMovieTitle
28
          Get
29
             Return m_strMovieTitle ' return movie title
30
           End Get ' end of Get accessor
31
32
           ' set movie title
33
          Set(ByVal Value As String)
34
35
              ' if title is greater than 20 characters
36
             If Value.Length > 20 Then
37
38
                 ' take first 20 characters
39
                Value = Value.Substring(0, 20)
40
                m_strMovieTitle = Value
41
             Else
42
43
                 ' set title
44
                m_strMovieTitle = Value
45
              End If
46
47
           End Set ' end of Set accessor
48
```

```
49
        End Property ' MovieTitle
50
51
        ' ReadOnly get property
52
        Public ReadOnly Property BonusMaterials() As String
53
54
           ' display bonus material information
55
           Get
56
57
               ' information on bonus materials
              Dim strBonusMaterial As String = ""
58
59
              Dim intCount As Integer ' counter variable
60
61
              ' loop through each bonus material
62
              For intCount = 0 To (m_objBonusMaterial.Length - 1)
63
64
                  ' format String to contain minutes and seconds
65
                 strBonusMaterial &= _
66
                    m_objBonusMaterial(intCount).Name() & ": " & _
67
                    m_objBonusMaterial(intCount).ItemLength & _
68
                     " minute(s)." & ControlChars.CrLf
69
              Next
70
71
              Return strBonusMaterial ' return String
72
           End Get ' end of Get accessor
73
74
        End Property ' BonusMaterials
75
76
        ' set and get movie length
77
        Public Property MovieLength() As Integer
78
79
           ' return m_intMovieLength
80
           Get
81
              Return m_intMovieLength ' return length of movie
82
           End Get ' end of Get accessor
83
84
           ' set minutes for movie
85
           Set(ByVal Value As Integer)
86
87
              ' make sure minute is nonegative
88
              If Value > 0 Then
89
                 m_intMovieLength = Value
90
              Else
91
                 m_intMovieLength = 0
92
              End If
93
94
           End Set ' end of Set accessor
95
96
        End Property ' MovieLength
97
98
     End Class ' DVDObject
```

1	' Exercise 19.17 Solution
2	' DVDBurner.vb
3	
4	Public Class FrmDVDBurner
5	Inherits System.Windows.Forms.Form
6	
7	Private m_objDVD As DVDObject ' create instance variable
8	
9	' Windows Form Designer generated code

# Microwave Oven Application

10	
10	Definite Cale has Careta Click (D.) (1) and a factor Object
11	Private Sub btnCreate_Click(ByVal sender As System.Object, _
12	ByVal e As System.EventArgs) Handles btnCreate.Click
13	
14	Dim objBonus(2) As Bonus ' array of Bonus
15	Dim intBonusLength As Integer ' bonus material minutes
16	
17	' store movie name
18	<pre>Dim strMovieTitle As String = txtTitle.Text</pre>
19	, and the second s
20	' movie minutes
21	Dim intMovieMinutes As Integer = _
22	Convert.ToInt32(Val(txtMovieMinute.Text))
23	Convert. Tornesz (var(cxchovreminate.rext))
23	L honus motorial description (nome)
	' bonus material description (name)
25	<pre>Dim strBonus1 As String = txtDescription1.Text</pre>
26	<pre>Dim strBonus2 As String = txtDescription2.Text</pre>
27	<pre>Dim strBonus3 As String = txtDescription3.Text</pre>
28	
29	' store minutes from TextBox
30	intBonusLength = Convert.ToInt32(Val(txtMinutes1.Text))
31	
32	' add bonus material name and time to array
33	<pre>objBonus(0) = New Bonus(strBonus1, intBonusLength)</pre>
34	
35	' store minutes from TextBox
36	<pre>intBonusLength = Convert.ToInt32(Val(txtMinutes2.Text))</pre>
37	
38	' add bonus material name and time to array
39	
	<pre>objBonus(1) = New Bonus(strBonus2, intBonusLength)</pre>
40	
41	' store minutes from TextBox
42	intBonusLength = Convert.ToInt32(Val(txtMinutes3.Text))
43	
44	' add bonus material name and time to array
45	objBonus(2) = New Bonus(strBonus3, intBonusLength)
46	
47	' call constructor for new object
48	m_objDVD = _
49	<pre>New DVDObject(strMovieTitle, objBonus, intMovieMinutes)</pre>
50	
51	' let the user know about progress
52	lblDisplay.Text = "Your DVD was created successfully!"
53	
54	' enable Information Button
55	btnInformation.Enabled = True
56	
57	End Sub 1 http://www.click
	End Sub ' btnCreate_Click
58 50	I dienles information closet DVD
59	' display information about DVD
60	<pre>Private Sub btnInformation_Click(ByVal sender As System.Object, _</pre>
61	ByVal e As System.EventArgs) Handles btnInformation.Click
62	
63	Dim strInformation As String ' output String
64	
65	lblDisplay.Text = ""
66	
67	' add title and length to String
68	' add information about bonus materials
69	strInformation = m_objDVD.MovieTitle & ": " & _
70	<pre>m_objDVD.MovieLength &amp; " minute(s)" &amp; _</pre>

71	ControlChars.CrLf & "Bonus Materials:" & _
72	<pre>ControlChars.CrLf &amp; m_objDVD.BonusMaterials</pre>
73	
74	' display output in a MessageBox
75	<pre>MessageBox.Show(strInformation, "DVD Description", _</pre>
76	MessageBoxButtons.OK, MessageBoxIcon.Information)
77	End Sub ' btnInformation_Click
78	
79	End Class ' FrmDVDBurner





# Shipping Hub Application

Introducing Collections, the For Each...Next Statement and Access Keys Solutions

$\supset$	Instructor's Manual Exercise Solutions Tutorial 20		
	MULTIPLE-CHOICE	20.1 are specifically designed to	store groups of values.
	QUESTIONS	a) Collections	b) Properties
	QUEUNONU	c) Accessors	d) None of the above.
		<b>20.2</b> The key provides a quick on a Form.	and convenient way to navigate through controls
		a) <i>Tab</i>	b) Enter
		c) Caps Lock	d) Alt
		<b>20.3</b> An ArrayList differs from an array i	n that an ArrayList can
		a) store objects of any type	b) resize dynamically
		c) be accessed programmatically	d) All of the above.
		<b>20.4</b> The element in a For EachNext sta	tement .
		a) must be of type Integer	ame type as the collection or array type d) None of the above.
		<b>20.5</b> The control that receives the focus the erty set to	ne first time Tab is pressed has a TabIndex prop-
		a) First	b) 0
		c) Next	d) 1
		<b>20.6</b> Users should be able to use the <i>Tab</i> k	ey to transfer the focus to
		<ul><li>a) only Buttons</li><li>c) only controls that have an AcceptT</li><li>d) only the controls that receive user i</li></ul>	b) only TextBoxes ab property
		, <b>.</b>	tain the focus when the <i>Tab</i> key is pressed, use the
		a) TabIndex property	b) TabStop and TabIndex properties
		c) TabStop property	d) Focus property
		<b>20.8</b> To add a value to the end of an Array	List, call the method.
		a) Add	b) AddToEnd
		c) AddAt	d) InsertAt
		<b>20.9</b> To remove a value from a specific ind	ex in the ArrayList, use method
		a) Remove	b) RemoveAt
		c) Delete	d) DeleteAt
		<b>20.10</b> To display an ampersand character of	on a control, type a in its Text property.
		a) &	b) &
		c) &&	d) _&
		<b>Answers:</b> 20.1) a. 20.2) a. 20.3) b. 20.4) b.	20.5) b. 20.6) d. 20.7) b. 20.8) a. 20.9) b. 20.10) c.
$\overline{)}$	EXERCISES		<i>m</i> ) Modify the <b>Salary Survey</b> application you cre-

ated in Exercise 17.12 by using a For Each...Next loop to replace the For...Next loop that is used in Tutorial 17 (Fig. 20.27).



Figure 20.27 Modified Salary Survey GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial20\Exercises\SalarySurveyModified directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click SalarySurveyModified.sln in the SalarySurvey directory to open the application.
- c) Locating the event handler. In Design View, double click the Show Totals Button to bring up the event handler. The code to handle the Click event should include two statements, one to clear the items in the ListBox and the other to add a header.
- d) *Creating a counter variable*. The For Each...Next loop allows you to loop through each element in a specified collection. The For...Next loop from Exercise 17.12 handles the String (m\_strSalaryRanges) and Integer (m\_intSalaries) arrays. This presents a problem. You cannot loop through both of these arrays using the same element reference. (One is an Integer, and the other is a String.) To handle this you need to create a common counter variable, one that you will use to loop through the indices of both arrays. This is possible because the lengths of both arrays are the same.
- e) *Adding an element reference*. It does not matter which array you decide to use in this exercise, because these arrays are of the same length. Declare an element reference with the correct data type.
- f) Create the For Each...Next loop. Use the new element reference that you have created along with the array of your choice to create the For Each...Next loop statement.
- g) *Adding text to the ListBox*. Adding the statement to output to the ListBox is exactly the same as the one from Exercise 17.12. The only difference will be the name of the counter variable that you decide to use.
- h) *Increment the counter variable*. To successfully loop through both arrays and output the data, you need to increment the counter variable. This ensures that the proper data is added to the ListBox through each iteration.
- Running the application. Select Debug > Start to run your application. Enter several sales amounts using the Calculate Button. Click the Show Totals Button and verify that the proper amounts are displayed for each salary range, based on the salaries calculate from your input.
- j) *Closing the application.* Close your running application by clicking its close box.
- k) Closing the IDE. Close Visual Studio .NET by clicking its close box.

3

2 ' SalarySurvey.vb

<sup>&#</sup>x27; Exercise 20.11 Solution

4 Public Class FrmSalarySurvey 5 Inherits System.Windows.Forms.Form 6 7 ' Windows Form Designer generated code 8 9 ' salary ranges 10 Private m\_strSalaryRanges As String() = New String() { \_ "\$200 - \$299", "\$300 - \$399", "\$400 - \$499", \_ "\$500 - \$599", "\$600 - \$699", "\$700 - \$799", \_ 11 12 "\$800 - \$899", "\$900 - \$999", "\$1000 + "} 13 14 15 ' number of employees in each salary range 16 Private m\_intSalaries As Integer() = New Integer( \_ 17 m\_strSalaryRanges.GetUpperBound(0)) {} 18 19 ' handles Calculate Button's Click event 20 Private Sub btnCalculate\_Click(ByVal sender As System.Object, \_ 21 ByVal e As System.EventArgs) Handles btnCalculate.Click 22 23 ' obtain total sales 24 Dim decSales As Decimal = Convert.ToDecimal( \_ 25 Val(txtInputSales.Text)) 26 27 ' employee's base salary 28 Dim decTotalSalary As Decimal = 200 29 30 ' add commision to total salary 31 decTotalSalary += Convert.ToDecimal(decSales \* 0.09) 32 33 ' display salary in a Label 34 lblTotalSalary.Text = String.Format("{0:C}", decTotalSalary) 35 36 ' increment the correct counter in array m\_intSalaries 37 Select Case decTotalSalary 38 39 Case Is < 300**40**  $m_intSalaries(0) += 1$ 41 Case Is < 40042 43  $m_intSalaries(1) += 1$ 44 45 Case Is < 50046  $m_intSalaries(2) += 1$ 47 48 Case Is < 60049  $m_intSalaries(3) += 1$ 50 51 Case Is < 70052  $m_intSalaries(4) += 1$ 53 54 Case Is < 80055  $m_intSalaries(5) += 1$ 56 57 Case Is < 90058  $m_intSalaries(6) += 1$ 59 60 Case Is < 100061  $m_intSalaries(7) += 1$ 62 63 Case Is >= 1000 64  $m_intSalaries(8) += 1$ 

```
65
66
           End Select
67
68
           ' clear TextBox
69
           txtInputSales.Clear()
70
        End Sub ' btnCalculate_Click
71
72
        ' handles click event for btnShowTotals Button
73
        Private Sub btnShowTotals_Click(ByVal sender As System.Object, _
           ByVal e As System.EventArgs) Handles btnShowTotals.Click
74
75
76
           Dim intCounter As Integer = 0 ' counter variable
77
           Dim strRange As String ' each range in m_strSalaryRanges
78
79
           ' clear all items in the ListBox
80
           lstSalaryTotals.Items.Clear()
81
82
           ' add header to ListBox
83
           lstSalaryTotals.Items.Add("Salary range:" & _
84
              ControlChars.Tab & "Total:")
85
           ' add each element from two arrays to the ListBox
86
87
           For Each strRange In m_strSalaryRanges
88
              lstSalaryTotals.Items.Add(m_strSalaryRanges(intCounter) & _
89
                 ControlChars.Tab & m_intSalaries(intCounter))
90
              intCounter += 1 ' increment the counter
91
           Next
92
93
        End Sub ' btnShowTotals_Click
94
95
     End Class ' FrmSalarySurvey
```

**20.12** (*Modified Shipping Hub Application*) Modify the Shipping Hub application created in this tutorial, so that the user can double click a package in the lstPackages ListBox. When a package number is double clicked, the package's information should be displayed in a MessageBox (Fig. 20.28).

Shipping Hub		
Arrived at:	12/18/2002 10:17:01 AM	
-Package Information	on	Packages by State
Package ID:	39423	AL
Address:	318 Indiana Street	39423 k
City:	Miland State: AL Zip: 91142	
< <u>B</u> ACK	Scan New Add Remove Edit NEXT	>
	Package Information	
	Package 39423 Arrived at: 12/18/2002 10:17:01 AM Address: 318 Indiana Street City: Miland State: AL Zip code: 91142	
	ок	

Figure 20.28 Modified Shipping Hub application GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial20\Exercises\ShippingHubModified directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click ShippingHubModified.sln in the ShippingHubModified directory to open the application.
- c) Viewing the event handler. Click ShippingHub.vb in the Solution Explorer and select View > Code. Scroll to the end of code listing to locate the ListBox's Double-Click event handler.
- d) Initializing necessary variables. To loop through the packages in the ArrayList of Packages, you need to create a reference of type Package. It is also helpful to create a String variable to store the information about the given package. Write code in the DoubleClick event handler to declare the String strPackage. A ListBox's DoubleClick event is raised when the control is double clicked.
- e) Check whether the user has selected a valid item. To determine whether the user has selected a valid item (and not an empty element in the ListBox), write an If...Then statement to make sure that the ListBox is not empty when the user selected an item. [*Hint*: A SelectedIndex value of -1 means that no item is currently selected.]
- f) Writing a For Each...Next loop. Use the Package reference you declared in Step c to create a For Each...Next loop with the m\_objList collection.
- g) **Determining whether the current selected package is correct.** Insert an If...Then statement to determine whether the current object that is selected from the m\_objList collection matches the selected item from the ListBox. Because the packages are listed in the ListBox by their package number, use that information in your If...Then statement. Once the correct package is matched, store that package's information in the strPackage String.
- h) Inserting the Else statement. Make sure to notify the user if an invalid item has been selected from the ListBox. If this occurs, add a message to the strPackage String that will be displayed in the MessageBox.
- Displaying the MessageBox. Call the MessageBox's Show method to display the text you have added to the strPackage String. This displays either the information for the package they have selected or the message telling them they have selected an invalid package.
- j) Running the application. Select Debug > Start to run your application. Add several packages. In the Packages by State GroupBox, select a state for which there are packages being sent. Double click one of the packages listed in the Packages by State ListBox, and verify that the correct information is displayed in a MessageBox.
- k) *Closing the application.* Close your running application by clicking its close box.
- 1) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 20.12 Solution
2	' ShippingHub.∨b
3	
4	Public Class FrmShippingHub
5	Inherits System.Windows.Forms.Form
6	
7	Private m_objList As Collections.ArrayList ' list of packages
8	Private m_objPackage As Package ' current package
9	Private m_intPosition As Integer ' position of current package
0	Private m_objRandom As Random ' random number for package id
11	Private m_intPackageID As Integer ' individual package number
2	
3	' Windows Form Designer generated code
4	
15	' Form Load event
6	Private Sub FrmShippingHub_Load(ByVal sender As _
7	System.Object, ByVal e As System.EventArgs) _

18	Handles MyBase.Load
19	
20	<pre>m_intPosition = 0 ' set initial position to zero</pre>
21	m_objRandom = New Random ' create new Random object
22	<pre>m_intPackageID = m_objRandom.Next(1, 100000) ' new package ID</pre>
23	
24	' show first state in ComboBox (using the Items property)
25	<pre>cboState.Text = Convert.ToString(cboState.Items.Item(0))</pre>
26	
	<pre>m_objList = New Collections.ArrayList ' list of packages</pre>
27	End Sub ' FrmShippingHub_Load
28	
29	' Scan New Button Click event
30	<pre>Private Sub btnNew_Click(ByVal sender As System.Object, _</pre>
31	ByVal e As System.EventArgs) Handles btnNew.Click
32	
33	<pre>m_intPackageID += 1 ' increment package id</pre>
34	<pre>m_objPackage = New Package(m_intPackageID) ' create package</pre>
35	m_objrackage = new rackage(m_nrtrackagerb) = create package
36	ClearControls() ' clear fields
37	lblPackageNumber.Text = _
38	m_objPackage.PackageNumber.ToString ' package number
39	lblArrivalTime.Text = _
40	<pre>m_objPackage.ArrivalTime.ToString ' display arrival time</pre>
41	
42	' only allow user to add package
43	fraAddress.Enabled = True ' disable GroupBox and its controls
44	SetButtons(False) ' enable/disable Buttons
45	<pre>btnAdd.Enabled = True ' enable Add Button</pre>
46	<pre>btnNew.Enabled = False ' disable Scan New Button</pre>
47	<pre>txtAddress.Focus() ' transfer the focus to txtAddress TextBox</pre>
48	End Sub ' btnNew_Click
49	
50	' Add Button Click event
51	<pre>Private Sub btnAdd_Click(ByVal sender As System.Object, _</pre>
52	ByVal e As System.EventArgs) Handles btnAdd.Click
53	byvar e As system. EventArgs) handres benadi.errek
54	Cat Dealers () I and Dealers menories from Tout Daves
	SetPackage() ' set Package properties from TextBoxes
55	m_objList.Add(m_objPackage) ' add package to ArrayList
56	
57	<pre>fraAddress.Enabled = False ' disable GroupBox and its controls</pre>
58	SetButtons(True) ' enable appropriate Buttons
59	
60	' package cannot be added until Scan New is clicked
61	btnAdd.Enabled = False ' disable Add Button
62	
63	' if package's state displayed, add ID to ListBox
64	If cboState.Text = cboViewPackages.Text Then
65	lstPackages.Items.Add(m_objPackage.PackageNumber)
66	End If
67	
68	cboViewPackages.Text = m_objPackage.State ' list packages
69	<pre>btnNew.Enabled = True ' enable Scan New Button</pre>
70	End Sub ' btnAdd_Click
71	
72	' Back Button Click event
73	
	Private Sub btnBack_Click(ByVal sender As System.Object, _
74	ByVal e As System.EventArgs) Handles btnBack.Click
74 75	
74 75 76	' move backward one package in the list
74 75 76 77	<pre>' move backward one package in the list If m_intPosition &gt; 0 Then</pre>
74 75 76	' move backward one package in the list

© Copyright 1992-2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```
79
           Else ' wrap to end of list
80
              m_{intPosition} = m_{objList.Count} - 1
81
           End If
82
83
           LoadPackage() ' load package data from item in list
84
        End Sub ' btnBack_Click
85
86
        ' Next Button Click event
87
        Private Sub btnNext_Click(ByVal sender As System.Object, _
88
           ByVal e As System. EventArgs) Handles btnNext. Click
89
90
            ' move forward one package in the list
91
           If m_intPosition < m_objList.Count - 1 Then</pre>
92
              m_{intPosition += 1}
93
           Else
              m_intPosition = 0 ' wrap to beginning of list
94
95
           End If
96
97
           LoadPackage() ' load package data from item in list
98
        End Sub ' btnNext_Click
99
100
        ' Remove Button click event
101
        Private Sub btnRemove_Click(ByVal sender As _
102
           System.Object, ByVal e As System.EventArgs) _
103
           Handles btnRemove.Click
104
105
           ' remove ID from ListBox if state displayed
106
           If cboState.Text = cboViewPackages.Text Then
107
              lstPackages.Items.Remove(m_objPackage.PackageNumber)
108
           End If
109
110
           m_objList.RemoveAt(m_intPosition) ' remove package from list
111
112
           ' load next package in list if there is one
113
           If m_objList.Count > 0 Then
114
115
              ' if not at first position, go to previous one
116
              If m_intPosition > 0 Then
117
                 m_{intPosition} -= 1
118
              End If
119
120
              LoadPackage() ' load package data from item in list
121
           Else
122
              ClearControls() ' clear fields
123
           End If
124
125
           SetButtons(True) ' enable appropriate Buttons
126
        End Sub ' btnRemove_Click
127
128
        ' Edit/Update Button Click event
129
        Private Sub btnEditUpdate_Click(ByVal sender As _
130
           System.Object, ByVal e As System.EventArgs) _
131
           Handles btnEditUpdate.Click
132
           ' when Button reads "Edit", allow user to
133
134
           ' edit package information only
135
           If btnEditUpdate.Text = "&Edit" Then
136
              fraAddress.Enabled = True
137
              SetButtons(False)
138
              btnEditUpdate.Enabled = True
139
```

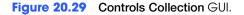
140	' change Button text from "Edit" to "Update"
141	<pre>btnEditUpdate.Text = "&amp;Update"</pre>
142	Else
	EISE
143	
144	' when Button reads "Update" remove the old package
145	' data and add new data from TextBoxes
146	SetPackage()
147	<pre>m_objList.RemoveAt(m_intPosition)</pre>
148	m_objList.Insert(m_intPosition, m_objPackage)
149	··· <u>_</u> j
150	' display state in ComboBox
151	
	<pre>cboViewPackages.Text = m_objPackage.State</pre>
152	
153	' when done, return to normal operating state
154	<pre>fraAddress.Enabled = False ' disable GroupBox</pre>
155	SetButtons(True) ' enable appropriate Buttons
156	
157	' change Button text from "Update" to "Edit"
158	<pre>btnEditUpdate.Text = "&amp;Edit"</pre>
159	End If
160	
	Fred Cub I have Ediation dealer Clinic
161	End Sub ' btnEditUpdate_Click
162	
163	' set package properties
164	<pre>Private Sub SetPackage()</pre>
165	m_objPackage.Address = txtAddress.Text
166	m_objPackage.City = txtCity.Text
167	<pre>m_objPackage.State = _</pre>
168	Convert.ToString(cboState.SelectedItem)
169	<pre>m_objPackage.Zip = Convert.ToInt32(Val(txtZip.Text))</pre>
170	End Sub ' SetPackage
171	Ellu Sub Selfackage
172	' load package information into Form
173	Private Sub LoadPackage()
174	
175	' retrieve package from list
176	<pre>m_objPackage = CType(m_objList.Item(m_intPosition), _</pre>
177	Package)
178	
179	' display package data
180	txtAddress.Text = m_objPackage.Address
181	
	<pre>txtCity.Text = m_objPackage.City</pre>
182	<pre>cboState.Text = m_objPackage.State</pre>
183	<pre>txtZip.Text = m_objPackage.Zip.ToString("00000")</pre>
184	lblArrivalTime.Text = _
185	m_objPackage.ArrivalTime.ToString
186	lblPackageNumber.Text = _
187	m_objPackage.PackageNumber.ToString
188	End Sub ' LoadPackage
189	
190	' clear all the input controls on the Form
191	Private Sub ClearControls()
192	txtAddress.Clear()
192	
	txtCity.Clear()
194	txtZip.Clear()
195	<pre>cboState.SelectedText = ""</pre>
196	lblArrivalTime.Text = ""
197	<pre>lblPackageNumber.Text = ""</pre>
198	End Sub ' ClearControls
199	
200	' enable/disable Buttons

```
201
        Private Sub SetButtons(ByVal blnState As Boolean)
202
           btnRemove.Enabled = blnState
203
           btnEditUpdate.Enabled = blnState
204
           btnNext.Enabled = blnState
205
           btnBack.Enabled = blnState
206
207
           ' disable navigation if not multiple packages
208
           If m_objList.Count < 2 Then</pre>
209
              btnNext.Enabled = False
210
              btnBack.Enabled = False
211
           End If
212
213
           ' if no items, disable Remove and Edit/Update Buttons
214
           If m_objList.Count = 0 Then
215
              btnEditUpdate.Enabled = False
216
              btnRemove.Enabled = False
217
           End If
218
        End Sub ' SetButtons
219
220
221
        ' event raised when user selects a new state in ComboBox
222
        223
           ByVal sender As System.Object, ByVal e As System.EventArgs) _
224
           Handles cboViewPackages.SelectedIndexChanged
225
226
           Dim objViewPackage As Package ' control variable package
227
           Dim strState As String = _
228
              Convert.ToString(cboViewPacakges.SelectedItem)
229
230
           lstPackages.Items.Clear() ' clear ListBox
231
232
           ' list all packages for current state in ListBox
233
           For Each objViewPackage In m_objList
234
235
              ' determine if state package is being shipped to
236
              ' matches the state selected in the ComboBox
237
              If objViewPackage.State = strState Then
238
239
                 ' add package number to the ListBox
240
                 lstPackages.Items.Add(objViewPackage.PackageNumber)
241
              End If
242
243
           Next
244
245
        End Sub ' cboViewPackages_SelectedIndexChanged
246
247
        ' display package information for selected package
248
        Private Sub lstPackages_DoubleClick(ByVal sender As _
249
           System.Object, ByVal e As System.EventArgs) _
250
           Handles lstPackages.DoubleClick
251
252
           Dim objPackageInformation As Package ' temporary package
253
           Dim strPackage As String = "" ' String for package information
254
255
           ' check if the lstPackages ListBox is empty
256
           If lstPackages.SelectedIndex <> -1 Then
257
258
              For Each objPackageInformation In m_objList
259
260
                 ' if the package currently in objPackageInformation
261
                 ' matches the user's selected package
```

262	<pre>If objPackageInformation.PackageNumber = _</pre>
263	Convert.ToInt32(lstPackages.SelectedItem) Then
264	strPackage &= ("Package " & _
265	objPackageInformation.PackageNumber & _
266	ControlChars.CrLf & _
267	"Arrived at: " & _
268	objPackageInformation.ArrivalTime & _
269	ControlChars.CrLf & _
270	"Address: " & _
271	objPackageInformation.Address & _
272	ControlChars.CrLf & _
273	"City: " & _
274	objPackageInformation.City & _
275	ControlChars.CrLf & _
276	<pre>"State: " &amp; objPackageInformation.State &amp; _</pre>
277	ControlChars.CrLf & _
278	"Zip code: " & _
279	<pre>objPackageInformation.Zip).ToString("00000")</pre>
280	
281	End If
282	
283	Next
284	
285	Else
286	
287	' if the user select a blank item in the ListBox
288	strPackage = "Please select a package"
289	End If
290	
291	<pre>MessageBox.Show(strPackage, "Package Information", _</pre>
292	MessageBoxButtons.OK, MessageBoxIcon.Information)
293	
294	End Sub ' lstPackages_DoubleClick
295	
296	End Class ' FrmShippingHub

**20.13** (*Controls Collection Application*) Visual Basic .NET provides many different types of collections. One such collection is the Controls collection, which is used to provide access to all of the controls on a Form. Create an application that uses the Controls collection and a For Each...Next loop to iterate through each control on the Form. As each control is encountered, add the control's name to a ListBox, and change the control's background color (in Fig. 20.29, Color.Wheat is used).

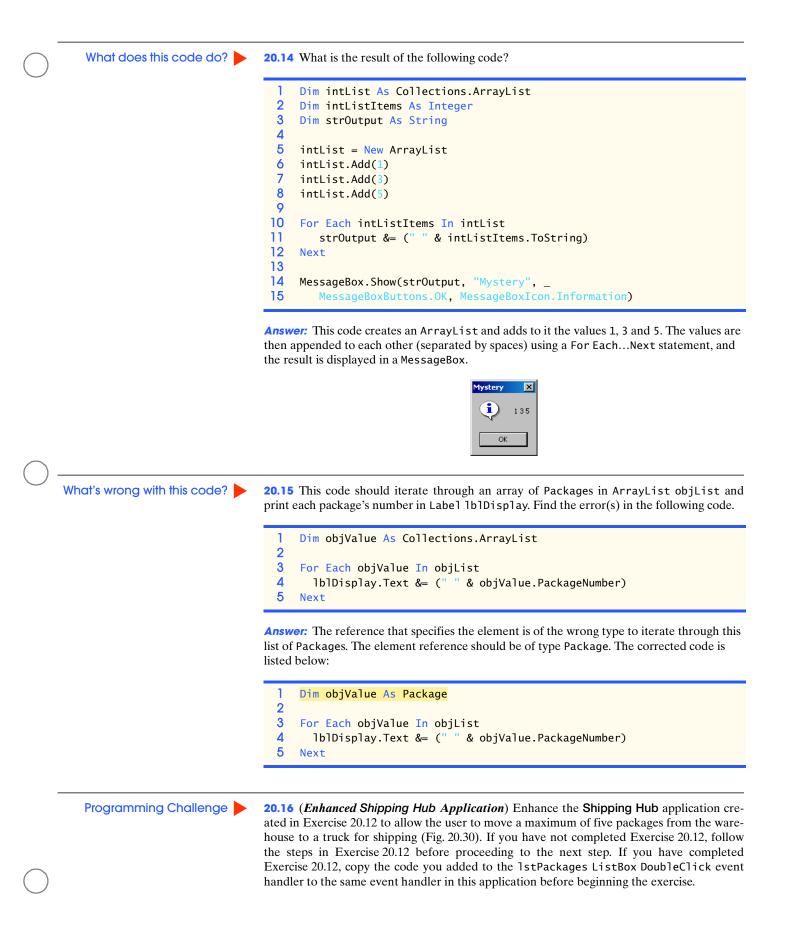
🔜 Contro	s Collection	<u>-                                    </u>
Name:		List of controls:
Book:	·	btnSubmit picBook cboBook
Picture:	BEREL BONDONE SHOES	IstList IstList Iblist IblCover IblBook IblName
		Submit



a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial20\Exercises\ControlsCollection directory to your C:\SimplyVB directory.

- b) **Opening the application's template file.** Double click ControlsCollection.sln in the ControlsCollection directory to open the application.
- c) *Generating an event handler*. Switch to Design view. Double click the **Submit** Button in design view to create an event handler for the click event.
- d) **Declaring a control variable**. Declare a reference of type Control. This reference represents each element in the For Each...Next statement as it iterates through each Control on the Form.
- e) *Clearing the ListBox*. To ensure that the information in the ListBox is updated each time the **Submit** Button is clicked, clear the ListBox of all items.
- f) *Writing a For Each...Next loop*. To create the For Each...Next loop, use the control variable that you created to iterate through the Form's Controls collection.
- g) Adding each control's name to the ListBox. Use the ListBox's Add method to insert the name of each control on the Form. Recall that a control's Name property contains the name of the control.
- h) Changing the control's background color. Use the Control's BackColor property to change the control's background color. Set the property to a new color using a member of the Color structure. [Hint: Type the word Color followed by the member-access operator to display a list of predefined colors using the Intellisense feature.] Note that the color of the PictureBox does not appear to change because its image displays in the control's foreground.
- i) *Running the application.* Select **Debug > Start** to run your application. Click the **Submit** Button. Verify that the controls' background colors change, and that all the controls are listed in the **List of controls:** ListBox.
- j) Closing the application. Close your running application by clicking its close box.
- k) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 20.13 Solution
2 3	' ControlsCollection.vb
4	Public Class FrmControlsCollection
5 6	Inherits System.Windows.Forms.Form
7	' Windows Form Designer generated code
8	
9 10	Private Sub btnSubmit_Click(ByVal sender As System.Object,
11	ByVal e As System.EventArgs) Handles btnSubmit.Click
12	Dim objControl As Control
13	
14 15	<pre>lstList.Items.Clear() ' clear ListBox</pre>
16 16	' iterate through controls collection
17	For Each objControl In Controls
8	5
19	' list name of each control
20	lstList.Items.Add(objControl.Name)
21 22	
22	' change background color
23 24	objControl.BackColor = Color.Wheat Next
25	Next
26	End Sub ' btnSubmit_Click
27 28	
20	End Class ' FrmControlsCollection



🚛 Shipping Hub		
Arrived at:	12/16/2002 10:18:33 AM	Packages by State
Package Informatio	n	SC 💌
Package ID:	22611	22611
Address: City:	9 Some Road Goose Creek State: SC Y Zip: 29445	Stip
< BACK	Scan New Add Remove Edit NEXT >	Packages to Ship  Package ID:  22604  22605  22606  22607

Figure 20.30 Enhanced Shipping Hub GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial20\Exercises\ShippingHubEnhanced directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click ShippingHubEnhanced.sln in the ShippingHubEnhanced directory to open the application.
- c) **Enabling the Ship Button.** The Ship Button should not be enabled until a package is selected in the lstPackage ListBox. Double click the lstPackage ListBox from design view to define the event handler. Use the Button's Enabled property to enable the Button if the SelectedIndex of the ListBox is not -1. This means that when the user selects a package from the ListBox, the user can send the package to the truck by clicking the Ship Button. Also, insert a line of code after the For Each...Next statement in the SelectedIndexChanged event handler to disable the Ship Button when a user chooses a different state.
- d) *Defining the Ship Button's Click Event*. Double click the Ship Button in Design View to define the Click event.
- e) *Incrementing the counter*. Because you are only allowing five packages to be "shipped," declare an instance variable that will track how many packages have been placed onto the truck. Increment the variable each time the **Ship** Button is clicked.
- f) Creating temporary variables. Create two temporary Package references to store the correct package's information. Use objTempPackage as the reference to the element in the collection type of a For Each...Next statement, and the objTruckPackage as a reference to the package added to the truck.
- g) *Using the If...Then...Else statement*. Use an If...Then...Else statement to allow packages to be placed onto the truck if the number of packages on the truck is less than five.
- h) Using the For Each...Next loop. Use a For Each...Next loop to iterate through the values in m\_objList. Each iteration should determine whether the current package is the one selected from the ListBox.
- Adding the package to the truck. When the For Each...Next loop has located the correct package, add that package to the truck by adding the reference to objTemp-Package to the truck's ArrayList, m\_objTruckList. Then assign the value in objTempPackage (the package sent to the truck) to objTruckPackage.
- j) Removing the package. When the For Each...Next loop completes, remove the package meant for the truck from m\_objList and the lstPackages ListBox.
- k) Displaying the package in the ListBox. Use a For Each...Next loop that iterates through each package in the m\_objTruckListArrayList and displays each package in the lstTruckListBox.

- Refreshing the GUI. Call the ClearControls and SetButtons methods to clear the TextBoxes and enable the appropriate Buttons. Also, set the Ship Button's Enabled property to False.
- m) *Coding the Else statement.* Display a MessageBox that notifies the user if the number of packages on the truck is already five. Then disable the **Ship** Button.
- n) Running the application. Select Debug > Start to run your application. Add several packages. In the Packages by State GroupBox, select several packages and add them to the Packages to Ship ListBox. Verify that you can add only 5 packages to this ListBox.
- o) *Closing the application.* Close your running application by clicking its close box.
- p) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
       Exercise 20.16 Solution
 2
     ' ShippingHub.vb
 3
 4
    Public Class FrmShippingHub
 5
        Inherits System.Windows.Forms.Form
 6
 7
        Private m_objList As Collections.ArrayList ' list of packages
 8
        Private m_objPackage As Package ' current package
       Private m_intPosition As Integer ' position of current package
Private m_objRandom As Random ' random number for package id
9
10
        Private m_intPackageID As Integer ' individual package number
11
12
        Private m_objTruckList As Collections.ArrayList ' shipment list
13
        Private m_intCounter As Integer = 0 ' count packages on truck
14
15
        ' Windows Form Designer generated code
16
17
        ' Form Load event
18
        Private Sub FrmShippingHub_Load(ByVal sender As _
19
           System.Object, ByVal e As System.EventArgs) _
20
           Handles MyBase.Load
21
22
           m_intPosition = 0 ' set initial position to zero
23
           m_objRandom = New Random ' create new Random object
24
           m_intPackageID = m_objRandom.Next(1, 100000) ' new package ID
25
26
           ' show first state in ComboBox (using the Items property)
27
           cboState.Text = Convert.ToString(cboState.Items.Item(0))
28
           m_objList = New Collections.ArrayList ' list of packages
29
           m_objTruckList = New Collections.ArrayList ' truck list
30
        End Sub ' FrmShippingHub_Load
31
32
        ' Scan New Button Click event
33
        Private Sub btnNew_Click(ByVal sender As System.Object, _
34
           ByVal e As System. EventArgs) Handles btnNew. Click
35
36
           m_intPackageID += 1 ' increment package ID
37
           m_objPackage = New Package(m_intPackageID) ' create package
38
39
           ClearControls() ' clear fields
40
           lblPackageNumber.Text = _
41
              m_objPackage.PackageNumber.ToString ' package number
42
           lblArrivalTime.Text = _
43
              m_objPackage.ArrivalTime.ToString ' display arrival time
44
45
           ' only allow user to add package
           fraAddress.Enabled = True ' disable GroupBox and its controls
46
```

<pre>48 btnAdd.Enabled = True ' enable Add Button 49 btnAdd.Enabled = False ' disable Scan New Button 50 txtAddress.Focus() ' transfer the focus to txtAddress TextBox 51 End Sub ' btnNew_Click 52 ' Add Button Click event 53 Private Sub btnAdd_Click(ByVal sender As System.Object, _ 54 ByVal e As System.EventArgs) Handles btnAdd.Click 55 SetPackage() ' set Package properties from TextBoxes 56 m_objList.Add(m_objPackage) ' add package to ArrayList 56 fraAddress.Enabled = False ' disable GroupBox and its controls 56 SetPackage() ' set Package properties Buttons 57 SetButtons(True) ' enable appropriate Buttons 58 ' package cannot be added until Scan New is clicked 59 btnAdd.Enabled = False ' disable Add Button 50 ' if package's state displayed, add ID to ListBox 50 / if cobSite.Text = chob/werPackage.Text Then 50 listPackages.Items.Add(m_objPackage.PackageNumber) 50 End If 51 cboViewPackages.Text = m_objPackage.State ' list packages 52 btnNew.Enabled = True ' enable Scan New Button 53 End Sub ' btnAdd_Click 54 ' move backward one package in the list 54 ' move backward one package in the list 55 ' move backward one package in the list 56 LoadPackage() ' load package data from item in list 56 End Sub ' btnBack_Click(ByVal sender As System.Object, _ 56 ByVal e As System.EventArgs) Handles btnNext.Click 56 LoadPackage() ' load package data from item in list 56 End Sub ' btnBack_Click(ByVal sender As System.Object, _ 57 ByVal e As System.EventArgs) Handles btnNext.Click 58 ' Next Button Click event 59 Private Sub btnNext_Click(ByVal sender As System.Object, _ 50 ByVal e As System.EventArgs) Handles btnNext.Click 59 ' Next Button Click event 50 Private Sub btnNext_Click(ByVal sender As System.Object, _ 50 ByVal e As System.EventArgs) Handles btnNext.Click 50 ' Next Button Click event 50 Private Sub btnNext_Click(ByVal sender As _ 51 System.Object, ByVal e As System.EventArgs) _ 50 LoadPackage() ' load package data from item in list 51 End Sub ' btnNext_Click(ByVal sender As _ 52 Sy</pre>	47	SetButtons(False) ' enable/disable Buttons
<pre>50 txtAddress.Focus() ' transfer the focus to txtAddress TextBox 51 End Sub ' btnNew_Click 52 53 ' Add Button Click event 54 55 Private Sub btnAdd_Click(ByVal sender As System.Object, _ 55 ByVal e As System.EventArgs) Handles btnAdd.Click 55 56 setPackage() ' set Package properties from TextBoxes 57 module as System.EventArgs) Handles btnAdd.Click 56 57 fraAddress.Enabled = False ' disable GroupBox and its controls 58 59 fraAddress.Enabled = False ' disable GroupBox and its controls 50 50 fraAddress.Enabled = False ' disable Add Button 50 50 fraAdd.Enabled = False ' disable Add Button 51 51 fooState.Text = choViewPackage.Text Then 52 53 frackages.state displayed, add ID to ListBox 54 froeState.Text = choViewPackage.Text Then 55 for LoState.Text = choViewPackage.State ' list packages 55 btnRew.Enabled = Frue ' enable Scan New Button 56 frighter System.EventArgs) Handles btnBack.Click 57 59 btnRew.Enabled = Frue ' enable Scan New Button 58 End Sub ' btnAdd_Click 57 59 frivate Sub btnBack.Click(ByVal sender As System.Object, _ 59 ByVal e As System.EventArgs) Handles btnBack.Click 57 59 frivate Sub btnBack.Click(ByVal sender As System.Object, _ 59 ByVal e As System.EventArgs) Handles btnBack.Click 57 59 frivate Sub btnBack.Click(ByVal sender As System.Object, _ 50 ByVal e As System.EventArgs) Handles btnBack.Click 58 59 frivate Sub btnRest.Click(ByVal sender As System.Object, _ 50 ByVal e As System.EventArgs) Handles btnNext.Click 59 50 frivate Sub btnRest.Click(ByVal sender As System.Object, _ 50 ByVal e As System.EventArgs) Handles btnNext.Click 50 50 frivate Sub btnRest.Click(ByVal sender As System.Object, _ 50 ByVal e As System.EventArgs) Handles btnNext.Click 50 50 frivate Sub btnRest.Click(ByVal sender As _ 50 System.Object, ByVal e As System.EventArgs) _ 50 50 Handles btnRemove.Click(ByVal sender As _ 50 System.Object, ByVal e As System.EventArgs) _ 50 50 Handles btnRemove.Click(ByVal sender As _ 50 System.Object, ByVal e As System.EventArgs) _ 50 50 Handles btnRemove.Click(ByVal sender As _ 50 System.O</pre>	48	<pre>btnAdd.Enabled = True ' enable Add Button</pre>
<pre>51 End Sub ' btnNew_Click 52 ' Add Button Click event 53 Private Sub btnAdd_Click(ByVal sender As System.Object, _ 54 ByVal e As System.EventArgs) Handles btnAdd.Click 55 SetPackage() ' set Package properties from TextBoxes 56 m_objList.Add(m_objPackage) ' add package to ArrayList 56 fraAddress.Enabled = False ' disable GroupBox and its controls 57 SetButtons(True) ' enable appropriate Buttons 58 ' package cannot be added until Scan New is clicked 59 btnAdd.Enabled = False ' disable Add Button 50 ' package cannot be added until Scan New is clicked 50 btnAdd.Enabled = False ' disable Add Button 51 SetButtons(True) ' enable appropriate Buttons 52 ' package state displayed, add ID to ListBox 53 If cboState.Text = cboViewPackages.Text Then 54 IstPackages.Text = m_objPackage.PackageNumber) 55 End If 56 ' Back Button Click event 57 ByVal e As System.EventArgs) Handles btnBack.Click 58 ' Back Button Click event 59 Frivate Sub btnBack_Click(ByVal sender As System.Object, _ 59 ByVal e As System.EventArgs) Handles btnBack.Click 50 ' move backward one package in the list 51 If m_intPosition = 1 52 Else ' wrap to end of list 53 m_intPosition = m_objList.Count - 1 54 End Sub ' btnBack_Click(ByVal sender As System.Object, _ 54 Wal e As System.EventArgs) Handles btnNext.Click 56 ' Next Button Click event 57 Private Sub btnNext_Click(ByVal sender As System.Object, _ 55 Wyal e As System.EventArgs) Handles btnNext.Click 56 ' Next Button Click event 57 Private Sub btnNext_Click(ByVal sender As System.Object, _ 55 Wyal e As System.EventArgs) Handles btnNext.Click 57 ' move forward one package in the list 58 If m_intPosition += 1 59 M_aintPosition = 0 ' wrap to beginning of list 59 End If 50 LoadPackage() ' load package data from item in list 50 Else 50 m_intPosition = 0 ' wrap to beginning of list 50 End If 50 LoadPackage() ' load package data from item in list 50 Else 50 m_intPosition = 0 ' wrap to beginning of list 50 End If 50 LoadPackage() ' load package data from item in list</pre>	49	btnNew.Enabled = False ' disable Scan New Button
<pre>51 End Sub ' btnNew_Click 52 ' Add Button Click event 53 Private Sub btnAdd_Click(ByVal sender As System.Object, _ 54 ByVal e As System.EventArgs) Handles btnAdd.Click 55 SetPackage() ' set Package properties from TextBoxes 56 m_objList.Add(m_objPackage) ' add package to ArrayList 56 fraAddress.Enabled = False ' disable GroupBox and its controls 57 SetButtons(True) ' enable appropriate Buttons 58 ' package cannot be added until Scan New is clicked 59 btnAdd.Enabled = False ' disable Add Button 50 ' package cannot be added until Scan New is clicked 50 btnAdd.Enabled = False ' disable Add Button 51 SetButtons(True) ' enable appropriate Buttons 52 ' package state displayed, add ID to ListBox 53 If cboState.Text = cboViewPackages.Text Then 54 IstPackages.Text = m_objPackage.PackageNumber) 55 End If 56 ' Back Button Click event 57 ByVal e As System.EventArgs) Handles btnBack.Click 58 ' Back Button Click event 59 Frivate Sub btnBack_Click(ByVal sender As System.Object, _ 59 ByVal e As System.EventArgs) Handles btnBack.Click 50 ' move backward one package in the list 51 If m_intPosition = 1 52 Else ' wrap to end of list 53 m_intPosition = m_objList.Count - 1 54 End Sub ' btnBack_Click(ByVal sender As System.Object, _ 54 Wal e As System.EventArgs) Handles btnNext.Click 56 ' Next Button Click event 57 Private Sub btnNext_Click(ByVal sender As System.Object, _ 55 Wyal e As System.EventArgs) Handles btnNext.Click 56 ' Next Button Click event 57 Private Sub btnNext_Click(ByVal sender As System.Object, _ 55 Wyal e As System.EventArgs) Handles btnNext.Click 57 ' move forward one package in the list 58 If m_intPosition += 1 59 M_aintPosition = 0 ' wrap to beginning of list 59 End If 50 LoadPackage() ' load package data from item in list 50 Else 50 m_intPosition = 0 ' wrap to beginning of list 50 End If 50 LoadPackage() ' load package data from item in list 50 Else 50 m_intPosition = 0 ' wrap to beginning of list 50 End If 50 LoadPackage() ' load package data from item in list</pre>		
<pre>52 53 'Add Button Click event 54 Private Sub btnAdd_Click(ByVal sender As System.Object, _ 55 ByVal e As System.EventArgs) Handles btnAdd.Click 56 57 SetPackage() 'set Package properties from TextBoxes 58 m_objlist.Add(m_objPackage) 'add package to ArrayList 59 50 fraAddress.Enabled = False 'disable GroupBox and its controls 51 SetButtons(True) 'enable appropriate Buttons 52 53 'package cannot be added until Scan New is clicked 54 btnAdd.Enabled = False 'disable Add Button 55 54 'package cannot be added until Scan New is clicked 55 'if package's state displayed, add ID to ListBox 56 'if packages.Items.Add(m_objPackage.Text Then 57 lcoState.Text = cboViewPackages.Text Then 58 lstPackages.Items.Add(m_objPackage.State ' list packages 59 btnAdd_Click 51 'Back Button Click event 52 'move backward one package in the list 53 m_intPosition = 1 54 lefs 'way to end of list 55 m_intPosition = 0 Then 56 lask ' btnBack_Click(ByVal sender As System.Object, _ 57 ByVal e As System.EventArgs) Handles btnBack.Click 52 'move backward one package in the list 53 m_intPosition = 1 54 lefs 'way to end of list 55 m_intPosition = 0 Then 56 m_intPosition = 0 Then 57 m_intPosition = 0 'wrap to beginning of list 58 end If 59 m_intPosition = 0 'wrap to beginning of list 59 m_intPosition = 0 'wrap to beginning of list 50 lefs 51 fm_intPosition = 0 'wrap to beginning of list 55 end If 55 end If 55 lefs 'wrap to end of list 55 m_intPosition = 0 'wrap to beginning of list 56 end If 57 end Sub 'btnRewt_Click(ByVal sender As _system.Object, _ 55 m_intPosition = 0 'wrap to beginning of list 56 end If 57 loadPackage() 'load package data from item in list 57 end Sub 'btnRewt_Click 57 'move Button click event 57 // ByVal e As System.EventArgs) Handles btnRewt.Click 57 'move forward one package data from item in list 58 end If 59 loadPackage() 'load package data from item in list 59 end If 50 loadPackage() 'load package data from item in list 59 end If 50 loadPackage() 'load package data from item in list 50 elds 'btnNext_Click 50 'Remove Button</pre>		
<pre>53 ' Add Button Click event 54 Private Sub btnAdd_Click(ByVal sender As System.Object, _ 55 ByVal e As System.EventArgs) Handles btnAdd.Click 56 57 SetPackage() ' set Package properties from TextBoxes 58 m.objList.Add(m_objPackage) ' add package to ArrayList 59 60 fraAddress.Enabled = False ' disable CroupBox and its controls 51 SetButtons(True) ' enable appropriate Buttons 52 63 ' package cannot be added until Scan New is clicked 64 btnAdd.Enabled = False ' disable Add Button 65 66 ' if package's state displayed, add ID to ListBox 67 If cboState.Text = cboViewPackages.Text Then 68 lstPackages.Items.Add(m_objPackage.PackageNumber) 69 End If 70 71 cboViewPackages.Text = m_objPackage.State ' list packages 72 btnNew.Enabled = True ' enable Scan New Button 73 End Sub ' btnAdd_Click 74 75 ' Back Button Click event 76 / rivate Sub btmRack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnBack.Click 78 79 ' move backward one package in the list 71 fintPosition = 1 79 Else ' wrap to end of list 70 m_intPosition = m_objList.Count - 1 71 End If 70 71 Next Button Click event 72 ' Next Button Click(ByVal sender As System.Object, _ 73 ByVal e As System.EventArgs) Handles btnNext.Click 74 75 ' Next Button Click event 76 ' nove forward one package in the list 76 If m_intPosition = m_objList.Count - 1 77 ByVal e As System.EventArgs) Handles btnNext.Click 78 79 ' move forward one package in the list 70 Private Sub btnNext_Click(ByVal sender As System.Object, _ 71 ByVal e As System.EventArgs) Handles btnNext.Click 73 74 ' move forward one package in the list 75 fm_intPosition = 0 ' wrap to beginning of list 75 End Sub ' btnNext_Click 76 76 m_intPosition = 0 ' wrap to beginning of list 76 End Sub ' btnNext_Click 77 78 Mandles btnRemove.Click(ByVal sender As _ 78 System.Object, ByVal e As System.EventArgs) _ 78 Handles btnRemove.Click event 79 Private Sub btnRemove.Click System.EventArgs) _ 78 Handles btnRemove.Click event 79 Private Sub btnRemove.Click event 70 Private Sub btnRemove.Click even</pre>		
<pre>54 Private Sub btnAdd_Click(ByVal sender As System.Object, _ 55 65 66 67 68 SetPackage() ' set Package properties from TextBoxes 68 m_objList.Add(m_objPackage) ' add package to ArrayList 69 60 61 fraAddress.Enabled = False ' disable GroupBox and its controls 62 63 ' package cannot be added until Scan New is clicked 64 btnAdd.Enabled = False ' disable Add Button 65 66 ' if package's state displayed, add ID to ListBox 67 If cboState.Text = cboViewPackages.Text Then 78 IstPackages.Items.Add(m_objPackage.State ' list packages 79 btnAdd.Enabled = Frue ' enable Scan New Button 70 End If 70 71 cboViewPackages.Text = m_objPackage.State ' list packages 72 btnAdd.Click 74 75 ' Back Button Click event 76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnBack.Click 79 ' move backward one package in the list 70 m_intPosition &gt; 0 Then 71 m_intPosition = m_objList.Count - 1 72 Else ' wrap to end of list 73 move forward one package in the list 74 if add If 75 76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnNext.Click 78 ' Next Button Click event 79 Private Sub btnNext_Click(ByVal sender As System.Object, _ 71 ByVal e As System.EventArgs) Handles btnNext.Click 79 ' move forward one package in the list 70 If m_intPosition = m_objList.Count - 1 71 ByVal e As System.EventArgs) Handles btnNext.Click 72 ' move forward one package in the list 73 If m_intPosition = 0 ' wrap to beginning of list 74 End If 75 End Sub btnNext_Click 76 ' Remove Button click event 77 Private Sub btnRemove.Click(ByVal sender As _ 78 System.Object, System.Object, Lick 79 ' move forward one package data from item in list 70 End If 70 LoadPackage() ' load package data from item in list 71 End Sub ' btnNext_Click 72 ' move forward one package in the list 73 fm.intPosition = 0 ' wrap to beginning of list 74 End Sub ' btnNext_Click 75 ' Remove Button click event 75 / Private Sub btnRemove.Click(ByVal sender As _ 75 / System.Object, ByVal e As System.E</pre>		' Add Button Click event
55       ByVal e As System.EventArgs) Handles btnAdd.Click         56       SetPackage() ' set Package properties from TextBoxes         57       SetPackage() ' set Package properties from TextBoxes         58       m_objList.Add(m_objPackage) ' add package to ArrayList         59       fraAddress.Enabled = False ' disable GroupBox and its controls         51       SetButtons(True) ' enable appropriate Buttons         52       ' package cannot be added until Scan New is clicked         54       btnAdd.Enabled = False ' disable Add Button         56       ' if package's state displayed, add ID to ListBox         57       If cboState.Text = cboViewPackages.Text Then         68       'stpackages.Text = m_objPackage.PackageNumber)         59       End If         70       cboViewPackages.Text = m_objPackage.State ' list packages         71       cboViewPackages.Text = m_objPackage.State ' list packages         72       btnNew.Enabled = True ' enable Scan New Button         73       cboViewPackages.Text = m_objPackage.State ' list packages         74       ' back Button Click event         75       ' Back Button Click event         76       ' move backward one package in the list         77       ByVal e As System.EventArgs) Handles btnNext.Click         78       ' Next Button Click event <td></td> <td></td>		
<pre>56 SetPackage() ' set Package properties from TextBoxes 57 SetPackage() ' set Package ' add package to ArrayList 58 m_objList.Add(m_objPackage) ' add package to ArrayList 59 fraAddress.Enabled = False ' disable GroupBox and its controls 50 SetButtons(True) ' enable appropriate Buttons 51 SetButtons(True) ' enable appropriate Buttons 52 ' package cannot be added until Scan New is clicked 52 btnAdd.Enabled = False ' disable Add Button 53 ' jackage's state displayed, add ID to ListBox 54 If cboState.Text = cboViewPackages.Text Then 55 If cboState.Text = cboViewPackage.State ' list packages 55 btnNew.Enabled = True ' enable Scan New Button 56 End If 57 cboViewPackages.Text = m_objPackage.State ' list packages 56 btnNew.Enabled = True ' enable Scan New Button 57 End Sub ' btnAdd_Click 57 Back Button Click event 58 Private Sub btnBack.Click(ByVal sender As System.Object, _ 59 ByVal e As System.EventArgs) Handles btnBack.Click 59 ' move backward one package in the list 50 If m_intPosition &gt; 0 Then 51 m_intPosition &gt; 0 Then 52 m_intPosition = m_objList.Count = 1 54 End If 55 LoadPackage() ' load package data from item in list 56 End Sub ' btnBack_Click 57 ' move forward one package in the list 58 If m_intPosition = m_objList.Count = 1 59 Private Sub btnNext_Click(ByVal sender As System.Object, _ 59 ByVal e As System.EventArgs) Handles btnNext.Click 56 ' next Button Click event 57 Private Sub btnNext_Click(ByVal sender As System.Object, _ 58 ByVal e As System.EventArgs) Handles btnNext.Click 59 ' move forward one package in the list 50 If m_intPosition = 0 ' wrap to beginning of list 50 End If 50 LoadPackage() ' load package data from item in list 50 End If 50 LoadPackage() ' load package data from item in list 50 End Sub ' btnNext_Click 50 Private Sub btnRemove.Click(ByVal sender As _ 50 System.Object, ByVal e As System.EventArgs) _ 50 Handles btnRemove.Click(ByVal sender As _ 50 System.Object, ByVal e As System.EventArgs) _ 51 Handles btnRemove.Click(ByVal sender As _</pre>		
57       SetPackage() ' set Package properties from TextBoxes         58       m_objlist.Add(m_objPackage) ' add package to ArrayList         59       fraAddress.Enabled = False ' disable GroupBox and its controls         51       SetButtons(True) ' enable appropriate Buttons         52       ' package cannot be added until Scan New is clicked         64       btnAdd.Enabled = False ' disable Add Button         65       ' if package's state displayed, add ID to ListBox         66       ' if packages.Text = cboViewPackages.Text Then         67       If cboState.Text = cboViewPackage.State ' list packages         68       lstPackages.Text = m_objPackage.State ' list packages         70       btnNew.Enabled = True ' enable Scan New Button         71       cboViewPackages.Text = m_objPackage.State ' list packages         72       btnNew.Enabled = True ' enable Scan New Button         73       End Sub ' btnAdd_Click         74       ' Back Button Click event         75       ' Back Button Click event         76       ' move backward one package in the list         77       ByVal e As System.EventArgs) Handles btnNat.Click         78       ' way to end of list         79       ' move backward one package data from item in list         76       LoadPackage() ' load package data from item in l		Byvar e AS System. EventArgs) handles blhadd. Crick
<pre>58 m_objList.Add(m_objPackage) ' add package to ArrayList 59 51 fraAddress.Enabled = False ' disable GroupBox and its controls 52 SetButtons(True) ' enable appropriate Buttons 53 ' package cannot be added until Scan New is clicked 54 btnAdd.Enabled = False ' disable Add Button 55 ' if package's state displayed, add ID to ListBox 56 ' if package's state displayed, add ID to ListBox 57 If cboState.Text = cboViewPackages.Text Then 58 lstPackages.Items.Add(m_objPackage.PackageNumber) 59 End If 50 cboViewPackages.Text = m_objPackage.State ' list packages 50 btnNew.Enabled = True ' enable Scan New Button 51 cboViewPackages.Text = m_objPackage.State ' list packages 52 btnNew.Enabled = True ' enable Scan New Button 53 End Sub ' btnAdd_Click 54 ' Back Button Click event 55 ' Back Button Click event 56 ' move backward one package in the list 56 ' move backward one package in the list 57 ' move backward one package data from item in list 58 End Sub ' btnBack_Click 59 ' move forward one package data from item in list 51 End Sub ' btnBack_Click 56 LoadPackage() ' load package data from item in list 57 End Sub ' btnBack_Click 58 ' Next Button Click event 59 ' nove forward one package in the list 50 If m_intPosition = 1 51 Else ' move forward one package in the list 51 If m_intPosition += 1 52 Else ' move forward one package in the list 53 If m_intPosition = 0' wrap to beginning of list 54 End If 55 End Sub ' btnNext_Click 56 LoadPackage() ' load package data from item in list 56 End If 57 m_intPosition = 0 ' wrap to beginning of list 58 End If 59 End If 50 LoadPackage() ' load package data from item in list 51 End Sub ' btnNext_Click 52 ' Remove Button click event 53 ' Remove Button click event 54 Private Sub btnRemove_Click(ByVal sender As_ 55 System.Object, ByVal e As System.EventArgs) _ 55 Handles btnRemove.Click</pre>		
<pre>fraAddress.Enabled = False ' disable GroupBox and its controls SetButtons(True) ' enable appropriate Buttons ' package cannot be added until Scan New is clicked btnAdd.Enabled = False ' disable Add Button ' if package's state displayed, add ID to ListBox If cboState.Text = cboViewPackages.Text Then lstPackages.Items.Add(m_objPackage.PackageNumber) End If cboViewPackages.Text = m_objPackage.State ' list packages btnNew.Enabled = True ' enable Scan New Button End Sub ' btnAdd_Click ' Back Button Click event Private Sub btnBack_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnBack.Click ' move backward one package in the list If m_intPosition &gt; 0 Then m_intPosition = 1 End Sub ' btnBack_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnBack.Click ' move backward one package in the list If m_intPosition = 1 End If LoadPackage() ' load package data from item in list End Sub ' btnBack_Click ' move forward one package in the list If m_intPosition = m_objList.Count - 1 End If ' Next Button Click event Private Sub btnNext_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnNext.Click ' move forward one package in the list If m_intPosition &lt; m_objList.Count - 1 Then m_intPosition += 1 Else ' move forward one package in the list If m_intPosition = 0 ' wrap to beginning of list End Sub ' btnNext_Click ' Remove Button click event Private Sub btnNext_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _ Handles btnRemove.Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _ Handles btnRemove.Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _ Handles btnRemove.Click(ByVal sender As System.Object, ByVal e As System.EventArgs) _ Handles btnRemove.Click</pre>		
<pre>60 fraAddress.Enabled = False ' disable GroupBox and its controls 61 SetButtons(True) ' enable appropriate Buttons 62 ' package cannot be added until Scan New is clicked 62 btnAdd.Enabled = False ' disable Add Button 63 ' if package's state displayed, add ID to ListBox 64 If cboState.Text = cboViewPackages.Text Then 65 lstPackages.Items.Add(m_objPackage.PackageNumber) 66 End If 70 cboViewPackages.Text = m_objPackage.State ' list packages 71 btnAdd_Click 72 btnNew.Enabled = True ' enable Scan New Button 73 End Sub ' btnAdd_Click 74 ' Back Button Click event 75 Private Sub btnBack_Click(ByVal sender As System.Object, _ 76 ByVal e As System.EventArgs) Handles btnBack.Click 77 move backward one package in the list 78 imutPosition &gt; 0 Then 79 mintPosition = 1 70 Else ' wrap to end of list 70 move backward one package data from item in list 71 Else ' wrap to end of list 72 move backage() ' load package data from item in list 73 End Sub ' btnBack_Click 74 ' move forward one package in the list 75 imutPosition = m_objList.Count - 1 76 End Sub ' btnBack_Click 77 move forward one package in the list 78 imitPosition &lt; m_objList.Count - 1 79 ByVal e As System.EventArgs) Handles btnNext.Click 70 ' move forward one package in the list 71 If m_intPosition &lt; m_objList.Count - 1 Then 72 m_intPosition = 0 ' wrap to beginning of list 78 End Sub ' btnNext_Click(ByVal sender As _ 79 MintPosition = 0 ' wrap to beginning of list 79 End If 70 LoadPackage() ' load package data from item in list 71 End Sub ' btnNext_Click 72 ' Remove Button click event 73 Private Sub btnNemove_Click(ByVal sender As _ 74 System.Object, ByVal e As System.EventArgs) _ 75 Andles btnRemove.Click(ByVal sender As _ 76 System.Object, ByVal e As System.EventArgs) _ 71 Andles btnRemove.Click</pre>		m_objList.Add(m_objPackage) ' add package to ArrayList
<pre>61 SetButtons(True) ' enable appropriate Buttons 62 ' package cannot be added until Scan New is clicked 63 btnAdd.Enabled = False ' disable Add Button 65 ' if package's state displayed, add ID to ListBox 66 ' if package's state displayed, add ID to ListBox 67 If cboState.Text = cboViewPackages.Text Then 68 lstPackages.Items.Add(m_objPackage.PackageNumber) 69 End If 70 cboViewPackages.Text = m_objPackage.State ' list packages 71 btnNew.Enabled = True ' enable Scan New Button 72 End Sub ' btnAdd_Click 73 End Sub ' btnAdd_Click 74 ' Back Button Click event 75 ' Back Button Click event 76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnBack.Click 78 ' move backward one package in the list 79 ' move backward one package in the list 70 m_intPosition &gt; 0 Then 71 m_intPosition = m_objList.Count - 1 72 Else ' wrap to end of list 73 m_intPosition = m_objList.Count - 1 74 End If 75 ' Next Button Click event 76 Private Sub btnNext_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnNext.Click 78 ' move forward one package in the list 79 If m_intPosition &lt; m_objList.Count - 1 Then 70 m_intPosition &lt;= 0 ' wrap to beginning of list 70 Else 71 m_intPosition = 0 ' wrap to beginning of list 72 End If 73 End Sub ' btnNext_Click(ByVal sender As _ 74 End Sub ' btnNext_Click 75 ' Remove Button click event 76 Private Sub btnNext_Click(ByVal sender As _ 76 System.Object, ByVal e As System.EventArgs) _ 76 Handles btnRemove.Click(ByVal sender As _ 77 System.Object, ByVal e As System.EventArgs) _ 70 Handles btnRemove.Click</pre>		
<pre>62 63 ' package cannot be added until Scan New is clicked 64 btnAdd.Enabled = False ' disable Add Button 65 66 ' if package's state displayed, add ID to ListBox 67 If cboState.Text = cboViewPackages.Text Then 68 1stPackages.Items.Add(m_objPackage.PackageNumber) 69 End If 70 71 cboViewPackages.Text = m_objPackage.State ' list packages 72 btnNew.Enabled = True ' enable Scan New Button 73 End Sub ' btnAdd_Click 74 75 ' Back Button Click event 76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnBack.Click 77 in ove backward one package in the list 78 if m_intPosition &gt; 0 Then 78 m_intPosition = m_objList.Count - 1 78 Else ' wrap to end of list 79 in Next Button Click event 79 Private Sub btnNext_Click(ByVal sender As System.Object, _ 70 ByVal e As System.EventArgs) Handles btnNext.Click 79 i move forward one package in the list 70 If m_intPosition &lt; m_objList.Count - 1 71 End Sub ' btnBack_Click 79 i move forward one package in the list 70 if m_intPosition &lt; m_objList.Count - 1 Then 71 m_intPosition &lt; m_objList.Count - 1 Then 72 m_intPosition = 0 ' wrap to beginning of list 73 End Sub ' btnNext_Click 74 If m_intPosition = 0 ' wrap to beginning of list 75 End Sub ' btnNext_Click 76 If Remove Button click event 77 Private Sub btnNewt_Click(ByVal sender As _ 77 System.Object, ByVal e As System.EventArgs) _ 77 Handles btnRemove.Click(ByVal sender As _ 78 System.Object, ByVal e As System.EventArgs) _ 79 Handles btnRemove.Click(ByVal sender As _ 70 System.Object, ByVal e As System.EventArgs) _ 70 Handles btnRemove.Click 70 if Remove Button click event 71 Set System.EventArgs) _ 71 Set System.Object, Click 72 if Remove.Click(ByVal sender As _ 73 System.Object, ByVal e As System.EventArgs) _ 74 Set System.Object, Click 75 if Remove Button click event 75 if Remove.Click(ByVal sender As _ 75 System.Object, ByVal e As System.EventArgs) _ 75 System.Object, ByVal e As System.EventArgs) _ 75 System.Object, ByVal e As System.EventArgs) _ 75 System.Object, ByVal exectement</pre>		
<pre>63 ' package cannot be added until Scan New is clicked 64 btnAdd.Enabled = False ' disable Add Button 65 ' if package's state displayed, add ID to ListBox 66 If CboState.Text = cboViewPackages.Text Then 77 1 btoPackages.Itext = m_objPackage.PackageNumber) 78 201 79 201 70 201 70 201 70 201 71 201 72 201 73 201 74 201 75 201 76 201 76 201 77 201 78 201 79 201 70 201 70 201 70 201 70 201 70 201 70 201 71 201 71 201 71 201 72 201 72 201 73 201 74 201 74 201 75 201 76 201 76 201 76 201 77 201 76 201 77 201 76 201 77 201 78 201 78 201 78 201 79 201 70 201 70 201 70 201 70 201 70 201 70 201 71 201 71 201 71 201 71 201 72 201 72 201 72 201 73 201 74 201 74 201 75 201 76 201 76 201 77 201 77 201 78 201 78 201 78 201 79 201 70 201</pre>		SetButtons(True) ' enable appropriate Buttons
<pre>64 btnAdd.Enabled = False ' disable Add Button 66 ' if package's state displayed, add ID to ListBox 67 If cboState.Text = cboViewPackages.Text Then 68 lstPackages.Items.Add(m_objPackage.PackageNumber) 69 End If 70 71 cboViewPackages.Text = m_objPackage.State ' list packages 72 btnNew.Enabled = True ' enable Scan New Button 73 End Sub ' btnAdd_Click 74 75 ' Back Button Click event 76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnBack.Click 78 79 ' move backward one package in the list 70 If m_intPosition &gt; 0 Then 71 m_intPosition = m_objList.Count = 1 72 Else ' wrap to end of list 73 m_intPosition = m_objList.Count = 1 74 End If 75 Next Button Click event 76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnNext.Click 78 79 ' Next Button Click event 70 Private Sub btnNext_Click(ByVal sender As System.Object, _ 70 ByVal e As System.EventArgs) Handles btnNext.Click 76 77 Next Button Click event 79 Private Sub btnNext_Click(ByVal sender As System.Object, _ 70 ByVal e As System.EventArgs) Handles btnNext.Click 71 If m_intPosition = 0 ' wrap to beginning of list 72 End If 73 End If 74 End If 75 MaintPosition = 0 ' wrap to beginning of list 76 End If 77 Memove Button click event 78 Private Sub btnNext_Click(ByVal sender As _ 79 System.Object, ByVal e As System.EventArgs) _ 70 LoadPackage() ' load package data from item in list 71 End Sub ' btnNext_Click 73 ' Remove Button click event 74 Private Sub btnRemove_Click(ByVal sender As _ 74 System.Object, ByVal e As System.EventArgs) _ 75 System.Object, ByVal e As System.EventArgs) _ 76 Handles btnRemove.Click</pre>		
<pre>66 67 ' if package's state displayed, add ID to ListBox 67 If cboState.Text = cboViewPackages.Text Then 68 IstPackages.Items.Add(m_objPackage.PackageNumber) 69 End If 70 70 CoViewPackages.Text = m_objPackage.State ' list packages 71 botNew.Enabled = True ' enable Scan New Button 72 End Sub ' btnAdd_Click 73 ' Back Button Click event 74 ' Back Button Click event 75 Orivate Sub btnBack_Click(ByVal sender As System.Object, _ 76 ByVal e As System.EventArgs) Handles btnBack.Click 77 ' move backward one package in the list 78 ' move backward one package in the list 79 ' move backward one package in the list 79 ' move backward one package at from item in list 70 Else ' wrap to end of list 70 m_intPosition = m_objList.Count - 1 71 End Sub ' btnBack_Click 73 74 Next Button Click event 75 Private Sub btnNext_Click(ByVal sender As System.Object, _ 75 ByVal e As System.EventArgs) Handles btnNext.Click 76 ' move forward one package in the list 77 If m_intPosition += 1 78 Else 77 move forward one package in the list 78 If m_intPosition += 1 79 Else 70 m_intPosition = 0 ' wrap to beginning of list 70 End Sub ' btnNext_Click(ByVal sender As _ 70 System.Object, ByVal e As System.EventArgs) _ 70 Handles btnRemove_Click(ByVal sender As _ 78 System.Object, ByVal e As System.EventArgs) _ 79 Handles btnRemove.Click(ByVal sender As _ 70 System.Object, ByVal e As System.EventArgs) _ 70 Handles btnRemove.Click</pre>	63	' package cannot be added until Scan New is clicked
<pre>66 ' if package's state displayed, add ID to ListBox 67 If cboState.Text = cboViewPackages.Text Then 68 IstPackages.Itext = m_objPackage.PackageNumber) 69 End If 70 70 71 cboViewPackages.Text = m_objPackage.State ' list packages 72 btnNew.Enabled = True ' enable Scan New Button 73 End Sub ' btnAdd_Click 74 75 ' Back Button Click event 76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnBack.Click 78 79 ' move backward one package in the list 70 If m_intPosition &gt; 0 Then 71 m_intPosition = 1 72 Else ' wrap to end of list 73 m_intPosition = m_objList.Count - 1 74 End If 75 Next Button Click event 76 Private Sub btnBack_Click 77 78 79 / Next Button Click event 79 / Next Button Click event 70 Private Sub btnNext_Click(ByVal sender As System.Object, _ 70 ByVal e As System.EventArgs) Handles btnNext.Click 79 / move forward one package in the list 70 If m_intPosition &lt; m_objList.Count - 1 Then 71 m_intPosition += 1 72 Else 73 / move forward one package in the list 74 If m_intPosition += 0 ' wrap to beginning of list 75 End If 76 LoadPackage() ' load package data from item in list 77 End If 78 End If 79 LoadPackage() ' load package data from item in list 79 End If 70 LoadPackage() ' load package data from item in list 70 End Sub ' btnNext_Click(ByVal sender As _ 71 System.Object, ByVal e As System.EventArgs) _ 76 Handles btnRemove.Click(ByVal sender As _ 77 System.Object, ByVal e As System.EventArgs) _ 70 Handles btnRemove.Click</pre>	64	<pre>btnAdd.Enabled = False ' disable Add Button</pre>
<pre>67  If cboState.Text = cboViewPackages.Text Then 68</pre>	65	
<pre>67  If cboState.Text = cboViewPackages.Text Then 68</pre>		' if package's state displayed, add ID to ListBox
<pre>68 lstPackages.Items.Add(m_objPackage.PackageNumber) 69 End If 70 71 cboViewPackages.Text = m_objPackage.State ' list packages 72 btnNew.Enabled = True ' enable Scan New Button 73 End Sub ' btnAd_Click 74 75 ' Back Button Click event 76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnBack.Click 78 79 ' move backward one package in the list 79 ' move backward one package in the list 79 ' move backward one package in the list 79 m_intPosition &gt; 0 Then 78 m_intPosition = 1 79 Else ' wrap to end of list 79 m_intPosition = m_objList.Count - 1 70 End If 70 71 End Sub ' btnBack_Click 73 74 End Sub ' btnBack_Click 75 End Sub ' btnBack_Click 76 ' Next Button Click event 76 Private Sub btnNext_Click(ByVal sender As System.Object, _ 79 ByVal e As System.EventArgs) Handles btnNext.Click 79 ' move forward one package in the list 71 If m_intPosition += 1 72 m_intPosition = 0 ' wrap to beginning of list 73 End Sub ' btnNext_Click 74 End Sub ' btnNext_Click 75 m_intPosition = 0 ' wrap to beginning of list 76 Else 77 m_intPosition = 0 ' wrap to beginning of list 78 End If 79 LoadPackage() ' load package data from item in list 70 LoadPackage() ' load package data from item in list 70 End Sub ' btnNext_Click 71 Private Sub btnNext_Click 72 ' Remove Button click event 73 ' Remove Button click event 74 Private Sub btnNext_Click 75 System.Object, ByVal e As System.EventArgs) _ 76 Handles btnRemove.Click(ByVal sender As _ 77 System.Object, ByVal e As System.EventArgs) _ 78 Handles btnRemove.Click</pre>		
<pre>69 End If 70 71 cboViewPackages.Text = m_objPackage.State ' list packages 72 btnNew.Enabled = True ' enable Scan New Button 73 End Sub ' btnAdd_Click 74 75 ' Back Button Click event 76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnBack.Click 78 79 ' move backward one package in the list 79 ' move backward one package in the list 79 imove backward one package in the list 79 move backward one package in the list 79 mintPosition -= 1 70 Else ' wrap to end of list 79 mintPosition = m_objList.Count - 1 70 Else ' wrap to end of list 70 mintPosition = m_objList.Count - 1 71 End If 72 Else ' wrap to end of list 73 move forkage() ' load package data from item in list 74 End Sub ' btnBack_Click 75 ' Next Button Click event 76 Private Sub btnNext_Click(ByVal sender As System.Object, _ 71 ByVal e As System.EventArgs) Handles btnNext.Click 72 ' move forward one package in the list 73 intPosition = 0 ' wrap to beginning of list 74 Else 75 m_intPosition = 0 ' wrap to beginning of list 75 End If 76 LoadPackage() ' load package data from item in list 77 End Sub ' btnNext_Click 70 LoadPackage() ' load package data from item in list 70 Else 71 Remove Button click event 71 Private Sub btnRemove_Click(ByVal sender As _ 72 System.Object, ByVal e As System.EventArgs) _ 73 Handles btnRemove.Click</pre>		
<pre>cboViewPackages.Text = m_objPackage.State ' list packages btnNew.Enabled = True ' enable Scan New Button End Sub ' btnAdd_Click ' Back Button Click event Private Sub btnBack_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnBack.Click ' move backward one package in the list ' move backward one package in the list ff m_intPosition &gt; 0 Then m_intPosition -= 1 Else ' wrap to end of list m_intPosition = m_objList.Count - 1 End If LoadPackage() ' load package data from item in list End Sub ' btnBack_Click ' Next Button Click event Private Sub btnNext_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnNext.Click ' move forward one package in the list if m_intPosition += 1 Else ' move forward one package in the list If m_intPosition += 1 Else m_intPosition = 0 ' wrap to beginning of list End If LoadPackage() ' load package data from item in list End If ' Remove Button click event ' Remove Button click event Private Sub btnNext_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ Handles btnRemove_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ Handles btnRemove.Click</pre>		
<pre>71 cboViewPackages.Text = m_objPackage.State ' list packages 72 btnNew.Enabled = True ' enable Scan New Button 73 End Sub ' btnAdd_Click 74 75 ' Back Button Click event 76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnBack.Click 78 79 ' move backward one package in the list 79 ' move backward one package data from item in list 79 End If 70 End If 70 End If 71 End Sub ' btnBack_Click 73 74 End Sub ' btnBack_Click 74 75 End Sub ' btnNext_Click(ByVal sender As System.Object, _ 74 ByVal e As System.EventArgs) Handles btnNext.Click 75 ' move forward one package in the list 76 If m_intPosition += 1 77 End Sub ' btnNext_Click(ByVal sender As System.Object, _ 78 m_intPosition = 0 ' wrap to beginning of list 78 End If 79 LoadPackage() ' load package data from item in list 70 LoadPackage() ' load package data from item in list 71 End Sub ' btnNext_Click 72 ' Remove Button click event 73 ' Remove Button click event 74 Private Sub btnRemove_Click(ByVal sender As _ 75 System.Object, ByVal e As System.EventArgs) _ 70 Handles btnRemove.Click</pre>		
<pre>btnNew.Enabled = True ' enable Scan New Button End Sub ' btnAdd_Click ' Back Button Click event Private Sub btnBack_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnBack.Click ' move backward one package in the list ' move backward one package in the list If m_intPosition -= 1 Else ' wrap to end of list m_intPosition = m_objList.Count - 1 End If LoadPackage() ' load package data from item in list End Sub ' btnBack_Click ' Next Button Click event Private Sub btnNext_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnNext.Click ' move forward one package in the list If m_intPosition += 1 Else m_intPosition = 0 ' wrap to beginning of list End If LoadPackage() ' load package data from item in list End If LoadPackage() ' load package data from item in list End If LoadPackage() ' load package data from item in list End Sub ' btnNext_Click ' Remove Button click event Private Sub btnRemove_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ Handles btnRemove.Click</pre>		chol/jewPackages Text - m objPackage State ' list packages
<pre>Final Sub ' btnAdd_Click Final Sub ' btnAdd_Click Final Sub ' btnAdd_Click Final Sub System.EventArgs) Handles btnBack.Click Final System.EventArgs) Handles btnBack.Click Final System.EventArgs) Handles btnBack.Click Final System.EventArgs) Handles btnBack.Click Final System System.EventArgs) Handles btnBack.Click Final System System.EventArgs) Handles btnBack_Click Final Sub ' btnBack_Click Final System.EventArgs) Handles btnNext.Click Final System.EventArgs) Handles btnNext.Click Final System.EventArgs) Handles btnNext.Click Final System System.EventArgs) Handles btnNext.Click Final System System System System.Final System Syst</pre>		· · · ·
<pre>// // // // // // // // // // // // //</pre>		
<pre>/5 ' Back Button Click event /7 Private Sub btnBack_Click(ByVal sender As System.Object, _ /7 ByVal e As System.EventArgs) Handles btnBack.Click /7 ' move backward one package in the list /7 ' move backward one package data from item in list /7 End If /8 LoadPackage() ' load package data from item in list /7 End Sub ' btnBack_Click /7 Next Button Click event /7 Private Sub btnNext_Click(ByVal sender As System.Object, _ /7 ByVal e As System.EventArgs) Handles btnNext.Click /7 move forward one package in the list /7 If m_intPosition &lt; m_objList.Count - 1 Then /7 m_intPosition += 1 /8 Else /7 m_intPosition = 0 ' wrap to beginning of list /8 End If /9 LoadPackage() ' load package data from item in list /8 End If /9 LoadPackage() ' load package data from item in list /7 Remove Button click event /7 rivate Sub btnRemove_Click(ByVal sender As _ /7 System.Object, ByVal e As System.EventArgs) _ /7 Handles btnRemove_Click</pre>		
<pre>76 Private Sub btnBack_Click(ByVal sender As System.Object, _ 77 ByVal e As System.EventArgs) Handles btnBack.Click 78 79 ' move backward one package in the list 80 If m_intPosition &gt; 0 Then 81 m_intPosition = 1 82 Else ' wrap to end of list 83 m_intPosition = m_objList.Count - 1 84 End If 85 86 LoadPackage() ' load package data from item in list 87 End Sub ' btnBack_Click 88 97 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 89 ' Next Button click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 ' move forward one package in the list 93 ' move forward one package in the list 94 If m_intPosition += 1 95 m_intPosition = 0 ' wrap to beginning of list 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 ' Remove Button click event 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 91 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click 107 108 ' Remove Button click event 109 ' handles btnRemove.Click 100 ' Remove Button click event 101 Private Sub btnRemove.Click 102 ' Remove Button click event 103 ' Remove Button click event 104 Private Sub btnRemove.Click 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click 107 108 ' Button Click event 109 ' Button Click event 109 ' Button Click event 109 ' Button Click event 100 ' Button Click event 100 ' Button Click event 101 ' Button Click event 102 ' Button Click event 103 ' Remove Button Click event 104 Private Sub btnRemove.Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 ' Button Click event 107 ' Button Click event 108 ' Button Click event 109 ' Button Click event 109 ' Button Click event 100 ' Button Click event 101 ' Button Click event 102 ' Bu</pre>		L Back Button Click sugat
<pre>77 ByVal e As System.EventArgs) Handles btnBack.Click 78 79 ' move backward one package in the list 80 If m_intPosition &gt; 0 Then 81 m_intPosition -= 1 82 Else ' wrap to end of list 83 m_intPosition = m_objList.Count - 1 84 End If 85 86 LoadPackage() ' load package data from item in list 87 End Sub ' btnBack_Click 88 97 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition = 0 ' wrap to beginning of list 88 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click </pre>		
<pre>78 79 79 79 79 79 79 79 70 70 70 70 70 70 70 70 70 70 70 70 70</pre>		
<pre>79 ' move backward one package in the list 80 If m_intPosition &gt; 0 Then 81 m_intPosition -= 1 82 Else ' wrap to end of list 83 m_intPosition = m_objList.Count - 1 84 End If 85 86 LoadPackage() ' load package data from item in list 87 End Sub ' btnBack_Click 88 99 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 95 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		ByVal e As System.EventArgs) Handles btnBack.Click
<pre>80  If m_intPosition &gt; 0 Then 81  m_intPosition -= 1 82  Else ' wrap to end of list 83  m_intPosition = m_objList.Count - 1 84  End If 85 86  LoadPackage() ' load package data from item in list 87  End Sub ' btnBack_Click 88 89  ' Next Button Click event 90  Private Sub btnNext_Click(ByVal sender As System.Object, _ 91  ByVal e As System.EventArgs) Handles btnNext.Click 92 93  ' move forward one package in the list 94  If m_intPosition &lt; m_objList.Count - 1 Then 95  m_intPosition += 1 96  Else 97  m_intPosition = 0 ' wrap to beginning of list 98  End If 99 100  LoadPackage() ' load package data from item in list 101  End Sub ' btnNext_Click 102 103  ' Remove Button click event 104  Private Sub btnRemove_Click(ByVal sender As _ 97  System.Object, ByVal e As System.EventArgs) _ 106  Handles btnRemove.Click</pre>		
<pre>81 m_intPosition -= 1 82 Else ' wrap to end of list 83 m_intPosition = m_objList.Count - 1 84 End If 85 86 LoadPackage() ' load package data from item in list 87 End Sub ' btnBack_Click 88 89 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 95 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>82 Else ' wrap to end of list 83 m_intPosition = m_objList.Count - 1 84 End If 85 86 LoadPackage() ' load package data from item in list 87 End Sub ' btnBack_Click 88 89 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 95 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>83 m_intPosition = m_objList.Count - 1 84 End If 85 86 LoadPackage() ' load package data from item in list 87 End Sub ' btnBack_Click 88 89 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 88 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 97 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>84 End If 85 86 LoadPackage() ' load package data from item in list 87 End Sub ' btnBack_Click 88 89 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 97 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>85 86 LoadPackage() ' load package data from item in list 87 End Sub ' btnBack_Click 88 89 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 97 System.Object, ByVal e As System.EventArgs) _ 98 Handles btnRemove.Click</pre>	83	$m_{intPosition} = m_{objList.Count} - 1$
<pre>86 LoadPackage() ' load package data from item in list 87 End Sub ' btnBack_Click 88 89 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 97 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>	84	End If
<pre>87 End Sub ' btnBack_Click 88 89 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 97 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>	85	
<pre>87 End Sub ' btnBack_Click 88 89 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 97 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		LoadPackage() ' load package data from item in list
<pre>88 89 ' Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 595stem.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>89 'Next Button Click event 90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 'move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 'wrap to beginning of list 98 End If 99 100 LoadPackage() 'load package data from item in list 101 End Sub 'btnNext_Click 102 103 'Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>90 Private Sub btnNext_Click(ByVal sender As System.Object, _ 91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		' Next Button Click event
<pre>91 ByVal e As System.EventArgs) Handles btnNext.Click 92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 5ystem.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>92 93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>93 ' move forward one package in the list 94 If m_intPosition &lt; m_objList.Count - 1 Then 95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		Spear a no system eventingsy handles bellicketerrek
<pre>94  If m_intPosition &lt; m_objList.Count - 1 Then 95  m_intPosition += 1 96  Else 97  m_intPosition = 0 ' wrap to beginning of list 98  End If 99 100  LoadPackage() ' load package data from item in list 101  End Sub ' btnNext_Click 102 103 ' Remove Button click event 104  Private Sub btnRemove_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ 106  Handles btnRemove.Click</pre>		' move forward one package in the list
<pre>95 m_intPosition += 1 96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>96 Else 97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		-
<pre>97 m_intPosition = 0 ' wrap to beginning of list 98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		—
<pre>98 End If 99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>99 100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>100 LoadPackage() ' load package data from item in list 101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		End It
<pre>101 End Sub ' btnNext_Click 102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>102 103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>103 ' Remove Button click event 104 Private Sub btnRemove_Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		End Sub ' btnNext_Click
<pre>104 Private Sub btnRemove_Click(ByVal sender As _ 105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>		
<pre>105 System.Object, ByVal e As System.EventArgs) _ 106 Handles btnRemove.Click</pre>	103	' Remove Button click event
106 Handles btnRemove.Click	104	Private Sub btnRemove_Click(ByVal sender As _
	105	System.Object, ByVal e As System.EventArgs) _
107	106	Handles btnRemove.Click
	107	

```
108
             remove ID from ListBox if state displayed
109
           If cboState.Text = cboViewPackages.Text Then
110
              lstPackages.Items.Remove(m_objPackage.PackageNumber)
111
           End If
112
113
           m_objList.RemoveAt(m_intPosition) ' remove package from list
114
115
           ' load next package in list if there is one
116
           If m_objList.Count > 0 Then
117
118
               ' if not at first position, go to previous one
119
              If m_intPosition > 0 Then
120
                 m_intPosition -= 1
121
              End If
122
123
              LoadPackage() ' load package data from item in list
124
           Else
125
              ClearControls() ' clear fields
126
           End If
127
128
           SetButtons(True) ' enable appropriate Buttons
129
        End Sub ' btnRemove_Click
130
131
        ' Edit/Update Button Click event
132
        Private Sub btnEditUpdate_Click(ByVal sender As _
133
           System.Object, ByVal e As System.EventArgs) _
134
           Handles btnEditUpdate.Click
135
136
            ' when Button reads "Edit", allow user to
            ' edit package information only
137
138
           If btnEditUpdate.Text = "&Edit" Then
139
              fraAddress.Enabled = True
140
              SetButtons(False)
141
              btnEditUpdate.Enabled = True
142
143
               ' change Button text from "Edit" to "Update"
144
              btnEditUpdate.Text = "&Update"
145
           Else
146
147
               ' when Button reads "Update" remove the old package
148
               ' data and add new data from TextBoxes
149
              SetPackage()
150
              m_objList.RemoveAt(m_intPosition)
151
              m_objList.Insert(m_intPosition, m_objPackage)
152
153
               ' display state in ComboBox
154
              cboViewPackages.Text = m_objPackage.State
155
156
               ' when done, return to normal operating state
              fraAddress.Enabled = False ' disable GroupBox
157
158
              SetButtons(True) ' enable appropriate Buttons
159
160
               ' change Button text from "Update" to "Edit"
              btnEditUpdate.Text = "&Edit"
161
162
           End If
163
164
        End Sub ' btnEditUpdate_Click
165
166
        ' set package properties
167
        Private Sub SetPackage()
168
           m_objPackage.Address = txtAddress.Text
```

169	m_objPackage.City = txtCity.Text
170	m_objPackage.State = _
171	Convert.ToString(cboState.SelectedItem)
172	m_objPackage.Zip = Convert.ToInt32(Val(txtZip.Text))
173	End Sub ' SetPackage
174	·
175	' load package information into Form
176	Private Sub LoadPackage()
177	
178	' retrieve package from list
179	<pre>m_objPackage = CType(m_objList.Item(m_intPosition), _</pre>
180	Package)
181	Tackage)
182	' display package data
183	txtAddress.Text = m_objPackage.Address
184	
185	<pre>txtCity.Text = m_objPackage.City chaCtate Taxt = m_objPackage State</pre>
	<pre>cboState.Text = m_objPackage.State tutZin Tout = m_objPackage.Zin ToStmins("000000")</pre>
186	<pre>txtZip.Text = m_objPackage.Zip.ToString("00000") ltlActionalTime_Text</pre>
187	lblArrivalTime.Text = _
188	m_objPackage.ArrivalTime.ToString
189	<pre>lblPackageNumber.Text = _</pre>
190	m_objPackage.PackageNumber.ToString
191	End Sub ' LoadPackage
192	
193	' clear all the input controls on the Form
194	Private Sub ClearControls()
195	txtAddress.Clear()
196	txtCity.Clear()
197	txtZip.Clear()
198	<pre>cboState.SelectedText = ""</pre>
199	lblArrivalTime.Text = ""
200	lblPackageNumber.Text = ""
201	End Sub ' ClearControls
202	
203	' enable/disable Buttons
204	Private Sub SetButtons(ByVal blnState As Boolean)
205	<pre>btnRemove.Enabled = blnState</pre>
206	<pre>btnEditUpdate.Enabled = blnState</pre>
207	<pre>btnNext.Enabled = blnState</pre>
208	btnBack.Enabled = blnState
209	
210	' disable navigation if not multiple packages
211	<pre>If m_objList.Count &lt; 2 Then</pre>
212	btnNext.Enabled = False
213	btnBack.Enabled = False
214	End If
215	
216	' if no items, disable Remove and Edit/Update Buttons
217	<pre>If m_objList.Count = 0 Then</pre>
218	btnEditUpdate.Enabled = False
219	btnRemove.Enabled = False
220	End If
221	
222	End Sub ' SetButtons
223	
224	' event raised when user selects a new state in ComboBox
225	Private Sub cboViewPackages_SelectedIndexChanged( _
226	ByVal sender As System.Object, ByVal e As System.EventArgs) _
220	Handles cboViewPackages.SelectedIndexChanged
228	nanules cooview acrages. Selecteurinex Changen
220	Dim objViewPackage As Package ' control variable package
227	UTIL UN IVTEWFACKAGE AS FACKAGE CONTROL VARIADIE PACKAGE

230	Dim strState As String = _
231	Convert.ToString(cboViewPacakges.SelectedItem)
232	
233	<pre>lstPackages.Items.Clear() ' clear ListBox</pre>
234	Istrackages.items.crear() crear Listbox
235	' list all packages for current state in ListBox
236	For Each objViewPackage In m_objList
237	
238	' determine if state package is being shipped to
239	' matches the state selected in the ComboBox
240	<pre>If objViewPackage.State = strState Then</pre>
241	
242	' add package number to the ListBox
243	lstPackages.Items.Add(objViewPackage.PackageNumber)
244	End If
245	
	Next
246	Next
247	
248	<pre>btnShip.Enabled = False ' disable Ship Button</pre>
249	
250	End Sub ' cboViewPackages_SelectedIndexChanged
251	
	I disalering problems information for collected problems
252	' displaying package information for selected package
050	
253	Private Sub lstPackages_DoubleClick(ByVal sender As _
254	Object, ByVal e As System.EventArgs) _
255	Handles lstPackages.DoubleClick
256	
257	Dim objPackageInformation As Package ' temporary package
258	Dim strPackage As String = "" ' String for package information
259	bin striackage As string – string for package information
260	' check if the lstPackages ListBox is empty
261	<pre>If lstPackages.SelectedIndex &lt;&gt; -1 Then</pre>
262	
263	For Each objPackageInformation In m_objList
264	
265	' if the package currently in objPackageInformation
266	
	' matches the user's selected package
267	<pre>If objPackageInformation.PackageNumber = _</pre>
268	Convert.ToDouble(lstPackages.SelectedItem) Then
269	strPackage &= "Package " & _
270	<pre>objPackageInformation.PackageNumber &amp; _</pre>
271	ControlChars.CrLf & _
272	"Arrived at: " & _
272	
	objPackageInformation.ArrivalTime & _
274	ControlChars.CrLf & _
275	"Address: " & _
276	objPackageInformation.Address & _
277	ControlChars.CrLf & _
278	"City: " & _
279	objPackageInformation.City & _
	· · ·
280	ControlChars.CrLf & _
281	"State: " & objPackageInformation.State & _
282	ControlChars.CrLf & _
283	"Zip code: " & _
284	objPackageInformation.Zip.ToString("00000"))
285	
286	End If
287	
288	Next
288 289	Next

290	Else
291	
292	' if the user select a blank item in the ListBox
293	<pre>strPackage = "Please select a package"</pre>
294	End If
295	
	Manager Dev Charles Declares - "Declares - Ta Compation"
296	<pre>MessageBox.Show(strPackage, "Package Information", _</pre>
297	MessageBoxButtons.OK, MessageBoxIcon.Information)
298	
299	End Sub ' lstPackages_DoubleClick
300	
301	' allow packages to be shipped
302	Private Sub btnShip_Click(ByVal sender As _
303	System.Object, ByVal e As System.EventArgs) _
304	Handles btnShip.Click
305	
306	Dim shiTompDackaga Ac Dackaga ' tomponany packaga
307	Dim objTempPackage As Package ' temporary package
	Dim objTruckPackage As Package ' package to remove
308	
309	<pre>m_intCounter += 1 ' increment package count</pre>
310	
311	' if there is less than 6 packages in m_intCounter
312	<pre>If m_intCounter &lt;= 5 Then</pre>
313	' for each package from the m_objList ArrayList
314	For Each objTempPackage In m_objList
315	
316	' move package to truck
317	<pre>If objTempPackage.PackageNumber = _</pre>
318	Convert.ToInt32(1stPackages.SelectedItem) Then
319	m_objTruckList.Add(objTempPackage)
320	
321	objTruckPackage = objTempPackage
322	End If
323	Next
324	
325	' remove the package from warehouse
326	m_objList.Remove(objTruckPackage)
327	
328	' remove selected package
329	<pre>lstPackages.Items.Remove(lstPackages.SelectedItem)</pre>
330	
331	lstTruck.Items.Clear() ' clear ListBox
332	<pre>lstTruck.Items.Add("Package ID:") ' add header</pre>
333	
334	' list all packages in ListBox
335	For Each objViewPackage In m_objTruckList
336	for Each objerem ackage in m_objerackerse
337	' add package to lstTruck ListBox
338	
339	lstTruck.Items.Add(objViewPackage.PackageNumber)
	Next
340	
341	<pre>btnShip.Enabled = False ' disable the Ship Button</pre>
342	ClearControls() ' clear the TextBoxes
343	SetButtons(True) ' enable appropriate Buttons
344	
345	Else
346	<pre>MessageBox.Show("Truck can only hold 5 packages", _</pre>
347	"Limit Exceeded", _
348	MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
349	,, <u>,</u> ,
350	<pre>btnShip.Enabled = True ' enable the Ship Button</pre>

351	End If
352	
353	End Sub ' btnShip_Click
354	
355	' disable Ship button when no package is selected
356	<pre>Private Sub lstPackages_SelectedIndexChanged(ByVal sender As _</pre>
357	System.Object, ByVal e As System.EventArgs) _
358	Handles lstPackages.SelectedIndexChanged
359	
360	<pre>If lstPackages.SelectedIndex &lt;&gt; -1 Then</pre>
361	<pre>btnShip.Enabled = True ' enable the Ship Button</pre>
362	End If
363	
364	End Sub ' 1stPackages_SelectedIndexChanged
365	
366	End Class ' FrmShippingHub





# "Cat and Mouse" Painter Application

Introducing the Graphics Object and Mouse Events Solutions

Instructor's Manual Exercise Solutions Tutorial 21			
MULTIPLE-CHOICE	<b>21.1</b> The <i>x</i> - and <i>y</i> -coordinates of the MouseEventArgs object are relative to		
QUESTIONS	<ul><li>a) the screen</li><li>c) the Form or control that contains the o</li><li>d) None of the above.</li></ul>	b) the application control that raised the event	
	<b>21.2</b> The method of the Graphics class draws a solid ellipse.		
	a) FillEllipse	b) Ellipse	
	c) SolidEllipse	d) FilledEllipse	
	<b>21.3</b> The object passed to a mouse event handler contains information about the mouse event that was raised.		
	a) EventHandler	b) MouseEventHandler	
	c) MouseEventArgs	d) EventArgs	
	21.4 The event is raised when a n	nouse button is pressed.	
	a) MousePress	b) MouseClick	
	c) MouseDown	d) MouseButtonDown	
	<b>21.5</b> A is used to fill a shape with color using a Graphics object.		
	a) painter	b) brush	
	c) paint bucket	d) marker	
	<b>21.6</b> A(n) event is raised every ti	me the mouse interacts with a control.	
	a) control	b) mouse pointer	
	c) mouse	d) user	
	<b>21.7</b> The property of MouseEv pressed.	entArgs specifies which mouse button was	
	a) Source	b) Button	
	c) WhichButton	d) ButtonPressed	
	<b>21.8</b> The class contains methods shapes.	s for drawing text, lines, rectangles and other	
	a) Pictures	b) Drawings	
	c) Graphics	d) Illustrations	
	<b>21.9</b> An ellipse with its is a circle.		
	a) height twice the length of its width	b) width set to zero	
	c) height half the length of its width	d) height equal to its width	
	21.10 The method creates a Grap	hics object.	
	a) NewGraphics	b) CreateGraphics	
	c) PaintGraphics	d) InitializeGraphics	
	<b>Answers:</b> 21.1) c. 21.2) a. 21.3) c. 21.4) c. 21	.5) b. 21.6) c. 21.7) b. 21.8) c. 21.9) d. 21.10) b.	
	Exercise Solutions Tutorial 21 MULTIPLE-CHOICE	Exercise Solutions Tutorial 21         MULTIPLE-CHOICE QUESTIONS       21.1 The x- and y-coordinates of the MouseEx a) the screen c) the Form or control that contains the of d) None of the above.         21.2 The method of the Graphic: a) FillEllipse c) SolidEllipse         21.3 The object passed to a mous mouse event that was raised. a) EventHandler c) MouseEventArgs         21.4 The event is raised when a m a) MousePress c) MouseBound         21.5 A is used to fill a shape with a) painter c) paint bucket         21.6 A(m) event is raised every ti a) control c) mouse         21.7 The property of MouseEventers         21.8 The class contains methods shapes. a) Source c) WhichButton         21.8 The class contains methods shapes. a) Pictures c) Graphics         21.9 An ellipse with its is a circle. a) height twice the length of its width c) height half the length of its width c) height half the length of its width c) height half the length of its width	

EXERCISES

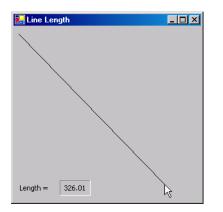
**21.11** (*Line Length Application*) The Line Length application should draw a straight black line on the Form and calculate the length of the line(Fig. 21.27). The line should begin at the coordinates where the mouse button is pressed and should stop at the point where the mouse button is released. The application should display the line's length (that is, the distance between the two endpoints) in the Label Length =. Use the following formula to calculate

the line's length, where  $(x_1, y_1)$  is the first endpoint (the coordinates where the mouse button is pressed) and  $(x_2, y_2)$  is the second endpoint (the coordinates where the mouse button is released). To calculate the distance (or length) between the two points, use the following equation:

$$d = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

To draw a straight line, you need to use the **DrawLine** method on a Graphics object. When drawing lines, you should use a **Pen** object, which is an object used to specify characteristics of lines and curves. Use the following method call to draw a black line between the two points using a Graphics object reference objGraphic:

objGraphic.DrawLine(New Pen(Color.Black),  $x_1$ ,  $y_1$ ,  $x_2$ ,  $y_2$ )





- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial21\Exercises\LineLength directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click LineLength.sln in the Line-Length directory to open the application.
- c) *Declaring instance variables.* Declare and initialize a reference to a Graphics object that you will use to draw a line. Then declare four Integers in which you will store the *x* and *y*-coordinates of the two points.
- d) *Adding a MouseDown event handler.* Create a MouseDown event handler. Add code to the MouseDown event handler to store the coordinates of the first endpoint of the line.
- e) Creating the Distance method. Define a Function procedure named Length that returns the distance between two endpoints as a Double. The Function procedure should use the following statement to perform the line length calculation, where intXDistance is the difference between the x-coordinates of the two points and intYDistance is the difference between the y-coordinates of the two points:

Math.Sqrt((intXDistance ^ 2) + (intYDistance ^ 2))

- f) Adding a MouseUp event handler. Create a MouseUp event handler. First store the coordinates of the line's second endpoint. Then call the Length method to obtain the distance between the two endpoints (the line's length). Finally, display the line on the Form and the line's length in the Length = Label, as in Fig. 21.27.
- g) *Running the application.* Select **Debug > Start** to run your application. Draw several lines and view their lengths. Verify that the length values are accurate.
- h) Closing the application. Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 21.11 Solution
 2
     ' LineLength.vb
 3
 4
    Public Class FrmLineLength
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' create and initialize Graphics object
 8
       Private m_objGraphic As Graphics = CreateGraphics()
 9
10
       Private m_intX1 As Integer ' first point's x-coordinate
       Private m_intY1 As Integer ' first point's y-coordinate
11
       Private m_intX2 As Integer ' second point's x-coordinate
12
       Private m_intY2 As Integer ' second point's y-coordinate
13
14
        ' Windows Form Designer generated code
15
16
17
        ' handles FrmLineLength's MouseUp event
18
       Private Sub FrmLineLength_MouseDown(ByVal sender As Object, _
19
          ByVal e As System.Windows.Forms.MouseEventArgs) _
20
          Handles MyBase.MouseDown
21
22
          lblLength.Text = "" ' clear Label
23
24
           ' get x- and y- coordinates of mouse click
25
          m intX1 = e.X
26
          m_intY1 = e.Y
27
       End Sub ' FrmLineLength_MouseDown
28
29
        ' returns distance between two points
30
       Private Function Length() As Double
31
32
           ' horizontal distance
33
          Dim intXDistance As Integer = m_intX1 - m_intX2
34
35
           ' vertical distance
36
          Dim intYDistance As Integer = m_intY1 - m_intY2
37
38
          Return Math.Sqrt((intXDistance ^ 2) + (intYDistance ^ 2))
39
       End Function ' Length
40
41
        ' handles FrmLineLength's MouseUp event
42
       Private Sub FrmLineLength_MouseUp(ByVal sender As Object, _
43
          ByVal e As System.Windows.Forms.MouseEventArgs) _
44
          Handles MyBase.MouseUp
45
46
           ' final point
47
          m_intX2 = e.X
48
          m_intY2 = e.Y
49
50
           ' distance between two points
          Dim dblDistance As Double = Length()
51
52
53
           ' draw line connecting the two points
54
          m_objGraphic.DrawLine(New Pen(Color.Black), _
55
             m_intX1, m_intY1, m_intX2, m_intY2)
56
57
           ' display distance in Label
           lblLength.Text = String.Format("{0:F}", dblDistance)
58
59
       End Sub ' FrmLineLength_MouseUp
```

60

## 61 End Class ' FrmLineLength

**21.12** (*Circle Painter Application*) The Circle Painter application should draw a blue circle with a randomly chosen size when the user presses a mouse button anywhere over the Form (Fig. 21.28). The application should randomly select a circle diameter in the range from 5 to 199, inclusive. To draw a blue circle with a given diameter (intDiameter), use the following statement:

```
objGraphic.DrawEllipse(New Pen(Color.Blue), e.X, e.Y, _
intDiameter, intDiameter)
```

The DrawEllipse method, when passed a Pen (instead of a brush) as an argument, draws the outline of an ellipse. Recall that an ellipse is a circle if the height and width arguments are the same (in this case, the randomly selected intDiameter). Use the x- and y-coordinates of the MouseDown event as the x- and y- coordinates of the circle's bounding box (that is, the second and third arguments to the DrawEllipse method). Notice that the first argument to the DrawEllipse method is a Pen object. See Exercise 21.11 for a description of Pen.

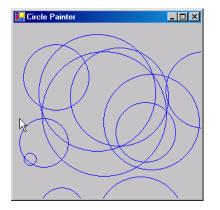


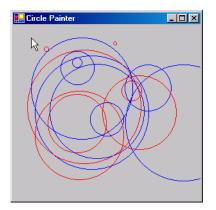
Figure 21.28 Circle Painter application's GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial21\Exercises\CirclePainter directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click CirclePainter.sln in the CirclePainter directory to open the application.
- c) Adding a MouseDown event handler. Create a MouseDown event handler. In the event handler, retrieve the x- and y-coordinates of the location the mouse pointer when a mouse button was pressed. Then generate a random number to use as the circle's diameter, using a Random object, and store it in a variable. Finally, call the DrawEl-lipse method on a reference to a Graphics object to draw a blue circle on the Form with the diameter generated by the Random object.
- d) *Running the application.* Select **Debug > Start** to run your application. Draw several blue circles and make sure that they are of different sizes.
- e) *Closing the application.* Close your running application by clicking its close box.
- f) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1 'Exercise 21.12 Solution
2 'CirclePainter.vb
3
4 Public Class FrmCirclePainter
5 Inherits System.Windows.Forms.Form
6
7 'Windows Form Designer generated code
```

8	
9	' handles MouseDown event for FrmCirclePainter
10	<pre>Private Sub FrmCirclePainter_MouseDown(ByVal sender As _</pre>
11	<pre>Object, ByVal e As System.Windows.Forms.MouseEventArgs) _</pre>
12	Handles MyBase MouseDown
13	
14	' initialize Graphics object
15	Dim objGraphic As Graphics = CreateGraphics()
16	
17	' initialize Random object
18	Dim objRandom As Random = New Random
19	
20	' generate a random circle diameter
21	Dim intDiameter As Integer = objRandom.Next(5, 200)
22	
23	' draw a blue circle
24	objGraphic.DrawEllipse(New Pen(Color.Blue), e.X, e.Y, _
25	intDiameter, intDiameter)
26	
27	End Sub ' FrmCirclePainter_MouseDown
28	
20	End Class ' FrmCirclePainter
29	

**21.13** (*Advanced Circle Painter Application*) In this exercise, you will enhance the application you created in Exercise 21.12. The advanced **Circle Painter** application should draw blue circles with a randomly generated diameter when the user presses the left mouse button. When the user presses the right mouse button, the application should draw a red circle with a randomly generated diameter (Fig. 21.29).





- a) Copying the template to your working directory. Make a copy of the Circle-Painter directory from Exercise 21.12 in your C:\SimplyVB directory. Rename the copied directory AdvancedCirclePainter. If you have not completed Exercise 21.12, follow the steps in Exercise 21.12 to complete the application.
- b) *Opening the application's template file.* Double click CirclePainter.sln file in the AdvancedCirclePainter directory to open the application.
- c) **Drawing the appropriate circle.** Use the Button property of the MouseEventArgs reference, e, to determine which mouse button was pressed. Finally, call the DrawEllipse method on a reference to a Graphics object to draw a blue circle on the Form if the left mouse button was clicked, or a red circle if the right mouse button was clicked.
- d) *Running the application.* Select **Debug > Start** to run your application. Draw several blue circles of different sizes using the left mouse button, then draw several red circles of different sizes using the right mouse button.
- e) *Closing the application.* Close your running application by clicking its close box.

f) Closing the IDE. Close Visual Studio .NET by clicking its close box.
 Answer:

1 ' Exercise 21.13 Solution 2 ' CirclePainter.vb 3 4 Public Class FrmCirclePainter 5 Inherits System.Windows.Forms.Form 6 7 ' Windows Form Designer generated code 8 9 ' handles MouseDown event for FrmCirclePainter 10 Private Sub FrmCirclePainter\_MouseDown(ByVal sender As \_ 11 Object, ByVal e As System.Windows.Forms.MouseEventArgs) \_ 12 Handles MyBase.MouseDown 13 14 ' initialize Graphics object 15 Dim objGraphic As Graphics = CreateGraphics() 16 17 ' initialize Random object 18 Dim objRandom As Random = New Random 19 20 ' generate a random circle diameter 21 Dim intDiameter As Integer = objRandom.Next(5, 200) 22 23 ' left mouse button pressed 24 If e.Button = MouseButtons.Left Then 25 26 ' draw a blue circle if left mouse button pressed 27 objGraphic.DrawEllipse(New Pen(Color.Blue), e.X, e.Y, \_ 28 intDiameter, intDiameter) 29 30 ' right mouse button pressed 31 ElseIf e.Button = MouseButtons.Right Then 32 33 ' draw a red circle if right mouse button pressed 34 objGraphic.DrawEllipse(New Pen(Color.Red), e.X, e.Y, \_ 35 intDiameter, intDiameter) 36 37 End If 38 39 End Sub ' FrmCirclePainter\_MouseDown 40 41 End Class ' FrmCirclePainter

```
What does this code do? 

21.14 Consider the code in Fig. 21.26. Suppose we change the MouseMove event handler to the code below. What happens when the user moves the mouse? Assume that a Label lblD-isplay has been placed on the Form.
```

Private Sub FrmPainter\_MouseMove(ByVal sender As Object, \_
ByVal e As System.Windows.Forms.MouseEventArgs)
Handles MyBase.MouseMove
IblDisplay.Text = "I'm at " & e.X & ", " & e.Y & "."
End Sub ' FrmPainter\_MouseMove

Answer: The Label continuously displays the mouse pointer's current position on the Form.

What's wrong with this code? **21.15** The following code should draw a BlueViolet circle of diameter 4 that corresponds to the movement of the mouse. Find the error(s) in the following code: 1 Private Sub FrmPainter\_MouseMove(ByVal sender As Object, \_ 2 ByVal e As System.Windows.Forms.MouseEventArgs) \_ 3 Handles MyBase.MouseMove 4 5 If m\_blnshouldPaint = True Then 6 Dim objGraphic As Graphics = Graphics() 7 8 objGraphic.FillEllipse = ( \_ 9 New SolidBrush(Color.BlueViolet), e.Y, e.X, 5, 4) 10 End If 11 End Sub ' FrmPainter\_MouseMove code is as follows: 1 2 ByVal e As System.Windows.Forms.MouseEventArgs) 3 Handles MyBase.MouseMove 4 5 If m\_blnshouldPaint = True Then

Answer: The position arguments in the FillEllipse method have been transposed. Use method CreateGraphics to initialize a Graphics object. A circle's height and width must be equal, so the fourth argument passed to method FillEllipse should be 4. There should be no assignment operator between the word FillEllipse and the parenthesis. The corrected

```
Private Sub FrmPainter_MouseMove(ByVal sender As Object, _
 6
          Dim objGraphic As Graphics = CreateGraphics()
 7
 8
          objGraphic.FillEllipse( _
 9
             New SolidBrush(Color.BlueViolet), e.X, e.Y, 4, 4)
10
       End If
11
    End Sub ' FrmPainter_MouseMove
```

```
Programming Challenge
```

21.16 (Advanced Painter Application) Extend the Painter application to enable a user to change the size and color of the circles drawn.

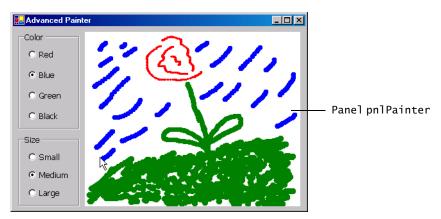


Figure 21.30 Advanced Painter application's GUI.

a) Copying the template to your working directory. Copy the C:\Examples\Tutorial21\Exercises\AdvancedPainter directory to your C:\SimplyVB directory.

- b) *Opening the application's template file.* Double click AdvancedPainter.sln in the AdvancedPainter directory to open the application (Fig. 21.30).
- c) Understanding the provided instance variables. The template already provides you with four instance variables. Variable m\_objBrushColor is a Color value that specifies the color of the brush used in the Advanced Painter application. The m\_blnShouldPaint and m\_blnShouldErase variable perform the same functions as in this tutorial's Painter application. The m\_intDiameter variable stores the diameter of the circle to be drawn.
- d) Declaring an enumeration to store the circle diameter sizes. Declare an enumeration Sizes to store the possible values of m\_intDiameter. Set constant SMALL to 4, MEDIUM to 8 and LARGE to 10.
- e) Adding event handlers for the Color RadioButtons. The Color RadioButton's event handlers should set m\_objBrushColor to their specified colors (Color.Red, Color.Blue, Color.Green or Color.Black).
- f) Adding event handlers for the Size RadioButtons. The Size RadioButton's event handlers should set m\_intDiameter to Sizes.SMALL (for the Small RadioButton), Sizes.MEDIUM (for the Medium RadioButton) or Sizes.LARGE (for the Large RadioButton).
- g) Adding a mouse event handler to a Panel. To associate mouse events with the Panel, select pnlPainter from the Class Name ComboBox. Then select the appropriate mouse event from the Method Name ComboBox.
- h) *Coding the MouseDown and MouseUp event handlers.* The MouseUp and MouseDown event handlers behave exactly as they do in the **Painter** application.
- i) Coding the MouseMove event handler. The MouseMove event handler behaves as the one in Painter application does. The color of the brush that draws the circle when m\_blnShouldPaint is True is specified by m\_objBrushColor. The eraser color is specified by the Panel's BackColor property.
- j) Running the application. Select Debug > Start to run your application. Start drawing on the Panel using different brush sizes and colors. Use the right mouse button to erase part of your drawing.
- k) Closing the application. Close your running application by clicking its close box.
- 1) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 21.16 Solution
 2
      AdvancedPainter.vb
 3
 4
    Public Class FrmAdvancedPainter
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' create Color value and initialize to Black
 8
       Private m_objBrushColor As Color = Color.Black
 9
10
        ' specify whether application should paint
11
        Private m_blnShouldPaint As Boolean = False
12
13
        ' specify whether application should erase
14
       Private m_blnShouldErase As Boolean = False
15
16
        ' diameter of MouseDown circle (initially set to small)
17
        Private m_intDiameter As Integer = 4
18
19
        ' size constants for diameter of MouseDown circle
20
        Private Enum Sizes
21
           SMALL = 4
22
          MEDIUM = 8
23
           LARGE = 10
24
       End Enum
```

25	
26	' Windows Form Designer generated code
27	
28	' handles radRed's CheckChanged event
29	<pre>Private Sub radRed_CheckedChanged(ByVal sender As _</pre>
30	System.Object, ByVal e As System.EventArgs) _
31	Handles radRed.CheckedChanged
	nundres i lanca encekca enangea
32	
33	' set brush color to red
34	<pre>If radRed.Checked = True Then</pre>
35	<pre>m_objBrushColor = Color.Red</pre>
36	End If
37	
38	End Sub ' radRed_CheckedChanged
39	- •
	L handling and Director Characterization and
40	' handles radBlue's CheckChanged event
41	<pre>Private Sub radBlue_CheckedChanged(ByVal sender As _</pre>
42	System.Object, ByVal e As System.EventArgs) _
43	Handles radBlue.CheckedChanged
44	handles had hereneekedendiged
45	' set brush color to blue
46	<pre>If radBlue.Checked = True Then</pre>
47	<pre>m_objBrushColor = Color.Blue</pre>
48	End If
49	
50	End Sub ' radBlue_CheckedChanged
51	
52	' handles radGreen's CheckChanged event
53	<pre>Private Sub radGreen_CheckedChanged(ByVal sender As _</pre>
54	System.Object, ByVal e As System.EventArgs) _
55	Handles radGreen.CheckedChanged
56	set the set of the set
	Lorent househouse the second
57	' set brush color to green
58	<pre>If radGreen.Checked = True Then</pre>
59	m_objBrushColor = Color.Green
60	End If
61	
62	End Sub ' radGreen_CheckedChanged
63	
64	' handles radBlack's CheckChanged event
65	
	Private Sub radBlack_CheckedChanged(ByVal sender As _
66	System.Object, ByVal e As System.EventArgs) _
67	Handles radBlack.CheckedChanged
68	
69	' set brush color to black
70	<pre>If radBlack.Checked = True Then</pre>
71	m_objBrushColor = Color.Black
72	End If
73	
74	End Sub L mad Dlack Charled Changed
	End Sub ' radBlack_CheckedChanged
75	
76	' handles radSmall's CheckChanged event
77	Private Sub radSmall_CheckedChanged(ByVal sender As _
78	System.Object, ByVal e As System.EventArgs) _
79	Handles radSmall.CheckedChanged
80	
81	' draw small circles
82	<pre>If radSmall.Checked = True Then</pre>
83	m_intDiameter = Sizes.SMALL
84	End If
85	

86	End Sub ' radSmall_CheckedChanged
87	
88	' handles radSMedium's CheckChanged event
89	Private Sub radMedium_CheckedChanged(ByVal sender As _
90	System.Object, ByVal e As System.EventArgs) _
91	Handles radMedium.CheckedChanged
92	
93	' draw medium circles
94	If radMedium.Checked = True Then
95	m_intDiameter = Sizes.MEDIUM
96	End If
	ENG IT
97	
98	End Sub ' radMedium_CheckedChanged
99	
100	' handles radLarge's CheckChanged event
101	<pre>Private Sub radLarge_CheckedChanged(ByVal sender As _</pre>
102	System.Object, ByVal e As System.EventArgs) _
103	Handles radLarge.CheckedChanged
104	
105	' draw large circles
106	If radLarge.Checked = True Then
107	m_intDiameter = Sizes.LARGE
108	End If
109	
110	End Sub ' radLarge_CheckedChanged
111	End Sub TadLarge_checkedchanged
112	' draw when mouse button pressed down
112	
	Private Sub pnlPainter_MouseDown(ByVal sender As _
114	Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
115	Handles pnlPainter.MouseDown
116	
117	' draw if left mouse button held down
118	<pre>If e.Button = MouseButtons.Left Then</pre>
119	m_blnShouldPaint = True
120	
121	' erase if right mouse button held down
122	ElseIf e.Button = MouseButtons.Right Then
123	m_blnShouldErase = True
124	End If
125	
126	End Sub ' pnlPainter_MouseDown
127	_
128	' stop drawing after mouse released
129	Private Sub pnlPainter_MouseUp(ByVal sender As _
130	Object, ByVal e As System.Windows.Forms.MouseEventArgs) _
131	Handles pnlPainter.MouseUp
132	nanures pitratiter.mouseop
	m blachouldDoint Folco I do not draw
133	<pre>m_blnShouldPaint = False ' do not draw m_blsShouldFaces = False ' do not draw</pre>
134	<pre>m_blnShouldErase = False ' do not erase</pre>
135	End Sub ' pnlPainter_MouseUp
136	
137	' draw when mouse moves if mouse down
138	Private Sub pnlPainter_MouseMove(ByVal sender As _
139	<pre>Object, ByVal e As System.Windows.Forms.MouseEventArgs) _</pre>
140	Handles pnlPainter.MouseMove
141	
142	' create Graphics object for the Panel
143	Dim objGraphic As Graphics = pnlPainter.CreateGraphics()
144	
145	' draw circles with specified brush color and size
146	If m_blnShouldPaint = True Then
	1S hishourtai arrie – rrue men

147	
148	objGraphic.FillEllipse(New SolidBrush(m_objBrushColor), _
149	e.X, e.Y, m_intDiameter, m_intDiameter)
150	
151	' draw circles with Panel's background color and specified size
152	<pre>ElseIf m_blnShouldErase = True Then</pre>
153	
154	objGraphic.FillEllipse( _
155	<pre>New SolidBrush(pnlPainter.BackColor), _</pre>
156	e.X, e.Y, m_intDiameter, m_intDiameter)
157	
158	End If
159	
160	End Sub ' pnlPainter_MouseMove
161	
162	End Class ' FrmAdvancedPainter





# **Typing Application**

Introducing Keyboard Events, Menus and Dialogs Solutions

$\bigcirc$	Instructor's Manual Exercise Solutions Tutorial 23		
	MULTIPLE-CHOICE QUESTIONS	<b>22.1</b> When creating a menu, typing a access shortcut for that item.	in front of a menu item name will create an
	QUESTIONS	a) &	b) !
		c) \$	d) #
		<b>22.2</b> Alt, Shift and Control are ke	eys.
		a) modifier	b) ASCII
		c) function	d) special
		<b>22.3</b> KeyChar is a property of	
		a) KeyEventArgs	b) Key
		c) KeyArgs	d) KeyPressEventArgs
		<b>22.4</b> Typing a hyphen (-) as a menu item's T	ext property will create a(n)
		a) separator bar	b) access shortcut
		c) new submenu	d) keyboard shortcut
		<b>22.5</b> A provides a group of relate	ed commands for Windows applications.
		a) separator bar	b) hot key
		c) menu	d) margin indicator bar
		<b>22.6</b> The enumeration specifies k	ey codes and modifiers.
$\bigcirc$		a) Keyboard	b) Key
$\bigcirc$		c) KeyboardTypes	d) Keys
		<b>22.7</b> The event is raised when a k	ey is pressed by the user.
		a) KeyPress	b) KeyHeld
		c) KeyDown	d) Both a and c.
		<b>22.8</b> Which of the following is not a keyboar	rd event?
		a) KeyPress	b) KeyDown
		c) KeyUp	d) KeyClicked
		<b>22.9</b> Which of the following is not a structure	2?
		a) Char	b) Color
		c) String	d) Date
		<b>22.10</b> The type allows you to det dialog.	termine which Button the user clicked to exit a
		a) DialogButtons	b) DialogResult
		c) Buttons	d) ButtonResult
		<b>Answers:</b> 22.1) a. 22.2) a. 22.3) d. 22.4) a. 22	2.5) c. 22.6) d. 22.7) d. 22.8) d. 22.9) c. 22.10) b.
	EXERCISES	that you developed in Tutorial 4 to prevent the Use keyboard events to allow the user to press	<i>rd Events</i> ) Enhance the <b>Inventory</b> application the user from entering input that is not a number. It is the number keys, the left and right arrows and is pressed, display a MessageBox instructing the

🛃 Inventory		Invalid Input
Cartons per shipment:	Total:	Enter numbers only
Items per carton: 0	Calculate Total	(OK)

Figure 22.30 Enhanced Inventory application.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial22\Exercises\KeyEventInventory directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click KeyEventInventory.sln in the KeyEventInventory directory to open the application.
- c) Adding the KeyDown event handler for the first TextBox. Add an empty KeyDown event handler for the Cartons per shipment: TextBox.
- d) *Adding a Select Case statement*. Add a Select Case statement to the KeyDown event handler that determines whether a number key, a left or right arrow or the *Backspace* key was pressed.
- e) Adding the Case Else statement. Add a Case Else statement that will determine whether a key other than a valid one for this application was pressed. If an invalid key was pressed, display a MessageBox that instructs the user to enter a number.
- f) Adding the KeyDown event handler for the second TextBox. Repeat Steps c-e, only this time create a KeyDown event handler for the Items per carton: TextBox. This event handler should perform the same functionality as the one for the Cartons per shipment: TextBox.
- g) Running the application. Select Debug > Start to run your application. Try entering letters or pressing the up and down arrow keys in the TextBoxes. A MessageBox should be displayed. Enter valid input and click the Calculate Total Button. Verify that the correct output is displayed.
- h) Closing the application. Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
Exercise 22.11 Solution
 1
 2
     ' Inventory.vb
 3
 4
    Public Class FrmInventory
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
       Private Sub btnCalculate_Click(ByVal sender As _
10
           System.Object, ByVal e As System.EventArgs) _
11
          Handles btnCalculate.Click
12
13
           ' multiply values input and display result in Label
14
           lblTotalResult.Text = _
15
              (Val(txtCartons.Text)
              * Val(txtItems.Text)).ToString
16
17
18
       End Sub ' btnCalculate_Click
19
20
       Private Sub txtCartons_KeyDown(ByVal sender As Object, _
21
          ByVal e As System.Windows.Forms.KeyEventArgs) _
22
          Handles txtCartons.KeyDown
23
24
          Dim result As DialogResult ' store result of MessageBox
25
```

26	Select Case e.KeyData
27	Case Keys.D0 To Keys.D9 ' numbers
28	
29	Case Keye Deels I beekenees
	Case Keys.Back ' backspace
30	
31	Case Keys.Enter ' enter
32	
33	<b>Case</b> Keys.Left, Keys.Right ' arrows
34	
35	Case Else ' all other keys
36	
37	' show MessageBox
38	result = MessageBox.Show("Enter numbers only", _
39	
	"Invalid Input", MessageBoxButtons.OK, _
40	MessageBoxIcon.Exclamation)
41	
<b>4</b> 2	' clear TextBox if invalid input entered
43	<pre>If result = DialogResult.OK Then</pre>
44	txtCartons.Clear()
45	End If
46	
47	End Select
48	
40	
	End Sub ' txtCartons_KeyDown
50	
51	<pre>Private Sub txtItems_KeyDown(ByVal sender As Object, _</pre>
52	<pre>ByVal e As System.Windows.Forms.KeyEventArgs) _</pre>
53	Handles txtItems.KeyDown
54	
55	Dim result As DialogResult ' store result of MessageBox
56	
57	Select Case e.KeyData
58	Case Keys.D0 To Keys.D9 ' numbers
59	
60	Case Keys.Back ' backspace
61	Case Reys. Back Backspace
	Const Keyes, Factors, London
62	Case Keys.Enter ' enter
63	
64	<b>Case</b> Keys.Left, Keys.Right ' arrows
65	
66	Case Else ' all other keys
67	
68	' show MessageBox
69	<pre>result = MessageBox.Show("Enter numbers only", _</pre>
70	"Invalid Input", MessageBoxButtons.OK, _
71	MessageBoxIcon.Exclamation)
72	Plessageboxicon. Excitaliacton)
	Lalass TextDev if invalid input entered
73	' clear TextBox if invalid input entered
74	<pre>If result = DialogResult.OK Then</pre>
75	<pre>txtItems.Clear()</pre>
76	End If
77	
78	End Select
79	
80	End Sub ' txtItems_KeyDown
81	
82	End Class ' FrmInventory
02	Line cruss i filitiventory

**22.12** (*Bouncing Ball Game*) Write an application that allows the user to play a game, the goal of which is to prevent a bouncing ball from falling off the bottom of the Form. When the

user presses the *S* key, a blue ball will bounce off the top, left and right sides (the "walls") of the Form. There should be a horizontal bar on the bottom of the Form, which serves as a paddle to prevent the ball from hitting the bottom of the Form. (The ball can bounce off the paddle, but not the bottom of the Form.) The user can move the paddle using the left and right arrow keys. If the ball hits the paddle, the ball should bounce up, and the game should continue. If the ball hits the bottom of the Form, the game should end. The paddle's width should decrease every 20 seconds to make the game more challenging. The GUI is provided for you (Fig. 22.31).

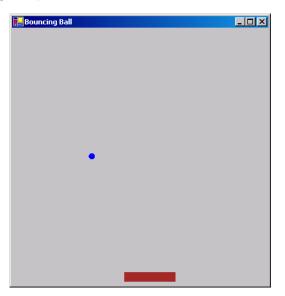


Figure 22.31 Bouncing Ball application.

- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial22\Exercises\BouncingBall directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click BouncingBall.sln in the BouncingBall directory to open the application.
- c) Creating the KeyDown event handler. Insert a KeyDown event handler for the Form.
- d) *Writing code to start the game*. Write an If...Then statement in the KeyDown event handler that tests whether the user presses the *S* key. You can use the KeyDown event handler for the *S* key in this case because you do not care whether the user presses an uppercase *S* or a lowercase *S*. If the user presses the *S* key, start the two Timers that are provided in the template.
- e) *Inserting code to move the paddle left*. Write an If...Then statement that tests if the user pressed the left-arrow key and if the paddle's horizontal position is greater than zero. If the paddle's horizontal position equals zero, the left edge of the paddle is touching the left wall and the paddle should not be allowed to move farther to the left. If both the conditions in the If...Then are true, decrease the paddle's *x*-position by 10.
- f) Inserting code to move the paddle right. Write an If...Then statement that tests if the user pressed the right-arrow key and whether the paddle's x-coordinate is less than the width of the Form minus the width of the paddle. If the paddle's x-coordinate equals the Form's width minus the width of the paddle, the paddle's right edge is touching the right wall and the paddle should not be allowed to move farther to the right. If both the conditions in the If...Then statement are true, increase the paddle's x-coordinate by 10.
- g) **Running the application.** Select **Debug > Start** to run your application. Press the *S* key to begin the game and use the paddle to keep the bouncing ball from dropping off the Form. Continue doing this until 20 seconds have passed, and verify that the paddle is decreased in size at that time.
- h) *Closing the application.* Close your running application by clicking its close box.

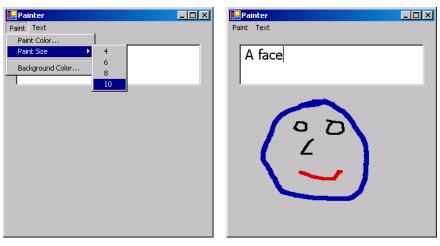
i) *Closing the IDE*. Close Visual Studio .NET by clicking its close box. *Answer:* 

```
1
     ' Exercise 22.12 Solution
 2
    ' BouncingBall.vb
 3
 4
    Public Class FrmBouncingBall
 5
       Inherits System.Windows.Forms.Form
 6
 7
       Private intX As Integer ' ball's x-coordinate
 8
       Private intY As Integer ' ball's y-coordinate
 9
       Private intRectangleX As Integer ' paddle's x-coordinate
10
       Private intRectangleWidth As Integer ' paddle's width
11
       Private intDeltaX As Integer ' ball's x rate of change
12
13
       Private intDeltaY As Integer ' ball's y rate of change
14
15
        Private blnXLeft As Boolean ' tests if ball can move left
16
       Private blnYUp As Boolean ' tests if ball can move up
17
18
       Private Const intMAX_X As Integer = 400 ' x boundary
19
       Private Const intMAX_Y As Integer = 400 ' y boundary
20
21
        ' object to generate random numbers
22
       Private objRandom As Random = New Random
23
24
        ' Windows Form Designer generated code
25
26
        Private Sub FrmBouncingBall_Load(ByVal sender As _
27
          System.Object, ByVal e As System.EventArgs) _
28
          Handles MyBase.Load
29
30
          intX = objRandom.Next(100, 301) ' ball's initial x
31
          intY = objRandom.Next(100, 301) ' ball's initial y
32
           intRectangleX = 175 ' rectangle's intial x position
33
          intRectangleWidth = 80 ' rectangle's intial width
34
35
          blnXLeft = False ' ball can move left
36
          blnYUp = False ' ball can move up
          intDeltaX = 2 ' move ball 2 positions right
37
           intDeltaY = 2 ' move ball 2 positions down
38
39
        End Sub ' FrmBouncingBall_Load
40
41
       Protected Overrides Sub OnPaint( _
42
          ByVal e As System.Windows.Forms.PaintEventArgs)
43
44
           ' create graphics object
45
          Dim objGraphic As Graphics = CreateGraphics()
46
47
           ' create new brush
48
          Dim objBrush As SolidBrush = New SolidBrush(Color.Blue)
49
50
           ' draw ball
51
          objGraphic.FillEllipse(objBrush, intX, intY, 10, 10)
52
53
           ' set color for, and draw paddle
54
          objBrush.Color = Color.Brown
55
           objGraphic.FillRectangle( _
56
             objBrush, intRectangleX, 380, intRectangleWidth, 15)
57
        End Sub ' OnPaint
```

```
58
59
        Private Sub tmrMoveBall_Tick(ByVal sender As System.Object, _
60
           ByVal e As System. EventArgs) Handles tmrMoveBall. Tick
61
62
           ' determine new x position
63
           If blnXLeft = True Then
64
               intX += intDeltaX
65
           Else
               intX -= intDeltaX
66
67
           End If
68
69
           ' determine new y position
70
           If blnYUp Then
71
               intY += intDeltaY
72
           Else
73
              intY -= intDeltaY
74
           End If
75
           If intY <= 0 Then</pre>
76
77
              blnYUp = True
78
               intDeltaY = objRandom.Next(2, 6)
79
           ElseIf intY >= 370 AndAlso intX >= intRectangleX _
80
              AndAlso intX <= (intRectangleX + intRectangleWidth) Then</pre>
81
               blnYUp = False
82
               intDeltaY = objRandom.Next(2, 6)
83
           ElseIf intY >= 410 Then ' end game if ball hits floor
84
               tmrMoveBall.Enabled = False
85
               tmrShrinkSlider.Enabled = False
86
              MessageBox.Show("Game Over")
87
           End If
88
89
           If intX <= 0 Then</pre>
90
               blnXLeft = True
91
              intDeltaX = objRandom.Next(2, 6)
92
           ElseIf intX >= intMAX_X - 10 Then
93
              blnXLeft = False
94
               intDeltaX = objRandom.Next(2, 6)
95
           End If
96
97
           Invalidate() ' Refresh Form
98
99
        End Sub ' tmrMoveBall_Tick
100
101
        ' shrinks the paddle every 20 seconds
102
        Private Sub tmrShrinkSlider_Tick(ByVal sender As _
103
           System.Object, ByVal e As System.EventArgs) _
104
           Handles tmrShrinkSlider.Tick
105
106
           ' shrink paddle if paddle greater than twice ball's width
107
           If intRectangleWidth >= 20 Then
108
               intRectangleWidth = Convert.ToInt32( _
109
                  intRectangleWidth / 2)
110
           End If
111
112
        End Sub ' tmrShrinkSlider_Tick
113
114
        ' handles KeyDown event
115
        Private Sub FrmBouncingBall_KeyDown(ByVal sender As Object, _
116
           ByVal e As System.Windows.Forms.KeyEventArgs) Handles _
117
           MyBase.KeyDown
118
```

119	If e.KeyCode = Keys.S Then ' start game if user presses S key
120	<pre>tmrMoveBall.Enabled = True ' start tmrMoveBall</pre>
121	tmrShrinkSlider.Enabled = True ' start tmrShrinkSlider
122	End If
123	
124	<pre>If e.KeyCode = Keys.Left AndAlso intRectangleX &gt;= 0 Then</pre>
125	<pre>intRectangleX -= 10 ' move paddle to the left</pre>
126	End If
127	
128	<pre>If e.KeyCode = Keys.Right AndAlso _</pre>
129	(intRectangleX <= intMAX_X - intRectangleWidth)                                 Then
130	
131	<pre>intRectangleX += 10 ' move paddle to the right</pre>
132	End If
133	
134	End Sub ' FrmBouncingBall_KeyDown
135	
136	End Class ' FrmBouncingBall

**22.13** (*Modified Painter Application*) Modify the **Painter** application that you developed in Tutorial 21 to include menus that allow the user to select the size and color of the painted ellipses and the color of the Form (Fig. 22.32). (The menus replace the RadioButtons.) Also, add a multiline TextBox that allows the user to type text to accompany the painting. The user should be able to use menus to select the font style and color of the text and the background color of the TextBox.





- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial22\Exercises\ModifiedPainter directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click ModifiedPainter.sln in the ModifiedPainter directory to open the application.
- c) Creating the menus. Create two menus. The first one should be titled Paint and should contain a Paint Color... menu item, a Paint Size submenu that contains menu items 4, 6, 8 and 10, a separator bar and a Background Color... menu item. The second menu should be titled Text and have Text Color... and Font... menu items, a separator bar and a TextBox Color... menu item.
- d) *Changing the paint color*. Add an event handler for the **Paint Color...** menu item. This event handler should display a **Color** dialog that allows the user to change the value stored in m\_paintColor.
- e) *Changing the paint size*. Add an event handler for each of the **Size** submenu's menu items. Each event handler should change the value stored in m\_intDiamter to the

value displayed on the menu (that is, clicking the **4** menu item will change the value of m\_intDiameter to 4).

- f) Changing the background color. Add an event handler for the Background Color... menu item. This event handler should display a Color dialog that allows the user to change the value stored in m\_backgroundColor and also change the BackColor property of the Form. To change the background color of the Form, assign the value specifying the background color to BackColor. For instance, the statement BackColor = Color.White changes the background color of the Form to white.
- g) *Changing the text color*. Add an event handler for the **Text Color**... menu item. This event handler should display a **Color** dialog that allows the user to change the color of the text displayed in the TextBox.
- h) *Changing the text style*. Add an event handler for the Font... menu item. This event handler should display a Font dialog that allows the user to change the style of the text displayed in the TextBox.
- i) *Changing the TextBox's background color*. Add an event handler for the TextBox Color... menu item. This event handler should display a Color dialog that allows the user to change the background color of the TextBox.
- j) Running the application. Select Debug > Start to run your application. Use the menus to draw shapes of various colors and brush sizes. Enter text to describe your drawing. Use the other menu options to change the color of the Form, the TextBox and the text in the TextBox.
- k) *Closing the application*. Close your running application by clicking its close box.
- 1) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 22.13 Solution
2	' Painter.vb
3	
4	Public Class FrmPainter
5	Inherits System.Windows.Forms.Form
6	
7	' specify whether moving the mouse should erase
8	Private m_blnShouldErase As Boolean
9	
10	' specify whether moving the mouse should draw
11	Private m_blnShouldPaint As Boolean
12	
13	' set diameter of MouseDown circle
14	Private m_intDiameter As Integer
15	Private m_paintColor As Color ' paint color
16	
17	Private m_backgroundColor As Color ' background color
18	
19	' create Graphics object
20	Private m_objGraphic As Graphics
21	
22	' Windows Form Designer generated code
23	
24	Private Sub FrmPainter_Load(ByVal sender As System.Object, _
25	ByVal e As System.EventArgs) Handles MyBase.Load
26	
27	<pre>m_blnShouldErase = False ' should not be painting</pre>
28	<pre>m_blnShouldPaint = False ' should not be erasing</pre>
29	m_intDiameter = 8 ' set paint size
30	m_paintColor = Color.BlueViolet ' set paint color
31	m_backgroundColor = FrmPainter.DefaultBackColor
32	<pre>m_objGraphic = CreateGraphics() ' initialize graphics object</pre>
33	End Sub ' FrmPainter_Load
34	

35	' handles FrmPainter's MouseDown event
36	<pre>Private Sub FrmPainter_MouseDown(ByVal sender As Object, _</pre>
37	ByVal e As System.Windows.Forms.MouseEventArgs) _
38	Handles MyBase.MouseDown
39	
40	' draw on Form if the left button is held down
41	If e.Button = MouseButtons.Left Then
42	
	<pre>m_blnShouldPaint = True</pre>
43	
44	' erase blue-violet circles if right button is held down
45	<pre>ElseIf e.Button = MouseButtons.Right Then</pre>
46	m_blnShouldErase = True
47	End If
48	
49	End Sub ' FrmPainter_MouseDown
50	
51	' handles FrmPainter's MouseUp event
52	Private Sub FrmPainter_MouseUp(ByVal sender As Object, _
53	ByVal e As System.Windows.Forms.MouseEventArgs) _
54	Handles MyBase.MouseUp
55	handles hybrise house op
56	<pre>m_blnShouldPaint = False ' do not draw on the Form</pre>
57	<pre>m_blnShouldErase = False ' do not erase</pre>
58	
59	End Sub ' FrmPainter_MouseUp
60	
61	' handles FrmPainter's MouseMove event
62	Private Sub FrmPainter_MouseMove(ByVal sender As Object, _
63	<pre>ByVal e As System.Windows.Forms.MouseEventArgs) _</pre>
64	Handles MyBase.MouseMove
65	
66	' draw circle if left mouse button is pressed
67	If m_blnShouldPaint Then
68	
69	e.X, e.Y, m_intDiameter, m_intDiameter)
70	
71	' mouse pointer "erases" if right mouse button is pressed
72	ElseIf m_blnShouldErase Then
73	<pre>m_objGraphic.FillEllipse(</pre>
74	<pre>New SolidBrush(m_backgroundColor), _</pre>
75	e.X, e.Y, m_intDiameter, m_intDiameter)
76	End If
77	
78	End Sub ' FrmPainter_MouseMove
79	
80	<pre>Private Sub mnuitmColor_Click(ByVal sender As System.Object, _</pre>
81	ByVal e As System.EventArgs) Handles mnuitmColor.Click
82	
83	Dim dlgColorDialog As New ColorDialog ' Color Dialog
84	Dim result As DialogResult ' stores Button clicked
85	
86	dlgColorDialog.FullOpen = True ' show all colors
87	result = dlgColorDialog.ShowDialog
88	-court - argeororbratog.onombratog
89	L do nothing if upon clicked dislarly Concel Butter
89 90	' do nothing if user clicked dialog's Cancel Button
	<pre>If result = DialogResult.Cancel Then Detuge</pre>
91	Return
92	End If
93	
94	' assign new color to Paint object
95	m_paintColor = dlgColorDialog.Color

96	End Sub ' mnuitmColor_Click
97	
<b>98</b>	<pre>Private Sub mnuitmFour_Click(ByVal sender As System.Object, _</pre>
99	ByVal e As System.EventArgs) Handles mnuitmFour.Click
100	
101	<pre>m_intDiameter = 4 ' set paint size to four</pre>
102	End Sub ' mnuitmFour_Click
103	
104	<pre>Private Sub mnuitmSix_Click(ByVal sender As System.Object, _</pre>
105	ByVal e As System.EventArgs) Handles mnuitmSix.Click
106	
107	m_intDiameter = 6 ' set paint size to six
108	End Sub ' mnuitmSix_Click
109	
110	<pre>Private Sub mnuitmEight_Click(ByVal sender As System.Object, _</pre>
111	ByVal e As System.EventArgs) Handles mnuitmEight.Click
112	
113	m_intDiameter = 8 ' set paint size to eight
114	End Sub ' mnuitmEight_Click
115	
116	<pre>Private Sub mnuitmTen_Click(ByVal sender As System.Object, _</pre>
117	ByVal e As System.EventArgs) Handles mnuitmTen.Click
118	
119	<pre>m_intDiameter = 10 ' set paint size to ten</pre>
120	End Sub ' mnuitmTen_Click
121	
122	Private Sub mnuitmBackground_Click(ByVal sender As _
123	System.Object, ByVal e As System.EventArgs) _
124 125	Handles mnuitmBackground.Click
125	Dim dlgColorDialog As New ColorDialog ' Color Dialog
120	Dim result As DialogResult ' stores Button clicked
128	Dhin resure As Dialogicesure Scores Ducton effected
129	dlgColorDialog.FullOpen = True ' show all colors
130	result = dlgColorDialog.ShowDialog()
131	
132	' do nothing if user clicked dialog's Cancel Button
133	<pre>If result = DialogResult.Cancel Then</pre>
134	Return
135	End If
136	
137	<pre>m_backgroundColor = dlgColorDialog.Color ' set "erase" color</pre>
138	BackColor = dlgColorDialog.Color ' set Form's color
139	End Sub ' mnuitmBackground_Click
140	
141	<pre>Private Sub mnuitmTextColor_Click(ByVal sender As System.Object, _</pre>
142	ByVal e As System.EventArgs) Handles mnuitmTextColor.Click
143	
144	Dim dlgColorDialog As New ColorDialog ' Color Dialog
145	Dim result As DialogResult ' stores Button clicked
146	
147	dlgColorDialog.FullOpen = True ' show all colors
148	result = dlgColorDialog.ShowDialog()
149	L de nothing if ween clicked dialogle Concel Dutter
150 151	<pre>' do nothing if user clicked dialog's Cancel Button If result = DialogResult.Cancel Then</pre>
152	Return
153	End If
154	
155	' assign new color to text
156	txtOutput.ForeColor = dlgColorDialog.Color

157	<pre>End Sub ' mnuitmTextColor_Click</pre>
158	
159	Private Sub mnuitmFont_Click(ByVal sender As System.Object,
160	ByVal e As System.EventArgs) Handles mnuitmFont.Click
161	
162	Dim dlgFontDialog As New FontDialog ' Font Dialog
163	Dim result As DialogResult ' stores Button clicked
164	
165	' show dialog and get result
166	result = dlgFontDialog.ShowDialog()
167	
168	' do nothing if user clicked dialog's Cancel Button
169	<pre>If result = DialogResult.Cancel Then</pre>
170	Return
171	End If
172	Lander and fast value to TautDay
173	' assign new font value to TextBox
174 175	<pre>txtOutput.Font = dlgFontDialog.Font Fond Sub l menuitmEart Click</pre>
175	End Sub ' mnuitmFont_Click
177	<pre>Private Sub mnuitmTextBoxColor_Click(ByVal sender As _</pre>
178	System.Object, ByVal e As System.EventArgs) _
179	Handles mnuitmTextBoxColor.Click
180	
181	Dim dlgColorDialog As New ColorDialog ' Color Dialog
182	Dim result As DialogResult ' stores Button clicked
183	bill resure is bialogicsure scores baccon erreked
184	dlgColorDialog.FullOpen = True ' show all colors
185	result = dlgColorDialog.ShowDialog()
186	
187	' do nothing if user clicked dialog's Cancel Button
188	If result = DialogResult.Cancel Then
189	Return
190	End If
191	
192	' assign background color of TextBox
193	txtOutput.BackColor = dlgColorDialog.Color
194	End Sub ' mnuitmTextBoxColor_Click
195	
196	End Class ' FrmPainter

What does this code do? 🕨	<b>22.14</b> What is the result of the following code?
	<pre>1 Private Sub mnuitmColor_Click(ByVal sender As _</pre>
	<pre>2 System.Object, ByVal e As System.EventArgs) _</pre>
	3 Handles mnuitmColor.Click
	4
	5 Dim dlgColorDialog As ColorDialog = New ColorDialog
	6 Dim result As DialogResult
	7
	<pre>8 dlgColorDialog.FullOpen = True</pre>
	9
	<pre>10 result = dlgColorDialog.ShowDialog()</pre>
	11
	12 If result = DialogResult.Cancel Then
	13 Return
	14 End If
	15

16 BackColor = dlgColorDialog.Color 17 End Sub ' mnuitmColor\_Click

**Answer:** When this event handler executes, the **Color** dialog is displayed. If the user chooses a color, that color is assigned to the BackColor property of lblDisplay. If the dialog's **Cancel** Button is clicked, the dialog closes and this event handler terminates.

What's wrong with this code?

**22.15** This code should allow a user to pick a font from a **Font** dialog and set the text in txt-Display to that font. Find the error(s) in the following code, assuming that a TextBox named txtDisplay exists on a Form.

```
1 Private Sub Fonts()
2 Dim dlgFontDialog As FontDialog
3
4 dlgFontDialog = New FontDialog
5 dlgFontDialog.ShowDialog()
6 txtDisplay.Font = dlgFontDialog.Font
7 End Sub ' Fonts
```

**Answer:** The code should check whether the user clicked the **Cancel** Button in the dialog. If the user clicked **Cancel**, the method should be terminated. If the user did not click **Cancel**, the text should be set to the selected style.

```
Private Sub Fonts()
 1
2
       Dim result As DialogResult
 3
       Dim dlgFontDialog As FontDialog
 Δ
 5
       dlgFontDialog = New FontDialog()
 6
       result = dlgFontDialog.ShowDialog()
 7
8
       If result = DialogResult.Cancel Then
9
           Return
10
        Else
11
           txtDisplay.Font = dlgFontDialog.Font
12
       End If
13
    End Sub ' Fonts
14
```

### Programming Challenge

**22.16** (*Dvorak Keyboard Application*) Create an application that simulates the letters on the Dvorak keyboard. A Dvorak keyboard allows faster typing by placing the most commonly used keys in the most accessible locations. Use keyboard events to create an application similar to the **Typing Application**, except that it simulates the Dvorak keyboard instead of the standard keyboard. The correct Dvorak key should be highlighted on the virtual keyboard and the correct character should be displayed in the TextBox. The keys and characters map as follows:

- On the top row, the P key of the Dvorak keyboard maps to the R key on a standard keyboard, and the L key of the Dvorak keyboard maps to the P key on a standard keyboard.
- On the middle row, the A key remains in the same position and the S key on the Dvorak keyboard maps to the semicolon key on the standard keyboard.
- On the bottom row, the Q key on the Dvorak keyboard maps to the X key on the standard keyboard and the Z key maps to the question mark key.
- All of the other keys on the Dvorak keyboard map to the location shown in Fig. 22.33.



Figure 22.33 Dvorak Keyboard GUI.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial22\Exercises\DvorakKeyboard directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click DvorakKeyboard.sln in the DvorakKeyboard directory to open the application.
- c) *Creating the KeyPress event handler*. Add a KeyPress event handler for the Text-Box.
- d) *Creating a Select Case statement*. Add a Select Case statement to the KeyPress event handler. The Select Case statement should test whether all of the letter keys on the Dvorak keyboard were pressed except for the *S*, *W*, *V* and *Z* keys. If a Dvorak key was pressed, highlight it on the GUI and display the character in the TextBox.
- e) *Creating a KeyDown event handler*. Add a KeyDown event handler for the TextBox. The *S*, *W*, *V* and *Z* keys do not map to a letter key on the standard keyboard; therefore, a KeyDown event handler must be used to determine whether one of these keys was pressed.
- f) Adding a Select Case statement. Add a Select Case statement to your KeyDown event handler that determines whether S, W, V or Z was pressed. If one of these keys was pressed, highlight the key, and add the character to the TextBox.
- g) Running the application. Select Debug > Start to run your application. Use your keyboard to enter text. Verify that the text entered is correct based on the rules in the exercise description. Make sure the correct Buttons on the Form are highlighted as you enter text.
- h) Closing the application. Close your running application by clicking its close box.
- i) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 22.16 Solution
2	' DvorakKeyboard.vb
3	
4	Public Class FrmDvorakKeyboard
5	Inherits System.Windows.Forms.Form
6	
7	' reference to last Button pressed
8	Dim m_btnLastButton As Button
9	_
0	' Windows Form Designer generated code
1	
2	<pre>Private Sub txtOutput_KeyUp(ByVal sender As Object, _</pre>
3	ByVal e As System.Windows.Forms.KeyEventArgs)
4	Handles txtOutput.KeyUp
5	

16 ResetColor() 17 End Sub ' txtOutput\_KeyUp 18 19 ' highlight Button passed as argument 20 Private Sub ChangeColor(ByVal btnButton As Button) 21 22 ResetColor() 23 btnButton.BackColor = Color.LightGoldenrodYellow 24 m\_btnLastButton = btnButton 25 End Sub ' ChangeColor 26 27 ' changes m\_btnLastButton's color if it refers to a Button 28 Private Sub ResetColor() 29 30 If IsNothing(m\_btnLastButton) = False Then 31 m\_btnLastButton.BackColor = \_ 32 m\_btnLastButton.DefaultBackColor 33 34 End If ' m\_btnLastButton is not Nothing 35 End Sub ' ResetColor 36 37 38 ' handles Form KeyPress Event 39 Private Sub txtOutput\_KeyPress(ByVal sender As Object, \_ 40 ByVal e As System.Windows.Forms.KeyPressEventArgs) \_ 41 Handles txtOutput.KeyPress 42 43 ' convert pressed key to uppercase 44 Select Case Char.ToUpper(e.KeyChar) 45 46 ' following cases test if key pressed was a letter 47 Case Convert.ToChar(Keys.A) ' a maps to a key **48** ChangeColor(btnA) 49 txtOutput.Text &= "a" 50 51 Case Convert.ToChar(Keys.B) ' x maps to b key 52 ChangeColor(btnX) 53 txtOutput.Text &= "x" 54 55 Case Convert.ToChar(Keys.C) ' j maps to c key 56 ChangeColor(btnJ) 57 txtOutput.Text &= "j" 58 59 Case Convert.ToChar(Keys.D) ' e maps to d key 60 ChangeColor(btnE) 61 txtOutput.Text &= "e" 62 63 Case Convert.ToChar(Keys.F) ' u maps to f key 64 ChangeColor(btnU) 65 txtOutput.Text &= "u" 66 67 Case Convert.ToChar(Keys.G) ' i maps to g key 68 ChangeColor(btnI) 69 txtOutput.Text &= "i" 70 71 Case Convert.ToChar(Keys.H) ' d maps to h key 72 ChangeColor(btnD) 73 txtOutput.Text &= "d" 74 75 Case Convert.ToChar(Keys.I) ' c maps to i key 76 ChangeColor(btnC)

77	txtOutput.Text &= "c"
78	
79	<pre>Case Convert.ToChar(Keys.J) ' h maps to j key</pre>
80	
	ChangeColor(btnH)
81	txtOutput.Text &= "h"
82	
83	Case Convert.ToChar(Keys.K) ' t maps to k key
84	
	ChangeColor(btnT)
85	txtOutput.Text &= "t"
86	
87	<pre>Case Convert.ToChar(Keys.L) ' n maps to l key</pre>
88	
	ChangeColor(btnN)
89	txtOutput.Text &= "n"
90	
91	<pre>Case Convert.ToChar(Keys.M) ' m maps to m key</pre>
92	ChangeColor(btnM)
93	txtOutput.Text &= "m"
94	
95	Case Convert.ToChar(Keys.N) ' b maps to n key
96	ChangeColor(btnB)
97	txtOutput.Text &= "b"
98	
99	<pre>Case Convert.ToChar(Keys.0) ' r maps to o key</pre>
100	ChangeColor(btnR)
101	
	txtOutput.Text &= "r"
102	
103	Case Convert.ToChar(Keys.P) ' 1 maps to p key
104	ChangeColor(btnL)
105	txtOutput.Text &= "]"
	txtoutput.Text $\alpha = 1$
106	
107	Case Convert.ToChar(Keys.R) ' p maps to r key
108	ChangeColor(btnP)
109	txtOutput.Text &= "p"
110	
111	<pre>Case Convert.ToChar(Keys.S) ' o maps to s key</pre>
112	ChangeColor(btn0)
113	txtOutput.Text &= "o"
114	
115	<pre>Case Convert.ToChar(Keys.T) ' y maps to t key</pre>
116	ChangeColor(btnY)
117	txtOutput.Text &= "y"
118	
119	<pre>Case Convert.ToChar(Keys.U) ' g maps to u key</pre>
120	ChangeColor(btnG)
121	txtOutput.Text &= "g"
122	
123	Case Convert TaChar( $V_{ave} \setminus V_{ave}$ is reasonable v low
	Case Convert.ToChar(Keys.V) ' k maps to v key
124	ChangeColor(btnK)
125	txtOutput.Text &= "k"
126	
127	<pre>Case Convert.ToChar(Keys.X) ' q maps to x key</pre>
128	ChangeColor(btnQ)
129	txtOutput.Text &= "q"
130	
131	<pre>Case Convert.ToChar(Keys.Y) ' f maps to y key</pre>
132	ChangeColor(btnF)
133	txtOutput.Text &= "f"
134	
135	End Select ' ends test for letters
136	
137	End Sub ' txtOutput_KeyPress

138	
139	' handles KeyDown event
140	<pre>Private Sub txtOutput_KeyDown(ByVal sender As Object, _</pre>
141	ByVal e As System.Windows.Forms.KeyEventArgs) _
142	Handles txtOutput.KeyDown
143	
144	Select Case e.KeyData
145	
146	' use KeyDown for these keys because the Dvorak
147	' representation does not map to letters of QWERTY keyboard
148	Case Keys.OemSemicolon ' s maps to semicolon key
149	ChangeColor(btnS)
150	<pre>txtOutput.Text &amp;= "s"</pre>
151	
152	Case Keys.Oemcomma ' w maps to comma key
153	ChangeColor(btnW)
154	txtOutput.Text &= "w"
155	
156	Case Keys.OemPeriod ' y maps to period key
157	ChangeColor(btnV)
158	txtOutput.Text &= "v"
159	
160	<b>Case</b> Keys.OemQuestion ' z maps t question mark key
161	ChangeColor(btnZ)
162	txtOutput.Text &= "z"
163	
164	End Select
165	
166	End Sub ' txtOutput_KeyDown
167	
168	End Class ' FrmDvorakKeyboard



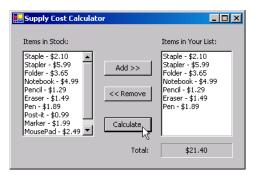


# Screen Scraping Application

Introducing String Processing Solutions

$\supset$	Instructor's Manual Exercise Solutions Tutorial 23		
_	MULTIPLE CHOICE	23.1 Extracting desired information from	Web pages is called
	QUESTIONS	<ul><li>a) Web crawling</li><li>c) querying</li></ul>	<ul><li>b) screen scraping</li><li>d) redirection</li></ul>
		<b>23.2</b> If IndexOf method does not find the	specified substring, it returns
		a) False c) -1	<ul><li>b) 0</li><li>d) None of the above.</li></ul>
		23.3 The String class allows you to	Strings.
		a) search c) replace characters in	<ul><li>b) retrieve characters from</li><li>d) All of the above.</li></ul>
		<b>23.4</b> is a technology for descri	bing Web content.
		a) Class String	b) A String literal
		c) HTML	d) A screen scraper
		<b>23.5</b> The String class is located in the	_
		a) String c) System.IO	<ul><li>b) System.Strings</li><li>d) System</li></ul>
		, <u>-</u>	new String object by copying part of an existing
		String object.	ice set ing object by copying part of an existing
$\mathcal{I}$		a) StringCopy	b) Substring
		c) CopyString	d) CopySubString
		<b>23.7</b> All String objects are	
		<ul><li>a) the same size</li><li>c) preceded by at least one whitespace</li><li>d) immutable</li></ul>	b) always equal to each other ce character
		23.8 The IndexOf method does not e	xamine any characters that occur prior to the
		a) starting index	b) first match
		c) last character of the String	d) None of the above.
		23.9 The method determines	whether a String ends with a particular substring.
		a) CheckEnd	b) StringEnd
		c) EndsWith	d) EndIs
		· ·	espace characters that appeara String.
		<ul><li>a) in</li><li>c) at the end of</li></ul>	<ul><li>b) at the beginning of</li><li>d) at the beginning and end of</li></ul>
		,	23.5) d. 23.6) b. 23.7) d. 23.8) a. 23.9) c. 23.10) d.
-	EXERCISES		<i>ion</i> ) Write an application that calculates the cost of ing list (Fig. 23.18). The application should contain

all the supplies added to the user's shopping list (Fig. 23.18). The application that calculates the cost of two ListBoxes. The first ListBox contains all the supplies offered and their respective prices. Users should be able to select the desired supplies from the first ListBox and add them to the second ListBox. Provide a **Calculate** Button that displays the total price for the user's shopping list (the contents of the second ListBox).





- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial23\Exercises\SupplyCalculator directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click SupplyCalculator.sln in the SupplyCalculator directory to open the application.
- c) Adding code to the Add >> Button. Double click the Add >> Button to create an empty event handler. Add code to the event handler that adds the selected item from the first ListBox to the lstStock ListBox. Make sure to check that at least one item is selected in the first ListBox before attempting to add an item to the lstStock ListBox.
- d) *Enabling the Buttons*. Once the user adds something to the lstStock ListBox, set the Enabled properties of the << Remove and Calculate Buttons to True.
- e) *Deselecting the items*. Once the items are added to the lstStock ListBox, make sure that those items are deselected in the lstSupply ListBox. Also, clear the **Total**: Label to indicate to the user that a new total price must be calculated.
- f) Adding code to the << Remove Button. Double click the << Remove Button to create an empty event handler. Use a Do While loop to remove any selected items in the lstStock ListBox. Make sure to check that at least one item is selected before attempting to remove an item. [*Hint*: Method lstStock.Items.RemoveAt(intIndex) will remove the item located at intIndex from the lstStock ListBox.]
- g) Adding code to the Calculate Button. Double click the Calculate Button to create an empty event handler. Use a For...Next statement to loop through all the items in the lstStock ListBox. Convert each item from the ListBox into a String. Then use the String method Substring to extract the price of each item.
- h) **Displaying the total**. Convert the String representing each item's price to a Decimal, and add this to the overall total (of type Decimal). Remember to output the value in currency format.
- i) Running the application. Select Debug > Start to run your application. Use the Add >> and << Remove Buttons to add and remove items from the Items in Your List: ListBox. Click the Calculate Button and verify that the total price displayed is correct.
- j) *Closing the application.* Close your running application by clicking its close box.
- k) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1 'Exercise 23.11 Solution
2 'SupplyCalculator.vb
3
4 Public Class FrmSupplyCalculator
5 Inherits System.Windows.Forms.Form
6
7 'Windows Form Designer generated code
8
9 'remove item from shopping list
```

```
TO
        Private Sub btnRemove_Click(ByVal sender As _
11
           System.Object, ByVal e As System.EventArgs) _
12
           Handles btnRemove.Click
13
14
            ' if there is at least one item selected
15
           If lstStock.SelectedIndex <> -1 Then
16
17
               ' remove items while there are selected items
18
              Do While lstStock.SelectedIndex <> -1
19
20
                  ' remove item at the selected index
21
                 lstStock.Items.RemoveAt(lstStock.SelectedIndex)
22
              Loop
23
24
           Else
25
               ' display message if there is no item selected
26
              MessageBox.Show("Please select item to remove", _
27
                  "Error Removing", MessageBoxButtons.OK, __
28
                 MessageBoxIcon.Exclamation)
29
           End If
30
31
           ' if there is no item
32
           If lstStock.Items.Count < 1 Then</pre>
33
34
               ' disable the Remove and Calculate Buttons
35
              btnRemove.Enabled = False
36
              btnCalculate.Enabled = False
37
           End If
38
39
           lblTotal.Text = "" ' clear lblTotal Label
40
        End Sub ' btnRemove_click
41
42
        ' add shopping item to list
43
        Private Sub btnAdd_Click(ByVal sender As _
           System.Object, ByVal e As System.EventArgs) _
44
45
           Handles btnAdd.Click
46
47
            ' if there is at least one item selected
48
           If lstSupply.SelectedIndex <> -1 Then
49
50
               ' add each item to lstStock ListBox
51
              lstStock.Items.Add(lstSupply.SelectedItem)
52
53
              btnRemove.Enabled = True ' enable the Remove Button
54
              btnCalculate.Enabled = True ' enable the Calculate Button
55
              lstSupply.SelectedIndex = -1 ' unselect items
56
              lblTotal.Text = "" ' clear lblTotal
57
58
           End If
59
60
        End Sub ' btnAdd_Click
61
62
        ' calculate total price for shopping list
63
        Private Sub btnCalculate_Click(ByVal sender As _
64
           System.Object, ByVal e As System.EventArgs) _
65
           Handles btnCalculate.Click
66
67
           Dim decTotal As Decimal ' total amount
68
           Dim strPrice As String ' temporary price variable
69
           Dim intCounter As Integer ' counter variable
70
```

```
71
             run through list of item(s)
72
           For intCounter = 0 To lstStock.Items.Count - 1
73
74
              ' retrieve price from items
75
              strPrice = lstStock.Items(intCounter).ToString
76
77
              ' get substring starting after the $
78
              strPrice = strPrice.Substring(strPrice.IndexOf("$") + 1)
79
80
              ' add price of each item to total
81
              decTotal += Convert.ToDecimal(strPrice)
82
           Next
83
84
           ' display total
85
           lblTotal.Text = String.Format("{0:C}", decTotal)
86
        End Sub ' btnCalculate_Click
87
88
     End Class ' FrmSupplyCalculator
```

**23.12** (*Encryption Application*) Write an application that encrypts a message from the user (Fig. 23.19). The application should be able to encrypt the message in two different ways: substitution cipher and transposition cipher (both described below). The user should be able to enter the message in a TextBox and select the desired method of encryption. Display the encrypted message in a Label.

In a substitution cipher, every character in the English alphabet is represented by a different character in the substitution alphabet. Every time a letter occurs in the English sentence, it is replaced by the letter in the corresponding index of the substitution string. In a transposition cipher, two Strings are created. The first new String contains all the characters at the even indices of the input String. The second new String contains all of the characters at the odd indices. The new Strings are the encrypted text. For example a transposition cipher for the word "code" would be: "cd oe."

Encryption	
Enter text to encrypt:	The brown fox jumps over the fence
C Substitution Cipher	Transposition Cipher
Encrypted text:	Tebonfxjmsoe h ec h rw o up vrtefne
	Encrypt

Figure 23.19 Text Encryption application's GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial23\Exercises\Encryption directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click Encryption.sln in the Encryption directory to open the application.
- c) Adding code to the Encrypt Button. Double click the Encrypt Button to create an empty event handler.
- d) *Determine the cipher method*. Use If...Then...Else statements to determine which method of encryption the user has selected and call the appropriate procedure.
- e) Locating the SubstitutionCipher method. Locate the SubstitutionCipher procedure. The English and substitution alphabet Strings have been defined for you in this procedure.
- f) Converting the text input to lowercase. Add code to the SubstitutionCipher method that uses the ToLower method of class String to make all the characters in the input string (txtPlainText.Text) lowercase.

- g) **Performing the substitution encryption**. Use nested For...Next loops to iterate through each character of the input String. When each character from the input String is found in the String holding the English alphabet, replace the character in the input String with the character located at the same index in the substitution String.
- h) Display the String. Now that the String has been substituted with all the corresponding cipher characters, assign the cipher String to the lblCipherText Label.
- i) *Locating the TranspositionCipher method*. Locate the TranspositionCipher method. Define three variables—a counter variable and two Strings (each representing a word).
- j) Extracting the first word. Use a Do While...Loop to retrieve all the "even" indices (starting from 0) from the input String. Increment the counter variable by 2 each time, and add the characters located at even indices to the first String created in Step h.
- k) Extracting the second word. Use another Do While...Loop to retrieve all the "odd" indices (starting from 1) from the same input String. Increment the counter variable by 2, and add the characters at odd indices to the second String that you created in Step h.
- Output the result. Add the two Strings together with a space in between, and output the result to the lblCipherText Label.
- m) *Running the application.* Select **Debug > Start** to run your application. Enter text into the **Enter text to encrypt:** TextBox. Select the **Substitution Cipher** RadioButton and click the **Encrypt** Button. Verify that the output is the properly encrypted text using the substitution cipher. Select the **Transposition Cipher** RadioButton and click the **Encrypt** Button. Verify that the output is the properly encrypted text using the transposition cipher.
- n) *Closing the application.* Close your running application by clicking its close box.
- o) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
    ' Exercise 23.12 Solution
2
    ' Encryption.vb
 3
 4
    Public Class FrmEncryption
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
9
        ' using the substitution cipher
10
        Private Sub SubstitutionCipher()
11
12
           ' normal alphabet String
          Dim strNormalAlphabet As String = .
13
14
              "abcdefghijklmnopqrstuvwxyz .!?,"
15
16
           ' substitution alphabet String
17
          Dim strCipherAlphabet As String = _
18
              "cdefg.hijk!lmn opgr?stuv,wxyzab"
19
20
          Dim intIndex1 As Integer ' index variable for For...Next loop
21
          Dim intIndex2 As Integer ' inner index variable
22
          Dim strPlain As String ' String entered by the user
23
          Dim strCipher As String ' encrypted String
24
          lblCipherText.Text = "" ' clear output TextBox
25
26
27
           ' change all the characters to lower case
28
           strPlain = txtPlainText.Text.ToLower
```

```
29
30
            ' iterate through the length of the String
31
           For intIndex1 = 0 To txtPlainText.Text.Length - 1
32
33
               ' iterate through alphabet and special character( .!?,")
34
              For intIndex2 = 0 To 30
35
36
                  ' compare characters
37
                 If strPlain.Chars(intIndex1) = _
38
                     strNormalAlphabet.Chars(intIndex2) Then
39
40
                     ' build encrypted text
41
                     strCipher &= strCipherAlphabet.Chars(intIndex2)
42
43
                 End If
44
45
              Next
46
47
           Next
48
49
           lblCipherText.Text = strCipher ' output the encrypted String
50
        End Sub ' SubstitutionCipher
51
52
        ' using the transposition cipher
53
        Private Sub TranspositionCipher()
54
           Dim intCounter As Integer = 0 ' counter variable
55
           Dim strFirstWord As String ' first word
56
           Dim strLastWord As String ' second word
57
58
           ' create first word from the "even" index
59
           Do While intCounter < txtPlainText.Text.Length</pre>
60
61
              ' add character from specified location to strFirstWord
62
              strFirstWord &= txtPlainText.Text.Chars(intCounter)
63
              intCounter += 2 ' increment counter by 2
64
           Loop ' loop through the entire String
65
           ' create second word from the "odd" indices
66
67
           intCounter = 1
68
69
           Do While intCounter < txtPlainText.Text.Length</pre>
70
              ' add character from specified location to strLastWord
71
72
              strLastWord &= txtPlainText.Text.Chars(intCounter)
73
              intCounter += 2 ' increment counter by 2
74
           Loop ' loop through the entire String
75
76
            ' output encrypted text
77
           lblCipherText.Text = strFirstWord & " " & strLastWord
78
        End Sub ' TranspositionCipher
79
80
        ' encrypt a String of characters
81
        Private Sub btnEncrypt_Click(ByVal sender As System.Object, _
82
           ByVal e As System. EventArgs) Handles btnEncrypt. Click
83
84
           ' determine the selected RadioButton
85
           If radSubstitution.Checked = True Then
86
              SubstitutionCipher() ' call SubstitutionCipher
87
           Else
88
              TranspositionCipher() ' call TranspositionCipher
89
           End If
```

90 91	End Sub ' btnEncrypt_Click
92	
93	End Class ' FrmEncryption

**23.13** (Anagram Game Application) Write an Anagram Game that contains an array of pre-set words (Fig. 23.20). The game should randomly select a word and scramble its letters. A Label displays the scrambled word for the user to guess. If the user guesses correctly, display a message, and repeat the process with a different word. If the guess is incorrect, display a message, and let the user try again.

📮 Anagram Game 📃 🗖 🗙	🔜 Anagram Game 📃 🗆 🗙
semlasby	eaastabd
Your guess: mallesay	Your guess: database
Submit Wrong answer. Try again!	You are Correct!

Figure 23.20 Anagram Game application's GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial23\Exercises\Anagram directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click Anagram.sln in the Anagram directory to open the application.
- c) *Locating the GenerateAnagram method*. Locate the GenerateAnagram method. It is the first method after the FrmAnagram\_Load event handler.
- d) **Picking a random word**. Generate a random number to use as the index of the word in the m\_strAnagram array. Retrieve word from the m\_strAnagram array, using the first random number as an index. Store the word in another String variable. Generate a second random number to store the index of a character to be moved.
- e) Generate the scrambled word. Use a For...Next statement to iterate through the word 20 times. Each time the loop executes, pass the second random number created in *Step c* to the Chars property of class String. Append the character returned by Chars to the end of the String, and remove it from its original position. Next, generate a new random number to move a different character during the next iteration of the loop. Remember to output the final word to the lblAnagram Label.
- f) *Defining the Submit Button*. Double click the **Submit** Button to generate an empty event handler.
- g) *Testing the user's input*. Use an If...Then...Else statement to determine whether the user's input matches the actual word. If the user is correct, clear and place the focus on the TextBox and generate a new word. Otherwise, select the user's text and place focus on the TextBox.
- h) *Running the application.* Select **Debug > Start** to run your application. Submit correct answers and incorrect answers, and verify that the appropriate message is displayed each time.
- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1 ' Exercise 23.13 Solution
2 ' Anagram.vb
3
4 Public Class FrmAnagram
5 Inherits System.Windows.Forms.Form
```

6 7	' Windows Form Designer generated code
8	
9	' array of words to be scrambled
10	<pre>Private m_strAnagram As String() = New String() {"controls", _</pre>
11	"events", "properties", "visual", "program", "application", _
12	"basic", "debugger", "database", "files", "inheritance", _
13	"assembly", "multimedia", "procedures", "functions", _
14	"arrays", "strings", "collections", "integration", _
15	"structures"}
16	
17	Private m_objRandom As Random = New Random ' random number
18	Private m_intRandomNumber As Integer ' random index variable
19	Private m_strScrambled As String ' randomly chosen word
20	
21	' generate new scrambled word
22	Private Sub FrmAnagram_Load(ByVal sender As _
23	System.Object, ByVal e As System.EventArgs) _
24 25	Handles MyBase.Load
25 26	GenerateAnagram() ' generate new word
27	End Sub ' FrmAnagram_Load
28	End Sub FiliiAnayi alii_Load
29	'scramble words
30	Private Sub GenerateAnagram()
31	
32	' generate new random number
33	<pre>m_intRandomNumber = m_objRandom.Next(0, 19)</pre>
34	··_···································
35	' select new word from array with m_intRandomNumber index
36	m_strScrambled = m_strAnagram(m_intRandomNumber)
37	
38	' generate new random index
39	Dim intRandomIndex As Integer = _
40	<pre>m_objRandom.Next(0, m_strScrambled.Length - 1)</pre>
41	
42	Dim intCounter As Integer ' loop counter variable
43	
44	' loop to generate scrambled word
45	For intCounter = $0$ To $20$
46	
47	' attach character at the end of string
48	m_strScrambled &= m_strScrambled.Chars(intRandomIndex)
49 50	' remove character from the word
51	$m_{strScrambled} = m_{strScrambled.Remove(intRandomIndex, 1)$
52	$   _S(I)S(I)                                $
53	' new random index
54	intRandomIndex =
55	<pre>m_objRandom.Next(0, m_strScrambled.Length - 1)</pre>
56	
57	Next
58	
59	lblAnagram.Text = m_strScrambled ' display scrambled word
60	End Sub ' GenerateAnagram
61	
62	' check if the user's answer is correct
63	<pre>Private Sub btnSubmit_Click(ByVal sender As _</pre>
64	System.Object, ByVal e As System.EventArgs) _
65	Handles btnSubmit.Click
66	

```
67
             answer is correct
68
           If txtGuess.Text = m_strAnagram(m_intRandomNumber) Then
69
              lblResult.Text = "You are Correct!"
70
              GenerateAnagram() ' generate new word
71
              txtGuess.Clear() ' clear the TextBox
              txtGuess.Focus() ' place focus on TextBox
72
73
           Else
74
75
              ' answer is incorrect
              lblResult.Text = "Wrong answer. Try again!"
76
77
              txtGuess.Focus() ' place focus on TextBox
78
              txtGuess.SelectAll() ' select the answer
79
           End If
80
81
        End Sub ' btnSubmit_Click
82
83
     End Class ' FrmAnagram
```

```
What does this code do?
                                   23.14 What is assigned to strResult when the following code executes?
                                      1
                                          Dim strWord1 As String = "CHORUS"
                                      2
                                          Dim strWord2 As String = "d i n o s a u r"
                                          Dim strWord3 As String = "The theme is string."
                                      3
                                      4
                                         Dim strResult As String
                                      5
                                      6
                                          strResult = strWord1.ToLower()
                                      7
                                          strResult = strResult.Substring(4)
                                      8
                                          strWord2 = strWord2.Replace(" ", "")
                                      9
                                          strWord2 = strWord2.Substring(4, 4)
                                    10
                                          strResult = strWord2 & strResult
                                    11
                                    12
                                          strWord3 = strWord3.Substring(strWord3.IndexOf(" ") + 1, 3)
                                    13
                                    14
                                          strResult = strWord3.Insert(3, strResult)
                                    Answer: After assigning initial values to Strings strWord1, strWord2 and strWord3, the
                                    code above changes the word "CHORUS" to all lowercase letters, resulting in "chorus" being
                                    assigned to strResult. strResult then is assigned the substring of "chorus" beginning at
                                    the fifth character (index 4), "u", resulting in strResult's value as "us". The next line of
                                    code deletes spaces from strWord2, resulting in the word "dinosaur". The substring of
                                    "dinosaur" beginning at the fifth character and of length 4 ("saur") is then assigned to
                                    strWord2. Then the value of strResult ("us") is appended to the end of strWord2 ("us")
                                    and placed in strResult, yielding "saurus". Following that, strWord3 is assigned a sub-
                                    string of itself beginning one character after the first space character and of length 3 ("the").
                                    Finally, the value "saurus" (strResult) is inserted into strWord3 at the location of the
                                    fourth character (the end of the string) and assigned to strResult. The final value of strRe-
                                    sult is the word "thesaurus".
What's wrong with this code?
                                    23.15 This code should remove all commas from strTest and convert all lowercase letters
                                    to uppercase letters. Find the error(s) in the following code.
                                          Dim strTest As String = "Bug,2,Bug"
                                      1
                                      2
                                      3
                                          strTest = strTest.ToUpper()
                                      4
                                          strTest = strTest.Replace("")
```

**Answer:** Replace method takes two arguments: one substring to search for, and another substring to replace all matching occurrences of the first argument. The proper call to method Replace is shown in the following code.

```
1 Dim strTest As String = "Bug,2,Bug"
2
3 strTest = strTest.ToUpper()
4 strTest = strTest.Replace(",", "")
```

Programming Challenge

**23.16** (*Pig Latin Application*) Write an application that encodes English language phrases into pig Latin. Pig Latin is a form of coded language often used for amusement. Many variations exist in the methods used to form pig Latin phrases. For simplicity, use the following method to form the pig Latin words:

To form a pig Latin word from an English-language phrase, the translation proceeds one word at a time. To translate an English word into a pig Latin word, place the first letter of the English word (if it is not a vowel) at the end of the English word and add the letters "ay." If the first letter of the English word is a vowel, place it at the end of the word and add "y." Using this method, the word "jump" becomes "umpjay", the word "the" becomes "hetay" and the word "ace" becomes "ceay." Blanks between words remain blanks.

Assume the following: The English phrase consists of words separated by blanks, there are no punctuation marks, and all words have two or more letters. Enable the user to input a sentence. The TranslateToPigLatin method should translate the sentence into pig Latin, word by word. [*Hint*: You will need to use the Join and Split methods of class String demonstrated in Fig. 23.16 to form the pig Latin phrases].

🖳 Pig Latin	
Enter a sentence:	This is a sentence to convert to pig Latin
Pig Latin:	histay siy ay entencesay otay onvertcay otay igpay atinlay
	Translate

Figure 23.21 Pig Latin Application.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial23\Exercises\PigLatin directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click PigLatin.sln in the PigLatin directory to open the application.
- c) *Splitting the sentence*. Use method Split on the String passed to the TranslateToPigLatin method. Assign the result of this operation to strWords.
- d) *Retrieving the word's first letter*. Declare a For...Next loop that iterates through your array of words. As you iterate through the array, store each word's first letter in strTemporary.
- e) *Determining the suffix*. Use If...Then...Else statements to determine the suffix for each word. Store this suffix in strSuffix.
- f) *Generating new words*. Generate the new words by arranging each word's pieces in the proper order.
- g) Returning the new sentence. When the For...Next loop finishes, use method Join to combine all of the elements in strWords, and Return the new pig Latin sentence.
- h) *Running the application.* Select **Debug > Start** to run your application. Enter a sentence and click the **Translate** Button. Verify that the sentence is correctly converted into Pig Latin.
- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
    ' Exercise 23.16 Solution
 2
    ' PigLatin.vb
 3
 4
    Public Class FrmPigLatin
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' receive sentence from user and send to TranslateToPigLatin
10
       Private Sub btnTranslate_Click(ByVal sender As System.Object, _
11
          ByVal e As System. EventArgs) Handles btnTranslate. Click
12
13
           ' retrieve English phrase from user
14
          Dim strPhrase As String = txtInput.Text
15
16
          ' display output
17
          txtOutput.Text = TranslateToPigLatin(strPhrase)
18
       End Sub ' btnTranslate_Click
19
20
        ' translates the string input by the user
21
        ' from English to pig Latin
22
       23
          ByVal strEnglishPhrase As String) As String
24
25
          Dim strWords As String() ' array to hold each word
26
          Dim strSuffix As String ' suffix for the end of each word
27
          Dim intIndex As Integer ' index to iterate through the array
28
          Dim strTemporary As String ' temporary string
29
30
          strWords = strEnglishPhrase.Split() ' split words
31
32
          For intIndex = 0 To strWords.Length - 1
33
34
              ' get first letter of each word
35
             strTemporary = strWords(intIndex).Substring(0, 1).ToLower
36
37
             ' check if each word starts with a vowel
38
             If strTemporary = "a" OrElse _
39
                strTemporary = "e" OrElse _
40
                strTemporary = "i" OrElse _
                strTemporary = "o" OrElse _
41
                strTemporary = "u" Then
42
43
44
                strSuffix = "y"
45
             Else ' if not, suffix is different
46
                strSuffix = "ay"
47
             End If
48
49
              ' swap letters to create new word
50
             strWords(intIndex) = strWords(intIndex).Substring(1) & _
51
                strTemporary & strSuffix
52
53
          Next
54
55
           ' put words together and return the whole sentence
56
          Return String.Join(" ", strWords)
57
       End Function ' TranslateToPigLatin
```

**58 59** End Class ' FrmPigLatin





## Ticket Information Application

Introducing Sequential-Access Files Solutions

)	Instructor's Manual Exercise Solutions Tutorial 24		
-	MULTIPLE-CHOICE	<b>24.1</b> Data maintained in a file is cal	led
	QUESTIONS	a) persistent data	b) bits
		c) secondary data	d) databases
		<b>24.2</b> Methods from the	class can be used to write data to a file.
		a) StreamReader	b) FileWriter
		c) StreamWriter	d) WriteFile
		<b>24.3</b> Namespace provid form file processing.	les the classes and methods that you need to use to per-
		a) System.IO	b) System.Files
		c) System.Stream	d) System.Windows.Forms
		<b>24.4</b> Sometimes a group of related	files is called a
		a) field	b) database
		c) collection	d) byte
		<b>24.5</b> A(n) allows the use	er to select a file to open.
		a) CreateFileDialog	b) OpenFileDialog
		c) MessageBox	d) None of the above.
)		24.6 Digits, letters and special symb	bols are referred to as
		a) constants	b) Integers
		c) Strings	d) characters
		24.7 The method reads	a line from a file.
		a) ReadLine	b) Read
		c) ReadAll	d) ReadToNewline
		<b>24.8</b> A contains inform file.	ation that is read in the order that it was written to the
		a) sequential-access file	b) text file
		c) StreamReader	d) StreamWriter
		<b>24.9</b> The smallest data item that a c	computer can support is called a
		a) character set	b) character
		c) special symbol	d) bit
		24.10 Methods from the	class can be used to read data from a file.
		a) StreamWriter	b) FileReader
		c) StreamReader	d) ReadFile
		<b>Answers:</b> 24.1) a. 24.2) c. 24.3) a. 2	24.4) b. 24.5) b. 24.6) d. 24.7) a. 24.8) a. 24.9) d. 24.10) c.
-	EXERCISES	24.11 (Birthday Saver Application	n) Create an application that stores people's names and

**24.11** (*Birthday Saver Application*) Create an application that stores people's names and birthdays in a file (Fig. 24.35). The user creates a file and inputs each person's first name, last name and birthday on the Form. The information is then written to the file.

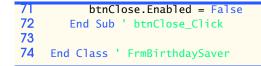
🖳 Birthday Sa	iver	_ <b>_</b> ×
First name:		Open File
Last name:		Enter
Birthday:	12/12/2002 +	Close File

Figure 24.35 Birthday Saver application's GUI.

- a) **Copying the template to your working directory.** Copy the C:\Examples\Tutorial24\Exercises\BirthdaySaver directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click BirthdaySaver.sln in the BirthdaySaver directory to open the application (Fig. 24.35).
- c) Adding and customizing an OpenFileDialog component. Add an OpenFileDialog component to the Form. Change its Name property to obj0penFileDialog. Set the CheckFileExists property to False.
- d) Importing namespace System. 10. Import System. 10 to allow file processing.
- e) Declaring a StreamWriter object. Declare a StreamWriter object that can be used throughout the entire class.
- f) Defining the Open File... Button's Click event handler. Double click the Open File... Button to create the btnOpen\_Click event handler. Write code to display the Open dialog. If the user clicks the Cancel Button in the dialog, then the event handler should perform no further actions. Otherwise, determine whether the user provided a file name that has the .txt extension (indicating a text file). If the user did not, display a MessageBox asking the user to select an appropriate file. If the user specified a valid file name, perform Step f.
- g) *Initializing the StreamWriter*. Initialize the StreamWriter in the btnOpenFile\_Click event handler, passing the user-input file name as an argument. Allow the user to append information to the file by passing the Boolean value True as the second argument to the StreamWriter.
- h) Defining the Enter Button's Click event handler. Double click the Enter Button to create the event handler btnEnter\_Click. This event handler should write the entire name of the person on one line in the file. Then the person's birthday should be written on the next line in the file. Finally, the TextBoxes on the Form should be cleared, and the DateTimePicker's value should be set back to the current date.
- i) **Defining the Close File Button's Click event handler**. Double click the **Close File** Button to create the btnClose\_Click event handler. Close the StreamWriter connection in this event handler.
- j) Running the application. Select Debug > Start to run your application. Open a file by clicking Open File... Button. After a file has been opened, use the input fields provided to enter birthday information. After each person's name and birthday are typed in, click the Enter Button. When you are finished, close the file by clicking the Close File Button. Browse to the file and ensure that its contents contain the birthday information that you entered.
- k) *Closing the application.* Close your running application by clicking its close box.
- 1) *Closing the IDE*. Close Visual Studio .NET by clicking its close box.

1	' Exercise 24.11 Solution
2	'BirthdaySaver.vb
3	
4	Imports System.IO
5	
6	Public Class FrmBirthdaySaver
7	Inherits System.Windows.Forms.Form
8	
9	' StreamWriter used to write to file

10	Private m_objOutput As StreamWriter
11	
12	' Windows Form Designer generated code
13	
14	' Open File Button's Click event
15	<pre>Private Sub btnOpen_Click(ByVal sender As System.Object, _</pre>
16	ByVal e As System.EventArgs) Handles btnOpen.Click
17	
18	' display Open File dialog
19	Dim result As DialogResult = objOpenFileDialog.ShowDialog()
20	bhill resure As brandykesure = objopen riebrandy.showbrandy()
21	' exit event handler if user clicked Cancel Button
22	<pre>If result = DialogResult.Cancel Then</pre>
23	Return
24	End If
25	
26	' get specified file name
27	Dim strFileName As String = objOpenFileDialog.FileName
28	
29	' show error if user specified invalid file
30	<pre>If strFileName.EndsWith(".txt") = False Then</pre>
31	<pre>MessageBox.Show("File name must end with .txt", _</pre>
32	"Invalid File Type", _
33	MessageBoxButtons.OK, MessageBoxIcon.Error)
34	······································
35	Else
36	btnOpen.Enabled = False
37	btnEnter.Enabled = True
38	btnClose.Enabled = True
39	buictose.Ellableu = Thue
40	<pre>m_objOutput = New StreamWriter(strFileName, True)</pre>
41	End If
42	
43	End Sub ' btnOpen_Click
44	
45	' clear all user input
46	Sub ClearUserInput()
47	txtFirstName.Clear()
48	txtLastName.Clear()
49	dtpBirthday.Value = Date.Now
50	End Sub ' ClearUserInput
51	
52	' handles Enter Button's Click event
53	<pre>Private Sub btnEnter_Click(ByVal sender As System.Object, _</pre>
54	ByVal e As System.EventArgs) Handles btnEnter.Click
55	
56	' write user input to file
57	m_objOutput.WriteLine(txtFirstName.Text & " " & _
58	txtLastName.Text)
59	<pre>m_objOutput.WriteLine(dtpBirthday.Value.Month &amp; "/" &amp; _</pre>
60	dtpBirthday.Value.Day & "/" & dtpBirthday.Value.Year)
61	ClearUserInput()
62	End Sub ' btnEnter_Click
63	
	L handles Class File Dutter to Click such
64	' handles Close File Button's Click event
65	Private Sub btnClose_Click(ByVal sender As System.Object, _
66	ByVal e As System.EventArgs) Handles btnClose.Click
67	
68	<pre>m_objOutput.Close() ' close stream</pre>
69	btnOpen.Enabled = True
70	btnEnter.Enabled = False



**24.12** (*Photo Album Application*) Create an application that displays images for the user, as shown in Fig. 24.36. This application should display the current image in a large Picture-Box and display the previous and next images in smaller PictureBoxes. A description of the book represented by the large image should be displayed in a multiline TextBox. The application should use the Directory class's methods to facilitate the displaying of the images.



Figure 24.36 Photo Album application GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial24\Exercises\PhotoAlbum directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click PhotoAlbum.sln in the PhotoAlbum directory to open the application.
- c) *Creating instance variables.* Create instance variable m\_intCurrent to represent the current image that is displayed, and set the variable to 0. Create the m\_strLargeImage array (to store the path names of five large images), the m\_strSmallImage array (to store the path names of five small images) and the m\_strDescriptions array (to store the descriptions of the five books represented by the images).
- d) Defining the RetrieveData procedure. Create a Sub procedure named RetrieveData to store the path names of the larger images in m\_strLargeImages and the path names of the smaller images in m\_strSmallImage. Use the Directory class's Get-CurrentDirectory method to determine the directory path for the images\large and images\small folders. Sequential-access file books.txt stores the file name of each image. The file is organized such that the file name of the small and large images are on the first line. These files have similar names. The small image's file name ends with \_thumb.jpg (that is, filename\_thumb.jpg) while the large image's file name ends with \_large.jpg (that is, filename\_large.jpg). The description of the book, which should be stored in array m\_strDescriptions, follows the file name.
- e) **Defining the DisplayPicture procedure**. Create a Sub procedure named Display-Picture to display the current image in the large PictureBox and to display the previous and next images in the smaller PictureBoxes.

- f) Using If...Then...Else in the DisplayPicture procedure. Use an If...Then...Else statement to display the images on the Form. If the Integer instance variable is 0, display the image of the first book. Also, display the next book's image in the next image PictureBox. However, because there is no previous image, nothing should be displayed in the previous image PictureBox, and the Previous Image Button should be disabled. If the last image is displayed in the large PictureBox, then disable the Next Image Button, and do not display anything in the next image PictureBox. Otherwise, all three PictureBoxes should display their corresponding images, and the Previous Image and Next Image Buttons should be enabled.
- g) **Defining the FrmPhotoAlbum\_Load event handler**. Double click the Form to create the FrmPhotoAlbum\_Load event handler. Invoke methods RetrieveData and DisplayPicture in this event handler.
- h) Defining the btnPrevious\_Click event handler. Double click the Previous Image Button to create the btnPrevious\_Click event handler. In this event handler, decrease the Integer instance variable by 1 and invoke procedure DisplayPicture.
- i) **Defining the btnNext\_Click event handler.** Double click the **Next Image** Button to create the btnNext\_Click event handler. In this event handler, increment the Integer instance variable by 1 and invoke the DisplayPicture procedure.
- j) Running the application. Select Debug > Start to run your application. Click the Previous Image and Next Image Buttons to ensure that the proper images and descriptions are displayed.
- k) *Closing the application.* Close your running application by clicking its close box.
- 1) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 24.12 Solution
 2
     ' PhotoAlbum.vb
 3
 4
    Imports System.IO
 5
 6
    Public Class FrmPhotoAlbum
 7
       Inherits System.Windows.Forms.Form
8
9
        ' represents current image's index
10
        Private m_intCurrent As Integer = 0
11
12
       Private m_strLargeImage As String() = New String(5) {}
13
       Private m_strSmallImage As String() = New String(5) {}
14
       Private m_strDescriptions As String() = New String(5) {}
15
16
        ' Windows Form Designer generated code
17
18
        ' handles Form's Load event
19
        Private Sub FrmPhotoAlbum_Load(ByVal sender As System.Object, _
20
          ByVal e As System. EventArgs) Handles MyBase. Load
21
22
          RetrieveData()
23
          DisplayPicture() ' display first image
24
        End Sub ' FrmPhotoAlbum_Load
25
26
        ' handles Previous Image Button's Click event
27
       Private Sub btnPrevious_Click(ByVal sender As System.Object, __
28
          ByVal e As System. EventArgs) Handles btnPrevious. Click
29
30
          m_intCurrent -= 1
31
          DisplayPicture()
                             ' display new images
32
        End Sub ' btnPrevious_Click
33
34
        ' handles Next Image Button's Click event
```

```
35
        Private Sub btnNext_Click(ByVal sender As System.Object, _
36
           ByVal e As System. EventArgs) Handles btnNext. Click
37
38
           m_intCurrent += 1
39
           DisplayPicture() ' display new images
        End Sub ' btnNext_Click
40
41
42
        ' extract descriptions from file and images from the directory
43
        Private Sub RetrieveData()
44
45
           ' create directory path for large images
46
           Dim strLargeDirectory As String = _
47
              (Directory.GetCurrentDirectory & "\images\large\")
48
49
           ' create directory path for small images
50
           Dim strSmallDirectory As String = _
51
              (Directory.GetCurrentDirectory & "\images\small\")
52
53
           ' initialize StreamReader to read lines from file
54
           Dim objInput As StreamReader = New StreamReader("books.txt")
55
56
           ' read first image name before entering loop
57
           Dim strImageName As String = objInput.ReadLine
58
59
           Dim intCounter As Integer = 0
60
           ' loop through lines in file
61
62
           Do While strImageName <> ""
63
64
              m_strLargeImage(intCounter) = (strLargeDirectory _
                 & strImageName & "_large.jpg")
65
66
              m_strSmallImage(intCounter) = (strSmallDirectory _
67
                 & strImageName & "_thumb.jpg")
68
              m_strDescriptions(intCounter) = objInput.ReadLine
69
70
              ' read next line in file
71
              strImageName = objInput.ReadLine
72
73
              intCounter += 1
74
           Loop
75
76
        End Sub ' RetrieveData
77
78
        ' displays images
79
        Private Sub DisplayPicture()
80
81
           ' set main image
82
           picMain.Image = _
83
                 Image.FromFile(m_strLargeImage(m_intCurrent))
84
85
           ' if index is 0 (first image), do not show previous image
86
           If m_intCurrent = 0 Then
87
              picPrevious.Image = Nothing ' do not show previous image
88
89
              ' preview next image
90
              picNext.Image = _
91
                 Image.FromFile(m_strSmallImage(m_intCurrent + 1))
92
              btnPrevious.Enabled = False ' disable Previous Button
93
94
95
           ' if index corresponds to last item in array,
```

```
96
             do not show next image
97
           ElseIf m_intCurrent = m_strLargeImage.GetUpperBound(0) Then
98
              picPrevious.Image = _
99
                 Image.FromFile(m_strSmallImage(m_intCurrent - 1))
100
101
              picNext.Image = Nothing ' do not show Next image
102
              btnNext.Enabled = False ' disable Next Button
103
104
           ' show previous, current and next image
105
           Else
106
              picPrevious.Image = _
107
                 Image.FromFile(m_strSmallImage(m_intCurrent - 1))
108
109
              picNext.Image =
110
                 Image.FromFile(m_strSmallImage(m_intCurrent + 1))
111
112
              ' enable Buttons
113
              btnPrevious.Enabled = True
              btnNext.Enabled = True
114
115
           End If
116
117
           ' set description
118
           txtDescription.Text = m_strDescriptions(m_intCurrent)
119
        End Sub ' DisplayPicture
120
121 End Class ' FrmPhotoAlbum
```

**24.13** (*Car Reservation Application*) Create an application that allows a user to reserve a car for the specified day. The small car reservation company can rent out only four cars per day. Let the application allow the user to specify a certain day. If four cars have already been reserved for that day, then indicate to the user that no vehicles are available.

•		Ju	<b>ly,</b> 20	03		►
Sun	Mon	Tue	Wed	Thu	Fri	Sat
29	30	1	2	3	4	5
6	7	8	9	10	11	12
<b>3</b>	14	15	16	17	18	19
20	21	22	23	24	25	26
27	28	29	30	31	1	2
3	4	5	6	7	8	9
2	Toda	ay: 7,	/13/2	003		
Name	. F	-	-	-	-	

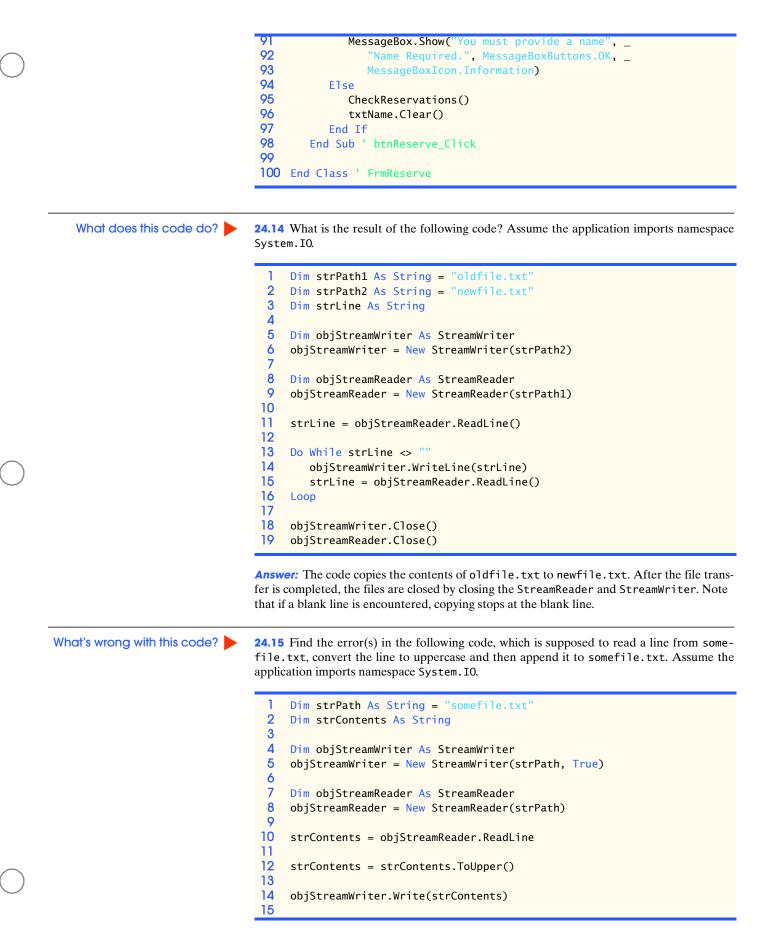
Figure 24.37 CarReservation application's GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial24\Exercises\CarReservation directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click CarReservation.sln in the CarReservation directory to open the application.
- c) Adding a MonthCalendar control to the Form. Drag and drop a MonthCalendar control on the Form. Set the Location property of the control to 16, 32.
- d) Importing System. IO namespace. Import namespace System. IO to allow file processing.
- e) *Defining the FrmReserve\_Load event handler*. Double click the Form to create the FrmReserve\_Load event handler.

- f) Defining a Function procedure. Create a Function procedure named NumberOf-Reservations that takes one argument of type Date. The procedure should create a StreamReader that reads from the reservations.txt file. Use a Do While Loop to allow the StreamReader to search through the entire reservations.txt file to see how many cars have been rented for the day selected by the user. The procedure should close the StreamReader connection and return the number of cars rented for the day selected.
- g) **Defining a Sub procedure**. Create a Sub procedure named CheckReservations. This procedure should invoke the NumberOfReservations Function procedure, passing in the user-selected day as an argument. The CheckReservations method should then retrieve the number returned by NumberOfReservations and determine if four cars have been rented for that day. If four cars have been rented, then display a message dialog to the user stating that no cars are available that day for rental. If fewer than four cars have been rented for that day, create a StreamWriter object, passing reservations.txt as the first argument True as the second argument. Write the day and the user's name to the reservations.txt file and display a message dialog to the user stating that a car has been reserved.
- h) Defining the btnReserve\_Click event handler. Double click the Reserve Car Button to create the btnReserve\_Click event handler. In this event handler, invoke the CheckReservations procedure and clear the Name: TextBox.
- i) *Running the application.* Select **Debug > Start** to run your application. Enter several reservations, including four reservations for the same day. Enter a reservation for a day that already has four reservations to ensure that a message dialog will be displayed.
- j) Closing the application. Close your running application by clicking its close box. Open reservations.txt to ensure that the proper data has been stored (based on the reservations entered in Step i).
- k) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 24.13 Solution
 2
     ' CarReservation.vb
 3
 4
    Imports System.IO
 5
 6
    Public Class FrmReserve
 7
       Inherits System.Windows.Forms.Form
 8
 9
       ' Windows Form Designer generated code
10
11
       ' method to determine number of cars rented for that day
12
       Function NumberOfReservations(ByVal objDay As Date) As Integer
13
           ' set to the value of selected day in month view
14
15
          Dim intChosenDay As Integer = objDay.Day
16
          Dim intFileDay As String
17
          Dim intCars As Integer = 0
18
19
           ' StreamReader reads lines from file to strLine
20
          Dim objFile As StreamReader
21
          Dim strLine As String
22
23
           ' initialize StreamReader
24
           objFile = New StreamReader("reservations.txt")
25
26
           ' read the first line before entering loop
27
           strLine = objFile.ReadLine
28
29
           ' loop through all file data
```

30	Do While strLine <> ""
31	
32	intFileDay = strLine
33	
34	' if days match, increment count
35	<pre>If intFileDay = intChosenDay.ToString Then</pre>
36	
	intCars += 1
37	
38	End If
39	
40	' read name in
41	objFile.ReadLine()
42	
43	' read day of next event in file
44	strLine = objFile.ReadLine
45	
	Loop
46	
47	objFile.Close()
48	
49	Return intCars
50	End Function ' NumberOfReservations
51	
52	' method to check reservations for chosen day
53	Sub CheckReservations()
54	v
55	Dim intCount As Interes
	Dim intCount As Integer
56	Dim intReservations As Integer
57	
58	' gets data for selected day and stores in intReservations
59	<pre>intReservations = NumberOfReservations(mvwDate.SelectionStart)</pre>
60	
61	' determine if user can reserve car that day
62	If intReservations >= 4 Then
63	<pre>MessageBox.Show("Sorry, all cars have been reserved" &amp; _</pre>
64	" for this day. Please select another day.", _
65	"No cars available.", _
	, _
66	MessageBoxButtons.OK, MessageBoxIcon.Information)
67	
68	Else
69	
70	' create StreamWriter to write to file
71	Dim objWrite As StreamWriter = _
72	New StreamWriter("Reservations.txt", True)
73	
74	objWrite.WriteLine(mvwDate.SelectionStart.Day)
75	objWrite.WriteLine(txtName.Text)
76	objWrite.Close()
77	
78	MessageBox.Show("A car has been reserved for you", _
79	"Car reserved", MessageBoxButtons.OK, _
80	MessageBoxIcon.Information)
81	End If
82	
83	End Sub ' CheckReservations
84	
85	' invoke when Reserve Button clicked
86	
	Private Sub btnReserve_Click(ByVal sender As System.Object, _
87	ByVal e As System.EventArgs) Handles btnReserve.Click
88	
89	' determine if name was provided
90	<pre>If txtName.Text = "" Then</pre>



16 17	objStreamWriter.Close() objStreamReader.Close()
open Stre readi	Ver: Once the StreamWriter opens the file specified by strPath, the file is marked . While open, no other object can open the file. Thus an exception is thrown when amReader tries to open the same file. A solution to this problem involves performing the ng operations before the writing operations. Notice in the code below that the Stream- er is closed before the StreamWriter is opened.
1	Dim strPath As String = "somefile.txt"
2	Dim strContents As String
3	, and the second s
4	Dim objStreamReader As StreamReader
5	objStreamReader = New StreamReader(strPath)
6	-
7	strContents = objStreamReader.ReadLine
8	
9	<pre>strContents = strContents.ToUpper()</pre>
10	
11	<pre>objStreamReader.Close()</pre>
12	
13	Dim objStreamWriter As StreamWriter
14	<pre>objStreamWriter = New StreamWriter(strPath, True)</pre>
15	
16	objStreamWriter.Write(strContents)
17	
	objStreamWriter.Close()

Programming Challenge

**24.16** (*File Scrape Application*) Create an application similar to the screen scraping application of Tutorial 23, that opens a user-specified file and searches the file for the price of a book, returning it to the user (Fig. 24.38). [*Hints*: You will need to use the ReadToEnd method of class StreamReader to retrieve the entire contents of the files. The book price appears, for example, in the sample booklist.htm file as Our Price: <br/> <b style="text-align: center;">style="text-align: center;"style="text-align: center;"style="text-a

🛃 File Scrape	
File path:	Open
The price of the book is:	Search

Figure 24.38 File scrape application GUI.

- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial24\Exercises\FileScrape directory to your C:\SimplyVB directory. Notice that two HTML files—booklist.htm and bookpool.htm—are provided for you.
- b) *Opening the application's template file.* Double click FileScrape.sln in the File-Scrape directory to open the application.
- c) *Creating an event handler*. Create an event handler for the **Open...** Button that allows the user to select a file to search for prices.
- d) *Creating a second event handler*. Create an event handler for the **Search** Button. This event handler should search the specified HTML file for the book price. When the price is found, display it in the output Label.
- e) Running the application. Select Debug > Start to run your application. Click the Open... Button and select one of the .htm files provided in the FileScrape directory. Click the Search Button and view the price of the book. For booklist.htm, the price should be \$59.99 and bookpool.htm the price should be \$39.50.

- f) Closing the application. Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
    ' Exercise 24.16 Solution
 2
    ' FileScrapeFig. 24.35.vb
 3
 4
    Imports System.IO
 5
 6
    Public Class FrmScreenScrape
 7
       Inherits System.Windows.Forms.Form
 8
 9
        ' Windows Form Designer generated code
10
        ' display OpenFile dialog when Open... Button is clicked
11
12
       Private Sub btnOpen_Click(ByVal sender As System.Object, _
13
          ByVal e As System. EventArgs) Handles btnOpen. Click
14
15
          Dim result As DialogResult
16
17
           ' show dialog to user for selecting file
18
           result = obj0penFileDialog.ShowDialog()
19
20
          If result = DialogResult.Cancel Then
21
22
              Return ' if user cancels, do nothing
23
          Else
24
25
              ' store file name that user selected
26
              txtPath.Text = objOpenFileDialog.FileName
27
           End If
28
29
       End Sub ' btnOpen_Click
30
31
        ' search file for price
32
       Private Sub btnSearch_Click(ByVal sender As System.Object, _
33
          ByVal e As System. EventArgs) Handles btnSearch. Click
34
35
          Dim strFilePath As String
36
          Dim intMatchLocation As Integer
37
          Dim intRankBegin As Integer
38
          Dim intRankEnd As Integer
39
40
           ' if TextBox is empty, show an error message and return
          If txtPath.Text = "" Then
41
42
             MessageBox.Show("No path selected", __
43
                 "Path Not Chosen", MessageBoxButtons.OK, _
44
                MessageBoxIcon.Exclamation)
45
              Return
46
           Else
47
48
              ' set filepath to value in txtPath
49
              strFilePath = txtPath.Text
50
           End If
51
52
           ' create stream reader to open and read text file
53
          Dim objReader As StreamReader = New StreamReader(strFilePath)
54
55
           ' read file from beginning to end
56
          Dim strContents As String = objReader.ReadToEnd()
```

' close the file to free resource
objReader.Close()
' find locations of text in file
<pre>intMatchLocation = strContents.IndexOf("Our Price:", 0)</pre>
intRankBegin = strContents.IndexOf("\$", intMatchLocation)
intRankEnd = strContents.IndexOf("<", intRankBegin)
' extract price from file
<pre>txtPrice.Text = strContents.Substring(intRankBegin, _</pre>
(intRankEnd - intRankBegin))
End Sub ' btnSearch_Click
End Class ' FrmScreenScrape





# **ATM Application**

Introducing Database Programming Solutions

$\bigcirc$	Instructor's Manual Exercises for Tutorial 25	[ <i>Note:</i> The solutions for this tutorial database. We have done this so that the so student has only learned to connect to dat tions must be run from a specific location.]	abases using absolute paths, so their solu-	
	MULTIPLE-CHOICE	<b>25.1</b> A provides mechanisms for st consistent with a database's format.	toring and organizing data in a manner that is	
	QUESTIONS	<ul><li>a) relational database</li><li>c) data command</li></ul>	<ul><li>b) connection object</li><li>d) database management system</li></ul>	
		<b>25.2</b> An entire row in a database table is know	/n as a	
		<ul><li>a) record</li><li>c) column</li></ul>	b) field d) primary key	
		<ul><li>25.3 A primary key is used to</li><li>a) create rows in a database</li><li>c) distinguish between records in a table</li></ul>	<ul><li>b) identify fields in a database</li><li>d) read information from a database</li></ul>	
		<ul> <li>25.4 A data command object allows you to</li> <li>a) connect to a database</li> <li>c) execute a statement to retrieve or modify a database</li> </ul>	b) read information from a database d) create a database	
$\bigcirc$		<ul><li>25.5 A data reader can</li><li>a) retrieve information from a database</li><li>c) establish a connection to a database</li></ul>	<ul><li>b) modify information stored in a database</li><li>d) close a connection to a database</li></ul>	
$\bigcirc$		<b>25.6</b> In a SELECT statement, what follows the SELECT keyword?		
		<ul><li>a) the name of the table</li><li>c) the name of the database</li></ul>	<ul><li>b) the name of the field</li><li>d) the criteria that the record must meet</li></ul>	
		<b>25.7</b> What does the following SELECT statement	nt do?	
		SELECT Age FROM People WHERE La	astName = 'Purple'	
		a) It selects the age of the person (or peop ple table.	ble) with the last name Purple from the Peo-	
		<ul><li>b) It selects the value Purple from the Ag</li><li>c) It selects the age of the person with the</li><li>d) It selects the People field from the Age</li></ul>	last name Purple from the People database.	
		25.8 The SQL modifies informatio	n in a database.	
		<ul><li>a) SELECT statement</li><li>c) CHANGE statement</li></ul>	<ul><li>b) MODIFY statement</li><li>d) UPDATE statement</li></ul>	
		<b>25.9</b> Which of the following statements mo account number is 2?	difies the Accounts table's PIN field? The	
		<ul> <li>a) SELECT PIN FROM Accounts WHERE Accounts</li> <li>b) SELECT Accounts FROM AccountNumber</li> <li>c) UPDATE Accounts SET PIN=1243 WHERE</li> <li>d) UPDATE PIN=1243 SET AccountNumber</li> </ul>	r = 2 WHERE PIN E AccountNumber = 2	
		25.10 A is an organized collection	of data.	
$\bigcirc$		a) record	b) database	
$\smile$		c) data reader	d) primary key	
		<b>Answers:</b> 25.1) d. 25.2) a. 25.3) c. 25.4) c. 25.5	b) a. 25.6) b. 25.7) a. 25.8) d. 25.9) c. 25.10) b.	

## **EXERCISES**

**25.11** (Stock Portfolio Application) A stock broker wants an application that will display a client's stock portfolio (Fig. 25.37). All the companies that the user holds stock in should be displayed in a ComboBox when the application is loaded. When the user selects a company from the ComboBox and clicks the **Stock Information** Button, the stock information for that company should be displayed in Labels.

ect the name of the stock for which you want rmation, and then press the Stock Info button.	Select the name of the sto information, and then pre	
	ACME Airlines	•
Bug2Bug Deitel Sporting Goods ACME Airlines Books, Inc.	Stock In	formation
Stock Bugs Rugs	Stock Info	
Stock name:	Stock name:	ACME Airlines
Stock symbol:	Stock symbol:	AAIR
Number of shares:	Number of shares:	100
Price per share:	Price per share:	\$21.26
Total value:	Total value:	\$2,126.00

Figure 25.37 Stock Portfolio application.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial25\Exercises\StockPortfolio directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click StockPortfolio.sln in the StockPortfolio directory to open the application.
- c) Copying the database to your working directory. Copy the stocks.mdb database from your C:\Examples\Tutorial25\Exercises\Databases directory to your C:\SimplyVB\StockPortfolio directory.
- d) Adding a data connection to the Server Explorer. Click the Connect to Database icon in the Server Explorer, and add a data connection to the stock.mdb database. Add an OleDbConnection object to the Form.
- e) Adding command objects to the Form. Add two command objects to the Form, and set both their Connection properties to the database connection object. Name the command objects objSelectStockNameCommand and objSelectStockInformationCommand. The first object will be used to retrieve the name of a stock and the second item will be used to retrieve all of a stock's information, based on the name of the stock.
- f) Setting the command objects' CommandText properties. Select the objSelectStock-NameCommand object and click the ellipses Button that appears to the right of the CommandText property in the Properties window. In the Query Builder, select stock-Name from the stocks table and click OK. Select the objSelectStockInformathe Query Builder tionCommand and open as you did for objSelectStockNameCommand. This time, select the stockSymbol, shares and price items from the stocks table. Then, select the stockName item and provide it with the =? criteria value. Finally, uncheck the stockName item from the stocks table. Click OK to dismiss the Query Builder.
- g) Adding a Load event to the Form. Add a Load event handler for the Form. Add code to this event handler that opens a connection to the database. Use the objSelect-StockNameCommand to retrieve the StockNames, and add them to the ComboBox.
- h) Adding a Click event handler for the btnStockInformation Button. Add a Click event handler for the Stock Information Button. Add code to the event handler that passes the SelectedItem to the StockData method as a String. Then close the connection.

- i) **Defining the StockData method.** Create a StockData method that takes a String representing the name of the stock as an argument. Connect to the database, and retrieve the information for the stock passed as an argument (using objSelect-StockInformationCommand). Display the information in the corresponding Labels and close the connection to the database. Call the ComputeTotalValueString method, which you define in the next step, to calculate the total value.
- j) **Defining the ComputeTotalValueString method.** Create the ComputeTotalValueString method to compute the total value by multiplying the number of shares by the price per share.
- k) Running the application. Select Debug > Start to run your application. Select a company from the ComboBox and click the Stock Information Button. Verify that the information displayed in the Stock Info GroupBox is correct, based on the information stored in stock.mdb. Repeat this process for the other companies.
- 1) *Closing the application.* Close your running application by clicking its close box.
- m) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
Exercise 25.11 Solution
 1
 2
    ' StockPortfolio.vb
 3
 4
    Imports System.Data.OleDb
 5
 6
    Public Class FrmStockPortfolio
 7
        Inherits System.Windows.Forms.Form
 8
 9
        ' Windows Form Designer generated code
10
11
        ' called when Form is loaded
12
       Private Sub FrmStockPortfolio_Load(ByVal sender _
13
          As System.Object, ByVal e As System.EventArgs) _
14
          Handles MyBase.Load
15
          obj0leDbConnection.0pen() ' open connection
16
17
18
           ' create data reader to read from database
19
          Dim objReader As OleDbDataReader
20
21
          objReader = objSelectStockNameCommand.ExecuteReader()
22
23
          Do While objReader.Read
24
25
              ' add all stock names in database to the ComboBox
26
              cboStockNames.Items.Add(objReader("stockName"))
27
           Loop
28
29
          obj0leDbConnection.Close() ' close database connection
30
       End Sub ' FrmStockPortfolio_Load
31
32
        ' handles click event for btnStockInformation Button
33
       Private Sub btnStockInformation_Click(ByVal sender As _
          System.Object, ByVal e As System.EventArgs) _
34
35
          Handles btnStockInformation.Click
36
37
           ' String representing stock selected
38
          Dim strSelection As String = _
39
              Convert.ToString(cboStockNames.SelectedItem)
40
41
           ' display stock information
42
           StockData(strSelection)
```

ATM Application 307

43	End Sub ' btnStockInformation_Click
44	
45	' retrieve stock information from database
46	' and set corresponding output Labels with data
47	Private Sub StockData(ByVal strStock As String)
48	
49	<pre>obj0leDbConnection.0pen() ' open connection</pre>
50	objorebbconnectron.open() open connectron
	Low and Concluded in the de Communities of a state of the
51	' specify which stock's information to retrieve
52	<pre>objSelectStockInformationCommand.Parameters( _</pre>
53	<pre>"stockName").Value = strStock</pre>
54	
55	' create data reader to read from database
56	Dim objReader As OleDbDataReader
57	
58	<pre>objReader = objSelectStockInformationCommand.ExecuteReader()</pre>
59	objReader.Read() ' read from the database
60	
61	' set all output Labels
62	' with coresponding stock information
63	lblNameOutput.Text = strStock
64	•
	<pre>lblSymbolOutput.Text =</pre>
65	<pre>Convert.ToString(objReader("stockSymbol"))</pre>
66	
67	<pre>lblShareNumberOutput.Text = _</pre>
68	Convert.ToString(objReader("shares"))
69	
70	Dim decSharePrice As Decimal = _
71	Convert.ToDecimal(objReader("price"))
72	
73	lblSharePriceOutput.Text = _
74	<pre>String.Format("{0:C}", decSharePrice)</pre>
75	
76	objReader.Close() ' close OleDbDataReader
77	
78	lblTotalOutput.Text = ComputeTotalValueString( _
79	<pre>1blShareNumberOutput.Text, Convert.ToString(decSharePrice))</pre>
80	
81	<pre>obj0leDbConnection.Close() ' close database connection</pre>
82	
	End Sub ' StockData
83	L holoon procedure to compute total value of start
84	' helper procedure to compute total value of stock
85	' and return it as a String
86	Private Function ComputeTotalValueString(ByVal strShareNumber _
87	As String, ByVal strSharePrice As String) As String
88	
89	Dim intShareNumber As Integer = _
90	Convert.ToInt32(strShareNumber)
91	Dim decSharePrice As Decimal = _
92	Convert.ToDecimal(strSharePrice)
93	
94	' return the total value formatted as a currency
95	Return String.Format("{0:C}", _
96	(intShareNumber * decSharePrice))
97	
98	End Function ' ComputeTotalValueString
90 99	End Punction Computerotarvaruestring
	End Class I EmpStackDontfalia
100	End Class ' FrmStockPortfolio

**25.12** (Restaurant Bill Calculator Application) A restaurant wants you to develop an application that calculates a table's bill (Fig. 25.38). The application should display all the menu items from the restaurant's database in four ComboBoxes. Each ComboBox should contain a category of food offered by the restaurant (Beverage, Appetizer, Main course and Dessert). The user can choose from one of these ComboBoxes to add an item to a table's bill. When the table is finished ordering, the user can click the Calculate Bill Button to display the Subtotal:, Tax: and Total: for the table.

🖳 Restaurant Bill Calculator 📃 🗖 🗙	🔜 Restaurant Bill Calculator 📃 🗖 🗙
Restaurant	Restaurant
Waiter Information	Waiter Information
Table number:	Table number:
Waiter name:	Waiter name: Sue
Menu Items	Menu Items
Beverage:	Beverage: Soda 💌
Appetizer:	Appetizer: Buffalo Wings
Main course:	Main course: Prime Rib
Dessert:	Dessert: Sundae
Calculate Bill	Calculate Bill
Subtotal:	Subtotal: \$32.80
Tax:	Tax: \$1.64
Total:	Total: \$34.44

Figure 25.38 Restaurant Bill Calculator application.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial25\Exercises\RestaurantBillCalculator directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click RestaurantBillCalculator.sln in the RestaurantBillCalculator directory to open the application.
- c) Copying the database to your working directory. Copy the menu.mdb database from C:\Examples\Tutorial25\Exercises\Databases to your C:\SimplyVB\RestaurantBillCalculator directory.
- d) Adding a data connection to the Server Explorer. Click the Connect to Database icon in the Server Explorer, and add a data connection to the menu.mdb database. Add an OleDbConnection object to the Form.
- e) Adding command objects to the Form. Add two command objects to the Form, and set both their Connection properties to the database connection object. Name the command objects objSelectNameCommand and objSelectPriceCommand. The first object will be used to retrieve the name of a menu item, based on category (for example, appetizer). The second command object will be used to retrieve a menu item's price, based on the item's name.
- f) Setting the command objects' CommandText properties. Select the objSelectName-Command object and open the Query Builder. Add the menu table and select the name and category items. Provide the category item with the =? criteria value, then deselect category in the menu table. Click OK. Select the objSelectPriceCommand and open the Query Builder. This time, select the items marked price and name from the menu table. Provide the name item with the =? criteria value. Finally, uncheck the name item in the menu table. Click OK to dismiss the Query Builder.
- g) Adding a Load event to the Form. Create the Load event handler for the Form. Add code to the event handler that opens a connection to the database. Call the LoadCat-egory method four times, each time passing a different category and ComboBox as arguments. Close the connection to the database.

- h) Coding the LoadCategory method. Create a method LoadCategory that takes a String representing the Category to load and the name of the ComboBox to add items to as arguments. Because the Form's Load event handler is calling this method before it closes the connection to the database, the connection should still be open. Create a data reader to read all the items from the database for the specified Category, using objSelectNameCommand. Close the reader before exiting the method, so that a new reader can be created when the method is invoked again.
- Adding SelectedIndexChanged event handler for the ComboBoxes. Add a SelectedIndexChanged event handler for all the ComboBoxes. Add code to the event handler that adds the String representation of the SelectedItem to the ArrayList.
- j) Adding a Click event handler for the btnCalculateBill Button. Add a Click event handler for the Calculate Bill Button. Add code to the event handler that ensures that a table number and waiter name have been entered. If one of these fields is empty, display a MessageBox informing the user that both fields must contain information. The event handler should then call the CalculateSubtotal method to calculate the subtotal of the bill. Display the subtotal, tax and total of the bill in the appropriate Labels.
- k) Coding the CalculateSubtotal method. The CalculateSubtotal method should open a connection to the database and retrieve the Price field for all the menu items in the m\_objBillItems ArrayList (using objSelectPriceCommand). This method should then calculate the total price of all the items in the ArrayList and return this value as a Decimal. Remember to close the connection to the database.
- Running the application. Select Debug > Start to run your application. Enter a table number and waiter name, and select different menu items from the ComboBoxes. Click the Calculate Bill Button and verify that the subtotal, tax and total values are correct. Select more items from the ComboBoxes and again click the Calculate Button. Verify that the price of the new items has been added to the bill.
- m) *Closing the application.* Close your running application by clicking its close box.
- n) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 25.12 Solution
 2
    ' RestaurantBillCalculator.vb
 3
 4
    Imports System.Data.OleDb
 5
 6
    Public Class FrmRestaurantBillCalculator
 7
       Inherits System.Windows.Forms.Form
 8
9
        ' Windows Form Designer generated code
10
11
       ' hold all items on running bill
12
       Dim m_objBillItems As ArrayList = New ArrayList()
13
14
        ' invoked when application is loaded
15
       Private Sub FrmRestaurantBillCalculator_Load(ByVal sender As _
16
           System.Object, ByVal e As System.EventArgs) _
17
          Handles MyBase.Load
18
19
          obj0leDbConnection.0pen() ' open connection to the database
20
21
           ' load all ComboBoxes with appropriate items
22
          LoadCategory("Beverage", cboBeverage)
23
          LoadCategory("Appetizer", cboAppetizer)
24
          LoadCategory("Main Course", cboMainCourse)
25
          LoadCategory("Dessert", cboDessert)
26
27
          obj0leDbConnection.Close() ' close connection to the database
28
       End Sub ' FrmRestaurantBillCalculator_Load
```

29	
30	' loads the specified category of menu items in
31	' their corresponding ComboBox
32	Private Sub LoadCategory(ByVal strCategory As String, _
33	ByVal cboCategory As ComboBox)
	byvar chocategory AS combobox)
34	
35	' specify category parameter
36	objSelectNameCommand.Parameters( _
37	<pre>"category").Value = strCategory</pre>
38	
39	' declare a reader for the database
40	Dim objMenuReader As OleDbDataReader
41	
42	objMenuReader = _
43	objSelectNameCommand.ExecuteReader()
44	
45	Do While objMenuReader.Read()
46	
47	I metalize some of items in the energified externel.
	' retrieve names of items in the specified category
48	' from database, then add to specified ComboBox
49	<pre>cboCategory.Items.Add(objMenuReader("Name"))</pre>
50	Loop
51	
52	objMenuReader.Close() ' close the reader
53	End Sub ' LoadCategory
54	Lifu Sub LoadCategory
55	' handles SelectedIndexChanged event for cboBeverage ComboBox
56	<pre>Private Sub cboBeverage_SelectedIndexChanged(ByVal sender As _</pre>
57	System.Object, ByVal e As System.EventArgs) _
58	Handles cboBeverage.SelectedIndexChanged
59	5
60	' add selected Beverage to m_objBillItems ArrayList
61	
	<pre>m_objBillItems.Add(cboBeverage.SelectedItem)</pre>
62	End Sub ' cboBeverage_SelectedIndexChanged
63	
64	' handles SelectedIndexChanged event for cboAppetizer ComboBox
65	<pre>Private Sub cboAppetizer_SelectedIndexChanged(ByVal sender As _</pre>
66	System.Object, ByVal e As System.EventArgs) _
67	Handles cboAppetizer.SelectedIndexChanged
68	handres ebonppeerzer isereeteaindexendiged
	I wild as Teached Associations as an abatrian teacher Association
69	' add selected Appetizer to m_objBillItems ArrayList
70	<pre>m_objBillItems.Add(cboAppetizer.SelectedItem)</pre>
71	End Sub ' cboAppetizer_SelectedIndexChanged
72	
73	' handles SelectedIndexChanged event for cboMainCourse ComboBox
74	Private Sub cboMainCourse_SelectedIndexChanged(ByVal sender As _
75	System.Object, ByVal e As System.EventArgs) _
76	Handles cboMainCourse.SelectedIndexChanged
77	
78	' add selected Main Course to m_objBillItems ArrayList
79	m_objBillItems.Add(cboMainCourse.SelectedItem)
80	End Sub ' cboMainCourse_SelectedIndexChanged
81	
82	' handles SelectedIndexChanged event for cboDessert ComboBox
83	
	Private Sub cboDessert_SelectedIndexChanged(ByVal sender As _
84	System.Object, ByVal e As System.EventArgs) _
85	Handles cboDessert.SelectedIndexChanged
86	
87	' add selected Dessert to m_objBillItems ArrayList
88	<pre>m_objBillItems.Add(cboDessert.SelectedItem)</pre>
89	End Sub ' cboDessert_SelectedIndexChanged
~ /	

90	
91	' handles click event for btnCalculateBill Button
92	<pre>Private Sub btnCalculateBill_Click(ByVal sender As _</pre>
93	System.Object, ByVal e As System.EventArgs) _
94	Handles btnCalculateBill.Click
95	
96	' must enter waiter name and table number
97	<pre>If txtWaiterName.Text = "" OrElse _</pre>
98	txtTableNumber.Text = "" Then
99	
100	MessageBox.Show( _
101	"Table Number and Waiter Name must be entered", _
102	"Empty Field", MessageBoxButtons.OK, _
103	MessageBoxIcon.Exclamation)
104	hessageboxicomexchanactony
105	Else ' calculate the bill
106	
107	' calculate the Subtotal
108	Dim decSubtotal As Decimal = CalculateSubtotal()
109	
110	' display the Subtotal in Label
111	lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal)
112	ibisubcocarnesarc.rext - Scring.ronmac( {V.C; , uecsubcocar)
112	' calculate tax and display in Label
114	Dim decTax As Decimal = Convert.ToDecimal( _
115	decSubtotal * 0.05)
116	uecsubtotal * 0.05)
117	<pre>lblTaxPacult Taxt = String Format("{0.Cl" docTax)</pre>
118	<pre>lblTaxResult.Text = String.Format("{0:C}", decTax)</pre>
119	' calculate total and display in Label
120	lblTotalResult.Text =
120	<pre>String.Format("{0:C}", (decSubtotal + decTax))</pre>
122	$String.Format(\{0,C\},(decSubtotar + decrax))$
122	End If
123	
124	End Sub ' btnCalculateBill_Click
125	
120	' calculate the subtotal of the bill
127	Private Function CalculateSubtotal() As Decimal
120	Private Function CarculateSubtotar() As Decimar
130	' open connection to the database
130	
131	obj0leDbConnection.Open()
	' declare a reader for the database
133 134	
134	Dim objMenuReader As OleDbDataReader
136	Dim intCounton Ac Integen
130	Dim intCounter As Integer Dim decSubtotal As Decimal = 0
137	DIM decsublolal AS Decimal = 0
139	For intervention ( To (m objPillTome Count 1)
139	<pre>For intCounter = 0 To (m_objBillItems.Count - 1)</pre>
	I succifie name of them to notations
141	' specify name of item to retrieve
142	<pre>objSelectPriceCommand.Parameters("name").Value =</pre>
143	Convert.ToString(m_objBillItems(intCounter))
144	ahiManuDaadan
145	objMenuReader =
146	objSelectPriceCommand.ExecuteReader()
147	
148	objMenuReader.Read() ' read from the database
149	
150	' retrieve price of items with specified name

1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1	
151	' and add price to dblSubtotal
152	<pre>decSubtotal += Convert.ToDecimal(objMenuReader("Price"))</pre>
153	
154	objMenuReader.Close() ' close the reader
155	Next
156	
157	' close connection to the database
158	<pre>obj0leDbConnection.Close()</pre>
159	
160	Return decSubtotal
161	End Function ' CalculateSubtotal
162	
163	End Class ' FrmRestaurantBillCalculator

**25.13** (Airline Reservation Application) An airline company wants you to develop an application that displays flight information (Fig. 25.39). The database contains two tables, one containing information about the flights, the other containing passenger information. The user should be able to choose a flight number from a ComboBox. When the View Flight Information Button is clicked, the application should display the date of the flight, the flight's departure and arrival cities and the names of the passengers schedule to take the flight.

🔜 Airline Reservatio	on		<u> </u>		
Choose a Flight:	900	View Flight Inform	. I Airline Reservat	ion	
Flight Information	600 700 X 350	Passenger List —	Choose a Flight:	600 💌	View Flight Information
Date: Departure City:	250		Flight Information		Passenger List
Arrival City:		-	Date:	8/24/2003	Green, Joe Gray, Chloe
			Departure City:	Toronto	Pink, Gina
			Arrival City:	Miami	

Figure 25.39 Airline Reservation application.

- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial25\Exercises\AirlineReservation directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click AirlineReservation.sln in the AirlineReservation directory to open the application.
- c) Copying the database to your working directory. Copy the reservations.mdb database from C:\Examples\Tutorial25\Exercises\Databases to your C:\SimplyVB\AirlineReservation directory.
- d) Adding a data connection to the Server Explorer. Click the Connect to Database icon in the Server Explorer, and add a data connection to the reservations.mdb database. Add an OleDbConnection object to the Form.
- e) Adding command objects to the Form. Add three command objects to the Form, and set all their Connection properties to the database connection object. Name the command objects objSelectFlightNumberCommand (used to retrieve flight numbers), objSelectFlightInformationCommand (used to retrieve information about a flight based on the flight's number) and objSelectPassengerInformationCommand (used to retrieve information about a flight's passengers based on the flight's number).
- f) Setting the command objects' CommandText properties. Select the objSelect-FlightNumberCommand object and open the Query Builder. Select FlightNumber from the flights table and click OK. Select the objSelectFlightInformationCommand and open the Query Builder. This time, select the Date, DepartureCity and

ArrivalCity items from the flights table. This action causes all items from the table to be returned. Then, select the FlightNumber item and provide it with the =? criteria value. Finally, uncheck the FlightNumber item from the flights table. Click OK to dismiss the Query Builder. Select the objSelectPassengerInformationCommand and open the Query Builder. Select the LastName and FirstName items from the reservations table. Then, select the FlightNumber item and provide it with the =? criteria value. Finally, uncheck the FlightNumber item from the reservations table. Click OK to dismiss the Query Builder.

- g) Adding a Load event to the Form. Create a Load event handler for the Form that opens a connection to the database. Retrieve all the FlightNumbers from the Flights table in the reservations.mdb database (using objSelectFlightNumber-Command), and add those FlightNumbers to the ComboBox.
- h) Adding a Click event handler for the btnViewFlightInformation Button. Add a Click event handler for the View Flight Information Button. Add code to the event handler to pass the SelectedItem to the DisplayFlightInformation method.
- i) Defining the DisplayFlightInformation method. The DisplayFlightInformation method should take as an argument a String representing the flight number chosen. You will need to define two readers in this method, to read from the two tables in the database. Once you open the connection to the database, create a reader that reads the specified flight information from the flights table (using objSelect-FlightInformationCommand). Display the flight information in the correct Label. Close this reader, and create a second reader that reads passenger information from the reservations table (using objPassengerInformationCommand). Retrieve from the table all the passengers scheduled to take the specified flight. Clear any old items from the ListBox, and display passengers' names in the ListBox.
- j) Running the application. Select Debug > Start to run your application. Select a flight and click the View Flight Information Button. Verify that the flight information is correct. Repeat this process for the other flights.
- k) Closing the application. Close your running application by clicking its close box.
- 1) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 25.13 Solution
2	' AirlineReservation.vb
3	
4	Imports System.Data.OleDb
5	
6	Public Class FrmAirlineReservation
7	Inherits System.Windows.Forms.Form
8	
9	'Windows Form Designer generated code
10	
11	' invoked when Form is loaded
12	Private Sub FrmAirlineReservation_Load(ByVal sender As _
3	System.Object, ByVal e As System.EventArgs) _
4	Handles MyBase.Load
15	
16	<pre>obj0leDbConnection.0pen() ' open database connection</pre>
17	Louiste meden to med information from detailers
8  9	' create reader to read information from database
20	Dim objReader As OleDbDataReader
21	<pre>objReader = objSelectFlightNumberCommand.ExecuteReader()</pre>
22	objkeader = objserectringittkumbercommand.ExecuteReader()
23	Do While objReader.Read()
24	
25	' retrieve flight number from database
26	' and add it to cboChooseAFlight
27	choChooseAFlight.Items.Add(

28

29 30

31 32

33

34 35

36

37 38

39 40

41

42

43 44

45

46 47

48 49

50

51

52 53

54 55

56

57 58

59

60 61

62

63 64

65 66

67

68 69

70

71

72

73

74 75

76 77

78

79

80 81

82

83 84

85

86 87

88

Convert.ToString(objReader("FlightNumber"))) Loop obj0leDbConnection.Close() ' close database connection End Sub ' FrmAirlineReservation\_Load ' handles click event for btnViewFlightInformation Button Private Sub btnViewFlightInformation\_Click(ByVal sender \_ As System.Object, ByVal e As System.EventArgs) \_ Handles btnViewFlightInformation.Click ' retrieve selected index Dim strFlightNumber As String = \_ Convert.ToString(cboChooseAFlight.SelectedItem) ' display new flight information DisplayFlightInformation(strFlightNumber) End Sub ' btnFlightView\_Click ' display flight information in GroupBox's Labels Private Sub DisplayFlightInformation( \_ ByVal strFlightNumber As String) obj0leDbConnection.Open() ' open database connection objSelectFlightInformationCommand.Parameters( \_ "FlightNumber").Value = strFlightNumber ' create data reader to read information from database Dim objFlightReader As OleDbDataReader objFlightReader = \_ objSelectFlightInformationCommand.ExecuteReader() Do While objFlightReader.Read() ' retrieve date, departure city, arrival city lblDateOutput.Text = \_ Convert.ToString(objFlightReader("Date")) lblDepartureOutput.Text = \_ Convert.ToString(objFlightReader("DepartureCity")) lblArrivalOutput.Text = \_ Convert.ToString(objFlightReader("ArrivalCity")) Loop objFlightReader.Close() ' close the data reader ' specify flight number to retrieve passenger information objSelectPassengerInformationCommand.Parameters( \_ "flightNumber").Value = strFlightNumber ' create data reader to read information from database Dim objPassengerReader As OleDbDataReader objPassengerReader = \_ objSelectPassengerInformationCommand.ExecuteReader() Dim strFirstName As String Dim strLastName As String

89	
90	<pre>lstDisplay.Items.Clear() ' clear previous passenger list</pre>
91	
92	<pre>Do While objPassengerReader.Read()</pre>
93	
94	strLastName =
95	Convert.ToString(objPassengerReader("LastName"))
96	<pre>strFirstName = _</pre>
97	Convert.ToString(objPassengerReader("FirstName"))
98	
99	lstDisplay.Items.Add(strLastName & ", " & strFirstName)
100	Loop
101	
102	objPassengerReader.Close() ' close the data reader
103	
104	<pre>obj0leDbConnection.Close() ' close database connection</pre>
105	
106	End Sub ' DisplayFlighInformation
107	
108	End Class ' FrmAirlineReservation

What does this code do? **25.14** What does the following code do?

```
1
    objSelectAgeData.Parameters("Name").Value = "Bob"
 2
 3
    obj0leDbConnection.0pen()
 4
5
    Dim objReader As OleDbDataReader = _
 6
       objSelectAgeData.ExecuteReader
 7
 8
    objReader.Read()
 9
10
    m_intAge = Convert.ToInteger(objReader("Age"))
11
12
    objReader.Close()
13
    obj0leDbConnection.Close()
```

Answer: This code segment connects to a database and retrieves the Age data for the person with the name Bob. This data is then stored in m\_intAge. The code then closes both the data reader and the connection to the database.

What's wrong with this code?	<b>25.15</b> Find the error(s) in the following code. This method should modify the Age field of strUserName.				
	<pre>objUpdateAge("Age").Value = _</pre>				
	2 intAge				
	<pre>4 objUpdateBalance.Parameters("Original_NAME").Value = _ 5 strUserName</pre>				
	6				
	<pre>7 objUpdateAge.ExecuteNonQuery() 8</pre>				
	<pre>9 obj0leDbConnection.Close()</pre>				

objUpdateAge must be used. Also, no connection is made to the database before the UPDATE is attempted.

1	<pre>objUpdateAge.Parameters("Age").Value = _</pre>
2	intAge
3 4 5 6	<pre>objUpdateBalance.Parameters("Original_NAME").Value = _    strUserName</pre>
7	<pre>obj0leDbConnection.0pen()</pre>
8 9 10	objUpdateAge.ExecuteNonQuery()
11	obj0leDbConnection.Close()

Programming Challenge

**25.16** (Enhanced Restaurant Bill Calculator) Modify the application you developed in Exercise 25.12 to keep track of multiple table bills at the same time. The user should be able to calculate a bill for a table and save that table's subtotal and waiter's name. The user should also be able to retrieve that information at a later time. [Hint: This database contains two tables, one for the menu items, as before, and another for all the tables in the restaurant.] Sample outputs are shown in Fig. 25.40.

Re

estaurant Bill Calculator	K Restaurant Bill Calculator
Restaurant	Restaurant
Waiter Information	Waiter Information
Table number:	Table number: 5 💌
Waiter name:	Waiter name: Jon
4 5 6 7	Menu Items
Beverage: 8	Beverage: Soda 💌
Appetizer:	Appetizer: Potato Skins
Main course:	Main course: Seafood Alfredo
Dessert:	Dessert:
btotal:	Sundae Carrot Cake Mud Pie Apple Crisp
Calculate Bil	Tax: Calculate Bill
tal: Pay Bill	Total: Pay Bill
-Waiter Inf Tab Wai -Menu Item	le number: 5 7
Beverage: Appetizer:	
Main cours	
Dessert:	Carrot Cake
Subtotal:	\$32.80 Save Table \$1.64 Calculate Bill
Total:	\$34.44 Pay Bill

Figure 25.40 Enhanced Restaurant Bill Calculator application's GUI.

- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial25\Exercises\RestaurantBillCalculatorEnhanced directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click RestaurantBillCalculator.sln in the RestaurantBillCalculatorEnhanced directory to open the application.
- c) *Copying the database to your working directory*. Copy the menu2.mdb database from C:\Examples\Tutorial25\Exercises\Databases to your C:\Sim-plyVB\RestaurantBillCalculatorEnhanced directory.
- d) Adding a data connection to the Server Explorer. Click the Connect to Database icon in the Server Explorer, and add a data connection to the menu2.mdb database. Add an OleDbConnection object to the Form.
- e) Adding command objects to the Form. Add five command objects to the Form, and set their Connection properties to the database connection object. Name the command objects objSelectNameCommand, objSelectPriceCommand, objSelectTableNumberCommand, objSelectTableInfoCommand and objUpdateSubtotal-Command.

- f) Setting the command objects' CommandText properties. Set the CommandText properties of objSelectNameCommand and objSelectPriceCommand as you did in Exercise 25.12. Set objSelectTableNumberCommand to retrieve table numbers from the tables table. Set objSelectTableInfoCommand to retrieve the name of the waiter and the subtotal of a table, based on that table's number. Set objUpdateSubtotalCommand to modify the subtotal for a table, also based on that table's number. [Note: For the last command object, you will need to change the type of query in the Query Builder, as you did earlier in this tutorial.]
- g) *Copying your existing code.* Copy the code for the application you created in Exercise 25.12 into the template application for this exercise. Place this code before method ResetForm. Disregard any syntax errors that may appear in the **Task List** at this point.
- h) Adding an instance variable. Add an instance variable (after the declaration of m\_objBilltems) called m\_decSubtotal, that will hold the subtotal for each table when it is loaded in the application.
- i) *Modifying methods* btnCalculateBill\_Click and CalculateSubtotal. Remove the portion of the btnCalculateBill\_Click event handler that checked for a table number and waiter name—this information will be displayed shortly. Modify the CalculateSubtotal method to update the table's subtotal based on the table's previous subtotal and any new items selected.
- j) Creating a method. Create method LoadTables that reads the table numbers from the database and adds them to the Table number: ComboBox. This method should be called in FrmRestaurantBillCalculator\_Load directly after the connection to the database is opened.
- k) Adding an event handler. Add an event handler for the Table number: ComboBox. When a table is selected from the ComboBox, that table's data should be loaded from the database.
- Creating an event handler for the Save Table Button. Create an event handler for the Save Table Button. This event handler should calculate the subtotal for the selected table. The event handler should then call method UpdateTable, passing the subtotal and table number as arguments. Finally, call the ResetForm method to reset the data displayed in the GUI.
- m) Creating an event handler for the Pay Bill Button. Create an event handler for the Pay Bill Button. This event handler should retrieve the current table number then call method UpdateTable, passing a subtotal of 0 (for new customers) and table number as arguments. Finally, call the ResetForm method to reset the data displayed in the GUI.
- n) **Creating method UpdateTable**. Create a method UpdateTable that takes the subtotal and table number as arguments. This method should save the table data in the database.
- o) Running the application. Select Debug > Start to run your application. Select a table number and various menu items from the ComboBoxes. Click the Calculate Bill Button and verify that the subtotal, tax and total values are correct. Select more items from the ComboBoxes and again click the Calculate Bill Button. Verify that the price of the new items has been added to the bill. Click the Save Table Button. Select a different table and various menu items. Click the Calculate Bill Button and verify that the price of the new items has been added to the bill. Click the Save Table Button. Select the first table and verify that the subtotal is the same as it was when the table was saved. Select various menu items and Click the Calculate Bill Button. Verify that the subtotal, tax and total values are correct (and now include prices of the new menu items). Click the Pay Bill Button and verify that the subtotal is not reset to \$0.00.
- p) Closing the application. Close your running application by clicking its close box.
- q) *Closing the IDE*. Close Visual Studio .NET by clicking its close box.

' Exercise 25.15 Solution 2 ' RestaurantBillCalculator.vb 3 4 Imports System.Data.OleDb 5 6 Public Class FrmRestaurantBillCalculator 7 Inherits System.Windows.Forms.Form 8 9 ' Windows Form Designer generated code 10 11 ' hold all items on running bill 12 Dim m\_objBillItems As ArrayList = New ArrayList 13 14 Dim m\_decSubtotal As Decimal = 0 ' current table subtotal 15 16 ' invoked when application is loaded 17 Private Sub FrmRestaurantBillCalculator\_Load(ByVal sender As \_ 18 System.Object, ByVal e As System.EventArgs) \_ 19 Handles MyBase.Load 20 21 obj0leDbConnection.Open() ' open connection to the database 22 23 ' load cboTables ComboBox with table numbers 24 LoadTables() 25 26 ' load all ComboBoxes with appropriate items 27 LoadCategory("Beverage", cboBeverage) 28 LoadCategory("Appetizer", cboAppetizer) 29 LoadCategory("Main Course", cboMainCourse) 30 LoadCategory("Dessert", cboDessert) 31 32 obj0leDbConnection.Close() ' close connection to the database 33 End Sub ' FrmRestaurantBillCalculator\_Load 34 35 ' loads the specified category of menu items in 36 ' their corresponding ComboBox 37 Private Sub LoadCategory(ByVal strCategory As String, \_ 38 ByVal cboCategory As ComboBox) 39 40 ' specify category parameter 41 42 "category").Value = strCategory 43 44 ' declare a reader for the database 45 Dim objMenuReader As OleDbDataReader 46 47 objMenuReader = \_ 48 objSelectNameCommand.ExecuteReader() 49 50 Do While objMenuReader.Read() 51 52 ' retrieve names of items in the specified category 53 ' from database, then add to specified ComboBox 54 cboCategory.Items.Add(objMenuReader("Name")) 55 Loop 56 57 objMenuReader.Close() 58 End Sub ' LoadCategory 59 60 ' handles SelectedIndexChanged event for cboBeverage ComboBox

61	Private Sub cboBeverage_SelectedIndexChanged(ByVal sender As _
62	System.Object, ByVal e As System.EventArgs) _
63	Handles cboBeverage.SelectedIndexChanged
64	······································
65	' add selected Beverage to m_objBillItems ArrayList
66	<pre>m_objBillItems.Add(cboBeverage.SelectedItem)</pre>
67	End Sub ' cboBeverage_SelectedIndexChanged
68	
69	' handles SelectedIndexChanged event for cboAppetizer ComboBox
70	<pre>Private Sub cboAppetizer_SelectedIndexChanged(ByVal sender As _</pre>
71	System.Object, ByVal e As System.EventArgs) _
72	Handles cboAppetizer.SelectedIndexChanged
73	
74	' add selected Appetizer to m_objBillItems ArrayList
75	<pre>m_objBillItems.Add(cboAppetizer.SelectedItem)</pre>
76	End Sub ' cboAppetizer_SelectedIndexChanged
77	
78	' handles SelectedIndexChanged event for cboMainCourse ComboBox
79	<pre>Private Sub cboMainCourse_SelectedIndexChanged(ByVal sender As _</pre>
80	System.Object, ByVal e As System.EventArgs) _
81	Handles cboMainCourse.SelectedIndexChanged
82	
83	' add selected Main Course to m_objBillItems ArrayList
84	m_objBillItems.Add(cboMainCourse.SelectedItem)
85	End Sub ' cboMainCourse_SelectedIndexChanged
86	End Sub Coomarneourse_serectedIndexenanged
87	handles SelectedInderChanged event for sheDessent CombeDer
88	' handles SelectedIndexChanged event for cboDessert ComboBox
	Private Sub cboDessert_SelectedIndexChanged(ByVal sender As _
89	System.Object, ByVal e As System.EventArgs) _
90	Handles cboDessert.SelectedIndexChanged
91	
92	' add selected Dessert to objBillItems ArrayList
93	m_objBillItems.Add(cboDessert.SelectedItem)
94	End Sub ' cboDessert_SelectedIndexChanged
95	
96	' handles click event for btnCalculateBill Button
97	Private Sub btnCalculateBill_Click(ByVal sender As _
98	System.Object, ByVal e As System.EventArgs) _
99	Handles btnCalculateBill.Click
100	handles beneared accornication
101	' calculate the Subtotal
102	
102	<pre>Dim decSubtotal As Decimal = CalculateSubtotal()</pre>
	1 display the Cubratel is takel
104	' display the Subtotal in Label
105	<pre>lblSubtotalResult.Text = String.Format("{0:C}", decSubtotal)</pre>
106	
107	' calculate tax and display in Label
108	<pre>Dim decTax As Decimal = Convert.ToDecimal(decSubtotal * 0.05)</pre>
109	
110	lblTaxResult.Text = String.Format("{0:C}", decTax)
111	
112	' calculate total and display in Label
113	lblTotalResult.Text = _
114	<pre>String.Format("{0:C}", (decSubtotal + decTax))</pre>
115	
116	End Sub ' btnCalculateBill_Click
117	
118	' calculate the subtotal of the bill
119	Private Function CalculateSubtotal() As Decimal
120	
121	' open connection to the database

321 ATM Application

122	obj0leDbConnection.Open()
	objorebbconnectron.open()
123	
124	' declare a reader for the database
125	Dim objMenuReader As OleDbDataReader
126	
127	Dim intCounter As Integer
128	Dim decSubtotal As Decimal = m_decSubtotal
	Dim decoubcolar As bechnar = m_decoubcolar
129	
130	<pre>For intCounter = 0 To (m_objBillItems.Count - 1)</pre>
131	
132	' specify name of item to retrieve
133	objSelectPriceCommand.Parameters("name").Value = _
134	Convert.ToString(m_objBillItems(intCounter))
135	
136	objMenuReader = _
137	objSelectPriceCommand.ExecuteReader()
138	
139	' read from the database
140	<pre>Do While objMenuReader.Read()</pre>
141	
142	' retrieve price of items with specified name
143	' and add price to decSubtotal
144	<pre>decSubtotal += Convert.ToDecimal(objMenuReader("Price"))</pre>
145	Loop
146	
147	objMenuReader.Close() ' close the reader
148	Next
149	
150	' close database connection
151	<pre>obj0leDbConnection.Close()</pre>
152	
153	Return decSubtotal
154	End Function ' CalculateSubtotal
	End Function CalculateSublocal
155	
156	' reset all controls on the Form
157	Private Sub ResetForm()
158	
159	' clear all Labels
160	<pre>lblSubtotalResult.Text = ""</pre>
161	lblTaxResult.Text = ""
162	lblTotalResult.Text = ""
163	lblWaiterNameOutput.Text = ""
164	
165	' set Selected index to -1 so
166	' no item is selected in ComboBoxes
167	cboBeverage.SelectedIndex = -1
168	cboAppetizer.SelectedIndex = -1
169	cboMainCourse.SelectedIndex = -1
170	cboDessert.SelectedIndex = -1
171	cboTables.SelectedIndex = -1
	coolables.SelectedIndex = -1
172	
173	<pre>fraMenuItems.Enabled = False ' disable Menu Items GroupBox</pre>
174	
175	' enable Waiter Information GroupBox
176	fraWaiterInformation.Enabled = True
177	
178	' disable all Buttons
179	btnSaveTable.Enabled = False
180	btnCalculateBill.Enabled = False
181	btnPayBill.Enabled = False
182	

© Copyright 1992-2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

183	m decSubtotal = 0 ' clear subtotal
184	
	Landada (17) and the DOMEST and America (17)
185	' reinitialize m_objBillItems ArrayList
186	m_objBillItems = New ArrayList
187	End Sub ' ResetForm
188	
189	' load tables from database
190	Private Sub LoadTables()
	Private Sub Loaurabres()
191	
192	' declare a reader for the database
193	Dim objTableReader As OleDbDataReader
194	
195	objTableReader = _
196	objSelectTableNumberCommand.ExecuteReader()
	objserectrabrendinber command. Executereader ()
197	
198	Do While objTableReader.Read()
199	
200	' retrieve names of items in the specified category
201	' from database, then add to specified ComboBox
202	cboTables.Items.Add(objTableReader("TableNumber"))
203	
204	Loop
205	
206	objTableReader.Close()
207	End Sub ' LoadTables
208	
209	' handles SelectedIndexChanged event for cboTables ComboBox
210	<pre>Private Sub cboTables_SelectedIndexChanged(ByVal sender _</pre>
211	As System.Object, ByVal e As System.EventArgs) _
212	Handles cboTables.SelectedIndexChanged
213	
214	<pre>fraMenuItems.Enabled = True ' enable all Menu Items GroupBox</pre>
215	
216	Loughla all Dutters
	'enable all Buttons
217	btnSaveTable.Enabled = True
218	btnCalculateBill.Enabled = True
219	btnPayBill.Enabled = True
220	
221	' clear Tax and Total output Labels
222	lblTaxResult.Text = ""
223	lblTotalResult.Text = ""
224	
225	obj0leDbConnection.Open() ' open connection to the database
226	
227	' SELECT statement used to retrieve item names
228	<pre>objSelectTableInfoCommand.Parameters( _</pre>
229	"tableNumber").Value = _
230	Convert.ToInt32(cboTables.SelectedItem)
231	
232	' declare a reader for the database
233	Dim objTableReader As OleDbDataReader
234	-
235	objTableReader = _
236	objSelectTableInfoCommand.ExecuteReader()
237	
238	Do While objTableReader.Read()
239	
240	' retrieve waiter name and subtotal from database
241	lblWaiterNameOutput.Text = _
242	Convert.ToString(objTableReader("WaiterName"))
243	

323 ATM Application

244	m_decSubtotal = _
245	Convert.ToDecimal(objTableReader("Subtotal"))
246	lblSubtotalResult.Text = _
247	<pre>String.Format("{0:C}", m_decSubtotal)</pre>
248	
249	Loop
	shiTshlaDaadaa (lass()   slass the moder
250	objTableReader.Close() ' close the reader
251	
252	' close connection to the database
253	obj0leDbConnection.Close()
254	
255	' disable Waiter Information GroupBox
256	<pre>fraWaiterInformation.Enabled = False</pre>
257	End Sub ' cboTables_SelectedIndexChanged
258	
259	' handles click event for btnSaveTable Button
260	Private Sub btnSaveTable_Click(ByVal sender As _
261	System.Object, ByVal e As System.EventArgs) _
262	Handles btnSaveTable.Click
263	
264	Dim decSubtotal As Decimal = CalculateSubtotal()
265	Dim intTableNumber As Integer = _
266	Convert.ToInt32(cboTables.SelectedItem)
267	
268	' update table
269	UpdateTable(decSubtotal, intTableNumber)
270	opualerable(decoublocal, inclablendinber)
271	<pre>ResetForm() ' reset all controls</pre>
272	End Sub ' btnSaveTable_Click
	End Sub Densaverable_entek
273	L handles all the second free handless Table Destroy
274	' handles click event for btnSaveTable Button
275	Private Sub btnPayBill_Click(ByVal sender As _
276	System.Object, ByVal e As System.EventArgs) _
277	Handles btnPayBill.Click
278	
279	Dim intTableNumber As Integer = _
280	Convert.ToInt32(cboTables.SelectedItem)
281	
282	UpdateTable(0, intTableNumber) ' update table
283	ResetForm() ' reset all controls
284	End Sub ' btnPayBill_Click
285	
286	' update table information in database
287	Private Sub UpdateTable(ByVal decSubtotal As Decimal, _
288	ByVal intTableNumber As Integer)
289	
290	<pre>obj0leDbConnection.Open() ' open database connection</pre>
291	
292	' specify parameters for UPDATE statement
293	objUpdateSubtotalCommand.Parameters( _
294	"subtotal").Value = decSubtotal
295	objUpdateSubtotalCommand.Parameters( _
296	"Original_TableNumber").Value = _
297	Convert.ToString(intTableNumber)
298	
299	' execute update statement
300	objUpdateSubtotalCommand.ExecuteNonQuery()
301	
302	<pre>obj0leDbConnection.Close() ' close database connection</pre>
303	End Sub ' UpdateTable
000	

**304 305** End Class ' FrmRestaurantBillCalculator





# **CheckWriter Application**

Introducing Graphics and Printing Solutions

)	Instructor's Manual Exercise Solutions Tutorial 26			
_	MULTIPLE-CHOICE	<b>26.1</b> The RGB value (0, 0, 255) represents	<b>26.1</b> The RGB value (0, 0, 255) represents .	
	QUESTIONS	a) Color.Red c) Color.Blue	b) Color.Green d) Color.Yellow	
		<b>26.2</b> The property of the Prin smoother.	tPreviewDialog object makes text appear	
		a) AntiAlias c) Alias	b) UseAntiAlias d) UseAlias	
		<ul> <li>26.3 Use a object to allow the users</li> <li>a) PrintPreviewDialog</li> <li>c) Print</li> </ul>	s to preview a document before it is printed. b) PrintDocument d) PrintPreviewControl	
		<b>26.4</b> The event handler specifies where the specifies wher		
		a) OnPaint c) Document	b) Print d) PrintPage	
		<ul><li>26.5 To display the preview dialog of the</li><li>a) PrintPreviewDialog</li><li>c) PrintDialog</li></ul>	object, call method ShowDialog. b) PrintDocument d) Both a and b.	
)		<ul><li>26.6 Set the property to False to in a) Document</li><li>c) TerminatePrint</li></ul>	ndicate that there are no more pages to print. b) HasMorePages d) Both a and b.	
		<b>26.7</b> The Print method sends a object to the printer for printing.		
		a) Graphics c) PrintPreviewDialog	b) PrintDocument d) Brush	
		<b>26.8</b> Keyword references the current object.		
		a) This c) Me	b) Class d) Property	
		26.9 Opacity is the value of a color		
		a) red c) dithering	<ul><li>b) transparency</li><li>d) blue</li></ul>	
		<b>26.10</b> Design units are used to specify the of a Font.		
		a) Size c) FontFamily	b) Name d) Style	
		<b>Answers:</b> 26.1) c. 26.2) b. 26.3) a. 26.4) d. 26.5) a. 26.6) b. 26.7) a. 26.8) c. 26.9)		

**EXERCISES** 

**26.11** (CheckWriter Modified to Print Background Images) Modify the CheckWriter application to display and print a background for the check. The GUI should look similar to Fig. 26.31. Users can select a background image. The image should appear in the Print preview dialog box and also should print as a background to the check.

🖶 Check Writer	
	No. Date July 28, 2003 V Amount \$
Pay to the Order Of	Dollars
Memo 000000000 123456789	Signed ————
<ul><li>○ Wood</li><li>⊙ Brick</li></ul>	Preview Print



- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial26\Exercises\ModifiedCheckWriter to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click CheckWriter.sln in the CheckWriter directory to open the application.
- c) *Create the CheckedChanged event handler*. Double click the **Wood** RadioButton to create its CheckedChanged event handler.
- d) **Defining the CheckedChanged event handler**. Define the RadioButton's Checked-Changed event handler to notify the application when users have made a background selection. If the **Wood** RadioButton is selected, then a preview of the wooden background should display in the picPreview PictureBox. Otherwise, if the **Brick** RadioButton is selected, then a preview of the brick background should display in the picPreview PictureBox.
- e) *Modifying the objPrintDocument\_PrintPage event handler*. Modify the objPrintDocument\_PrintPage event handler to print the background image. [*Hint*: Use the DrawImage method to display the background image to print. DrawImage takes five arguments: The image file, the *x*-coordinate, the *y*-coordinate, the width and the height.] To print the image in the background, the DrawImage method must be the first method called on the Graphics object.
- f) Running the application. Select Debug > Start to run your application. Enter data into the input fields and select either the Wood or Brick RadioButton. Verify that the appropriate image is displayed to the left of the RadioButtons. Click the Preview Button and verify that the check is displayed with the proper background. Close the preview and repeat this process selecting the background you had not selected before.
- g) Closing the application. Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 26.11 Solution
 2
    ' CheckWriter.vb
 3
 4
    Imports System.Drawing.Printing
 5
 6
    Public Class FrmCheckWriter
 7
       Inherits System.Windows.Forms.Form
 8
 9
       Private m_objFont As Font ' instance variable to store font
10
       Private m_strPath As String
11
12
        ' Windows Form Designer generated code
13
```

14	' PrintPage event raised for each page to be printed
15	Private Sub objPrintDocument_PrintPage(ByVal sender _
16	As System.Object, ByVal e As PrintPageEventArgs)
17	
18	Dim sngYPosition As Single
19	Dim sngXPosition As Single
20	brin slight of retoin his strigte
21	' represent left margin of page
22	<pre>Dim sngLeftMargin As Single = e.MarginBounds.Left</pre>
23	
24	' represent top margin of page
25	Dim sngTopMargin As Single = e.MarginBounds.Top
26	
27	Dim staling As Chains Nathing
	Dim strLine As String = Nothing
28	Dim objControl As Control
29	
30	' if m_strPath has value, display
31	' specified image on Image control
32	If m_strPath <> "" Then
33	Dim objImage As Image
34	objImage = Image.FromFile(m_strPath)
35	
36	' print image so it is the rear-most object
37	e.Graphics.DrawImage(Image.FromFile(m_strPath), _
38	<pre>sngLeftMargin, sngTopMargin, Me.Width, _</pre>
39	Me.Height - 60)
40	
41	End If
42	
43	' iterate over the form, printing each control
44	For Each objControl In Me.Controls
45	5
46	' we do not want to print Buttons or RadioButtons
47	
	<pre>If objControl.GetType.Name &lt;&gt; "Button" AndAlso _</pre>
48	objControl.GetType.Name <> "RadioButton" Then
49	<pre>strLine = objControl.Text</pre>
50	
51	Select Case objControl.Name
52	5
53	' underline the date
54	
	Case "dtpDate"
55	<pre>m_objFont = New Font("Tahoma", 8.25, _</pre>
56	FontStyle.Underline)
57	
58	' draw a box around amount
59	Case "txtAmount"
60	e.Graphics.DrawRectangle(Pens.Black, _
61	
	<pre>txtAmount.Location.X + sngLeftMargin, _</pre>
62	txtAmount.Location.Y + sngTopMargin - 4, $\_$
63	<pre>txtAmount.Width, txtAmount.Height)</pre>
64	
65	<pre>m_objFont = objControl.Font ' default font</pre>
66	
67	Case Else
68	<pre>m_objFont = objControl.Font ' default font</pre>
69	
70	End Select
71	
72	' set string positions relative to page margins
73	<pre>sngXPosition = sngLeftMargin + _</pre>
74	objControl.Location.X

75	
75	
76	<pre>sngYPosition = sngTopMargin + _</pre>
77	objControl.Location.Y
78	
79	' draw text in graphics object
80	e.Graphics.DrawString(strLine, m_objFont, _
81	Brushes.Black, sngXPosition, sngYPosition)
	Brushes. Black, sngArostition, sngtrostition)
82	
83	End If
84	
85	Next ' control
86	
87	' draw box around check
88	e.Graphics.DrawRectangle(Pens.Black, sngLeftMargin, _
89	sngTopMargin, Me.Width, Me.Height - <u>60</u> )
90	
91	I findforte that there are no more proved to write
	' indicate that there are no more pages to print
92	e.HasMorePages = False
93	
94	End Sub L shiDrintDesument DrintDess
	End Sub ' objPrintDocument_PrintPage
95	
96	' print document
97	Private Sub btnPrint_Click(ByVal sender As _
98	System.Object, ByVal e As System.EventArgs) _
99	Handles btnPrint.Click
100	
	the second second state and second second second second second
101	' create new object to assist in printing
102	Dim objPrintDocument As New PrintDocument
103	
	L toll DriveDesument where to find DriveDess such handles
104	' tell PrintDocument where to find PrintPage event handler
105	AddHandler objPrintDocument.PrintPage, _
106	AddressOf objPrintDocument_PrintPage
107	
108	' if no printers installed, display error message
109	<pre>If PrinterSettings.InstalledPrinters.Count = 0 Then</pre>
110	ErrorMessage()
	5.0
111	Return ' exit event handler
112	End If
113	
	the second se
114	' print the document
115	objPrintDocument.Print()
116	
117	End Sub ! http://www.click
	End Sub ' btnPrint_Click
118	
119	' display document in print preview dialog
120	Private Sub btnPreview_Click(ByVal sender As _
121	System.Object, ByVal e As System.EventArgs) _
122	Handles btnPreview.Click
123	
124	L create new object to acciet in provincing
	' create new object to assist in previewing
125	Dim objPrintDocument As New PrintDocument
126	
127	' tell PrintDocument where to find PrintPage event handler
128	AddHandler objPrintDocument.PrintPage, _
129	AddressOf objPrintDocument_PrintPage
130	· · · · · · · · · · · · · · · · · · ·
	1 if an anistone installed distaller and
131	' if no printers installed, display error message
132	<pre>If PrinterSettings.InstalledPrinters.Count = 0 Then</pre>
133	ErrorMessage()
133	ErrorMessage() Return ' exit event handler
134	Return ' exit event handler

```
136
137
           objPreview.Document = objPrintDocument ' specify document
138
           objPreview.ShowDialog() ' show print preview
139
140
        End Sub ' btnPreview_Click
141
142
        ' display an error message to the user
143
        Sub ErrorMessage()
144
145
           MessageBox.Show("No printers installed. You must " & _
146
              "have a printer installed to preview or print " & _
147
              "the document.", "Print Error", _
              MessageBoxButtons.OK, MessageBoxIcon.Information)
148
149
150
        End Sub ' ErrorMessage
151
152
        ' display selected image in Image control
153
        Private Sub radWood_CheckedChanged(ByVal sender As _
154
            System.Object, ByVal e As System.EventArgs) _
155
            Handles radWood.CheckedChanged
156
157
           ' if Wood RadioButton is selected
158
           ' then set m_strPath to wood image
159
           If radWood.Checked = True Then
160
              m_strPath = "wood.jpg"
161
162
              ' otherwise set m_strPath to brick image
163
           Else
164
              m_strPath = "bricks.jpg"
165
           End If
166
           ' set PictureBox image
167
168
           picPreview.Image = Image.FromFile(m_strPath)
169
        End Sub ' radWood_CheckedChanged
170
171
172 End Class ' FrmCheckWriter
```

**26.12** (Company Logo Designer Application) Develop a Company Logo application that allows users to design a company logo (Fig. 26.32). The application should provide the user with RadioButtons to allow the selection of the next shape to draw. TextBoxes should be provided to allow the user to enter the dimensions of the shapes.

E Company Logo	
Shape	Color
C Rectangle C Filled Rectangle C Line	
C Ellipse C Filled Ellipse	
X1: Y1: X2: Y2:	
Settings	Add
X: Y: Width: Height:	Clear



- a) *Copying the template to your working directory.* Copy the C:\Examples\Tutorial26\Exercises\CompanyLogo directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click CompanyLogo.sln in the CompanyLogo directory to open the application.
- c) **Defining the Add Button's Click event handler.** Create the Add Button's Click event handler. Define the event handler so that the shape that users specify is drawn on the PictureBox. Use the CreateGraphics method on the PictureBox to retrieve the Graphics object used to draw on the PictureBox.
- d) **Defining the Clear Button's Click event handler**. Create the **Clear** Button's Click event handler, and define it so that the PictureBox is cleared. [*Hint*: To clear the entire PictureBox, use the PictureBox's Invalidate method. The Invalidate method is often used to refresh (update) graphics of a control. By using the Invalidate method without specifying a graphic to draw, the PictureBox clears.] Also ensure that all TextBoxes are cleared when the **Clear** Button is clicked.
- e) *Running the application.* Select **Debug > Start** to run your application. Use the RadioButtons and TextBoxes to display at least one of each type of shape. Use different colors for the different shapes. Click the **Clear** Button to clear the shapes.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

1	' Exercise 26.12 Solution
2	' CompanyLogo.vb
3	
4	Public Class FrmLogo
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' adds shapes specified by users to the PictureBox
10	<pre>Private Sub btnAdd_Click(ByVal sender As System.Object, _</pre>
11	ByVal e As System.EventArgs) Handles btnAdd.Click
12	
13	' create a graphics object to draw shapes
14	Dim objGraphics As Graphics = picImage.CreateGraphics
15	
16	' create brush object used for solid shapes

17	Dim objBrush As SolidBrush = New SolidBrush( _
18	Color.FromName(Convert.ToString(cboColor.SelectedItem)))
19	, , , , , , , , , , , , , , , , , , ,
20	L emoste new object used for unfilled change
	' create pen object used for unfilled shapes
21	Dim objPen As Pen = New Pen(objBrush)
22	
23	' create rectangle
24	<pre>If radRectangle.Checked = True Then</pre>
25	
26	' determine if all values are provided
27	If (txtX1.Text $\Rightarrow$ "" AndAlso txtY1.Text $\Rightarrow$ "" AndAlso _
28	<pre>txtX2.Text &lt;&gt; "" AndAlso txtY2.Text &lt;&gt; "") Then</pre>
29	
30	objgraphics.DrawRectangle(objPen, Convert.ToInt32( _
31	<pre>txtXPosition.Text), Convert.ToInt32( _</pre>
32	<pre>txtYPosition.Text), Convert.ToInt32(txtWidth.Text), _</pre>
33	Convert.ToInt32(txtHeight.Text))
34	Else
35	
36	<pre>MessageBox.Show("You did not provide all the setting" &amp; _</pre>
37	" values", "Missing setting values", _
38	MessageBoxButtons.OK, MessageBoxIcon.Information)
39	
40	End If
41	
42	ElseIf radFilledRectangle.Checked = True Then
	Liseli Tauriffeukectangie.checkeu = fiue men
43	
44	' determine if all values are provided
45	<pre>If (txtX1.Text &lt;&gt; "" AndAlso txtY1.Text &lt;&gt; "" AndAlso _</pre>
46	txtX2.Text <> "" AndAlso txtY2.Text <> "") Then
47	
48	' create filled rectangle
49	<pre>objgraphics.FillRectangle(objBrush, Convert.ToInt32( _</pre>
50	txtXPosition.Text), Convert.ToInt32( _
51	<pre>txtYPosition.Text), Convert.ToInt32(txtWidth.Text), _</pre>
52	Convert.ToInt32(txtHeight.Text))
53	Else
54	
55	<pre>MessageBox.Show("You did not provide all the setting" &amp; _</pre>
56	" values", "Missing setting values", _
57	MessageBoxButtons.OK, MessageBoxIcon.Information)
58	hessageboxbacconstort, hessageboxicontintermactory
59	Full TE
	End If
60	
61	ElseIf radEllipse.Checked = True Then
62	
63	' determine if all values are provided
64	<pre>If (txtX1.Text &lt;&gt; "" AndAlso txtY1.Text &lt;&gt; "" AndAlso _</pre>
65	txtX2.Text <> "" AndAlso txtY2.Text <> "") Then
66	
67	' draw ellipse
68	<pre>objgraphics.DrawEllipse(objPen, Convert.ToInt32( _</pre>
69	<pre>txtXPosition.Text), Convert.ToInt32( _</pre>
70	txtYPosition.Text), Convert.ToInt32(txtWidth.Text), $\_$
71	Convert.ToInt32(txtHeight.Text))
72	Else
73	
74	<pre>MessageBox.Show("You did not provide all the setting" &amp; _</pre>
75	" values", "Missing setting values", _
76	MessageBoxButtons.OK, MessageBoxIcon.Information)
77	

78	End If
79	Flootf madrilladFllings Charled True Then
80 81	ElseIf radFilledEllipse.Checked = True Then
82	' determine if all values are provided
83	If (txtX1.Text $\Leftrightarrow$ "" AndAlso txtY1.Text $\Leftrightarrow$ "" AndAlso _
84	txtX2.Text <> "" AndAlso txtY2.Text <> "") Then
85	CXCAZ.TEXC <> ANDATSO CXCTZ.TEXC <> ) THEN
86	' create filled ellipse
87	objgraphics.FillEllipse(objBrush, Convert.ToInt32( _
88	txtXPosition.Text), Convert.ToInt32( _
89	<pre>txtYPosition.Text), Convert.ToInt32(txtWidth.Text), _</pre>
90	Convert.ToInt32(txtHeight.Text))
91	Else
92	
93	<pre>MessageBox.Show("You did not provide all the setting" &amp; _</pre>
94	" values", "Missing setting values", _
95	MessageBoxButtons.OK, MessageBoxIcon.Information)
96	
97	End If
98	
99	ElseIf radLine.Checked = True Then
100	
101	' determine if all values are provided
102	<pre>If (txtX1.Text &lt;&gt; "" AndAlso txtY1.Text &lt;&gt; "" AndAlso _</pre>
103	txtX2.Text <> "" AndAlso txtY2.Text <> "") Then
104	
105	' draw line
106	<pre>objGraphics.DrawLine(objPen, Convert.ToInt32(txtX1.Text), _</pre>
107	Convert.ToInt32(txtY1.Text), Convert.ToInt32( _
108	<pre>txtX2.Text), Convert.ToInt32(txtY2.Text))</pre>
109	Else
110	
111	MessageBox.Show("You did not provide all the X and Y" & _
112	" values", "Missing X Y values", _
113	MessageBoxButtons.OK, MessageBoxIcon.Information)
114	
115	End If
116	
117	End If
118	
119	End Sub ' btnAdd_Click
120	
121	' clear the application GUI
122	<pre>Private Sub btnClear_Click(ByVal sender As System.Object, _</pre>
123	ByVal e As System.EventArgs) Handles btnClear.Click
124	
125	' clear all fields
126	txtX1.Text = ""
127	txtX2.Text = ""
128	txtY1.Text = ""
129	txtY2.Text = ""
130	txtXPosition.Text = ""
131	txtYPosition.Text =
132	txtWidth.Text = ""
133	<pre>txtHeight.Text = ""</pre>
134	
135	' clear PictureBox
136	picImage.Invalidate()
137	
138	End Sub ' btnClear_Click

139 140 End Class ' FrmLogo

**26.13** (*Letter Head Designer Application*) Create a LetterHead application that allows users to design stationery for company documents (Fig. 26.33). Allow users to specify the image that will serve as the letterhead.

	🔜 Letter Head	
PictureBox displays image ——		
User enters contact information here	Image Location:	Preview Print



- a) Copying the template to your working directory. Copy the C:\Examples\Tutorial26\Exercises\LetterHead directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click LetterHead.sln in the LetterHead directory to open the application.
- c) *Creating a PrintPreviewDialog control*. Add a PrintPreviewDialog control to allow users to preview the letterhead before it is printed.
- d) Defining the PrintPage event handler. Allow users to print the document by defining the PrintPage event handler as you did in the CheckWriter application.
- e) **Defining the btnPrint\_Click event handler**. The btnPrint\_Click event handler should tell the PrintDocument where to find the PrintPage event handler, as in the **CheckWriter** application, and print the document.
- f) Defining the btnPreview\_Click event handler. The btnPreview\_Click event handler should tell the PrintDocument where to find the PrintPage event handler, as in the CheckWriter application, and then show the preview dialog.
- g) *Testing the application.* The Letterhead.png image file, located in C:\Examples\Tutorial26\Exercises\Images has been provided for you to test the application's letter head image capability.
- h) *Running the application.* Select Debug > Start to run your application. Enter your contact information and specify the location of an image. [Note: An image has been supplied in an Images directory, located in your C:\Examples\Tutorial26\Exercises directory. The image should be displayed in the PictureBox at the top of the Form. Click the Preview Button and verify that the image and contact information is displayed in the preview. Finally, click the Print Button to verify that the letterhead prints with the appropriate image and contact information.
- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1 'Exercise 26.13 Solution
2 'LetterHead.vb
3
4 Imports System.Drawing.Printing
5
6 Public Class FrmLetterHead
```

```
Inherits System.Windows.Forms.Form
7
 8
9
        ' create font object
10
       Private m_objFont As Font
11
12
        ' Windows Form Designer generated code
13
14
        ' PrintPage event raised for each page to be printed.
15
       Private Sub objPrintDocument_PrintPage(ByVal sender _
16
          As Object, ByVal e As PrintPageEventArgs)
17
18
          Dim sngYPosition As Single
19
          Dim sngXPosition As Single
20
21
          Dim sngLeftMargin As Single = e.MarginBounds.Left
22
          Dim sngTopMargin As Single = e.MarginBounds.Top
23
24
          Dim strPath As String
25
26
           ' get location of image
27
           strPath = txtImage.Text
28
29
           ' make sure image location was provided
          If strPath <> "" Then
30
31
             Dim objImage As Image
32
              objImage = Image.FromFile(strPath)
33
34
              ' print image so it is on top of page
35
              e.Graphics.DrawImage(Image.FromFile(strPath), _
36
                 sngLeftMargin + picImage.Location.X, _
37
                 sngTopMargin + picImage.Location.Y, _
38
                 picImage.Size.Width, picImage.Size.Height)
39
40
           End If
41
42
           ' if contact information is provided, print data
43
           If txtInformation.Text <> "" Then
44
45
              ' specifies font of text
46
              m_objFont = New Font("Tahoma", 12, _
47
                 FontStyle.Bold)
48
49
              sngXPosition = sngLeftMargin + _
50
                 txtInformation.Location.X
51
              sngYPosition = sngTopMargin + _
52
53
                 txtInformation.Location.Y
54
55
              ' print information
56
              e.Graphics.DrawString(txtInformation.Text, m_objFont, _
57
                 Brushes.Black, sngXPosition, sngYPosition)
58
           End If
59
60
           ' indicate there are no more pages to print
61
           e.HasMorePages = False
62
63
        End Sub ' objPrintDocument_PrintPage
64
65
        ' print the document
66
        Private Sub btnPrint_Click(ByVal sender As _
67
           System.Object, ByVal e As System.EventArgs)
```

68	Handles btnPrint.Click
69	
70	' create new object to assist in printing
71	Dim objPrintDocument As New PrintDocument
72	
73	' tell printer where to find PrintPage event handler
74	AddHandler objPrintDocument.PrintPage, _
75	AddressOf objPrintDocument_PrintPage
76 77	' print the document
78	<pre>' print the document objPrintDocument.Print()</pre>
79	
80	End Sub ' btnPrint_Click
81	
82	' display document in print preview dialog
83	Private Sub btnPreview_Click(ByVal sender As _
84	System.Object, ByVal e As System.EventArgs) _
85	Handles btnPreview.Click
86	
87	Dim objPrintDocument As PrintDocument = _
88	New PrintDocument
89	
90	AddHandler objPrintDocument.PrintPage, _
91	AddressOf objPrintDocument_PrintPage
92	
93	<pre>objPreview.Document = objPrintDocument</pre>
94	objPreview.ShowDialog()
95	
96	End Sub ' btnPreview_Click
97	
98	' add the specified image to the Image control
99	Private Sub btnAdd_Click(ByVal sender As System.Object, _
100	ByVal e As System.EventArgs) Handles btnAdd.Click
101 102	Limment the image energified
102	' import the image specified If txtImage.Text <> "" Then
103	II LALIMAYE. TEXT <> THEM
104	<pre>picImage.Image = Image.FromFile(txtImage.Text)</pre>
105	picimage.image - image.ifOmFile(tktimage.lekt)
107	Else
108	
109	MessageBox.Show("You must enter a path for your image.", _
110	"Input Error", MessageBoxButtons.OK, _
111	MessageBoxIcon.Information)
112	
113	EndIf
114	
115	End Sub ' btnAdd_Click
116	
117	End Class ' FrmLetterHead

What does this code do?

26.14 What is the result of the following code? Assume that objOutput\_PrintPage is defined.

1	<pre>Private Sub btnPrint_Click(ByVal sender As System.Object, _</pre>
2	ByVal e As System.EventArgs) Handles btnPrint.Click
3	
4	Dim objOutput As New PrintDocument
5	

6 AddHandler objOutput.PrintPage, \_
7 AddressOf objOutput\_PrintPage
8
9 objPrintOutput.Print()
10
11 End Sub ' btnPrint\_Click

**Answer:** The code indicates that the PrintPage event for objOutput should invoke the objOutput\_PrintPage event handler.

What's wrong with this code?

**26.15** Find the error(s) in the following code. This is the definition for a Click event handler for a Button. This event handler should draw a rectangle on a PictureBox control.

```
1
    Private Sub btnDrawImage_Click(ByVal sender As System.Object, _
 2
       ByVal e As System. EventArgs) Handles btnDrawImage. Click
 3
 4
        ' create an orange colored brush
 5
       Dim objBrush As SolidBrush = New SolidBrush(Orange)
 6
 7
        ' create a Graphics object to draw on the PictureBox
 8
       Dim objGraphics As Graphics = picPictureBox.AcquireGraphics
 9
10
        ' draw a filled rectangle
11
       objGraphics.FillRectangle(objBrush, 2, 3, 40, 30)
12
13
    End Sub ' btnDrawImage_Click
```

**Answer:** When specifying a color for the SolidBrush you must precede the color name with "Color." You cannot just write the color name. Also, to retrieve the Graphics object, the CreateGraphics method must be used, not AcquireGraphics. The corrected code is shown below.

```
1
    Private Sub btnDrawImage_Click(ByVal sender As System.Object, _
 2
       ByVal e As System.EventArgs) Handles btnDrawImage.Click
 3
 4
        ' create an orange colored brush
 5
       Dim objBrush As SolidBrush = New SolidBrush(Color.Orange)
 6
 7
        ' create a Graphics object to draw on the PictureBox
 8
       Dim objGraphics As Graphics = picPictureBox.CreateGraphics
 9
10
        ' draw a filled rectangle
11
       objGraphics.FillRectangle(objBrush, 2, 3, 40, 30)
12
13
    End Sub ' btnDrawImage_Click
```

Programming Challenge

**26.16** (Screen Saver Simulator Application) Develop an application that simulates a screen saver. This application should add random-colored, random-sized, solid and hollow shapes at different positions of the screen (Fig. 26.34). Copy the C:\Exercises\Tutorial26\ScreenSaver directory, and place it in your C:\SimplyVB directory. The design of the Form has been created, which consists of a black Form and a Timer control. In the ScreenSaver.vb code view, the DisplayShape method has been provided and the Timer's tick event handler has already been defined for you.

You must write the rest of the DisplayShape method code. Create the Graphics object from the Form using the Form's CreateGraphics method, and specify random colors, sizes and positions for the filled and hollow shapes that will be displayed on the screen. The width and height of the shapes should be no larger than 100 pixels.

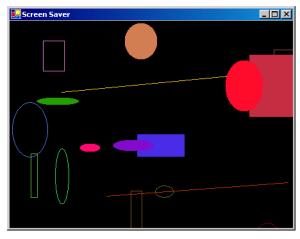


Figure 26.34 Screen Saver running.

### Answer:

1	' Exercise 26.16 Solution
2	'ScreenSaver.vb
3	
4	Public Class FrmShapeChanger
5	Inherits System.Windows.Forms.Form
6	,
7	Private m_dblCount As Double = $0.0$
8	-
9	' Windows Form Designer generated code
10	
11	' create a specified graphic on the form
12	Private Sub DisplayShape()
13	
14	' create a Graphics object
15	Dim objGraphicsObject As Graphics = Me.CreateGraphics
16	
17	' create random object for random number generation
18	Dim objRandom As Random = New Random
19	-
20	' create random color
21	Dim objColor As Color = Color.FromArgb( _
22	objRandom.Next(0, 255), objRandom.Next(0, 255), _
23	objRandom.Next(0, 255))
24	
25	' create brush object used for solid shapes
26	Dim objBrush As SolidBrush = New SolidBrush( _
27	objColor)
28	
29	' create pen object used for unfilled shapes
30	Dim objPen As Pen = New Pen(objBrush)
31	
32	' create random number used to create random shape
33	<pre>Dim intShape As Integer = objRandom.Next(0, 5)</pre>
34	
35	' set to width of form
36	Dim intWidth As Integer = Me.Size.Width
37	
38	' set to height of form
39	Dim intHeight As Integer = Me.Size.Height
40	
41	' decide which shape to draw

43

44 45

46

47

**48** 

49

50

51 52

53 54

55

56

57

**58** 

59

61

62 63

64

65

66

67

68

69

71 72

73

74

75

76

77

78 79

80 81

82

83

84

85

86

87

88

91

92

93

94

```
Select Case intShape
              Case 0
                 ' create filled rectangle
                 objGraphicsObject.FillRectangle(objBrush, _
                    objRandom.Next(0, Width), _
                    objRandom.Next(0, Height), _
                    objRandom.Next(10, 100), _
                    objRandom.Next(10, 100))
              Case 1
                 ' create filled ellipse
                 objGraphicsObject.FillEllipse(objBrush, _
                    objRandom.Next(0, Width), _
                    objRandom.Next(0, Height), _
                    objRandom.Next(10, 100), _
                    objRandom.Next(10, 100))
60
              Case 2
                 ' draw ellipse
                 objGraphicsObject.DrawEllipse(objPen, _
                    objRandom.Next(0, Width), _
                    objRandom.Next(0, Height), _
                    objRandom.Next(10, 100), _
                    objRandom.Next(10, 100))
70
              Case 3
                 ' draw rectangle
                 objGraphicsObject.DrawRectangle(objPen, _
                    objRandom.Next(0, Width), _
                    objRandom.Next(0, Height), _
                    objRandom.Next(10, 100), _
                    objRandom.Next(10, 100))
              Case 4
                 ' draw line
                 objGraphicsObject.DrawLine(objPen, _
                    objRandom.Next(0, Width), _
                    objRandom.Next(0, Height), _
                    objRandom.Next(10, Width), _
                    objRandom.Next(10, Height))
           End Select
89
       End Sub ' DisplayShape
90
        ' invoked each time tmrScreenSaver ticks
        Private Sub tmrScreenSaver_Tick(ByVal sender As System.Object, _
           ByVal e As System. EventArgs) Handles tmrScreenSaver. Tick
95
           m_dblCount += 0.25
96
           ' draw shape every half second
98
           If m_dblCount Mod 2.5 = 0 Then
99
              DisplayShape() ' draw another shape
100
           End If
101
102
       End Sub ' tmrScreenSaver_Tick
```

104 End Class ' FrmShapeChanger

26.17 (Screen Saver Simulator Enhancement Application) Enhance the Screen Saver Simulator application from Exercise 26.16 by modifying the Timer control's Tick event handler. Add code to this event handler so that after a specified amount of time, the screen should clear the displayed shapes. After the screen clears, random shapes should continue to display. Also, modify the code so that you can specify random opacity (alpha values) for the colors by using Color structure's FromArgb method. You should pass four arguments to this method. The first argument is the alpha value, the second is the red value, the third is the green value and the fourth is the blue value.

```
Answer:
```

103

```
1
     ' Exercise 26.17 Solution
 2
    ' ScreenSaver.vb
 3
 4
    Public Class FrmShapeChanger
 5
       Inherits System.Windows.Forms.Form
 6
 7
       Private m_dblCount As Double = 0.0
 8
9
        ' Windows Form Designer generated code
10
11
        ' create a specified graphic on the form
12
       Private Sub DisplayShape()
13
14
           ' create a Graphics object
15
          Dim objGraphicsObject As Graphics = Me.CreateGraphics
16
17
           ' create random object for random number generation
18
          Dim objRandom As Random = New Random
19
20
           ' create random color with random opacity
21
          Dim objColor As Color = Color.FromArgb( _
22
             objRandom.Next(0, 255), objRandom.Next(0, 255), _
23
             objRandom.Next(0, 255), objRandom.Next(0, 255))
24
25
           ' create brush object used for solid shapes
26
          Dim objBrush As SolidBrush = New SolidBrush(objColor)
27
28
           ' create pen object used for unfilled shapes
29
          Dim objPen As Pen = New Pen(objBrush)
30
31
           ' create random number used to create random shape
32
          Dim intShape As Integer = objRandom.Next(0, 5)
33
34
           ' set to width of form
35
          Dim intWidth As Integer = Me.Size.Width
36
37
           ' set to height of form
38
          Dim intHeight As Integer = Me.Size.Height
39
40
           ' decide which shape to draw
41
          Select Case intShape
42
             Case 0
43
44
                 ' create filled rectangle
45
                objGraphicsObject.FillRectangle(objBrush, _
46
                    objRandom.Next(0, Width), _
                    objRandom.Next(0, Height),
```

© Copyright 1992-2004 by Deitel & Associates, Inc. and Pearson Education Inc. All Rights Reserved.

```
objRandom.Next(10, 100), _
48
49
                     objRandom.Next(10, 100))
50
51
              Case 1
52
53
                  ' create filled ellipse
54
                  objGraphicsObject.FillEllipse(objBrush, _
55
                     objRandom.Next(0, Width), _
56
                     objRandom.Next(0, Height), _
                     objRandom.Next(10, 100), _
57
58
                     objRandom.Next(10, 100))
59
60
              Case 2
61
                  ' draw ellipse
62
                  objGraphicsObject.DrawEllipse(objPen, _
63
64
                     objRandom.Next(0, Width), _
65
                     objRandom.Next(0, Height), _
                     objRandom.Next(10, 100), _
66
67
                     objRandom.Next(10, 100))
68
69
              Case 3
70
71
                  ' draw rectangle
72
                  objGraphicsObject.DrawRectangle(objPen, _
73
                     objRandom.Next(0, Width), _
74
                     objRandom.Next(0, Height), _
75
                     objRandom.Next(10, 100), _
76
                     objRandom.Next(10, 100))
77
78
              Case 4
79
80
                  ' draw line
                  objGraphicsObject.DrawLine(objPen, _
81
82
                     objRandom.Next(0, Width), _
83
                     objRandom.Next(0, Height), _
84
                     objRandom.Next(10, Width),
85
                     objRandom.Next(10, Height))
86
           End Select
87
88
        End Sub ' DisplayShape
89
90
        ' invoked each time tmrScreenSaver ticks
91
        Private Sub tmrScreenSaver_Tick(ByVal sender As System.Object, _
92
           ByVal e As System. EventArgs) Handles tmrScreenSaver. Tick
93
94
           m_dblCount += 0.25
95
96
           ' draw shape every half second
97
           If m_dblCount Mod 2.5 = 0 Then
              DisplayShape() ' draw another shape
98
99
           End If
100
101
           If m_dblCount Mod 200 = 0 Then
102
              Me.Invalidate() ' clear screen
103
           End If
104
105
        End Sub ' tmrScreenSaver_Tick
106
107 End Class ' FrmShapeChanger
```





# **Phone Book Application**

Introducing Multimedia Using Microsoft Agent Solutions

$\bigcirc$	Instructor's Manual Exercise Solutions Tutorial 27			
-	MULTIPLE-CHOICE	27.1 The metho	od is used to specify what the Microsoft Agent will say.	
	QUESTIONS	a) Speak c) Command	b) Say d) Voice	
			,	
		a) Show	od is used to activate a Microsoft Agent character's animation b) Play	n.
		c) Speak	d) Appear	
		27.3 Method MoveTo takes	s two arguments. What do these arguments represent?	
			ich the Agent should move (left, right, up, down).	
		b) The name of the cha	-	
		c) The x-coordinate an move.	nd y-coordinate of the location to which the Agent object sho	ula
		d) The name of the cha	aracter and the direction of movement.	
		<b>27.4</b> Which method of IAge screen?	entCtlCharacter displays the Microsoft Agent character or	n the
		a) Play	b) Show	
		c) Speak	d) Appear	
$\bigcirc$		<b>27.5</b> Use the er character context menu.	event handler to execute code when users click Hide the A	gent
$\bigcirc$		a) Hide	b) HideEvent	
		c) Command	d) Disappear	
		<b>27.6</b> The Add method of the	e Commands property	
		a) adds a new comman		
		<ul><li>b) joins two commands</li><li>c) displays the Command</li></ul>	-	
		d) both a and c	ius pop-up window.	
		27.7 The event	handler controls what occurs when users speak to the Agent	
		a) Command	b) ClickEvent	
		c) Click	d) SelectedIndexChanged	
		27.8 specifies th	he <i>x</i> -coordinate of the mouse cursor on the screen.	
		a) Cursor.Location.2	X b) Cursor.Position.X	
		c) Mouse.Location.X	( d) Mouse.Position.X	
			as a parameter to Peedy's Play method causes him to smile.	
		a) "Think"	b) "Smile"	
		c) "Pleased"	d) "Happy"	
			as a parameter to Peedy's Play method causes him to rest.	
		a) "RestPose"	b) "Rest"	
		c) "Think"	d) "Pose"	
$\bigcirc$		<b>Answers:</b> 27.1) a. 27.2) b. 2	27.3) c. 27.4) b. 27.5) b. 27.6) a. 27.7) a. 27.8) b. 27.9) c. 27.1	<mark>0)</mark> a.

## EXERCISES

**27.11** (*Appointment Book Application Using Microsoft Agent*) Write an application that allows users to add appointments to an appointment book that uses Microsoft Agent. When users speak a person's name, Merlin returns the time and date of the appointment that users have with that person. If users say "Today", Merlin returns a list of the users' appointments for the day.

🛃 Appointment Book		
Appointment with: Appointment date: Appointment time:	January 01, 2003 💌	



- a) Copying the template to your working directory. Copy the directory C:\Examples\Tutorial27\Exercises\AppointmentBook to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click AppointmentBook.sln in the AppointmentBook directory to open the application.
- c) *Downloading the Merlin Microsoft Agent*. Download the Merlin.acs character file from the Microsoft Web site.
- d) Adding the Agent Control to the Form. Add the Microsoft Agent control to the Form.
- e) *Creating module-level variables.* Create three module-level variables of type Array-List to store the date, time and person with which the user has an appointment. Create a module-level variable of type AgentObjects.IAgentCtlCharacter (as you did in the **Phone Book** application).
- f) Defining the FrmAppointments\_Load event handler. Load Merlin's character file, display him on the screen and add the "Today" command to the command list.
- g) **Defining the btnAdd\_Click event handler**. Define this event handler so that the information provided by the user is added to its corresponding ArrayList. The **Appointment With:** TextBox input should be added to the ArrayList containing the names of people with whom the user has an appointment. The input for the appointment date and time should also be added to their respective ArrayLists. Display an error message if the user leaves the **Appointment With:** or the **Appointment Time:** TextBox empty.
- h) Adding voice-enabled commands. Within the btnAdd\_Click event handler, add a voice-enabled command that allows a user to speak the name of the person with whom the user has an appointment to the command list. This allows a user to check for whether there is an appointment with someone by speaking the person's name. The command should also appear in the Commands context menu.
- i) Defining the Agent's Command event handler. As you did in the Phone Book application, define what occurs when a user speaks or selects a command. If the user specifies the Today command, Merlin should tell the user the names of all the people with whom the user has an appointment today. If the user specifies a specific name, Merlin should state the time and date at which the user has an appointment with this person. If the user did not schedule any appointments, then Merlin should inform the user that no appointments were scheduled.
- j) Running the application. Select Debug > Start to run your application. Enter various appointments, where at least two of the appointments are scheduled for the current day. Input the name of the person you are meeting at one of the appointments by either speaking the name into your microphone, or right-clicking the agent and selecting that person's name. Verify that the agent repeats back correct information about that appointment. Input the value "Today" by either speaking it into the microphone or right-clicking the agent and selecting Today. Verify that the agent repeats back all the appointments for the current day.
- k) Closing the application. Close your running application by clicking its close box.

1) *Closing the IDE.* Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 27.11 Solution
 2
    ' AppointmentBook.vb
 3
 4
    Public Class FrmAppointments
 5
       Inherits System.Windows.Forms.Form
 6
 7
       ' create three Arraylists
 8
       Private m_objPerson As ArrayList = New ArrayList
 9
       Private m_objDate As ArrayList = New ArrayList
10
       Private m_objTime As ArrayList = New ArrayList
11
12
        ' represent current Agent
13
       Private m_objMSpeaker As AgentObjects.IAgentCtlCharacter
14
15
        ' Windows Form Designer generated code
16
17
        ' invoked when Form is loaded
18
       Private Sub FrmAppointments_Load(ByVal sender As _
19
          System.Object, ByVal e As System.EventArgs) _
20
          Handles MyBase.Load
21
22
           ' load agent character file
23
          objMainAgent.Characters.Load("Merlin", "Merlin.acs")
24
25
          ' specify current agent
26
          m_objMSpeaker = objMainAgent.Characters("Merlin")
27
28
           ' show Merlin on screen
29
          m_objMSpeaker.Show(0)
30
31
          ' add voice enabled command to Agent object
32
          m_objMSpeaker.Commands.Add("Today", "Today", _
33
              "Today", True, True)
34
       End Sub 'FrmAppointments_Load
35
36
        ' add appointments to ArrayLists
37
       Private Sub btnAdd_Click(ByVal sender As System.Object, _
38
          ByVal e As System. EventArgs) Handles btnAdd. Click
39
40
          If txtWith.Text = "" OrElse txtTime.Text = "" Then
41
42
             43
                "You left one or more fields empty above", _
44
                45
                MessageBoxIcon.Error)
46
          Else
47
              add information to ArrayLists
48
             m_objPerson.Add(txtWith.Text)
49
             m_objDate.Add(dtpAddDate.Value.ToLongDateString)
50
             m_objTime.Add(txtTime.Text)
51
52
             m_objMSpeaker.Commands.Add(txtWith.Text,
53
                txtWith.Text, txtWith.Text, True, True)
54
55
             ' clear TextBoxes
56
             txtWith.Clear()
57
             txtTime.Clear()
```

59

60 61 62

63

64

65

66 67

68

69

70 71

72 73

74

75

76

77 78

79

80 81

82 83

84 85

86

87 88

89

90

91

92

93

94 95

96

97 98

99 100

101

102

103

104

105

106

107

108 109

110 111

112

113 114

115

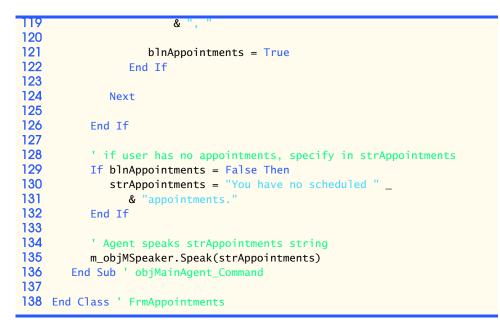
116

117 118 End If

End Sub ' btnAdd\_Click ' determine what Agent does when it hears command Private Sub objMainAgent\_Command(ByVal sender As Object, \_ ByVal e As AxAgentObjects.\_AgentEvents\_CommandEvent) \_ Handles objMainAgent.Command ' get UserInput object Dim objCommand As AgentObjects.IAgentCtlUserInput = \_\_\_\_\_\_ CType(e.userInput, AgentObjects.IAgentCtlUserInput) Dim intCount As Integer ' boolean to determine if user has appointments Dim blnAppointments As Boolean = False Dim strAppointments As String = \_ "Today you have an appointment with:" ' determine if user spoke Today command If objCommand.Name = "Today" Then ' search objDate arraylist for appointments ' set for today For intCount = 0 To m\_objDate.Count - 1 If Convert.ToString(m\_objDate(intCount)) = \_ Today.Date.ToLongDateString Then ' add the appointment to string Agent speaks strAppointments = strAppointments & \_ Convert.ToString(m\_objPerson(intCount)) \_ & " at " & \_ Convert.ToString(m\_objTime(intCount)) \_ & ", blnAppointments = True End If Next Else ' specify string Agent speaks strAppointments = "You have to " & \_ "meet with " & \_ Convert.ToString(objCommand.Name) & \_ " on: " ' determine if user made command using ' person's name For intCount = 0 To m\_objPerson.Count - 1 If objCommand.Name = \_ Convert.ToString(m\_objPerson(intCount)) Then ' add date and time to string Agent speaks strAppointments = strAppointments & \_ Convert.ToString(m\_objDate(intCount)) \_

& " at " & \_

Convert.ToString(m\_objTime(intCount))



**27.12** (Craps Game Application Using Microsoft Agent) Modify the Craps Game application from Tutorial 16 to include a Microsoft Agent character.

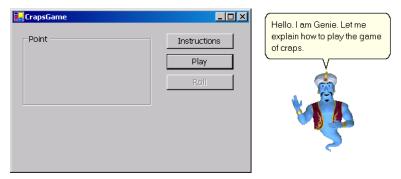


Figure 27.33 Modified Craps Game GUI.

- a) **Copying the template to your working directory.** Copy the directory C:\Examples\Tutorial27\Exercises\CrapsGameEnhancement to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click CrapsGame.sln in the Craps-GameEnhancement directory to open the application.
- c) *Downloading the Genie Microsoft Agent*. Download the Genie.acs character file from the Microsoft Web site.
- d) Adding the Agent control to the Form. Add the Microsoft Agent control to the Form.
- e) *Creating a module-level variable*. Create a module-level variable of type AgentObjects.IAgentCtlCharacter (as you did in the **Phone Book** application).
- f) Defining the FrmCrapsGame\_Load event handler. Load Genie's character file, and display him on the screen.
- g) Modifying the btnPlay\_Click event handler. Add code to the btnPlay\_Click event handler to control the Agent. When the user wins the game, Genie should play his Pleased animation and congratulate the user. If the user loses, Genie should play his Confused animation and say that the user lost. If the user neither wins nor loses, Genie should tell the user to roll again. Make sure to reset him to his RestPose after he plays any animation.
- h) Defining the btnRol1\_Click event handler. Add code to the btnRol1\_Click event handler to control the Agent. If users "make their point," Genie should play his Pleased animation and state that the user won. If the user rolls a 7, Genie should

play his Confused animation and say that the user lost. Otherwise, Genie should tell the user to roll again.

- i) **Defining** the **btnInstructions\_Click** event handler. Define the **btnInstructions\_Click** event handler to make Genie introduce himself to the user. Genie should then explain the rules to the game of craps.
- j) Running the application. Select Debug > Start to run your application. Click the Instructions Button and allow the agent character to tell you the rules of the game. Use the Play and Roll Buttons to play a few games of craps. When you need to roll again, verify that the agent tells you to roll again. Also, verify that the agents informs you whether you won or lost at the end of each game.
- k) *Closing the application.* Close your running application by clicking its close box.
- 1) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
     ' Exercise 27.12 Solution
 2
    ' CrapsGame.vb
 3
 4
    Imports System.IO
 5
 6
    Public Class FrmCrapsGame
 7
       Inherits System.Windows.Forms.Form
 8
 9
        ' die-roll constants
10
       Enum DiceNames
          SNAKE_EYES = 2
11
12
           TREY = 3
13
          CRAPS = 7
14
          LUCKY_SEVEN = 7
15
          YO_LEVEN = 11
          BOX\_CARS = 12
16
17
       End Enum
18
19
        ' filename and directory constants
20
       Const m_strFILE_PREFIX As String = "/images/die"
21
       Const m_strFILE_SUFFIX As String = ".png"
22
23
        ' instance variables
24
        Private m_intMyPoint As Integer = 0
25
       Private m_objRandom As Random = New Random
26
27
        ' create object that represents current agent
28
       Private m_objMSpeaker As AgentObjects.IAgentCtlCharacter
29
30
        ' Windows Form Designer generated code
31
32
        ' begin new game and determine point
33
       Private Sub btnPlay_Click(ByVal sender As System.Object, _
34
             ByVal e As System. EventArgs) Handles btnPlay. Click
35
36
           ' initialize variables for new game
37
          m_intMyPoint = 0
           fraPointDiceGroup.Text = "Point"
38
39
          lblStatus.Text = "
40
41
           ' remove point-die images
42
           picPointDie1.Image = Nothing
43
           picPointDie2.Image = Nothing
44
45
          Dim intSum As Integer = RollDice() ' roll dice
```

40	
47	' check die roll
48	Select Case intSum
49	
50	Case DiceNames.LUCKY_SEVEN, _
51	DiceNames.YO_LEVEN ' win on first roll
52	
53	<pre>btnRoll.Enabled = False ' disable roll button</pre>
54	lblStatus.Text = "You win!!!"
55	
56	' play Agent's Pleased animation
57	<pre>m_objMSpeaker.Play("Pleased")</pre>
58	
59	' make Agent speak
60	<pre>m_objMSpeaker.Speak("Congratulations, you won!")</pre>
	m_objmspeaker.speak( congraturations, you won! )
61	
62	m_objMSpeaker.Play("RestPose")
63	
64	Case DiceNames.SNAKE_EYES, _
65	DiceNames.TREY, _
66	DiceNames.BOX_CARS ' lose on first roll
67	
68	btnRoll.Enabled = False
69	lblStatus.Text = "Sorry. You lose."
70	
71	' nlaw Agent's Confused animation
	' play Agent's Confused animation
72	m_objMSpeaker.Play("Confused")
73	
74	' make Agent speak
75	
	<pre>m_objMSpeaker.Speak("Sorry, you lost!")</pre>
76	
77	<pre>m_objMSpeaker.Play("RestPose")</pre>
78	<u>_</u> ,,,,,,, _
79	<b>Case Else '</b> player must match point
80	<pre>m_intMyPoint = intSum</pre>
81	fraPointDiceGroup.Text = "Point is " & intSum
82	
	lblStatus.Text = "Roll Again!"
83	<pre>m_objMSpeaker.Speak("Please roll again.")</pre>
84	picPointDie1.Image = picDie1.Image
85	picPointDie2.Image = picDie2.Image
86	DisplayDie(picPointDie2, m_intMyDie2)
87	btnPlay.Enabled = False ' disable Play Button
88	<pre>btnRoll.Enabled = True ' enable Roll Button</pre>
89	End Select
90	
91	End Sub ' btnPlay_Click
92	
93	Laboranda contacto de acosta de 11
	' determine outcome of next roll
94	<pre>Private Sub btnRoll_Click(ByVal sender As System.Object, _</pre>
95	ByVal e As System.EventArgs) Handles btnRoll.Click
96	,
97	Dim intSum As Integer = RollDice()
98	
99	' determine outcome of roll
100	
	<pre>If intSum = m_intMyPoint Then ' player matches point</pre>
101	lblStatus.Text = "You win!!!"
102	<pre>m_objMSpeaker.Play("Pleased")</pre>
103	<pre>m_objMSpeaker.Speak("Congratulations, you won!")</pre>
104	<pre>m_objMSpeaker.Play("RestPose")</pre>
105	
106	

107	btnPlay.Enabled = True
108	<pre>ElseIf intSum = DiceNames.CRAPS Then ' player loses</pre>
109	<pre>lblStatus.Text = "Sorry. You lose."</pre>
110	<pre>m_objMSpeaker.Play("Confused")</pre>
111	<pre>m_objMSpeaker.Speak("Sorry, you lost!")</pre>
112	<pre>m_objMSpeaker.Play("RestPose")</pre>
113	
114	<pre>btnRoll.Enabled = False</pre>
115	btnPlay.Enabled = True
116	,
117	Else
118	<pre>m_objMSpeaker.Speak("Please roll again.")</pre>
119	End If
120	
121	End Sub ' btnRoll_Click
122	
122	L semente moder die wille
	' generate random die rolls
124	Private Function RollDice() As Integer
125	
126	' roll the dice
127	<pre>Dim intDie1 As Integer = m_objRandom.Next(1, 7)</pre>
128	<pre>Dim intDie2 As Integer = m_objRandom.Next(1, 7)</pre>
129	
130	' display image corresponding to each die
131	DisplayDie(picDie1, intDie1)
132	DisplayDie(picDie2, intDie2)
133	
134	Return (intDie1 + intDie2) ' return sum of dice values
135	
136	End Function ' RollDice
137	
138	' display die image
139	Private Sub DisplayDie(ByVal picDie As PictureBox, _
140	ByVal intFace As Integer)
141	byvar micrace As integer)
142	' assign die images to PictureBoxes
142	picDie.Image = _
143	
144	<pre>Image.FromFile(Directory.GetCurrentDirectory &amp;</pre>
	<pre>m_strFILE_PREFIX &amp; intFace &amp; m_strFILE_SUFFIX)</pre>
146	
147	End Sub ' DisplayDie
148	
149	' invoked when Form is loaded
150	<pre>Private Sub FrmCrapsGame_Load(ByVal sender As System.Object, _</pre>
151	ByVal e As System.EventArgs) Handles MyBase.Load
152	
153	' load Genie character file
154	objMainAgent.Characters.Load("Genie", "Genie.acs")
155	
156	m_objMSpeaker = objMainAgent.Characters("Genie")
157	
158	' show Genie on the screen
159	m_objMSpeaker.Show(0)
160	End Sub ' FrmCrapsGame_Load
161	End Sub TrincrupSound_E0dd
162	Agent explains instructions of the same
	' Agent explains instructions of the game
163	Private Sub btnInstructions_Click(ByVal sender As System.Object,
164	ByVal e As System.EventArgs) Handles btnInstructions.Click
165	
166	m_objMSpeaker.Play("Wave")
167	

168	m_objMSpeaker.Speak("Hello. I am Genie. Let me " & _
169	"explain how to play the game of craps.")
170	
171	m_objMSpeaker.Speak("Click the Play button to begin " & _
172	"the game.")
173	
174	<pre>m_objMSpeaker.Speak("Clicking Play causes you to roll" _</pre>
175	& " two dice.")
176	
177	m_objMSpeaker.Speak("If the sum of your dice roll is " _
178	& "7 or 11 on your first throw, then you win.")
179	
180	<pre>m_objMSpeaker.Speak("However, if the sum is 2, 3, or " _</pre>
181	& "12 on your first throw, then you lose.")
182	
183	m_objMSpeaker.Speak("If the sum is 4, 5, 6, 7, 8, 9 " _
184	& "or 10 on your first throw, then that sum " _
185	& "becomes your 'point'")
186	
187	m_objMSpeaker.Speak("To win, you must continue rolling" _
188	& " the dice until you roll the point value again.")
189	
190	m_objMSpeaker.Speak("You lose if you roll a 7 before " _
191	& "reaching your point value.")
192	
193	m_objMSpeaker.Play("RestPose")
194	End Sub ' btnInstructions_Click
195	
196	End Class ' FrmCrapsGame

**27.13** (Security Panel Application Using Microsoft Agent) Modify the Security Panel application from Tutorial 12 to include Microsoft Agent.

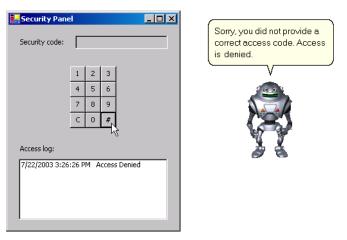


Figure 27.34 Robby from modified Security Panel application.

- a) *Copying the template to your working directory*. Copy the directory C:\Examples\Tutorial27\Exercises\SecurityPanelEnhancement to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click SecurityPanel.sln in the SecurityPanelEnhancement directory to open the application.
- c) *Downloading the Robby Microsoft Agent*. Download the Robby.acs character file from the Microsoft Web site.
- d) Adding the Agent control to the Form. Add the Microsoft Agent control to the Form.

- e) *Creating a module-level variable*. Create a module-level variable of type AgentObjects.IAgentCtlCharacter (as you did in the **Phone Book** application).
- f) Defining the FrmSecurityPane1\_Load event handler. Load Robby's character file, and display him on the screen. Command Robby to tell users to input their access codes.
- g) Modifying the btnEnter\_Click event handler. Add code to the btnEnter\_Click event handler to use the Microsoft Agent. If the user enters a valid access code, Robby should welcome the user and state the type of employee that the access code represents. If the access code is invalid, then Robby should state that an invalid code was provided and that access is denied.
- h) *Running the application.* Select **Debug > Start** to run your application. Enter various access codes. For correct access codes, verify that the agent tells you what type of employee the access code represents. For incorrect access codes, verify that the agents tells you that access is denied.
- i) *Closing the application.* Close your running application by clicking its close box.
- j) Closing the IDE. Close Visual Studio .NET by clicking its close box.

```
1
      Exercise 27.13 Solution
 2
    ' SecurityPanel.vb
 3
 Δ
    Public Class FrmSecurityPanel
 5
       Inherits System.Windows.Forms.Form
 6
 7
       Private m_objMSpeaker As AgentObjects.IAgentCtlCharacter
 8
 9
        ' Windows Form Designer generated code
10
11
       Private Sub btnEnter_Click(ByVal sender As System.Object, _
12
         ByVal e As System. EventArgs) Handles btnEnter. Click
13
14
          Dim intAccessCode As Integer ' stores access code entered
15
          Dim strMessage As String ' displays access status of users
16
17
           intAccessCode = Convert.ToInt32(txtSecurityCode.Text)
18
           txtSecurityCode.Clear()
19
20
           Select Case intAccessCode ' check access code input
21
22
              ' access code less than 10
23
              Case Is < 10
24
                 strMessage = "Restricted Access"
25
                 m_objMSpeaker.Speak("Welcome. You have " _
26
                     & "restricted access.")
27
28
              ' access code between 1645 and 1689
29
              Case 1645 To 1689
30
                 strMessage = "Technicians"
31
                 m_objMSpeaker.Speak("Welcome. Your access" _
32
                     & " code indicates that you are "
33
                     & "part of the Technician Personnel.")
34
35
              ' access code equal to 8345
36
             Case 8345
37
                 strMessage = "Custodians"
38
                 m_objMSpeaker.Speak("Welcome. Your access" _
39
                     & " code indicates that you are "
40
                     & "part of Custodial Services.")
41
```

43

44 45

46

47

48

49 50

51

52

53

54

55

56 57

58 59

60

61 62 63

64

65

66 67

68 69

70

71 72

73

74 75

76

77 78

79

80 81

82

83 84

85

86 87

88

89 90

91

92 93

94

95 96

97

```
access code equal to 9998 or between
              ' 1006 and 1008
              Case 9998, 1006 To 1008
                 strMessage = "Scientists"
                m_objMSpeaker.Speak("Welcome. Your access" _
                     & " code indicates that you are " _
                     & "part of the Scientific Personnel.")
              ' if no other Case is True
              Case Else
                 strMessage = "Access Denied"
                m_objMSpeaker.Speak("Sorry, you did not " _
                     & "provide a correct access code." _
                     & " Access is denied.")
          End Select
           ' display time and message in ListBox
          lstLogEntry.Items.Add(Date.Now & " " & strMessage)
        End Sub ' btnEnter_Click
        Private Sub btnZero_Click(ByVal sender As System.Object, _
          ByVal e As System. EventArgs) Handles btnZero. Click
           txtSecurityCode.Text &= "0" ' concatenate "0" to display
        End Sub ' btnZero_Click
        Private Sub btnOne_Click(ByVal sender As System.Object, _
          ByVal e As System. EventArgs) Handles btnOne. Click
           txtSecurityCode.Text &= "1" ' concatenate "1" to display
       End Sub ' btnOne_Click
        Private Sub btnTwo_Click(ByVal sender As System.Object, _
          ByVal e As System.EventArgs) Handles btnTwo.Click
          txtSecurityCode.Text &= "2" ' concatenate "2" to display
       End Sub ' btnTwo_Click
        Private Sub btnThree_Click(ByVal sender As System.Object, _
          ByVal e As System. EventArgs) Handles btnThree. Click
           txtSecurityCode.Text &= "3" ' concatenate "3" to display
       End Sub ' btnThree_Click
        Private Sub btnFour_Click(ByVal sender As System.Object, _
          ByVal e As System. EventArgs) Handles btnFour. Click
           txtSecurityCode.Text &= "4" ' concatenate "4" to display
       End Sub ' btnFour_Click
       Private Sub btnFive_Click(ByVal sender As System.Object, _
          ByVal e As System. EventArgs) Handles btnFive. Click
           txtSecurityCode.Text &= "5" ' concatenate "5" to display
       End Sub ' btnFive_Click
        Private Sub btnSix_Click(ByVal sender As System.Object, _
100
          ByVal e As System. EventArgs) Handles btnSix. Click
101
          txtSecurityCode.Text &= "6" ' concatenate "6" to display
102
```

103	End Sub ' btnSix_Click
104	
105	<pre>Private Sub btnSeven_Click(ByVal sender As System.Object, _</pre>
106	ByVal e As System.EventArgs) Handles btnSeven.Click
107	
108	<pre>txtSecurityCode.Text &amp;= "7" ' concatenate "7" to display</pre>
109	End Sub ' btnSeven_Click
110	
111	<pre>Private Sub btnEight_Click(ByVal sender As System.Object, _</pre>
112	ByVal e As System.EventArgs) Handles btnEight.Click
113	
114	<pre>txtSecurityCode.Text &amp;= "8" ' concatenate "8" to display</pre>
115	End Sub ' btnEight_Click
116	
117	<pre>Private Sub btnNine_Click(ByVal sender As System.Object, _</pre>
118	ByVal e As System.EventArgs) Handles btnNine.Click
119	
120	<pre>txtSecurityCode.Text &amp;= "9" ' concatenate "9" to display</pre>
121	End Sub ' btnNine_Click
122	
123	<pre>Private Sub btnClear_Click(ByVal sender As System.Object, _</pre>
124	ByVal e As System.EventArgs) Handles btnClear.Click
125	
126	<pre>txtSecurityCode.Clear() ' clear text from TextBox</pre>
127	End Sub ' btnClear_Click
128	
129	' invoked when the Form is loaded
130	Private Sub FrmSecurityPanel_Load(ByVal sender As _
131	System.Object, ByVal e As System.EventArgs) _
132	Handles MyBase.Load
133	
134	' load Agent character file
135	<pre>objMainAgent.Characters.Load("Robby", "Robby.acs")</pre>
136	
137	' specify current agent
138	m_objMSpeaker = objMainAgent.Characters("Robby")
139	
140	' show Robby on screen
141	m_objMSpeaker.Show(0)
142	
143	' Robby speaks
144	<pre>m_objMSpeaker.Speak("Please enter your security " _</pre>
145	& "access code.")
146	End Sub ' FrmSecurityPanel_Load
147	
148	End Class ' FrmSecurityPanel

What does this code do? 🕨	27.14 After the user clicks the Call Button, what does the following event handler do?
	<pre>Private Sub btnCall_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnCall.Click</pre>
	<pre>3 4 objMainAgent.Characters.Load("Genie", "Genie.acs")</pre>
	<pre>5 6 objMSpeaker = objMainAgent.Characters("Genie")</pre>
	<pre>7 8 objMSpeaker.Show(0)</pre>
	<pre>9 10 objMSpeaker.Speak("Hello, I'm Genie the special agent!")</pre>

12 End Sub

ТТ

**Answer:** The agent object is loaded as "Genie." Genie appears and says, "Hello, I'm Genie the special agent!"

What's wrong with this code?

**27.15** Find the error(s) in the following code. The event handler should have an agent object appear and say, "Hello, my name is Merlin". This should happen when the user clicks the **Call** Button.

```
1
    Private Sub btnCall_Click(ByVal sender As System.Object, _
2
       ByVal e As System. EventArgs) Handles btnCall.Click
 3
 4
       objMainAgent.Characters.Load("Merlin", "Merlin.acs")
 5
 6
       objMSpeaker = objMainAgent.Characters("Merlin")
 7
 8
       Dim intNumber As Integer = 10
9
10
       objMSpeaker.Show(intNumber)
11
12
       objMSpeaker.Play("Hello, my name is Merlin")
13
14
    End Sub
```

**Answer:** The Play method is called with an invalid argument. To have the agent say "Hello, my name is Merlin", you must call the Speak method.

```
1
    Private Sub btnCall_Click(ByVal sender As System.Object, _
 2
       ByVal e As System. EventArgs) Handles btnCall.Click
 3
 4
       objMainAgent.Characters.Load("Merlin", "Merlin.acs")
 5
 6
       objMSpeaker = objMainAgent.Characters("Merlin")
 7
 8
       Dim intNumber As Integer = 10
 9
10
       objMSpeaker.Show(intNumber)
11
12
       objMSpeaker.Speak("Hello, my name is Merlin")
13
14
    End Sub
```

Programming Challenge 27.16 Car Payment Application Using Microsoft Agent) Enhance the Car Payment Calculator application from Tutorial 9 to use the Microsoft Agent, Robby. When the application is executed, Robby should appear on the screen and wave to users. He should then explain the purpose of the application. After the user enters information into each field of the Car Payment Calculator and clicks the Calculate Button, Robby should speak the calculated payment amounts and the period (number of months) over which they were calculated. The C:\Examples\Tutorial27\Exercises\CarPaymentCalculatorEnhancement directory contains the template application for this exercise. Copy it to your working directory and open the application to begin the exercise.

- 1 ' Exercise 27.16 Solution
- 2 ' CarPayment.vb
- 3

Δ Public Class FrmCarPayment 5 Inherits System.Windows.Forms.Form 6 7 Private m\_objMSpeaker As AgentObjects.IAgentCtlCharacter 8 9 ' Windows Form Designer generated code 10 11 ' handles Calculate Button's Click event 12 Private Sub btnCalculate\_Click( ByVal sender As System.Object, \_ 13 ByVal e As System. EventArgs) Handles btnEnter. Click 14 15 Dim intYears As Integer = 2 ' repetition counter 16 ' payment period Dim intMonths As Integer = 0 ' car price 17 Dim intPrice As Integer = 0 ' down payment 18 Dim intDownPayment As Integer = 0 ' interest rate 19 Dim dblInterest As Double = 0 20 Dim decMonthlyPayment As Decimal = 0 ' monthly payment 21 Dim intLoanAmount As Integer = 0 ' cost after down payment 22 Dim dblMonthlyInterest As Double = 0 ' monthly interest rate 23 Dim strAgentSpeak As String = "You will have to pay: " 24 25 ' remove text displayed in ListBox 26 lstPayments.Items.Clear() 27 28 ' add header to ListBox 29 lstPayments.Items.Add("Months" & ControlChars.Tab & \_ 30 ControlChars.Tab & "Monthly Payments") 31 32 ' retrieve user input and assign values 33 ' to their respective variables 34 intDownPayment = Convert.ToInt32(Val(txtDownPayment.Text)) 35 intPrice = Convert.ToInt32(Val(txtStickerPrice.Text)) 36 dblInterest = Convert.ToDouble(Val(txtInterest.Text)) / 100 37 38 ' determine amount borrowed and monthly interest rate 39 intLoanAmount = intPrice - intDownPayment 40 dblMonthlyInterest = dblInterest / 12 41 ' loop four times 42 43 Do While intYears <= 5 44 45 ' calculate payment period 46 intMonths = 12 \* intYears 47 48 ' calculate monthly payment using Pmt 49 decMonthlyPayment = Convert.ToDecimal( \_ 50 Pmt(dblMonthlyInterest, intMonths, -intLoanAmount)) 51 52 ' display payment value 53 lstPayments.Items.Add(intMonths & ControlChars.Tab & \_ 54 ControlChars.Tab & String.Format("{0:C}", \_ 55 decMonthlyPayment)) 56 57 If intYears < 5 Then 58 59 ' specify format of payment values for first 60 ' four years 61 strAgentSpeak = strAgentSpeak & String.Format("{0:C}", \_ 62 decMonthlyPayment) & " per month, over " & intMonths \_ 63 & " months, 64 Else

```
65
                    specify format for payment value of fifth year
66
                  strAgentSpeak = strAgentSpeak & "or " & _
                    String.Format("{0:C}", decMonthlyPayment) _
& " per month, over " & intMonths _
67
68
69
                    & " months."
70
              End If
71
72
              intYears += 1 ' increment counter
73
           Loop
74
75
           ' Robby speaks strAgentSpeak string
76
           m_objMSpeaker.Speak(strAgentSpeak)
77
        End Sub ' btnCalculate_Click
78
79
        ' invoked when Form is loaded
80
        Private Sub FrmCarPayment_Load( ByVal sender As System.Object, _
81
           ByVal e As System. EventArgs) Handles MyBase. Load
82
83
            ' load Robby character into agent object
84
           objMainAgent.Characters.Load("Robby", "Robby.acs")
85
86
           m_objMSpeaker = objMainAgent.Characters("Robby")
87
88
           ' show Robby on the screen
89
           m_objMSpeaker.Show(0)
90
91
           ' play Wave animation
92
           m_objMSpeaker.Play("Wave")
93
94
           ' make Robby speak instructions for application
95
           m_objMSpeaker.Speak("Hello, I will be your assistant.")
96
97
           m_objMSpeaker.Play("RestPose")
98
99
           m_objMSpeaker.Speak( _
100
               "You need to enter the price of a car," _
              & " the down-payment amount and " _
101
              & " the annual interest rate of the loan.")
102
103
104
           m_objMSpeaker.Speak( _
105
              "After you input this information, " _
106
              & "I will calculate the monthly payments you will " _
107
              & "need to make for two-, three-, four- and " _
108
              & "five-year loans.")
109
        End Sub ' FrmCarPayment_Load
110
111 End Class ' FrmCarPayment
```





# Bookstore Application: Web Applications

Introducing Internet Information Services Solutions

	Instructor's Manual		
	Exercise Solutions		
	Tutorial 28		
-	MULTIPLE-CHOICE	<b>28.1</b> ASPX pages have the	extension
		a) .html	b) .wbform
	QUESTIONS	c) .vbaspx	d) .aspx
		· ·	
		<b>28.2</b> applications divide	
		a) <i>n</i> -tier	b) Multi-tier
		c) Both a and b.	d) None of the above.
		<b>28.3</b> All tiers of a multi-tier applica	ation
		a) must be located on the same	-
		b) must be located on different	-
			computer or on different computers
		d) must be arranged so that the information tier is on a diffe	e client and middle tier are on the same computer and erent computer
		<b>28.4</b> The client tier interacts with	h the tier to access information from
		a) middle; information	b) information; middle
		c) information; bottom	d) bottom; information
		<b>28.5</b> A is specialized resources.	software that responds to client requests by provid
		a) host	b) host name
		c) DNS server	d) Web server
		<b>28.6</b> A(n) can be though resource on the Web.	ght of as an address that is used to direct a browser
		a) middle tier	b) ASPX page
		c) URL	d) query string
		<b>28.7</b> A represents a gro	oup of on the Internet.
		a) domain; hosts	-
		c) host name; hosts	d) None of the above.
		, , , , ,	.,
		<b>28.8</b> is a Web server.	b) localhact
		<ul><li>a) IIS</li><li>c) Visual Studio .NET</li></ul>	b) localhost d) wwwroot
		,	,
		<b>28.9</b> A is a Web server to Internet.	that is located on a computer across a network such as
		a) localhost	b) local Web server
		c) remote Web server	d) None of the above.
		<b>28.10</b> The tier is the ap	plication's user interface.
		a) middle	b) client
		c) bottom	d) information
		<b>Answers:</b> 28.1) d. 28.2) c. 28.3) c. 3	28.4) a. 28.5) d. 28.6) c. 28.7) a. 28.8) a. 28.9) c. 28.10
			, , ,,, _,, _

EXERCISES

**28.11** (*Phone Book Application*) Over the next three tutorials, you will create a **Phone-Book** application. This phone book should be a Web-based version of the **PhoneBook** appli-

cation created in Tutorial 27. [Note: This Web application will not use Microsoft Agent.] The **PhoneBook** application should consist of two ASPX pages, which will be named PhoneBook and PhoneNumber. The PhoneBook page displays a DropDownList (a Web control similar to a ComboBox Windows Form control) that contains the names of several people. The names are retrieved from the db\_Phone.mdb database. When a name is selected and the **Get Number** Button is clicked, the client browser is redirected to the PhoneNumber page. The telephone number of the selected name should be retrieved from a database and displayed in a Label on the PhoneNumber page. For this exercise, you need only organize the components (Phone-Book and PhoneNumber ASPX pages, db\_Phone.mdb database and the code that performs the specified functionality) of this Web application into separate tiers. Decide which components belong in which tiers. You will begin building the solution, using Visual Studio .NET, in the next tutorial.

**Answer:** The client tier should contain the ASPX pages' GUIs. One page will contain a DropDownList control. The middle tier should contain the code used to retrieve the names and phone numbers from the database. The information tier should contain the db\_Phone.mdb database where the phone-number information is stored.

**28.12** (US State Facts Application) Over the next three tutorials, you will create a USStateFacts application. This application is designed to allow users to review their knowledge about specific U.S. states. This application should consist of two ASPX pages. The first page (named States) should display a ListBox containing 10 different state names. These state names are stored in the db\_StateFacts.mdb database. The user should be allowed to select a state name and click a Button to retrieve information about the selected state from the database. The information should be displayed on a different ASPX page (named StateFacts). The StateFacts page should display an image of the state flag and list the state capital, state flower, state tree and state bird (retrieved from the database) in a Table. You will be provided with images of the state flags. For this exercise, you need only organize the components (States and StateFacts ASPX pages, db\_StateFacts.mdb database and the code that performs the specified functionality) of this Web application into separate tiers. Decide which components belong in which tiers. You will begin building the solution, using Visual Studio .NET, in the next tutorial.

**Answer:** The client tier should contain the ASPX pages' GUIs, including a ListBox for the different state names. The middle tier should contain the code used to retrieve the state names and information from the database. The information tier should contain the db\_StateFacts.mdb database where the state information is stored.

**28.13** (*Road Sign Review Application*) Over the next three tutorials, you will create a **RoadSignReview** application. The **RoadSignReview** application should consist of two ASPX pages. This application displays road signs for users to review and allows them to schedule a driving test. The first page (named RoadSigns) should display 15 road signs in a Table. You will be provided images of the road signs. When the mouse pointer is moved over a sign, the name of the sign will appear in a tooltip in the Web browser window. The table should display the images by retrieving their information from the db\_RoadSigns.mdb database. This page also will contain two TextBoxes and a Button that allow users to provide their information to register for a driving test. When users click the **Register** Button, the second page (RoadTestRegistered) displays confirmation information that the user has registered for a driving test. For this exercise, you need only organize the components (RoadSigns and RoadTestRegistered ASPX pages, db\_RoadSigns.mdb database and the code that performs the specified functionality) of this Web application into separate tiers. Decide which components belong in which tiers. You will begin building the solution, using Visual Studio .NET, in the next tutorial.

**Answer:** The client tier should contain the ASPX pages' GUIs, one of which will contain a Table control for the various road signs. The middle tier should contain the code used to retrieve the road sign names from the database. The information tier should contain the db\_RoadSigns.mdb database where the road-signs information is stored.





### Bookstore Application: Client Tier

Introducing Web Controls Solutions

$\bigcirc$	Instructor's Manual Exercise Solutions Tutorial 29		
-	MULTIPLE-CHOICE	<b>29.1</b> You change the property plays in the background of the page.	of the ASPX page to specify the color that dis-
	QUESTIONS	a) BackColor	b) bgColor
		c) BackgroundColor	d) Color
		<b>29.2</b> Button, Label and Table controls tab.	s for ASPX pages can be accessed from the
		a) Web Forms	b) Components
		c) Data	d) Both a and b.
		<b>29.3</b> The attribute is used to spage.	pecify the position of a Web control on an ASPX
		a) position	b) location
		c) style	d) coordinate
		<b>29.4</b> Unlike the Windows Form Designer, t	he Web Form Designer
		a) does not provide two viewing mode	-
		b) provides two viewing modes	
		c) allows you to design the graphical u	iser interface
		d) does not allow you to design the us	er interface
$\bigcirc$		<ul><li>29.5 The BorderStyle property of the Ima</li><li>a) specifies the color of the border</li><li>b) specifies the type of border that dis</li><li>c) specifies the width of the border</li></ul>	
		, <b>1</b>	,
		<b>29.6</b> Setting the BorderStyle property to	
		a) raised	b) with a bold border d) with the appeified herder color
		c) with the specified border width	d) with the specified border color
		<b>29.7</b> Every of a Table Web con	
		a) TableRow; TableColumns	b) TableColumn; TableRows
		c) TableRow; TableCells	d) TableCell; TableRows
		<b>29.8</b> For you to be able to create an <b>ASP</b> . N running.	NET <b>Web</b> application project, must be
		a) IIS	b) Microsoft Access
		c) Microsoft Word	d) Internet Explorer
		<b>29.9</b> The mode allows you to dropping controls on the page.	create the ASPX page's GUI by dragging and
		a) HTML	b) Design
		c) Visual	d) GUI
		the attribute.	ntrol, set the and values of
_		a) X, Y, style	b) X,Y, position
$\bigcirc$		c) TOP, LEFT, style	d) TOP, LEFT, position
$\bigcirc$		<b>Answers:</b> 29.1) b. 29.2) a. 29.3) c. 29.4) b.	29.5) b. 29.6) a. 29.7) c. 29.8) a. 29.9) b. 29.10) c.

[Note: In these exercises, we may ask you to set an ASPX page as the application's start page, meaning that this page will appear first when the application is run. You can set an ASPX page as the start page by right clicking the file in the <b>Solution Explorer</b> and selecting <b>Set As Start Page</b> .]
<b>29.11</b> ( <i>Phone Book Application: GUI</i> ) Create the user interface for the <b>Phone Book</b> application. The design for the two pages for this application is displayed in Fig. 29.25.
a) <i>Creating an ASP.NET Web application</i> . Create an ASP.NET Web application project, and name it PhoneBook. Rename the ASPX page to PhoneBook.aspx, and set Option Strict to On. Set PhoneBook.aspx as the start page.
b) Changing the background color. Change the background color of your ASPX page (PhoneBook.aspx) to the light-yellow Web Palette color (located in the sixth col- umn of the 12th row) by using the bgColor property as demonstrated in this tutorial. Change the title of the ASPX page to Phone Book.
c) Adding a Label. Create a Label, set the font size to X-Large and change the Text property to Phone Book Web Application. Set the LEFT: portion of the style attribute value to 40px and the TOP: portion to 17px. Name the control lblPhone-Book.
d) Adding another Label. Create another Label, and set the Text property to Select a name from the list and click the Get Number Button:. Set the LEFT: portion of the style attribute value to 30px and the TOP: portion to 65px. Name this Web con- trol lblInstructions.
e) Adding a DropDownList Web control. Create a DropDownList Web control by drag- ging and dropping it from the Toolbox onto the ASPX page. The DropDownList Web control looks similar to the ComboBox Windows Form control. Set the width to 190px, and set the LEFT: portion of the style attribute value to 134px and the TOP: portion to 108px. Name the DropDownList cboNames.
f) Adding a Button. Create a Button, set its width to 90px and change the Text property to Get Number. Set the LEFT: portion of the style attribute value to 175px and the TOP: portion to 200px. Name the Web control btnGet.
g) Adding another ASPX page to the Phone Book application. Add another ASPX page to the Phone Book application, name it PhoneNumber.aspx and change the background to the light-yellow color. Change the title property to Phone Number.
<ul> <li>h) Adding a Label to the PhoneNumber.aspx. Create a Label and name it lblPhone-Number. Set the font size to X-Large and change the Text property to Phone Number:. Set the LEFT: portion of the style attribute value to 20px and the TOP: portion to 15px.</li> </ul>
<ul> <li>Adding another Labe1. Create another Labe1, set its BorderStyle to Inset, and set its height and width to 50px and 380px, respectively. Clear the text of the Labe1. Name the Labe1 lb1Numbers, and set the LEFT: portion of the style attribute value to 25px and the TOP: portion to 80px.</li> </ul>
j) Adding a Button to the PhoneNumber.aspx page. Create a Button, set its width to 115px and change the Text property to Phone Book. Set the LEFT: portion of the style attribute value to 135px and the TOP: portion to 150px. Name the Button btnPhoneBook.
k) <i>Saving the solution file</i> . Save the solution file to the PhoneBook folder located in the root directory of your Web server, as you did in <i>Step 8</i> of the box, <i>Creating an</i>

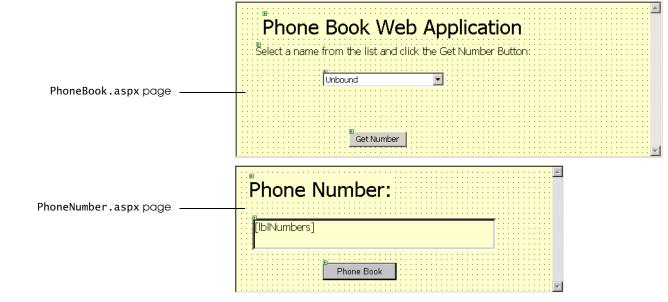


Figure 29.25 Phone Book application ASPX pages' design.

**Answer:** For this exercise, readers are asked to create the visual design of the page. They create the design by dragging and dropping controls on the page. There is no code paste-up for this exercise.

**29.12** (US State Facts Application: GUI) Create the user interface for the US State Facts application. The design for the two pages of this application is displayed in Fig. 29.26.

- a) *Creating an ASP.NET Web application*. Create a new ASP.NET Web application project, and name it USStateFacts. Rename the first ASPX page to States.aspx, and set Option Strict to On. Set States.aspx as the start page.
- b) Changing the background color. Change the background color of the States.aspx page to the light-blue Web Palette color (located in the sixth column of the second row) by using the bgColor property as demonstrated in this tutorial. Change the title property of the ASPX page to States.
- c) Adding a Label to States.aspx. Create a Label Web control, and place it on the page. Set the font size to XX-Large, and change the Text property to States. Change the LEFT: portion of its style attribute value to 390px, and set the TOP: portion to 15px. Name the Web control lblStates.
- d) Adding a Horizontal Rule to States.aspx. Create a Horizontal Rule, place it on the ASPX page and set its width to 150%. When setting its position, change the TOP: value to 80px, set the LEFT: value to 0px and specify the Height: as 4px. Name the Horizontal Rule hrzStates.
- e) Adding another Label to States.aspx. Create another Label, and place it beneath the Horizontal Rule. Change the font size to Medium, and set the Text property to Select a state from the list and click the button to view facts about that state:. Set its height to 16px and its width to 620px. Change the LEFT: portion of its style attribute value to 195px, and set the TOP: portion to 100px. Name this Web control lblInstructions.
- f) Adding a ListBox to States.aspx. Create a ListBox, and place it on the ASPX page. Set its Height property to 100px and its Width property to 155px. Set the LEFT: portion of the style attribute value to 365px and the TOP: portion to 150px. Name the ListBox lstStates.
- g) Adding a Button to States.aspx. Create a Button, and place it on the page. Set its Text property to Review Facts and its Width property to 130px. Change the LEFT: portion of the style attribute value to 375px and the TOP: portion to 270px. Name the Button btnFacts.

- h) Adding another ASPX page to the US State Facts application. Add another ASPX page to the US State Facts application, name it StateFacts.aspx and change the background color to light blue.
- i) Adding a Label to StateFacts.aspx. Create a Label, name it lblStateName, set its font size to XX-Large and change its ForeColor property to Blue. Clear the Label's text. Set its position by setting the LEFT: portion of the style attribute value to 20px and the TOP: portion to 15px.
- j) Adding a Horizontal Rule. Place the Horizontal Rule beneath the Label and set its TOP: position to 90px, its LEFT: position to 0px and its Height: to 4px. Change the width to 150%. Name the Horizontal Rule hrzStateFacts.
- k) Adding an Image control to StateFacts.aspx. Create an Image control and set its BorderStyle to Outset. Change the BorderWidth to 5px. Set its height to 200px and its width to 300px. Set the position of the Image by changing the LEFT: portion of the style attribute value to 20px and the TOP: portion to 110px. Name the Web control imgFlag.
- Adding a Table to StateFacts.aspx. Create a Table with four rows and two columns. Set the BorderStyle to Outset, the BorderWidth to 5px and GridLines to Both. Set the height and width of each TableCell of the first column to 70px and 200px, respectively, and set the Font property's Size to Large. Set the Text property of the cells in the first column to Capital:, Flower:, Tree: and Bird:, respectively. Change the LEFT: portion of the style attribute value to 335px and the TOP: portion to 110px. Name the Table control tblState.
- m)*Adding a Button to StateFacts.aspx*. Create a Button, change its text to State List, and change the LEFT: portion of the style attribute value to 285px and the TOP: portion to 425px. Name the Button control btnStateList.
- n) *Saving the solution file*. Save the solution file to the USStateFacts folder located in the root directory of your Web server, as you did in *Step 8* of the box, *Creating an ASP.NET Web Application*.

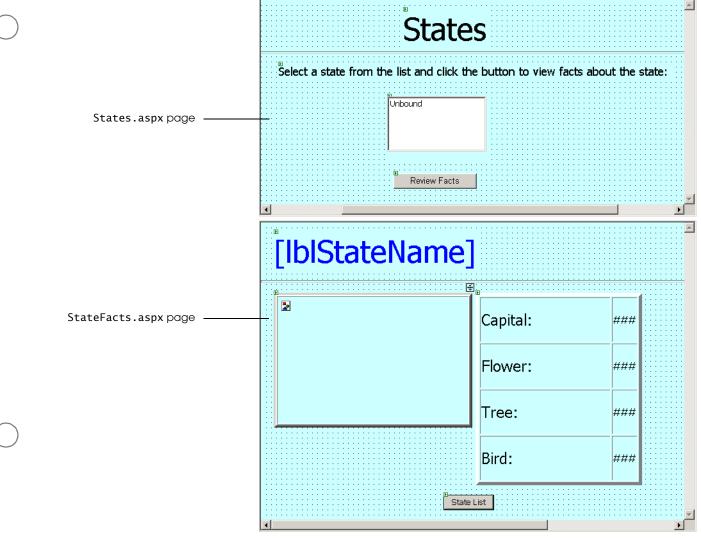


Figure 29.26 US State Facts application ASPX pages' design.

**Answer:** For this exercise, readers are asked to create the visual design of the page. They create the design by dragging and dropping controls on the page. There is no code paste-up for this exercise.

**29.13** (*Road Sign Review Application: GUI*) Create the user interface for the Road Sign Review application. The design for the two pages of this application is displayed in Fig. 29.27.

- a) *Creating an ASP.NET Web application.* Create a new ASP.NET Web application project, and name it **RoadSignReview**. Change the name of the existing ASPX page to RoadSigns.aspx, and set Option Strict to On. Set RoadSigns.aspx as the start page.
- b) Changing the background color. Change the background color of RoadSigns.aspx to the light-green Web Palette color (located in the sixth column of the 14th row) by using the bgColor property as demonstrated in this tutorial. Change the title of the ASPX page to RoadSigns.
- c) Adding a Label to RoadSigns.aspx. Create a Label, and set its font size to XX-Large. Change the Text property to Road Signs. Set its position by changing the style attribute value's LEFT: portion to 295px and the TOP: portion to 16px. Name the Label control lblRoadSigns.
- d) *Adding a Horizontal Rule to RoadSigns.aspx*. Create a Horizontal Rule. Set its width to 150%, and set the TOP: position to 80px, the LEFT: position to 0px and the height to 4px. Name the Horizontal Rule hrzRoadSigns.

- e) Adding a Table to RoadSigns.aspx. Create a Table with three rows and five columns. Set the BorderStyle to Outset, the BorderWidth to 5px and the GridLines property to Both. Also, set the Table's Height property to 279px and Width property to 626px. Set each row's height to 50px and each TableCell's width to 20px. Change the style attribute value by setting LEFT: to 70px and TOP: to 150px. Name the Table control tblRoadSigns.
- f) Adding a Label to RoadSigns.aspx. Create a Label, and set its font size to Large. Change the Text property to Register for Your Driving Test. Set its position by changing the style attribute value's LEFT: portion to 70px and TOP: portion to 470px. Name the Web control lblRegister.
- g) Adding a Label and TextBox to RoadSigns.aspx. Create a Label and set its text to Name:. Set its font size to Medium, and change its position to LEFT: 70px and TOP: 520px. Name the Label control lblName. Create a TextBox, and place it next to the Name: Label. Set its height to 20px and width to 115px. Change the position to LEFT: 135px and TOP: 520px. Name the TextBox control txtName.
- h) Adding another Label and TextBox pair to RoadSigns.aspx. Create a Label and set its text to Phone Number:. Set its font size to Medium, and change its position to LEFT: 275px and TOP: 520px. Name the Label control lblPhoneNumber. Create a TextBox, and place it next to the Phone Number: Label. Set its height to 20px and width to 115px. Change its position to LEFT: 410px and TOP: 520px. Name the TextBox control txtPhoneNumber.
- i) Adding a Button to RoadSigns.aspx. Create a Button, set its Text to Register, and change its height and width to 30px and 120px, respectively. Change the position of the Button by setting the LEFT: portion of the style attribute value to 555px and the TOP: portion to 520px. Name the Button control btnRegister.
- j) Adding another ASPX page to the Road Sign Review application. Add another ASPX page to the application, name it RoadTestRegistered.aspx and change the background color to light green.
- k) Adding a Label to RoadTestRegistered.aspx. Create a Label, setting its font size to XX-Large and its Text property to Registration Complete. Change its position by setting the LEFT: portion of its style attribute value to 200px and the TOP: portion to 15px. Name the Label control lblRegistration.
- 1) Adding a Horizontal Rule to RoadTestRegistered.aspx. Create a Horizontal Rule. Set its width to 150%, the TOP: position to 80px, the LEFT: position to 0px and the height to 4px. Name the Horizontal Rule hrzRoadTestRegistered.
- m)Adding another Label to RoadTestRegistered.aspx. Create a Label, name it lblConfirmation and set its font size to Medium. Delete the Text property value, and leave it blank. Change its position by setting the LEFT: portion of its style attribute value to 125px and the TOP: portion to 130px.
- n) *Saving the solution file*. Save the solution file to the RoadSignReview folder located in the root directory of your Web server, as you did in *Step 8* of the box, *Creating an ASP.NET Web Application*.

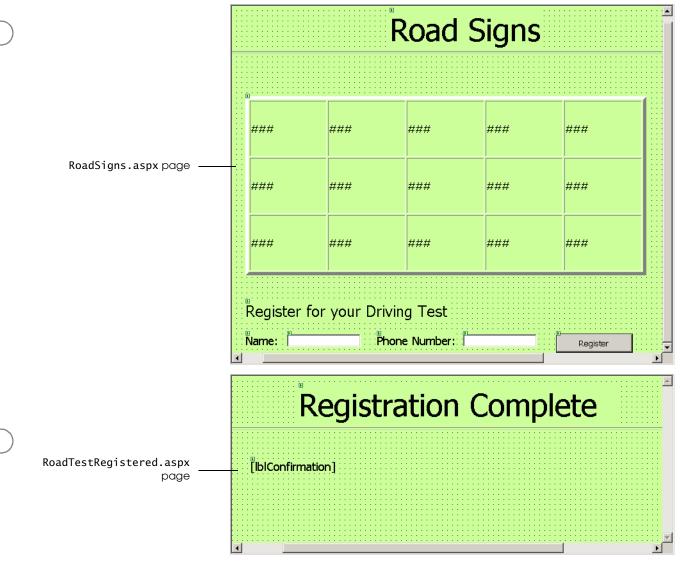


Figure 29.27 Road Signs application ASPX pages' design.

**Answer:** For this exercise, readers are asked to create the visual design of the page. They create the design by dragging and dropping controls on the page. There is no code paste-up for this exercise.





## Bookstore Application: Information Tier

Examining the Database and Creating Database Components Solutions

$\bigcirc$	Instructor's Manual Exercise Solutions Tutorial 30		
-	MULTIPLE-CHOICE	<b>30.1</b> is an example of a da	tabase product.
	QUESTIONS	<ul><li>a) Microsoft Access</li><li>c) Oracle</li></ul>	<ul><li>b) Microsoft SQL Server</li><li>d) All of the above.</li></ul>
		<ul><li>30.2 An advantage of using information</li><li>a) the data can be updated in read</li><li>b) information that changes need</li><li>c) Both a and b.</li><li>d) None of the above.</li></ul>	l time
		indicates that a) information will be updated in	that field according to specified criteria
		<b>30.4</b> The Parameters property of a) OleDbConnection	contains a collection of parameters. b) 01eDbDataConnection
		c) OleDbDataCommand	d) 01eDbCommand
$\bigcirc$		<b>30.5</b> The can be used to cr	eate an OleDbConnection.
0		a) Server Explorer window	b) Query Builder tool
		c) Both a and b.	d) None of the above.
		<b>30.6</b> You use the object database.	to create SQL statements for retrieving data from a
		a) OleDbConnection	b) OleDbDataReader
		c) 01eDbCommand	d) None of the above.
		<b>30.7</b> The is used when croobject's CommandText property.	eating SQL statements visually for the OleDbCommand
		a) Server Explorer window	b) Query Builder tool
		c) Both a and b.	d) None of the above.
		<b>30.8</b> You use the object to	open a connection to the database.
		a) OleDbConnection	b) OleDbDataReader
		c) OleDbCommand	d) None of the above.
		<b>30.9</b> You use the proper information that is not known in advan	ty of the OleDbCommand object to specify values for nce.
		<ul><li>a) Connection</li><li>c) Field</li></ul>	b) Parameters d) Name
		<b>30.10</b> Another name for the database	,
		a) the information tier	b) the bottom tier
$\frown$		c) Both a and b.	d) None of the above.
$\bigcirc$		<b>Answers:</b> 30.1) d. 30.2) c. 30.3) b. 30.4	4) d. 30.5) a. 30.6) c. 30.7) b. 30.8) a. 30.9) b. 30.10) c.

**EXERCISES** 30.11 (Phone Book Application: Database) Create the database connections and data command objects for the PhoneBook application by using the Server Explorer window and the Query Builder tool. a) **Opening the application**. Open the **PhoneBook** application that you created in Tutorial 29. b) Copying the db\_Phone.mdb database to the Databases folder. Copy the C:\Examples\Tutorial30\Exercises\Databases\db Phone.mdb database to the Databases folder in IIS's wwwroot folder. c) Using Server Explorer to add a connection to the database. In the Server Explorer window, add a connection to the db\_Phone.mdb database. Drag and drop the connection object onto the PhoneBook.aspx page. Name the connection object obi0leDbConnection. d) Using Query Builder for the PhoneBook.aspx page. Add an OleDbCommand to the PhoneBook.aspx page. Set the Connection property to the OleDbConnection object you added to the ASPX page, and use Query Builder to set the CommandText property of the OleDbCommand. This command should retrieve all the names of the people from the database. Name this command object objSelectNames. e) Adding a connection to the database to the PhoneNumber.aspx page. Using the Server Explorer window, drag and drop a database connection object onto the PhoneNumber.aspx page. Name this connection object obj01eDbConnection. f) Using Query Builder for PhoneBook.aspx. Add an OleDbCommand to the PhoneNumber.aspx page. Set the Connection property to the OleDbConnection object you added to the ASPX page, and use Query Builder to set the CommandText property of the 01eDbCommand. This configuration should retrieve the phone number of the person whose name will be selected, from the DropDownList in the PhoneBook.aspx page, by the user. You need to set the criteria to specify which person's phone number will be retrieved from the database. Name this command object objSelectPhoneNumber. g) Saving the solution. Select File > Save All to save the solution's files. Answer: Connect to the database by using the Server Explorer window, and specify data command objects by using the Query Builder tool. 30.12 (US State Facts Application: Database) Create the database connections and data command objects for the USStateFacts application by using the Server Explorer window and the Query Builder tool. a) **Opening the application**. Open the **USStateFacts** application that you created in Tutorial 29. b) Copying the db\_StateFacts.mdb database to the Databases folder. Copy the C:\Examples\Tutorial30\Exercises\Databases\db\_StateFacts.mdb database to the Databases folder in IIS's wwwroot folder. c) Using Server Explorer to add a connection to the database. In the Server Explorer window, add a connection to the db\_StateFacts.mdb database. Drag-and-drop the connection object onto the States.aspx page. Name this connection object obj01eDbConnection. d) Using Query Builder for the States.aspx page. Add an OleDbCommand to the States.aspx page. Set the Connection property to the OleDbConnection object you added to the ASPX page and use Query Builder to set the CommandText property of the 01eDbCommand. This command should retrieve the names of the states from the name field of the states table in the database. Name this command object objSelectNames. e) Adding a connection to the database to the StateFacts.aspx page. Using the Server Explorer window, drag-and-drop a database connection object onto the database on the StateFacts.aspx page. Name this connection object obj01eDbConnection. f) Using Query Builder for StateFacts.aspx. Add an OleDbCommand to the StateFacts.aspx page. Set the Connection property to the OleDbConnection object you added to the ASPX page, and use Query Builder to set the CommandText

property of the OleDbCommand. This configuration should retrieve all the information, from the states table of the database, about the state that is selected by the user. You need to set the criteria to specify which state's information will be retrieved from the database. Name this command object objSelectStateInformation.

g) Saving the solution. Select File > Save All to save the solution's files.

**Answer:** Connect to the database by using the **Server Explorer** window, and specify data command objects by using the **Query Builder** tool.

**30.13** (*Road Sign Review Application: Database*) Create the database connections and data command objects for the **RoadSignReview** application by using the **Server Explorer** window and the **Query Builder** tool.

- a) *Opening the application*. Open the **RoadSignReview** application that you created in Tutorial 29.
- b) *Copying the db\_RoadSigns.mdb database to the Databases folder*. Copy the C:\Examples\Tutorial30\Exercises\Databases\db\_RoadSigns.mdb database to the Databases folder in IIS's wwwroot folder.
- c) Using Server Explorer to add a connection to the database. In the Server Explorer window add a connection to the db\_RoadSigns.mdb database. Drag and drop the connection object onto the RoadSigns.aspx page. Name this command object obj0leDbConnection.
- d) Using Query Builder for the RoadSigns.aspx page. Add an OleDbCommand to the RoadSigns.aspx page. Set the Connection property to the OleDbConnection object that you added to the ASPX page, and use Query Builder to set the CommandText property of the OleDbCommand. This configuration should retrieve all the information about all the road signs from the signs table of the database. You will not need to specify a criterion for this exercise, because all the information from the database needs to be retrieved. Name this command object objSelectSignInformation.
- e) Saving the solution. Select File > Save All to save the solution's files.

**Answer:** Connect to the database by using the **Server Explorer** window, and specify data command objects by using the **Query Builder** tool.





## Bookstore Application: Middle Tier

Introducing Code-Behind Files Solutions

$\bigcirc$	Instructor's Manual Exercise Solutions Tutorial 31		
-	MULTIPLE-CHOICE QUESTIONS	<ul> <li>31.1 The Page_Load event handler</li> <li>a) redirects the client browser to dif</li> <li>b) defines the functionality when a B</li> <li>c) executes any processing necessary</li> <li>d) defines the functionality when a V</li> </ul>	ferent Web pages Button is clicked y to display a Web page
		<ul> <li>31.2 The Response.Redirect method _</li> <li>a) refreshes the current Web page</li> <li>b) sends the client browser to a species</li> <li>c) responds to user input</li> <li>d) responds to the click of a Button</li> </ul>	
		<ul> <li>31.3 Session items are used in the Bool</li> <li>a) variables in ASP.NET Web appli</li> <li>b) values need to be shared among v</li> <li>c) Session items are simpler to created</li> <li>d) Both a and b.</li> </ul>	cations must be created as Session items Web pages
$\bigcirc$		<ul> <li>31.4 Session state is used fori</li> <li>a) tracking user-specific data</li> <li>c) using a database</li> </ul>	n ASP.NET. b) running an application d) None of the above.
$\bigcirc$		<ul><li>31.5 The file extension for an ASPX cod</li><li>a) .asp</li><li>c) .aspx.vb</li></ul>	le-behind file is b) .aspx d) .code
		<ul><li>31.6 The Response object is a predefine</li><li>a) connects to a database</li><li>c) creates Web controls</li><li>d) provides methods for responding</li></ul>	b) retrieves information from a database
		<ul><li>31.7 The Response.Redirect method to</li><li>a) URL</li><li>c) Boolean value</li></ul>	
		<ul> <li>31.8 The property specifies the a) ImageGIF</li> <li>c) Image</li> </ul>	e image that an Image control displays. b) ImageURL d) Display
		<ul> <li>31.9 The Visual Basic .NET file that con the</li> <li>a) ASPX file</li> <li>c) class file</li> </ul>	<ul><li>tains the ASPX page's corresponding class is called</li><li>b) code-behind file</li><li>d) None of the above.</li></ul>
			cross Web pages by adding a to the Ses-
$\bigcirc$		<ul><li>a) key-value pair</li><li>c) database connection object</li></ul>	<ul> <li>b) number</li> <li>d) None of the above.</li> <li>a. 31.5) c. 31.6) d. 31.7) a. 31.8) b. 31.9) b. 31.10) a.</li> </ul>

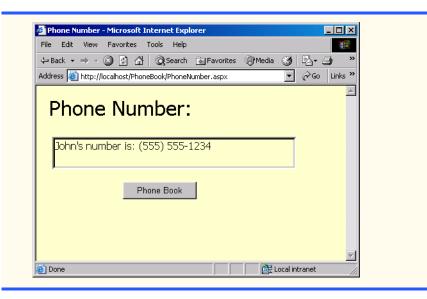
EXERCISES	<b>31.11</b> ( <i>Phone Book Application: Functionality</i> ) Define the middle tier for the <b>Phone Book</b> application.
	a) <b>Opening the application</b> . Open the <b>PhoneBook</b> application that you created in Tutorial 29 and continued to develop in Tutorial 30.
	b) <i>Importing System.Data.OleDb in PhoneBook.aspx.vb.</i> Import the System.Data.OleDb namespace in PhoneBook.aspx.vb.
	c) <b>Defining the Page_Load event handler of PhoneBook.aspx page</b> . Use the Open method to open the connection to the database. Create a data reader to read the information specified by the data command object.
	d) Populating the DropDownList with names. Add a Do WhileLoop to Phone-Book.aspx's Page_Load method. This loop should add to the DropDownList each person's name read by the data reader.
	e) <i>Closing the reader and connection.</i> Close the data reader and the connection to the database by invoking their Close methods.
	f) Creating the Get Number Button's Click event handler for the PhoneBook.aspx page. Double click the Get Number Button to create the Click event's event handler.
	g) <i>Creating a Session item.</i> In the Click event handler, create a Session item to store the selected name.
	h) <i>Redirecting to the PhoneNumber.aspx page.</i> In the Click event handler, use the Response.Redirect method to redirect the client browser to the PhoneNumber.aspx page.
	i) <i>Importing System.Data.OleDb in PhoneNumber.aspx.vb.</i> Import the System.Data.OleDb namespace in PhoneNumber.aspx.vb.
	j) Defining the Page_Load event handler for the PhoneNumber.aspx page. Use the Open method to open the connection to the database. Access the Session item to retrieve the selected name. Specify this name as the parameter value for the OleDb- Command object. Create a data reader to read the information specified by the data command object.
	k) Displaying the selected name and phone number. In the Page_Load event handler, read the desired phone number from the data reader. Display the selected name and corresponding phone number in the lblNumbers Label.
	<ol> <li>Closing the reader and connection. Close the data reader and the connection to the database by invoking their Close methods.</li> </ol>
	m) Creating the Phone Book Button's Click event handler for the PhoneNumber.aspx page. Double click the Phone Book Button to create the Click event's event handler.
	n) <i>Redirecting to the PhoneBook.aspx page.</i> In the Click event handler, use the Response.Redirect method to redirect the client browser to the PhoneBook.aspx page.
	l 'Solution 31.11 Solution
	2 ' PhoneBook.aspx.vb 3
	<pre>3 4 Imports System.Data.OleDb</pre>
	5 6 Public Class PhoneBook
	7 Inherits System.Web.UI.Page
	<pre>8 9 ' control declarations</pre>
	10
	<pre>11 'Web Form Designer Generated Code 12</pre>
	13 ' invoked when page is loaded
	14Private Sub Page_Load(ByVal sender As System.Object, _15ByVal e As System.EventArgs) Handles MyBase.Load

```
16
17
          obj0leDbConnection.0pen() ' open connection to the database
18
19
           ' declare reader to read from database
20
          Dim objReader As OleDbDataReader
21
22
           ' create reader to read from database
23
          objReader = objSelectNames.ExecuteReader()
24
25
           ' add names to DropDownList
26
          Do While objReader.Read()
27
28
              ' add item to DropDownList
29
             cboNames.Items.Add(Convert.ToString(objReader("Name")))
30
          Loop
31
32
          objReader.Close() ' close the reader
33
34
           ' close the connection to the database
35
          obj0leDbConnection.Close()
36
       End Sub ' Page_Load
37
38
        ' invoke when btnGet Button is clicked
       Private Sub btnGet_Click(ByVal sender As System.Object, _
39
40
          ByVal e As System.EventArgs) Handles btnGet.Click
41
42
          Session("Name") = Convert.ToString(cboNames.SelectedItem)
43
           ' redirect to another ASPX page
44
45
          Response.Redirect("PhoneNumber.aspx")
46
        End Sub ' btnGet_Click
47
48
    End Class ' PhoneBook
```

_			1icrosoft I		-	21				_ 🗆
File	Edit	View	Favorites	; Tools	Help					
d <b>e</b> B	ack 🔻	$\Rightarrow$ $\cdot$	🙆 🛃 1	3   Q	Search	🚡 Favorites	🛛 🛞 Media	3	₽s• ;	3
Addre	ess હ	http://	'localhost/Pl	noneBook	/PhoneE	look.aspx		-	∂ Go	Links
				-						_
	Ы			1		/_I_ A				
	PI	nor	ie B	OOk	(N)	/eb A	pplic	at	ion	
							•••			
	Selec	t a na	ame fro	m the	list ar	nd click the	e Get Nur	nber	butto	n:
							_			
			Joh	in			•			
			Joh	n						
				nifer		νž				
			Hov	ward						
				_		_				
				Ge	t Numb	per				
(11) D.	one								itranet	

1 ' Exercise 31.11 Solution
2 ' PhoneNumber.aspx.vb
3
4 Imports System.Data.OleDb
5
6 Public Class PhoneNumber

7	Inherits System.Web.UI.Page
8 9	' control declarations
10	
11	'Web Form Designer Generated Code
12	
13	' invoked when page is loaded
14	<pre>Private Sub Page_Load(ByVal sender As System.Object, _</pre>
15	ByVal e As System.EventArgs) Handles MyBase.Load
16	
17	' represents the name
18 19	<pre>Dim strName As String = Convert.ToString(Session("Name"))</pre>
20	<pre>obj0leDbConnection.Open() ' open connection to the database</pre>
21	objorebbconnectron.open() open connectron to the database
22	' specify name to retrieve phone number for
23	<pre>objSelectPhoneNumber.Parameters( _</pre>
24	<pre>"Name").Value = strName</pre>
25	
26	' declare reader to read from the database
27	Dim objReader As OleDbDataReader
28	
29 30	' create reader to read from the database
31	objReader = objSelectPhoneNumber.ExecuteReader()
32	objReader.Read() ' start data reader
33	objicadel includy state data reader
34	' display name and number in Label
35	<pre>lblNumbers.Text = strName &amp; "'s number is: " &amp; _</pre>
36	Convert.ToString(objReader("Phone_Numbers"))
37	
38	objReader.Close() ' close data reader
39	
40 41	' close the connection to the database
41	<pre>obj0leDbConnection.Close() End Sub ' Page_Load</pre>
43	Lifu Sub Tage_Load
44	' redirects user back to PhoneBook.aspx
45	<pre>Private Sub btnPhoneBook_Click(ByVal sender As System.Object, _</pre>
46	ByVal e As System.EventArgs) Handles btnPhoneBook.Click
47	
48	Response.Redirect("PhoneBook.aspx")
49	End Sub ' btnPhoneBook_Click
50 51	End Class L DheneNumber
51	End Class ' PhoneNumber



**31.12** (US State Facts Application: Functionality) Define the middle tier for the US State Facts application.

- a) *Opening the application*. Open the **USStateFacts** application that you created in Tutorial 29 and continued to develop in Tutorial 30.
- b) *Copying the FlagImages folder to your project folder.* Copy the C:\Examples\Tutorial31\Exercises\Images\FlagImages folder to the USStateFacts folder.
- c) *Importing* System.Data.OleDb in States.aspx.vb. Import the System.Data.OleDb namespace in States.aspx.vb before the class definition.
- d) *Defining the Page\_Load event handler for the States.aspx page.* Use the Open method to open the connection to the database. Create a data reader to read the information specified by the data command object.
- e) **Populating the ListBox with state names in the States.aspx page**. Add a Do While...Loop to States.aspx's Page\_Load method. This loop should add to the ListBox the name of each state read by the data reader.
- f) *Creating a Button's Click event handler for the States.aspx page.* Double click the **Review Facts** Button to create the Click event's event handler.
- g) *Creating a Session item*. Create a Session item in the Click event handler and assign it to the state name that the user selects from the ListBox.
- h) Redirecting to the StateFacts.aspx page. In the Click event handler, use the Redirect.Response method to redirect the client browser to the StateFacts.aspx page.
- i) *Importing System.Data.OleDb in StateFacts.aspx.vb.* Import the System.Data.OleDb namespace in StateFacts.aspx.vb.
- j) Defining the Page\_Load event handler of StateFacts.aspx page. Use the Open method to open the connection to the database. Access the Session object to retrieve the selected state name. Specify this name as a parameter value for the Ole-DbCommand object. Create a data reader to read the information specified by the data command object.
- k) Displaying the state facts in the Table. In the Page\_Load event handler, use the data reader to retrieve the desired state's facts. Display the selected state's name in the lblStateName Label. Set the ImageURL property of the Image control to the location of the selected state's flag image. Display the name of the state capital, flower, tree and bird in the Table on the StateFacts.aspx page.
- Closing the connection. Close the connection to the database by invoking the Close method.

- m) Creating the State List Button's Click event handler for the StateFacts.aspx page. Double click the State List Button to create the Click event handler.
- n) *Redirecting to the States.aspx page*. In the Click event handler use the Redirect.Response method to redirect the client browser to the States.aspx page.

```
1
     ' Exercise 31.12 Solution
 2
    ' States.aspx.vb
 3
 4
    Imports System.Data.OleDb
 5
 6
    Public Class States
 7
       Inherits System.Web.UI.Page
 8
 9
        ' control declarations
10
11
        ' Web Form Designer Generated Code
12
13
        ' invoked when page is loaded
14
        Private Sub Page_Load(ByVal sender As System.Object, _
15
           ByVal e As System. EventArgs) Handles MyBase. Load
16
17
           obj0leDbConnection.Open() ' open connection to the database
18
19
           ' declare reader to read from database
20
          Dim objReader As OleDbDataReader
21
22
           ' create reader to read from database
23
           objReader = objSelectNames.ExecuteReader()
24
25
           ' add names to the ListBox
26
          Do While objReader.Read()
27
28
              ' add item to ListBox
29
              lstStates.Items.Add(Convert.ToString(objReader("Name")))
30
           Loop
31
32
           objReader.Close() ' close the reader
33
34
           ' close the connection to the database
35
           obj0leDbConnection.Close()
36
        End Sub ' Page_Load
37
38
        ' handle click event
        Private Sub btnFacts_Click(ByVal sender As System.Object, _
39
40
           ByVal e As System. EventArgs) Handles btnFacts. Click
41
42
           ' create Session item named strName
43
           Session("strName") = lstStates.SelectedItem.Text
44
45
           ' redirect to another ASPX page
46
           Response.Redirect("StateFacts.aspx")
47
       End Sub ' btnFacts_Click
48
49
    End Class ' States
```



States - Microsoft Internet Explorer	
File Edit View Favorites Tools Help	
🔟 🕂 Back 🔻 🤿 💉 🙆 🚰 🧔 Search 📾 Favorites 🛞 Media 🎯 🗟 🖌 🎒 🛃 🚊	
Address 🧃 http://localhost/USStateFacts/States.aspx	💌 🧬 Go 🛛 Links
Chattan	
States	
014100	
Select a state from the list and click the button to view fact	about the state:
	s about une state.
Massachusetts	
Massachusetts 🔺 Texas	
Massachusetts	
Massachusetts Texas Kansas	
<mark>Massachusetts</mark> Texas Kansas Florida	
Massachusetts Texas Kansas Florida Arizona ▼	
<mark>Massachusetts</mark> Texas Kansas Florida	
Massachusetts Texas Kansas Florida Arizona ▼	
Massachusetts Texas Kansas Fiorida Arizona ▼	

1	' Solution 31.12 Solution
2 3	' StateFacts.aspx.vb
3	
4	Imports System.Data.OleDb
5	
6	Public Class StateFacts
7	Inherits System.Web.UI.Page
8	
9	' control declarations
10	
11	'Web Form Designer Generated Code
12	
13	' invoked when page is loaded
14	<pre>Private Sub Page_Load(ByVal sender As System.Object, _</pre>
15	ByVal e As System.EventArgs) Handles MyBase.Load
16	
17	' display name of selected book
18	lblStateName.Text = Convert.ToString(Session("strName"))
19	
20	obj0leDbConnection.Open() ' open connection to database
21	
22	' specify state name to retrieve information about
23	<pre>objSelectStateInformation.Parameters("Name").Value = _</pre>
24	Convert.ToString(Session("strName"))
25	
26	' declare database reader to read from database
27	Dim objReader As OleDbDataReader
28	
29	' create database reader to read from database
30	<pre>objReader = objSelectStateInformation.ExecuteReader()</pre>
31	
32	' while reader is reading database, retrieve data from
33	' specified positions and display them on page
34	Do While objReader.Read()
35	Latin Teo Chen Consellation and
36	' display flag for selected state
37	<pre>imgFlag.ImageUrl = "FlagImages/" &amp; _</pre>

38	Convert.ToString(objReader("flag"))
39	
40	' display information from database in Table
41	<pre>tblState.Rows(0).Cells(1).Text = _</pre>
42	Convert.ToString(objReader("capital"))
43	<pre>tblState.Rows(1).Cells(1).Text = _</pre>
44	Convert.ToString(objReader("flower"))
45	<pre>tblState.Rows(2).Cells(1).Text = _</pre>
46	Convert.ToString(objReader("tree"))
47	<pre>tblState.Rows(3).Cells(1).Text = _</pre>
<b>48</b>	Convert.ToString(objReader("bird"))
49	Loop
50	
51	<pre>obj0leDbConnection.Close() ' close connection to database</pre>
52	End Sub ' Page_Load
53	
54	' invoked when user clicks Button
55	<pre>Private Sub btnStateList_Click(ByVal sender As System.Object, _</pre>
56	ByVal e As System.EventArgs) Handles btnStateList.Click
57	
<b>58</b>	' redirects to States.aspx page
<b>59</b>	Response.Redirect("States.aspx")
60	End Sub ' btnStateList_Click
61	
62	End Class ' StateFacts

<sup>2</sup> StateFacts - Microsoft Internet Explorer             File Edit View Favorites Tools Help             ↓→ Back → → → ③ ② ③ △ ↓ ◎ Search ⓐ Favorites	영[Media 🎯 탄- 🧿 🖬 🗐 🤉				
Address Address http://localhost/USStateFacts/StateFacts.aspx		Co Links »			
<u>^</u>	Capital:	Boston			
	Flower:	Mayflower			
	Tree:	American Elm			
Bird: Chickadee					
Done     Done     Cocal intranet					

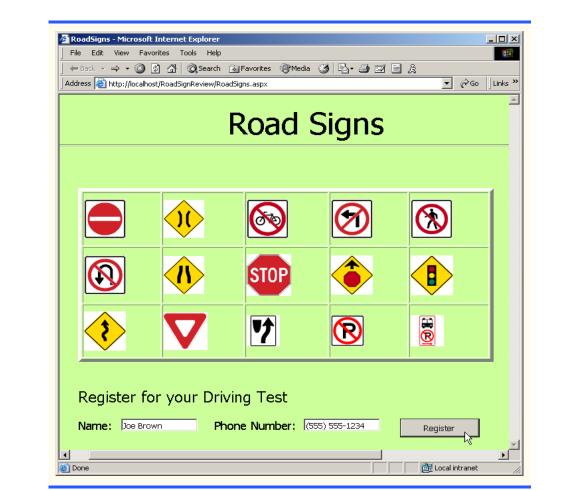
**31.13** (*Road Sign Review Application: Functionality*) Define the middle tier for the Road Sign Review application.

a) *Opening the application*. Open the **RoadSignReview** application that you created in Tutorial 29 and continued to develop in Tutorial 30.

- b) *Copying the SignImages folder to your project folder.* Copy the C:\Examples\Tutorial31\Exercises\Images\SignImages folder to the RoadSignReview folder.
- c) Importing System.Data.OleDb and System.Web.UI.WebControls to Road-Signs.aspx.vb. Import the System.Data.OleDb and System.Web.UI.WebControls namespaces to RoadSigns.aspx.vb. You need to import System.Web.UI.WebControls because you will be creating a Web control programmatically in this exercise.
- d) Defining the Page\_Load event handler for the RoadSigns.aspx page. Use the Open method to open the connection to the database. Create a data reader to read the information specified by the data command object.
- e) **Populating the Table with sign images in the RoadSigns.aspx page.** Add a Do While...Loop to RoadSigns.aspx's Page\_Load method. This loop should display an image of the sign and display the sign name in the ToolTip property. This property specifies the text that displays in a tooltip box when the mouse hovers over the Image. The sign image and name should be retrieved using the data reader. To display an Image in a cell of the Table, you need to create an Image control, specify a cell and use the cell's Controls.Add method to add an image to that cell. For example, to create an Image control programmatically, type Dim imgImageName As Image = New Image(). You then need to set the ImageURL property to the location of the desired image. To display an Image control in the first cell of the first row, you would write the line Table.Rows(0).Cells(0).Controls.Add(imgImageName). Also, if you wish to specify text for a tooltip, you must set the cell's ToolTip property—for example, Table.Rows(0).Cells(0).ToolTip = "This is a tooltip".
- f) *Closing the reader and connection.* Close the data reader and the connection to the database by invoking their Close methods.
- g) *Creating the Register Button's Click event handler for RoadSigns.aspx*. Double click the **Register** Button of RoadSigns.aspx to create the Click event handler.
- h) *Creating Session item*. Create two Session items in the Click event handler, and set the first one equal to the user input for the **Name:** TextBox. The second Session item should equal the user input for the **Phone Number:** TextBox.
- i) *Redirecting to the RoadTestRegistered.aspx page*. In the Click event handler, use the Redirect.Response method to redirect the client browser to the RoadTestRegistered.aspx page.
- j) Defining the Page\_Load method of RoadTestRegistered.aspx page. Use the Session items to display a confirmation to the user about the user's registration information. Display the confirmation using Label lblConfirmation. Display the user's name, and display text which states that the user will be contacted shortly at the phone number provided. This information should be displayed in a Label.

```
1
      Exercise 31.13 Solution
 2
     ' RoadSigns.aspx.vb
 3
 4
    Imports System.Data.0leDb
 5
    Imports System.Web.UI.WebControls
 6
 7
    Public Class RoadSigns
 8
       Inherits System.Web.UI.Page
 9
10
        ' control declarations
11
12
        ' Web Form Designer Generated Code
13
14
        ' invoked when page is loaded
15
       Private Sub Page_Load(ByVal sender As System.Object, _
16
           ByVal e As System. EventArgs) Handles MyBase. Load
17
18
           objOleDbConnection.Open() ' open connection to the database
```

```
19
20
           ' declare reader to read from database
21
          Dim objReader As OleDbDataReader
22
23
           ' create reader to read from database
24
           objReader = objSelectSignInformation.ExecuteReader()
25
26
           Dim intRow As Integer = 0
27
          Dim intColumn As Integer = 0
28
29
           ' display signs in Table
30
          Do While objReader.Read()
31
32
              ' move to next row
33
             If intColumn > 4 Then
34
                 intRow += 1
35
                 intColumn = 0
36
              End If
37
38
              ' create new Image control
39
             Dim imgCellImage As Image = New Image()
40
41
              ' set ImageURL property
42
              imgCellImage.ImageUrl = "SignImages/" & _
43
                 Convert.ToString(objReader("sign"))
44
45
              ' add image to table
46
              tblRoadSigns.Rows((intRow)).Cells( _
47
                 intColumn).Controls.Add(imgCellImage)
48
49
              ' add image name to ToolTip
50
              tblRoadSigns.Rows(intRow).Cells(intColumn).ToolTip = _
51
                 Convert.ToString(objReader("name"))
52
53
              intColumn += 1 ' increment column location
54
          Loop
55
56
           objReader.Close() ' close the reader
57
58
           ' close the connection to the database
59
           obj0leDbConnection.Close()
60
       End Sub ' Page_Load
61
62
        ' handles click event for btnRegister Button
63
       Private Sub btnRegister_Click(ByVal sender As System.Object, _
64
           ByVal e As System. EventArgs) Handles btnRegister. Click
65
66
           ' create Session item
67
           Session("strName") = txtName.Text
68
           Session("strPhoneNumber") = txtPhoneNumber.Text
69
70
           ' redirect to another ASPX page
71
           Response.Redirect("RoadTestRegistered.aspx")
72
       End Sub ' btnRegister_Click
73
74
    End Class ' RoadSigns
```



1	' Exercise 31.13 Solution
2	' RoadTestRegistered.aspx.vb
3	
4	Public Class RoadTestRegistered
5	Inherits System.Web.UI.Page
6	
7	' control declarations
8	
9	'Web Form Designer Generated Code
10	
11	' invoked when page is loaded
12	<pre>Private Sub Page_Load(ByVal sender As System.Object,</pre>
13	ByVal e As System.EventArgs) Handles MyBase.Load
14	
15	' display output
16	<pre>lblConfirmation.Text = Convert.ToString(Session("strName")) &amp; _</pre>
17	", you have registered to take your driving test. " & _
18	"We will call you back at " & _
19	<pre>Convert.ToString(Session("strPhoneNumber")) &amp; _</pre>
20	" to confirm a date."
21	End Sub ' Page_Load
22	
23	End Class ' RoadTestRegistered







# Enhanced Car Payment Calculator Application

Introducing Exception Handling Solutions

Instructor's Manual Exercise Solutions Tutorial 32	<ul><li>32.1 Dealing with exceptional situations as a</li><li>a) exception detection</li><li>c) exception resolution</li></ul>	an application executes is called b) exception handling d) exception debugging		
	<ul> <li>32.2 A(n) is always followed by</li> <li>a) if statement</li> <li>c) Try block</li> </ul>	<ul><li>at least one Catch block or a Finally block.</li><li>b) event handler</li><li>d) None of the above.</li></ul>		
	32.3 The method call Convert.ToInt32( a) FormatException c) DivideByZeroException	,		
	<b>32.4</b> If no exceptions are thrown in a Try block,			
	<ul><li>a) the Catch block(s) are skipped</li><li>c) an error occurs</li></ul>			
	<b>32.5</b> A(n) is an exception that do might cause the application to terminate exe	bes not have an exception handler, and therefore ecution.		
	<ul><li>a) uncaught block</li><li>c) error handler</li></ul>	<ul><li>b) uncaught exception</li><li>d) thrower</li></ul>		
	<b>32.6</b> A Try block can have associated with it.			
	a) only one Catch block	b) several Finally blocks		
	c) one or more Catch blocks	d) None of the above.		
	<b>32.7</b> The statement is used to re	throw an exception from inside a Catch block.		
	a) Rethrow	b) Throw		
	c) Try	d) Catch		
	<b>32.8</b> marks the end of a Try b blocks.	lock and its corresponding Catch and Finally		
	a) End Try	b) End Finally		
	c) End Catch	d) End Exception		
	32.9 A Finally block is located			
	<ul><li>a) after the Try block, but before each Catch block</li><li>b) before the Try block</li></ul>			
	<ul><li>c) after the Try block and the Try block</li><li>d) Either (b) or (c).</li></ul>	k's corresponding Catch blocks		
	<b>32.10</b> A is executed if an except is thrown.	ion is thrown from a Try block or if no exception		
	a) Catch block	b) Finally block		
	c) exception handler	d) All of the above.		
	<b>Answers:</b> 32.1) b. 32.2) c. 32.3) a. 32.4) a. 3	2.5) b. 32.6) c. 32.7) b. 32.8) a. 32.9) c. 32.10) b.		

**EXERCISES 32.11** (*Enhanced Miles Per Gallon Application*) Modify the Miles Per Gallon application (Exercise 13.13) to use exception handling to process the FormatExceptions that occur when converting the strings in the TextBoxes to Doubles (Fig. 32.16). The original application allowed the user to input the number of miles driven and the number of gallons used for a tank of gas, to determine the number of miles the user was able to drive on one gallon of gas.

Hiles Per Gallor					
Miles driven:	1000	Invalid Nu	ımber Format		×
Gallons used:	thirty	8	Please enter decim	al numbers for the miles o	driven and gallons.
Miles per gallon:			Yuurd	ОК	
	Calculate MPG			🖶 Miles Per Gallon	
				Miles driven:	1000
				Gallons used:	30
				Miles per gallon:	33.33
					Calculate MPG

Figure 32.16 Enhanced Miles Per Gallon application's GUI.

- a) *Copying the template to your working directory*. Copy the directory C:\Examples\Tutorial32\Exercises\EnhancedMilesPerGallon to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click EnhancedMilesPerGallon.sln in the EnhancedMilesPerGallon directory to open the application.
- c) *Adding a Try block.* Find the btnCalculateMPG\_Click event handler. Enclose all of the code in this event handler in a Try block.
- d) *Adding a Catch block.* After the Try block, add a Catch block to handle any FormatExceptions that may occur in the Try block. Inside the Catch block, add code to display an error message dialog.
- e) *Running the application.* Select **Debug > Start** to run your application. Enter invalid data as shown in Fig. 32.16 and click the **Calculate MPG** Button. A Message-Box should appear asking you to enter valid input. Enter valid input and click the **Calculate MPG** Button again. Verify that the correct output is displayed.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

#### **Answer:**

```
1
      Exercise 32.11 Solution
 2
    ' MilesPerGallon.vb
 3
 4
    Public Class FrmMilesPerGallon
 5
       Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
 8
 9
        ' calculate and return miles per gallon
10
       Private Function MilesPerGallon( _
11
          ByVal dblMilesDriven As Double,
12
          ByVal dblGallonsUsed As Double) As Double
13
14
          Return dblMilesDriven / dblGallonsUsed
15
       End Function ' MilesPerGallon
16
17
           ' handles Calculate Button's Click event
18
       Private Sub btnCalculateMPG_Click(ByVal sender As _
19
           System.Object, ByVal e As System.EventArgs) _
20
          Handles btnCalculateMPG.Click
21
22
           ' retrieve user input
```

```
23
           Try
24
25
               ' display miles per gallon
              lblOutputValue.Text = String.Format("{0:F}", _
26
27
                 MilesPerGallon(Convert.ToDouble(txtMilesDriven.Text), _
28
                 Convert.ToDouble(txtGallonsUsed.Text)))
29
30
           ' prompt for input in correct format
31
           Catch formatExceptionParameter As FormatException
32
33
              MessageBox.Show( _
34
                 "Please enter decimal numbers for " & _
35
                 "the miles driven and gallons.", _
                 "Invalid Number Format", MessageBoxButtons.OK, _
36
37
                 MessageBoxIcon.Error)
38
39
           End Try ' end Try...Catch statement
40
41
        End Sub ' btnCalculateMPG_Click
42
43
     End Class ' FrmMilesPerGallon
```

**32.12** (*Enhanced Prime Numbers Application*) Modify the **Prime Numbers** application (Exercise 13.17) to use exception handling to process the FormatExceptions that occur when converting the strings in the TextBoxes to Integers (Fig. 32.17). The original application took two numbers (representing a lower bound and an upper bound) and determined all of the prime numbers within the specified bounds, inclusive. An Integer greater than 1 is said to be prime if it is divisible by only 1 and itself. For example, 2, 3, 5 and 7 are prime numbers, but 4, 6, 8 and 9 are not.

🖳 Prime Numbers	_ 🗆 🗙	Invalid Number Format	X
Lower bound:	5	Please enter integers for the lower and upper bou	nds.
Upper bound:	29.9	CK]	
Prime numbers:			
	<b></b>	Prime Numbers	
		Lower bound: 5	
	Y	Upper bound: 30	
Calculate F	Primes	Prime numbers:	
		57 -	
		11	
		13 17	
		19	
		Calculate Primes	

Figure 32.17 Enhanced Prime Numbers application's GUI.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial32\Exercises\EnhancedPrimeNumbers directory to your C:\SimplyVB directory.
- b) *Opening the application's template file.* Double click EnhancedPrimeNumbers.sln in the EnhancedPrimeNumbers directory to open the application.
- c) *Adding a Try block*. Find the btnCalculatePrimes\_Click event handler. Enclose all the code following the variable declarations in a Try block.

- d) Adding a Catch block. Add a Catch block that catches any FormatExceptions that may occur in the Try block you added to btnCalculatePrimes\_Click in Step c. Inside the Catch block, add code to display an error message dialog.
- e) *Running the application.* Select **Debug > Start** to run your application. Enter invalid data as shown in Fig. 32.17 and click the **Calculate Primes** Button. A MessageBox should appear asking you to enter valid input. Enter valid input and click the **Calculate Primes** Button again. Verify that the correct output is displayed.
- f) *Closing the application.* Close your running application by clicking its close box.
- g) Closing the IDE. Close Visual Studio .NET by clicking its close box.

**Answer:** 

1	' Exercise 32.12 Solution
2	' PrimeNumbers.vb
3	
4	Public Class FrmPrimeNumbers
5	Inherits System.Windows.Forms.Form
6	· · · · · · · · · · · · · · · · · · ·
7	' Windows Form Designer generated code
8	······································
9	' determine if number is prime
10	Private Function Prime(ByVal intNumber As Integer) As Boolean
ii	Dim intCount As Integer ' declare counter
12	blim integant As integer decrare counter
13	' set square root of intNumber as limit
14	Dim intLimit As Integer = Convert.ToInt32(Math.Sqrt(intNumber))
15	Dim michaint As integer = convert. Toints2(math.sqrt(michainber))
16	Lleen until intCount meaches squame meat of intNumber
17	<pre>' loop until intCount reaches square root of intNumber For intCount = 2 To intLimit</pre>
18	For intcount = $2$ to intlimit
	The intlumber Med intCount of Them
19	If intNumber Mod intCount = 0 Then
20	Return False ' number is not prime
21	End If
22	
23	Next
24	
25	Return True ' number is prime
26	End Function ' Prime
27	
28	' handles Calculate Primes Button's Click event
29	Private Sub btnCalculatePrimes_Click(ByVal sender As _
30	System.Object, ByVal e As System.EventArgs) _
31	Handles btnCalculatePrimes.Click
32	
33	'declare variables
34	Dim intLowerBound As Integer
35	Dim intUpperBound As Integer
36	Dim intCounter As Integer
37	Dim strOutput As String
38	
39	' attempt to retrieve input from user
40	Try
41	<pre>intLowerBound = Convert.ToInt32(txtLowerBound.Text)</pre>
42	intUpperBound = Convert.ToInt32(txtUpperBound.Text)
43	
44	<pre>If intLowerBound &lt;= 0 OrElse intUpperBound &lt;= 0 Then</pre>
45	<pre>MessageBox.Show("Bounds must be greater than 0", _</pre>
46	"Invalid Bounds", MessageBoxButtons.OK, _
47	MessageBoxIcon.Exclamation)
<b>48</b>	<pre>ElseIf intUpperBound &lt; intLowerBound Then</pre>
49	<pre>MessageBox.Show("Upper bound cannot be less than " &amp; _</pre>

```
50
                     "lower bound", "Invalid Bounds", _
51
                    MessageBoxButtons.OK, MessageBoxIcon.Exclamation)
52
              Else
53
54
                  ' loop from lower bound to upper bound
55
                 For intCounter = intLowerBound To intUpperBound
56
57
                     ' if prime number, display in TextBox
58
                    If Prime(intCounter) = True Then
59
                       strOutput &= (intCounter & ControlChars.CrLf)
60
                    End If
61
62
                 Next
63
              End If
64
65
              txtPrimeNumbers.Text = strOutput
66
67
           Catch formatExceptionParameter As FormatException
68
              MessageBox.Show( _
69
                 "Please enter integers for the lower and upper " + _
                 "bounds.", "Invalid Number Format", _
70
71
                 MessageBoxButtons.OK, MessageBoxIcon.Error)
72
73
           End Try ' end Try...Catch statement
74
75
        End Sub ' btnCalculatePrimes_Click
76
77
     End Class ' FrmPrimeNumbers
```

**32.13** (*Enhanced Simple Calculator Application*) Modify the Simple Calculator application (Exercise 6.13) to use exception handling to process the FormatExceptions that occur when converting the strings in the TextBoxes to Integers and the DivideByZeroException when performing the division (Fig. 32.18). We will define what a DivideByZeroException is shortly. The application should still perform simple addition, subtraction, multiplication and division.

E Calculator						
Enter first number:	10	+ -				
Enter second number:	0	* 1				
Result:						
Arithmetic Error		X				
You cannot divide by 0. Please enter another value.						
	OK )					
Colculator						
🛄 Calculator						
Enter first number:	10	- • ×				
	10 2	+ - * []]				
Enter first number:		± •				

Figure 32.18 Enhanced Simple Calculator application.

a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial32\Exercises\EnhancedSimpleCalculator directory to your C:\SimplyVB directory.

- b) **Opening the application's template file.** Double click EnhancedSimpleCalculator.sln in the EnhancedSimpleCalculator directory to open the application.
- c) Adding a Try block to the btnAdd\_Click event handler. Find the btnAdd\_Click event handler. Enclose the body of btnAdd\_Click in a Try block.
- d) Adding a Catch block to the btnAdd\_Click event handler. Add a Catch block that catches any FormatExceptions that may occur in the Try block that you added in *Step c*. Inside the Catch block, add code to display an error message dialog.
- e) Adding a Try block to the btnSubtract\_Click event handler. Find the btnSubtract\_Click event handler, which immediately follows btnAdd\_Click. Enclose the body of the btnSubtract\_Click in a Try block.
- f) Adding a Catch block to the btnSubtract\_Click event handler. Add a Catch block that catches any FormatExceptions that may occur in the Try block that you added in *Step e*. Inside the Catch block, add code to display an error message dialog.
- g) Adding a Try block to the btnMultiply\_Click event handler. Find the btnMulitply\_Click event handler, which immediately follows btn\_Subtract\_Click. Enclose the body of the btnMultiply\_Click in a Try block.
- h) Adding a Catch block to the btnMultiply\_Click event handler. Add a Catch block that catches any FormatExceptions that may occur in the Try block that you added in *Step g*. Inside the Catch block, add code to display an error message dialog.
- i) Adding a Try block to the btnDivide\_Click event handler. Find the btnDivide\_Click event handler, which immediately follows btnMultiply\_Click. Enclose the body of the btnDivide\_Click in a Try block.
- j) Adding a Catch block to the btnDivide\_Click event handler. Add a Catch block that catches any FormatExceptions that may occur in the Try block that you added in Step i. Inside the Catch block, add code to display an error message dialog.
- k) Adding a second Catch block to the btnDivide\_Click event handler. Immediately following the first Catch block inside the btnDivide\_Click event handler, add a Catch block to catch any DivideByZeroExceptions. A DivideByZeroException is thrown when division by zero in integer arithmetic occurs. Inside the Catch block, add code to display an error message dialog.
- Running the application. Select Debug > Start to run your application. Enter valid input for the first number and 0 for the second number, then click the Button for division. A MessageBox should appear asking you not to divide by 0. Enter invalid input (such as letters) for the first and second number, then click any one of the Buttons provided. This time, a MessageBox should appear asking you to enter valid input. Enter valid input and click any one of the Buttons provided. Verify that the correct output is displayed.
- m) Closing the application. Close your running application by clicking its close box.
- n) *Closing the IDE*. Close Visual Studio .NET by clicking its close box.

#### **Answer:**

1	' Exercise 32.13 Solution
2	'SimpleCalculator.vb
3	
4	Public Class FrmCalculator
5	Inherits System.Windows.Forms.Form
6	
7	' Windows Form Designer generated code
8	
9	' handles addition Button's Click event
0	<pre>Private Sub btnAdd_Click(ByVal sender As System.Object, _</pre>
1	ByVal e As System.EventArgs) Handles btnAdd.Click
2	
3	' try to get user input
4	Try
5	
6	<pre>lblResult.Text = (Convert.ToInt32(txtFirstNumber.Text)</pre>

7	Convert.ToInt32(txtSecondNumber.Text)).ToString
8	
9	' handle case when user enters invalid input
20	Catch formatExceptionParameter As FormatException
21	
22	' prompt user for correct input
23	MessageBox.Show( _
24	"Please enter two integer values.", _
25	"Invalid Number Format", MessageBoxButtons.OK, _
26	MessageBoxIcon.Error)
	Messageboxicon.error
27	
28	End Try ' end TryCatch statement
.9	
30	End Sub ' btnAdd_Click
51	
	Liber die sechter staten Better die Glächersenst
2	' handles subtraction Button's Click event
33	<pre>Private Sub btnSubtract_Click(ByVal sender As System.Object, _</pre>
34	ByVal e As System.EventArgs) Handles btnSubtract.Click
5	
6	' try to get user input
7	Try
8	
9	<pre>lblResult.Text = (Convert.ToInt32(txtFirstNumber.Text)</pre>
0	Convert.ToInt32(txtSecondNumber.Text)).ToString
1	
2	L bandla casa when usan entens involid input
	' handle case when user enters invalid input
3	Catch formatExceptionParameter As FormatException
4	
5	' prompt user for correct input
6	MessageBox.Show(
7	
	"Please enter two integer values.", _
8	"Invalid Number Format", MessageBoxButtons.OK, _
9	MessageBoxIcon.Error)
0	
51	End Try ' end TryCatch statement
2	
3	End Sub ' btnSubtract_Click
4	
5	' handles multiplication Button's Click event
6	<pre>Private Sub btnMultiply_Click(ByVal sender As System.Object, _</pre>
7	ByVal e As System.EventArgs) Handles btnMultiply.Click
	bytar c ho bystan eventh go, handles beindreipty.effck
8	
9	' try to get user input
)	Try
1	
2	lblResult.Text = (Convert.ToInt32(txtFirstNumber.Text) * _
3	Convert.ToInt32(txtSecondNumber.Text)).ToString
4	
5	' handle case when user enters invalid input
6	Catch formatExceptionParameter As FormatException
7	
8	I prompt upon for connect input
	' prompt user for correct input
9	MessageBox.Show( _
0	"Please enter two integer values.", _
1	"Invalid Number Format", MessageBoxButtons.OK, _
2	MessageBoxIcon.Error)
3	hebbugeboxiconterrory
4	End Try ' end TryCatch statement
5	
6	End Sub ' btnMultiply_Click
7	

<pre>' handles division Button's Click event ' Private Sub bnDivide_Click(ByVal sender As System.Object, _ ByVal e As System.EventArgs) Handles btnDivide.Click ' try to retrieve user input and perform division Try ' Dim result As Integer = _ Convert.ToInt32(txtFirstNumber.Text) \ _ Convert.ToInt32(txtSecondNumber.Text) ' convert.ToInt32(txtSecondNumber.Text) ' handle case when user enters invalid input ' handle case when user of correct input ' handle case when user of correct input ' handle case when user tries to divide by zero ' "Invalid Number Format", MessageBoxButtons.OK, _ MessageBox.Show( _ ''Invalid Number for correct input ' handle case when user tries to divide by zero Catch divideByZeroExceptionParameter As DivideByZeroException ' handle case when user tries to divide by zero Catch divideByZeroExceptionParameter As DivideByZeroException ' handle case when user tries to divide by zero Catch divideByZeroExceptionParameter As DivideByZeroException ' handle case when user tries to divide by zero Catch divideByZeroExceptionParameter As DivideByZeroException ' handle case when user tries to divide by zero Catch divideByZeroExceptionParameter As DivideByZeroException ' handle case when user tries to divide by zero Catch divideByZeroExceptionParameter As DivideByZeroException ' handle case when user tries to divide by zero Catch divideByZeroExceptionParameter As DivideByZeroException ' handle case when user tries to divide by zero Catch divideByZeroExceptionParameter As DivideByZeroException ' handle case when user tries to divide by zero Catch divideByZeroExceptionParameter As DivideByZeroException ' handles tase conter tron', MessageBoxButtons.OK, _ MessageBox.Show( _ ''You cannot divide by 0. Please enter another value.'', _ As System.Object, ByVal e As System.EventArgs) _ Handles tastFirstNumber_TextChanged ' handles tastSecondNumber_TextChanged ' handles tastSecondNumbe</pre>		
<pre>80 ByVal e As System.EventArgs) Handles btnDivide.Click 81     ' try to retrieve user input and perform division 82    Try 84 85    Dim result As Integer = _ 86         Convert.ToInt32(txtFirstNumber.Text) \ _ 87         Convert.ToInt32(txtFirstNumber.Text) \ _ 88         TblResult.Text = result.ToString 90    ' handle case when user enters invalid input 91    ' handle case when user enters invalid input 92    Catch formatExceptionParameter As FormatException 93    ' prompt user for correct input 94    "Please enter two integer values.", _ 95    "Invalid Number Format", MessageBoxButtons.OK, _ 96    "Please enter two integer values.", _ 97    "Invalid Number Format", MessageBoxButtons.OK, _ 98    MessageBox.Show( _ 99 99 90 90 90 90 90 91    ' handle case when user tries to divide by zero 92 93    ' prompt user for correct input 94    MessageBox.Show( _ 95         "You cannot divide by 0. Please enter another value.", _ 96         "Arithmetic Error", MessageBoxButtons.OK, _ 97     MessageBoxIcon.Error) 98 99 99 99 99 99 99 99 90 90 90 90 99 99</pre>		' handles division Button's Click event
<pre>81 82 ' try to retrieve user input and perform division 83 Try 84 85 Dim result As Integer = _ 86 Convert.ToInt32(txtFirstNumber.Text) \ _ 87 Convert.ToInt32(txtSecondNumber.Text) 88 89 IblResult.Text = result.ToString 90 ' handle case when user enters invalid input 91 Catch formatExceptionParameter As FormatException 93 ' prompt user for correct input 94 messageBox.Show(_ 96 "Please enter two integer values.", _ 97 "Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 90 ' handle case when user tries to divide by zero 90 Catch divideByZeroExceptionParameter As DivideByZeroException 102 Catch divideByZeroExceptionParameter As DivideByZeroException 103 ' prompt user for correct input 104 MessageBox.Show(_ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 111 End Sub ' btnDivide_Click 121 ' handles TextChanged event 132 Private Sub txtFirstNumber.TextChanged(ByVal sender _ 133 As System.Object, ByVal e As System.EventArgs) _ 134 Handles TextChanged event 135 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 136 As System.Object, ByVal e As System.EventArgs) _ 137 Handles TextChanged event 138 DBResult.Text = "" 139 End Sub ' txtSecondNumber_TextChanged(ByVal sender _ 149 As System.Object, ByVal e As System.EventArgs) _ 140 Handles txtSecondNumber.TextChanged(ByVal sender _ 141 BiResult.Text = "" 142 End Sub ' txtSecondNumber.TextChanged(ByVal sender _ 143 As System.Object, ByVal e As System.EventArgs) _ 144 Handles txtSecondNumber.TextChanged(ByVal sender _ 145 As System.Object, ByVal e As System.EventArgs) _ 144 Handles txtSecondNumber.TextChanged(ByVal sender _ 145 As System.Object, ByVal e As System.EventArgs) _ 144 Handles txtSecondNumber.TextChanged(ByVal sender _ 145 As System.Object, ByVal e As System.EventArgs) _ 144 Handles txtSecondNumber.TextChanged(ByVal sender _ 145 As System.Object, ByVal e As System.EventArgs) _ 144 Handles txtSecondNum</pre>	79	
<pre>81 82 ' try to retrieve user input and perform division 83 Try 84 85 Dim result As Integer = _ 86 Convert.ToInt32(txtFirstNumber.Text) \ _ 87 Convert.ToInt32(txtSecondNumber.Text) 88 89 IblResult.Text = result.ToString 90 ' handle case when user enters invalid input 91 Catch formatExceptionParameter As FormatException 93 ' prompt user for correct input 94 messageBox.Show(_ 96 "Please enter two integer values.", _ 97 "Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 90 ' handle case when user tries to divide by zero 90 Catch divideByZeroExceptionParameter As DivideByZeroException 102 Catch divideByZeroExceptionParameter As DivideByZeroException 103 ' prompt user for correct input 104 MessageBox.Show(_ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 111 End Sub ' btnDivide_Click 121 ' handles TextChanged event 132 Private Sub txtFirstNumber.TextChanged(ByVal sender _ 133 As System.Object, ByVal e As System.EventArgs) _ 134 Handles TextChanged event 135 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 136 As System.Object, ByVal e As System.EventArgs) _ 137 Handles TextChanged event 138 DBResult.Text = "" 139 End Sub ' txtSecondNumber_TextChanged(ByVal sender _ 149 As System.Object, ByVal e As System.EventArgs) _ 140 Handles txtSecondNumber.TextChanged(ByVal sender _ 141 BiResult.Text = "" 142 End Sub ' txtSecondNumber.TextChanged(ByVal sender _ 143 As System.Object, ByVal e As System.EventArgs) _ 144 Handles txtSecondNumber.TextChanged(ByVal sender _ 145 As System.Object, ByVal e As System.EventArgs) _ 144 Handles txtSecondNumber.TextChanged(ByVal sender _ 145 As System.Object, ByVal e As System.EventArgs) _ 144 Handles txtSecondNumber.TextChanged(ByVal sender _ 145 As System.Object, ByVal e As System.EventArgs) _ 144 Handles txtSecondNumber.TextChanged(ByVal sender _ 145 As System.Object, ByVal e As System.EventArgs) _ 144 Handles txtSecondNum</pre>	80	ByVal e As System.EventArgs) Handles btnDivide.Click
<pre>83 Try 84 85 Dim result As Integer = _ 86 Convert.ToInt32(txtFirstNumber.Text) \ _ 87 Convert.ToInt32(txtSecondNumber.Text) 89 IblResult.Text = result.ToString 90 ' handle case when user enters invalid input 91 Catch formatExceptionParameter As FormatException 93 ' prompt user for correct input 94 ' prompt user for correct input 95 MessageBox.Show( _ 96 ''Please enter two integer values.", _ 97 ''Tunvalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 90 ' handle case when user tries to divide by zero 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 ' prompt user for correct input 104 MessageBox.Show( _ 105 ''You cannot divide by 0. Please enter another value.", _ 106 ''Arithmetic Error', MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 111 End Sub ' btnDivide_Click 122 ' handles TextChanged event 134 Private Sub txtFirstNumber.TextChanged(ByVal sender _ 135 As System.Object, ByVal e As System.EventArgs) _ 136 Handles txtFirstNumber_TextChanged(ByVal sender _ 137 As System.Object, ByVal e As System.EventArgs) _ 138 Handles TextChanged event 139 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 134 As System.Object, ByVal e As System.EventArgs) _ 135 As System.Object, ByVal e As System.EventArgs) _ 136 Handles txtSecondNumber_TextChanged(ByVal sender _ 137 As System.Object, ByVal e As System.EventArgs) _ 139 Handles txtSecondNumber_TextChanged(ByVal sender _ 140 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 155 As System.Object, ByVal e As System.EventArgs) _ 156 Handles txtSecondNumber_TextChanged(ByVal sender _ 157 As System.Object, ByVal e As System.EventArgs) _ 157 Handles txtSecondNumber_TextChanged 158 BiResult.Text = ''' 159 End Sub ' txtSecondNumber_TextChanged(ByVal sender _ 150 As System.Object, ByVal e As System.EventArgs) _ 150 Handles txtSecondNumber_TextChanged 150 BiResult.Text = ''' 151 End Sub ' txtSecondNumber_TextChanged 152 BiResult.Text = ''' 153 End Sub ' txtSecondNumber_TextChanged</pre>	81	
<pre>84 85 86 86 86 87 87 87 87 88 89 90 91 91 91 92 91 9 9 9 9 9 9 9 9 9 9 9 9</pre>	82	' try to retrieve user input and perform division
<pre>85 Dim result As Integer = _ 86 Convert.ToInt32(txtFirstNumber.Text) \ _ 87 Convert.ToInt32(txtSecondNumber.Text) 88 89 lblResult.Text = result.ToString 90 91 ' handle case when user enters invalid input 92 Catch formatExceptionParameter As FormatException 93 94 ' prompt user for correct input 95 MessageBox.Show( _ 96 "Flease enter two integer values.", _ 97 'Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 90 ' handle case when user tries to divide by zero 90 Catch divideByZeroExceptionParameter As DivideByZeroException 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 ' prompt user for correct input 103 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 ''Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 109 End Sub txFirstNumber_TextChanged(ByVal sender _ 118 IblResult.Text = "' 129 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 130 As System.Object, ByVal e As System.EventArgs) _ 131 IblResult.Text = "'' 132 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 133 As System.Object, ByVal e As System.EventArgs) _ 144 Handles TextChanged event 152 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 153 As System.Object, ByVal e As System.EventArgs) _ 154 Handles txtSecondNumber_TextChanged 155 IblResult.Text = "'' 156 End Sub ' txtSecondNumber_TextChanged 157 As System.Object, ByVal e As System.EventArgs) _ 158 Handles txtSecondNumber_TextChanged 159 IblResult.Text = "'' 150 End Sub ' txtSecondNumber_TextChanged 150 IblResult.Text = "'' 150 End Sub ' txtSecondNumber_TextChanged 151 IblResult.Text = "'' 152 End Sub ' txtSecondNumber_TextChanged 153 IblResult.Text = "'' 154 End Sub ' txtSecondNumber_TextChanged 155 IblResult.Text = "'' 155 IblResult.Text = "'' 155 IblResult.T</pre>	83	Try
<pre>85 Dim result As Integer = _ 86 Convert.ToInt32(txtFirstNumber.Text) \ _ 87 Convert.ToInt32(txtSecondNumber.Text) 88 89 lblResult.Text = result.ToString 90 91 ' handle case when user enters invalid input 92 Catch formatExceptionParameter As FormatException 93 94 ' prompt user for correct input 95 MessageBox.Show( _ 96 "Flease enter two integer values.", _ 97 'Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 90 ' handle case when user tries to divide by zero 90 Catch divideByZeroExceptionParameter As DivideByZeroException 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 ' prompt user for correct input 103 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 ''Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 109 End Sub txFirstNumber_TextChanged(ByVal sender _ 118 IblResult.Text = "' 129 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 130 As System.Object, ByVal e As System.EventArgs) _ 131 IblResult.Text = "'' 132 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 133 As System.Object, ByVal e As System.EventArgs) _ 144 Handles TextChanged event 152 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 153 As System.Object, ByVal e As System.EventArgs) _ 154 Handles txtSecondNumber_TextChanged 155 IblResult.Text = "'' 156 End Sub ' txtSecondNumber_TextChanged 157 As System.Object, ByVal e As System.EventArgs) _ 158 Handles txtSecondNumber_TextChanged 159 IblResult.Text = "'' 150 End Sub ' txtSecondNumber_TextChanged 150 IblResult.Text = "'' 150 End Sub ' txtSecondNumber_TextChanged 151 IblResult.Text = "'' 152 End Sub ' txtSecondNumber_TextChanged 153 IblResult.Text = "'' 154 End Sub ' txtSecondNumber_TextChanged 155 IblResult.Text = "'' 155 IblResult.Text = "'' 155 IblResult.T</pre>	84	
<pre>86 Convert.ToInt32(txtFirstNumber.Text) \ _ 87 Convert.ToInt32(txtSecondNumber.Text) 88 9 lblResult.Text = result.ToString 90 91 ' handle case when user enters invalid input 92 Catch formatExceptionParameter As FormatException 93 94 ' prompt user for correct input 95 MessageBox.Show( _ 96 "Please enter two integer values.", _ 97 "Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 90 ' handle case when user tries to divide by zero 100 ' handle case when user tries to divide by zero 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 103 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 'Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 110 110 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 118 lblResult.Text = "" 129 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 120 As System.Object, ByVal e As System.EventArgs) _ 121 Handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber_TextChanged 125 IblResult.Text = "" 126 End Sub ' txtSecondNumber_TextChanged 127 Handles txtSecondNumber_TextChanged 128 IblResult.Text = "" 129 End Sub ' txtSecondNumber_TextChanged 120 130 IblResult.Text = "" 131 End Sub ' txtSecondNumber_TextChanged 132 IblResult.Text = "" 133 End Sub ' txtSecondNumber_TextChanged 134 IblResult.Text = "" 135 End Sub ' txtSecondNumber_TextChanged 135 IblResult.Text = "" 136 End Sub ' txtSecondNumber_TextChanged 137 IblResult.Text = "" 138 End Sub ' txtSecondNumber_TextChanged 139 IblResult.Text = "" 130 End Sub ' txtSecondNumber_TextChanged 130 IblResult.Text = "" 131 End Sub ' txtSecondNumber_TextChanged 135 IblResult.Text = "" 136 End Sub ' txtSecondNumber_TextChanged 137 IblResult.Text = "" 137 End Sub</pre>		Dim result As Integer =
<pre>67 Convert.ToInt32(txtSecondNumber.Text) 78 79 lblResult.Text = result.ToString 70 71 ' handle case when user enters invalid input 72 Catch formatExceptionParameter As FormatException 73 74 ' prompt user for correct input 75 MessageBox.Show( _ 76 '' "Please enter two integer values.", _ 77 ''Invalid Number Format", MessageBoxButtons.OK, _ 78 MessageBoxIcon.Error) 79 70 ' handle case when user tries to divide by zero 70 71 Catch divideByZeroExceptionParameter As DivideByZeroException 71 72 '' prompt user for correct input 73 '' prompt user for correct input 74 MessageBox.Show( _ 75 '' You cannot divide by 0. Please enter another value.", _ 75 '' You cannot divide by 0. Please enter another value.", _ 76 '' You cannot divide by 0. Please enter another value.", _ 77 MessageBoxIcon.Error) 78 79 End Try ' end TryCatch statement 70 71 MessageBoxIcon.Error) 71 73 Handles TextChanged event 74 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 75 As System.Object, ByVal e As System.EventArgs) _ 76 Handles TextChanged event 77 78 79 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 71 Sub ' txtFirstNumber_TextChanged(ByVal sender _ 72 As System.Object, ByVal e As System.EventArgs) _ 73 Handles txtSecondNumber_TextChanged(ByVal sender _ 74 As System.Object, ByVal e As System.EventArgs) _ 75 Handles txtSecondNumber_TextChanged 76 LblResult.Text = ''' 77 End Sub ' txtSecondNumber_TextChanged 77 78 Buselt.Text = ''' 79 End Sub ' txtSecondNumber_TextChanged 70 73 As System.Object, ByVal e As System.EventArgs) _ 74 Handles txtSecondNumber_TextChanged 75 LblResult.Text = ''' 75 End Sub ' txtSecondNumber_TextChanged 76 LblResult.Text = ''' 77 End Sub ' txtSecondNumber_TextChanged 76 LblResult.Text = ''' 77 End Sub ' txtSecondNumber_TextChanged 77 LblResult.Text = ''' 77 End Sub ' txtSecondNumber_TextChanged 75 LblResult.Text = ''' 75 End Sub ' txtSecondNumber_TextChanged 75 LblResult.Text = ''' 75 End Sub ' txtSecondNumber_TextChanged 75 LblResult.Text = ''' 75 End Sub ' txtSecondNumber_TextChanged 75 LblResult.Te</pre>		
<pre>88 89 lblResult.Text = result.ToString 90 91 ' handle case when user enters invalid input 92 Catch formatExceptionParameter As FormatException 93 94 ' prompt user for correct input 95 MessageBox.Show(_ 96 ''Please enter two integer values.",_ 97 ''Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 90 ' handle case when user tries to divide by zero 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 103 ' prompt user for correct input 104 MessageBox.Show(_ 105 ''You cannot divide by 0. Please enter another value.",_ 106 ''Arithmetic Error', MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 111 End Sub ' btnDivide_Click 122 13 ' handles TextChanged event 14 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 15 As System.Object, ByVal e As System.EventArgs) _ 16 Handles TextChanged event 17 18 DilResult.Text = "" 19 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 102 103 ' handles TextChanged event 12 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 13 As System.Object, ByVal e As System.EventArgs) _ 14 Handles txtSecondNumber_TextChanged(ByVal sender _ 15 As System.Object, ByVal e As System.EventArgs) _ 16 Handles TextChanged event 17 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 18 As System.Object, ByVal e As System.EventArgs) _ 19 Handles txtSecondNumber_TextChanged(ByVal sender _ 12 As System.Object, ByVal e As System.EventArgs) _ 14 Handles txtSecondNumber_TextChanged(ByVal sender _ 15 As System.Object, ByVal e As System.EventArgs) _ 15 Handles txtSecondNumber_TextChanged(ByVal sender _ 16 IblResult.Text = "" 17 End Sub ' txtSecondNumber_TextChanged(ByVal sender _ 18 IblResult.Text = "" 19 End Sub ' txtSecondNumber_TextChanged(ByVal sender _ 19 IblResult.Text = "" 19 End Sub ' txtSecondNumber_TextChanged(ByVal sender _ 104 IblResult.Text = "" 105 IblResult.Text = "" 106 IblResult.Text = "" 107 End Sub ' txtSecondNumber_TextChanged 108 108 IblResult.Text = "" 109 End Sub ' txtSecondNumber_Text</pre>		
<pre>89 lblResult.Text = result.ToString 90 91 ' handle case when user enters invalid input 92 Catch formatExceptionParameter As FormatException 93 94 ' prompt user for correct input 95 MessageBox.Show(_ 96 "Please enter two integer values.",_ 97 "Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 100 ' handle case when user tries to divide by zero 91 Catch divideByZeroExceptionParameter As DivideByZeroException 92 101 Catch divideByZeroExceptionParameter As DivideByZeroException 93 ' prompt user for correct input 94 MessageBox.Show(_ 95 "You cannot divide by 0. Please enter another value.", _ 96 "Arithmetic Error", MessageBoxButtons.OK, _ 97 MessageBoxIcon.Error) 98 199 End Try ' end TryCatch statement 99 100 End Try ' end TryCatch statement 900 End Sub ' btnDivide_Click 911 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 912 As System.Object, ByVal e As System.EventArgs) _ 913 Handles txtFirstNumber_TextChanged 914 IblResult.Text = "" 915 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 916 As System.Object, ByVal e As System.EventArgs) _ 917 Handles TextChanged event 918 IblResult.Text = "" 919 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 920 As System.Object, ByVal e As System.EventArgs) _ 921 Handles TextCondNumber_TextChanged 922 ' handles TextChanged event 923 As System.Object, ByVal e As System.EventArgs) _ 924 Handles TextCondNumber_TextChanged 925 BiResult.Text = "" 926 BiResult.Text = "" 927 End Sub ' txtSecondNumber_TextChanged 928</pre>		
<pre>90 91 92 92 93 94 94 94 94 95 94 95 96 96 96 96 97 97 97 97 97 97 97 97 97 98 98 98 98 98 98 98 98 98 98 98 98 98</pre>		lblResult.Text = result ToString
<pre>91 ' handle case when user enters invalid input 92 Catch formatExceptionParameter As FormatException 93 ' prompt user for correct input 94 ' prompt user for correct input 95 MessageBox.Show( _ 96 "Please enter two integer values.", _ 97 "Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 100 ' handle case when user tries to divide by zero 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 ' prompt user for correct input 103 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 109 End Sub ' btnDivide_Click 113 ' handles TextChanged (ByVal sender _ 126 As System.Object, ByVal e As System.EventArgs) _ 127 Handles TextChanged event 128 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 139 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 140 Andles TextChanged event 141 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 142 As System.Object, ByVal e As System.EventArgs) _ 144 Handles TextCendged event 129 Private Sub txtSecondNumber_TextChanged 130 blResult.Text = "" 131 End Sub ' txtSecondNumber_TextChanged 132 blResult.Text = "" 133 End Sub ' txtSecondNumber_TextChanged 134 Handles TextCendyLoweder_TextChanged 135 blResult.Text = "" 134 End Sub ' txtSecondNumber_TextChanged 135 blResult.Text = "" 135 End Sub ' txtSecondNumber_TextChanged 136 blResult.Text = "" 137 End Sub ' txtSecondNumber_TextChanged 138</pre>		ibiliteourerreite – resulterrosering
<pre>22 Catch formatExceptionParameter As FormatException 23 24 ' prompt user for correct input 25 MessageBox.Show( _ 26 ''Please enter two integer values.", _ 27 ''Invalid Number Format", MessageBoxButtons.OK, _ 28 MessageBoxIcon.Error) 20 ' handle case when user tries to divide by zero 20 Catch divideByZeroExceptionParameter As DivideByZeroException 20 ' prompt user for correct input 20 '' prompt user for correct input 20 ''You cannot divide by 0. Please enter another value.", _ 20 ''You cannot divide by 0. Please enter another value.", _ 21 ''Arithmetic Error", MessageBoxButtons.OK, _ 22 ''Arithmetic Error'', MessageBoxButtons.OK, _ 23 ''You cannot divide by 0. Please enter another value.", _ 24 ''Arithmetic Error'', MessageBoxButtons.OK, _ 25 ''You cannot divide by 0. Please enter another value.", _ 26 ''You cannot divide by 0. Please enter another value.", _ 27 ''Arithmetic Error'', MessageBoxButtons.OK, _ 28 ''You cannot divide by 0. Please enter another value.", _ 29 ''Arithmetic Error'', MessageBoxButtons.OK, _ 20 ''Arithmetic Error'', MessageBoxButtons.OK, _ 21 ''Arithmetic Error'', MessageBoxButtons.OK, _ 22 ''Arithmetic Error'', MessageBoxButtons.OK, _ 23 ''Arithmetic Error'', MessageBoxButtons.OK, _ 24 ''Anithmetic Error'', MessageBoxButtons.OK, _ 25 ''Anoles TextChanged event 26 ''sandles txtFirstNumber_TextChanged(ByVal sender _ 27 ''sandles txtSecondNumber_TextChanged(ByVal sender _ 28 As System.Object, ByVal e As System.EventArgs) _ 29 Handles txtSecondNumber.TextChanged 20 ''handles txtSecondNumber.TextChanged 21 ''handles txtSecondNumber.TextChanged 22 ''handles txtSecondNumber.TextChanged 23 ''sandles txtSecondNumber.TextChanged 24 IblResult.Text = '''' 25 End Sub ''txtSecondNumber_TextChanged 26 IblResult.Text = '''' 27 End Sub ''txtSecondNumber_TextChanged 28</pre>		' handle case when user enters invalid input
<pre>93 ' prompt user for correct input 94 ' prompt user for correct input 95 MessageBox.Show( _ 96 "Please enter two integer values.", _ 97 "Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 ' handle case when user tries to divide by zero 90 Catch divideByZeroExceptionParameter As DivideByZeroException 102 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 101 End Sub ' btnDivide_Click 112 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber_TextChanged(ByVal sender _ 117 IblResult.Text = "" 128 Private Sub ' txtFirstNumber_TextChanged(ByVal sender _ 129 As System.Object, ByVal e As System.EventArgs) _ 120 Handles TextChanged event 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber_TextChanged 125 IblResult.Text = "" 126 End Sub ' txtSecondNumber_TextChanged 127 Handles txtSecondNumber_TextChanged 128 IblResult.Text = "" 129 End Sub ' txtSecondNumber_TextChanged</pre>		
<pre>94 ' prompt user for correct input 95 MessageBox.Show( _ 96 "Please enter two integer values.", _ 97 "Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 100 ' handle case when user tries to divide by zero 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 103 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 110 End Sub ' btnDivide_Click 120 ' handles TextChanged event 131 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 132 As System.Object, ByVal e As System.EventArgs) _ 133 Handles TextChanged event 134 IblResult.Text = "" 135 IblResult.Text = "" 136 IblResult.Text = "" 137 End Sub ' txtSecondNumber_TextChanged(ByVal sender _ 138 As System.Object, ByVal e As System.EventArgs) _ 139 Handles TextChanged event 130 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 131 As System.Object, ByVal e As System.EventArgs) _ 132 Handles txtSecondNumber_TextChanged 133 IblResult.Text = "" 134 End Sub ' txtSecondNumber_TextChanged 135 As System.Object, ByVal e As System.EventArgs) _ 136 Handles txtSecondNumber_TextChanged 137 IblResult.Text = "" 138 IblResult.Text = "" 139 End Sub ' txtSecondNumber_TextChanged 130 IblResult.Text = "" 130 For the SecondNumber_TextChanged 131 IblResult.Text = "" 132 End Sub ' txtSecondNumber_TextChanged 133 IblResult.Text = "" 134 IblResult.Text = "" 135 End Sub ' txtSecondNumber_TextChanged 136 IblResult.Text = "" 137 End Sub ' txtSecondNumber_TextChanged 138 IblResult.Text = "" 139 End Sub ' txtSecondNumber_TextChanged 130 IblResult.Text = "" 130 IblResult.Text = "" 131 IblResult.Text = "" 132 IblResult.Text = "" 133 IblResult.Text = "" 134 IblResult.Text = "" 135 IblResult.Text = "" 136 IblResult.Text = "" 137 IblResult.Text = "" 138 IblResult.</pre>		
<pre>95 MessageBox.Show(_ 96 "Please enter two integer values.", _ 97 "Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 100 ' handle case when user tries to divide by zero 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 103 ' prompt user for correct input 104 MessageBox.Show(_ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 109 End Sub ' btnDivide_Click 112 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber_TextChanged 117 118 End Sub ' txtFirstNumber_TextChanged 120 13 ' handles TextChanged event 149 Private Sub txtFirstNumber_TextChanged 150 161 Handles txtFirstNumber_TextChanged(ByVal sender _ 162 As System.Object, ByVal e As System.EventArgs) _ 162 Handles txtSecondNumber_TextChanged 173 174 Handles txtSecondNumber_TextChanged 175 As System.Object, ByVal e As System.EventArgs) _ 176 Handles txtSecondNumber_TextChanged 177 End Sub ' txtSecondNumber_TextChanged 178 BlResult.Text = "" 179 End Sub ' txtSecondNumber.TextChanged 170 As System.Object, ByVal e As System.EventArgs) _ 171 Handles txtSecondNumber.TextChanged 172 IblResult.Text = "" 173 End Sub ' txtSecondNumber.TextChanged 174 Handles txtSecondNumber.TextChanged 175 As System.Object, ByVal e As System.EventArgs) _ 176 Handles txtSecondNumber.TextChanged 177 As System.Object, ByVal e As System.EventArgs) _ 178 Handles txtSecondNumber.TextChanged 179 End Sub ' txtSecondNumber.TextChanged 170 As System.Object.ByVal e As System.EventArgs) _ 171 As System.Object.ByVal e As System.EventArgs) _ 172 End Sub ' txtSecondNumber.TextChanged 173 As System.Object.ByVal e As System.EventArgs) _ 174 As System.Object.ByVal e As System.EventArgs) _ 175 As System.Object.ByVal e As System.EventArgs) _ 176 As</pre>		I prompt user for connect input
<pre>96 "Please enter two integer values.", _ 97 "Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 99 90 ' handle case when user tries to divide by zero 100 Catch divideByZeroExceptionParameter As DivideByZeroException 102 103 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 109 End Sub ' btnDivide_Click 110 110 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber_TextChanged 125 126 IblResult.Text = "" 127 End Sub ' txtSecondNumber.TextChanged 128 129 120 121 ' handles txtSecondNumber_TextChanged 122 IblResult.Text = "" 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 IblResult.Text = "" 127 End Sub ' txtSecondNumber.TextChanged 128</pre>		
<pre>97 "Invalid Number Format", MessageBoxButtons.OK, _ 98 MessageBoxIcon.Error) 99 100 ' handle case when user tries to divide by zero 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 103 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 110 End Sub ' btnDivide_Click 121 13 ' handles TextChanged event 132 ' handles TextChanged event 133 As System.Object, ByVal e As System.EventArgs) _ 134 Handles txtFirstNumber_TextChanged 135 136 ' handles TextChanged event 136 1blResult.Text = "" 137 End Sub ' txtSecondNumber_TextChanged 138 139 Tivate Sub txtSecondNumber.TextChanged 139 140 151 As System.Object, ByVal e As System.EventArgs) _ 152 Handles TextChanged event 153 As System.Object, ByVal e As System.EventArgs) _ 154 Handles txtSecondNumber_TextChanged 155 155 TextChanged event 155 Private Sub txtSecondNumber_TextChanged 156 TextChanged 157 Text = "" 157 End Sub ' txtSecondNumber_TextChanged 158 TextSecondNumber.TextChanged 159 TextSecondNumber.TextChanged 150 TextSecondNumber.TextChanged 151 Text = "" 155 TextSecondNumber.TextChanged 155 TextSecondNumber.TextChanged 156 TextSecondNumber.TextChanged 157 TextSecondNumber.TextChanged 158 TextSecondNumber.TextChanged 159 TextSecondNumber.TextChanged 150 TextSecondNumber.TextChanged 150 TextSecondNumber.TextChanged 151 TextSecondNumber.TextChanged 152 TextSecondNumber.TextChanged 153 TextSecondNumber.TextChanged 154 TextSecondNumber.TextChanged 155 TextSecondNum</pre>		-
<pre>98 MessageBoxIcon.Error) 99 100 ' handle case when user tries to divide by zero 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 103 ' prompt user for correct input 104 MessageBox.Show(_ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 111 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber_TextChanged 117 118 IblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber_TextChanged 125 126 IblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128 128 129 130 140 140 141 140 141 141 141 141 141 14</pre>		
<pre>99 100 101 102 102 103 104 105 105 106 107 107 108 108 109 109 109 109 109 109 109 109 109 100 109 109</pre>		
<pre>100 ' handle case when user tries to divide by zero 101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 109 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 IblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 ' handles TextChanged event 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 IblResult.Text = "" 127 End Sub ' txtSecondNumber.TextChanged 128</pre>		MessageBox1con.Error)
<pre>101 Catch divideByZeroExceptionParameter As DivideByZeroException 102 103 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 111 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 IblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 125 126 IblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128 128 129 120 120 120 121 ' handles txtSecondNumber_TextChanged 120 121 ' handles txtSecondNumber_TextChanged 123 124 Handles txtSecondNumber.TextChanged 125 125 126 IblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128 128 129 120 120 120 120 120 120 120 121 ' handles txtSecondNumber_TextChanged 121 ' handles txtSecondNumber_TextChanged 122 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 125 IblResult.Text = "" 126 IblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128 128 129 120 120 120 120 120 120 120 120 120 120</pre>		
<pre>102 103 104 105 105 106 106 107 107 108 108 109 108 109 109 109 109 109 109 109 109 109 109</pre>		
<pre>103 ' prompt user for correct input 104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 100 109 End Try ' end TryCatch statement 100 101 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber_TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>104 MessageBox.Show( _ 105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 110 111 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber_TextChanged 117 118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber_TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128 128 13</pre>		
<pre>105 "You cannot divide by 0. Please enter another value.", _ 106 "Arithmetic Error", MessageBoxButtons.OK, _ 107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 110 111 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber_TextChanged 117 118 IblResult.Text = "" 129 End Sub ' txtFirstNumber_TextChanged(ByVal sender _ 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 IblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>106</pre>		
<pre>107 MessageBoxIcon.Error) 108 109 End Try ' end TryCatch statement 110 111 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>108 109 End Try ' end TryCatch statement 110 111 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 IblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 IblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>109 End Try ' end TryCatch statement 110 111 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 IblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 IblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>110 111 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>111 End Sub ' btnDivide_Click 112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115</pre>		
<pre>112 113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>113 ' handles TextChanged event 114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>114 Private Sub txtFirstNumber_TextChanged(ByVal sender _ 115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>115 As System.Object, ByVal e As System.EventArgs) _ 116 Handles txtFirstNumber.TextChanged 117 118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>116 Handles txtFirstNumber.TextChanged 117 118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>117 118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>118 lblResult.Text = "" 119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>	116	Handles txtFirstNumber.TextChanged
<pre>119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>	117	
<pre>119 End Sub ' txtFirstNumber_TextChanged 120 121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>	118	<pre>lblResult.Text = ""</pre>
<pre>121 ' handles TextChanged event 122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>	119	
<pre>122 Private Sub txtSecondNumber_TextChanged(ByVal sender _ 123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>	120	
<pre>123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>	121	' handles TextChanged event
<pre>123 As System.Object, ByVal e As System.EventArgs) _ 124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>	122	<pre>Private Sub txtSecondNumber_TextChanged(ByVal sender _</pre>
<pre>124 Handles txtSecondNumber.TextChanged 125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>	123	
<pre>125 126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>126 lblResult.Text = "" 127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
<pre>127 End Sub ' txtSecondNumber_TextChanged 128</pre>		
128		

$\supset$	What does this code do? 🕨	<b>32.14</b> What does the following code do, assuming that dblValue1 and dblValue2 are both declared as Doubles?		
		<pre>1 Try 2 3 dblValue1 = Convert.ToDouble(txtInput1.Text)</pre>		

```
4
                                          dblValue2 = Convert.ToDouble(txtInput2.Text)
                                   5
                                   6
                                          txtOutput.Text = (dblValue1 * dblValue2).ToString
                                   7
                                   8
                                       Catch formatExceptionParameter As FormatException
                                   9
                                  10
                                          MessageBox.Show( _
                                  11
                                              "Please enter decimal values.", _
                                  12
                                             "Invalid Number Format", _
                                  13
                                             MessageBoxButtons.OK, MessageBoxIcon.Error)
                                  14
                                  15
                                       End Try
                                 Answer: This code multiplies two Doubles if both inputs in txtInput1 and txtInput2 can
                                 be converted to type Double (that is, decimal or integer values). Otherwise it displays an
                                 error message dialog that informs the user to enter decimal values in the TextBoxes.
What's wrong with this code?
                                 32.15 The following code should add integers from two TextBoxes and display the result in
                                 txtResult. Assume that intValue1 and intValue2 are declared as Integers. Find the
                                 error(s) in the following code:
                                   1
                                       Try
                                   2
                                   3
                                          intValue1 = Convert.ToInt32(txtInput1.Text)
                                   4
                                          intValue2 = Convert.ToInt32(txtInput2.Text)
                                   5
                                   6
                                          txtOutput.Text = (intValue1 + intValue2).ToString
                                   7
                                   8
                                       End Try
                                   9
                                  10
                                       Catch formatExceptionParameter As FormatException
                                  11
                                  12
                                          "Please enter valid Integers.", _
                                  13
                                             "Invalid Number Format", _
                                  14
                                  15
                                             MessageBoxButtons.OK, MessageBoxIcon.Error)
                                  16
                                  17
                                       End Catch
```

**Answer:** The error in this code is that the Catch block appears after the End Try keywords. All Catch blocks must appear before these keywords. Also, there should be no End Catch following the Catch block.

```
1
    Try
2
 3
       intValue1 = Convert.ToInt32(txtInput1.Text)
 4
       intValue2 = Convert.ToInt32(txtInput2.Text)
 5
 6
       txtOutput.Text = (intValue1 + intValue2).ToString
 7
 8
       Catch formatExceptionParameter As FormatException
9
10
       MessageBox.Show( _
11
           "Please enter valid Integers.", _
12
          "Invalid Number Format", _
13
          MessageBoxButtons.OK, MessageBoxIcon.Error)
14
15
    End Try
```

Programming Challenge

**32.16** (Enhanced Vending Machine Application) The Vending Machine application from Tutorial 3 has been modified to use exception handling to process the IndexOutOfRangeExceptions that occur when selecting items out of the range 0 through 7 (Fig. 32.19). This type of exception will be defined shortly. To get a snack, the user must type the number of the desired snack in the TextBox, then press the **Dispense Snack**: Button. The name of the snack is displayed in the output Label.

	Vending Mach	ine			
	O Pretzels 4	1 cookie 5	Prefzels 2 5004 6	3 T	Please make a selection:          10         Dispense Snack    Array Index Dut of Bounds Error
]	Vending Mact	nine			Please enter a value between 0-7.
		1	Pretzels 2	S	Please make a selection: 5 Dispense Snack
	Pretzels	I COOKIE 5	5 5 6	3 (5111) 7	
[	Oat	meal Cookie h	as been dispense	ed	

Figure 32.19 Vending Machine application.

- a) *Copying the template to your working directory*. Copy the C:\Examples\Tutorial32\Exercises\EnhancedVendingMachine directory to your C:\SimplyVB directory.
- b) **Opening the application's template file.** Double click EnhancedVendingMachine.sln in the EnhancedVendingMachine directory to open the application.
- c) *Adding a Try block*. Find the btnDispense\_Click event handler. Enclose all of the code in the event handler in a Try block.
- d) Adding a Catch block. Add a Catch block that catches any FormatExceptions that may occur in the Try block that you added to btnDispense\_Click in Step c. Inside the Catch block, add code to display an error message dialog.
- e) Adding a second Catch block. Immediately following the Catch block you added in Step d, add a second Catch block to catch any IndexOutOfRangeExceptions that may occur. An IndexOutOfRangeException occurs when the application attempts to access an array with an invalid index. Inside the Catch block, add code to display an error message dialog.
- f) Running the application. Select Debug > Start to run your application. Make an out of range selection (for instance, 32) and click the Dispense Snack Button. Verify that the proper MessageBox is displayed for the invalid input. Enter letters for a

selection and click the **Dispense Snack** Button. Verify that the proper MessageBox is displayed for the invalid input.

- g) *Closing the application.* Close your running application by clicking its close box.
- h) Closing the IDE. Close Visual Studio .NET by clicking its close box.

Answers:

```
1
     ' Exercise 32.16 Solution
2
    ' VendingMachine.vb
 3
 4
    Public Class FrmVendingMachine
 5
         Inherits System.Windows.Forms.Form
 6
 7
        ' Windows Form Designer generated code
8
9
        Dim strSnacks As String() = New String() _
10
           {"Chocolate Chip Cookie", "Bubble Gum", _
           "Plain Pretzel", "Soda", "Salted Pretzel", _
"Oatmeal Cookie", "Diet Soda", "Sugar-free Gum"}
11
12
13
14
        ' method to dispense snack
15
        Private Sub btnDispense_Click(ByVal sender As _
16
           System.Object, ByVal e As System.EventArgs) _
17
           Handles btnDispense.Click
18
19
           ' try to get user input
20
           Try
21
22
              ' get user input
23
              Dim intSelection As Integer = _
24
                 Convert.ToInt32(txtSelection.Text)
25
26
              lblOutput.Text = strSnacks(intSelection).ToString & _
27
                 " has been dispensed"
28
29
           ' handle case when user enters invalid input
30
           Catch formatExceptionParameter As FormatException
31
32
              MessageBox.Show( _
33
                 "Please enter an integer value", _
34
                 "Number Format Exception", _
35
                 MessageBoxButtons.OK, MessageBoxIcon.Error)
36
37
           ' handle case when user inputs number not in array bounds
38
           Catch indexExceptionParamter As IndexOutOfRangeException
39
40
              MessageBox.Show( _
41
                 "Please enter a value between 0-7.", _
42
                 "Array Index Out of Bounds Error", _
43
                 MessageBoxButtons.OK, MessageBoxIcon.Error)
44
45
           End Try ' end Try...Catch statement
46
47
        End Sub ' btnDispense_Click
48
49
    End Class ' FrmVendingMachine
```