Preface

Visual Basic (VB) has become such a popular programming language for several reasons. VB is easy to learn, which makes it an excellent tool for understanding elementary programming concepts. In addition, it has evolved into such a powerful and popular product that skilled Visual Basic programmers are in demand in the job market.

Visual Basic is fully object-oriented and compatible with many other languages using the .NET Framework. This book incorporates object-oriented concepts throughout, as well as the syntax and terminology of the language.

Visual Basic is designed to allow the programmer to develop applications that run under Windows and/or in a Web browser without the complexity generally associated with programming. With very little effort, the programmer can design a screen that holds standard elements such as buttons, check boxes, radio buttons, text boxes, and list boxes. Each of these objects operates as expected, producing a "standard" Windows or Web user interface.

About This Text

This textbook is intended for use in an introductory programming course, which assumes no prior knowledge of computer programming. The later chapters are also appropriate for professional programmers who are learning a new language to upgrade their skills.

This text assumes that the student is familiar with the Windows operating environment and can use an Internet browser application.

Approach

This text incorporates the basic concepts of programming, problem solving and programming logic, as well as the design techniques of an object-oriented, event-driven language. VB is a fully object-oriented language, which includes inheritance and polymorphism. Object-oriented programming (OOP) is introduced in Chapter 1, and its features appear in every chapter of the book.

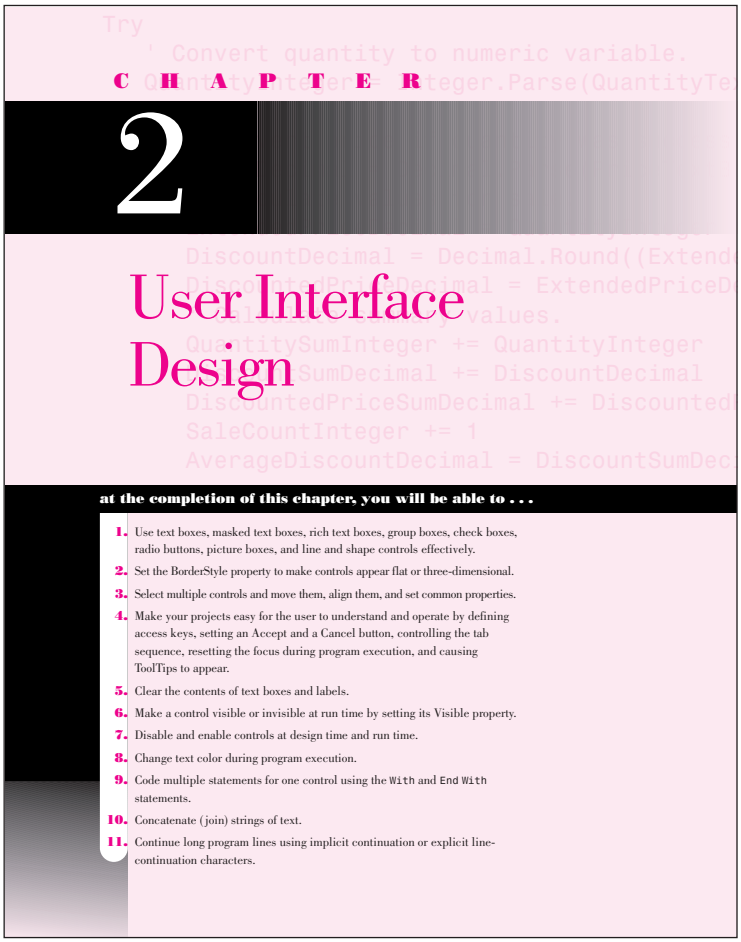
Chapter topics are presented in a sequence that allows the programmer to learn how to deal with a visual interface while acquiring important programming skills such as creating projects with objects, decisions, loops, and data management.

A high priority is given to writing applications that are easy for the user to understand and to use. Students are presented with interface design guidelines throughout the text.

Text Features

Object-Oriented Concepts

are presented throughout the text to offer students an introduction to object-oriented design before learning to create their own classes.



CHAPTER 2

User Interface Design

at the completion of this chapter, you will be able to . . .

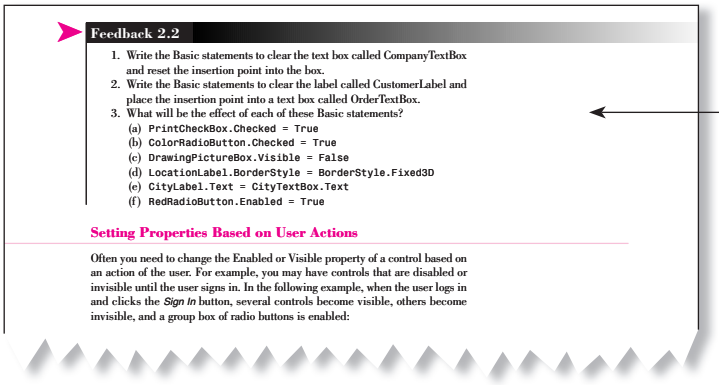
1. Use text boxes, masked text boxes, rich text boxes, group boxes, check boxes, radio buttons, picture boxes, and line and shape controls effectively.
2. Set the BorderStyle property to make controls appear flat or three-dimensional.
3. Select multiple controls and move them, align them, and set common properties.
4. Make your projects easy for the user to understand and operate by defining access keys, setting an Accept and a Cancel button, controlling the tab sequence, resetting the focus during program execution, and causing ToolTips to appear.
5. Clear the contents of text boxes and labels.
6. Make a control visible or invisible at run time by setting its Visible property.
7. Disable and enable controls at design time and run time.
8. Change text color during program execution.
9. Code multiple statements for one control using the `With` and `End With` statements.
10. Concatenate (join) strings of text.
11. Continue long program lines using implicit continuation or explicit line-continuation characters.

Good Programming Habits

1. Always test the tab order on your forms. Fix it if necessary by changing the TabIndex properties of the controls.
2. Provide visual separation for input fields and output fields and always make it clear to the user which are which.
3. Make sure that your forms can be navigated and entered from the keyboard. Always set an Accept button (AcceptButton property) for every form.
4. To make a label maintain its size regardless of the value of the Text property, set AutoSize to False.
5. To make the text in a text box right justified or centered, set the TextAlign property.
6. You can use the Checked property of a check box to set other properties that must be True or False.

Interface Guidelines

are presented to offer students a better understanding of meeting user needs and employing industry standards.



Feedback 2.2

1. Write the Basic statements to clear the text box called CompanyTextBox and reset the insertion point into the box.
2. Write the Basic statements to clear the label called CustomerLabel and place the insertion point into a text box called OrderTextBox.
3. What will be the effect of each of these Basic statements?
   (a) `PrintCheckBox.Checked = True`
   (b) `ColorRadioButton.Checked = True`
   (c) `DrawingPictureBox.Visible = False`
   (d) `LocationLabel.BorderStyle = BorderStyle.Fixed3D`
   (e) `CityLabel.Text = CityTextBox.Text`
   (f) `RedRadioButton.Enabled = True`

Setting Properties Based on User Actions

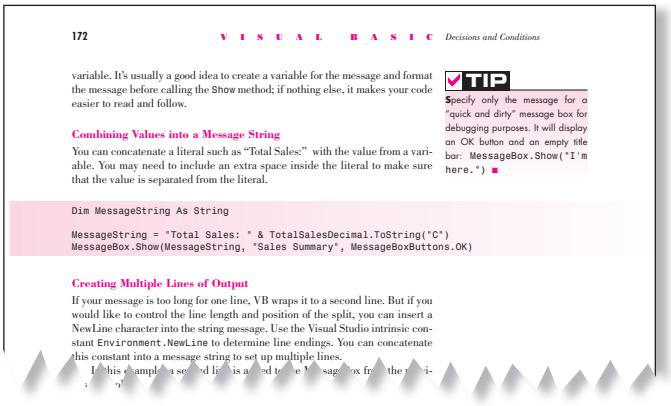
Often you need to change the Enabled or Visible property of a control based on an action of the user. For example, you may have controls that are disabled or invisible until the user signs in. In the following example, when the user logs in and clicks the *Sign In* button, several controls become visible, others become invisible, and a group box of radio buttons is enabled:

Feedback Questions

give students time to reflect on the current topic and to evaluate their understanding of details.

Tips

in the margins help students avoid potential trouble spots in their programs and encourage them to develop good programming habits.



variable. It's usually a good idea to create a variable for the message and format the message before calling the `Show` method; if nothing else, it makes your code easier to read and follow.

TIP

Specify only the message for a "quick and dirty" message box for debugging purposes. It will display an OK button and an empty title bar: `MessageBox.Show("I'm here.")` ■

Combining Values into a Message String

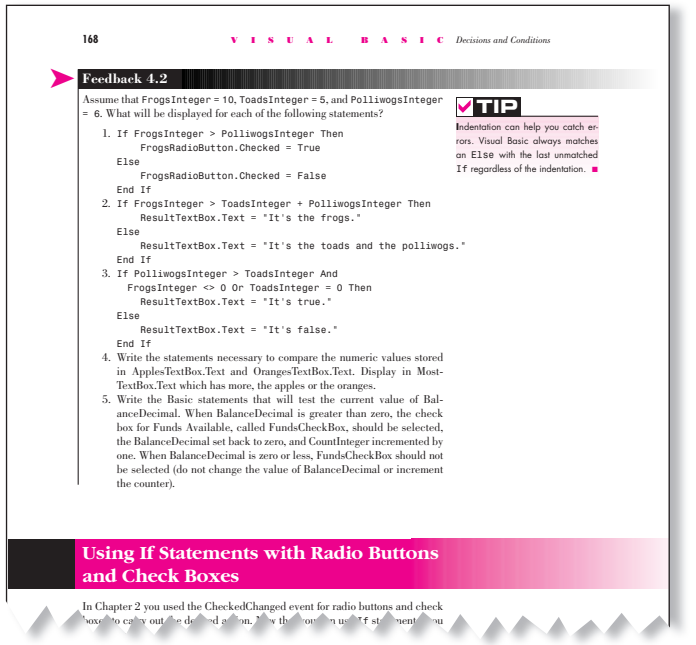
You can concatenate a literal such as "Total Sales:" with the value from a variable. You may need to include an extra space inside the literal to make sure that the value is separated from the literal.

```
Dim MessageString As String

MessageString = "Total Sales: " & TotalSalesDecimal.ToString("C")
MessageBox.Show(MessageString, "Sales Summary", MessageBoxButtons.OK)
```

Creating Multiple Lines of Output

If your message is too long for one line, VB wraps it to a second line. But if you would like to control the line length and position of the split, you can insert a NewLine character into the string message. Use the Visual Studio intrinsic constant `Environment.NewLine` to determine line endings. You can concatenate this constant into a message string to set up multiple lines.



Feedback 4.2

Assume that `FrogsInteger` = 10, `ToadsInteger` = 5, and `PolliwogsInteger` = 6. What will be displayed for each of the following statements?

1.
```
If FrogsInteger > PolliwogsInteger Then
    FrogsRadioButton.Checked = True
Else
    FrogsRadioButton.Checked = False
End If
```
2.
```
If FrogsInteger > ToadsInteger + PolliwogsInteger Then
    ResultTextBox.Text = "It's the frogs."
Else
    ResultTextBox.Text = "It's the toads and the polliwogs."
End If
```
3.
```
If PolliwogsInteger > ToadsInteger And
  FrogsInteger <> 0 Or ToadsInteger = 0 Then
    ResultTextBox.Text = "It's true."
Else
    ResultTextBox.Text = "It's false."
End If
```
4. Write the statements necessary to compare the numeric values stored in ApplesTextBox.Text and OrangesTextBox.Text. Display in MostTextBox.Text which has more, the apples or the oranges.
5. Write the Basic statements that will test the current value of BalanceDecimal. When BalanceDecimal is greater than zero, the check box for Funds Available, called FundsCheckBox, should be selected, the BalanceDecimal set back to zero, and CountInteger incremented by one. When BalanceDecimal is zero or less, FundsCheckBox should not be selected (do not change the value of BalanceDecimal or increment the counter).

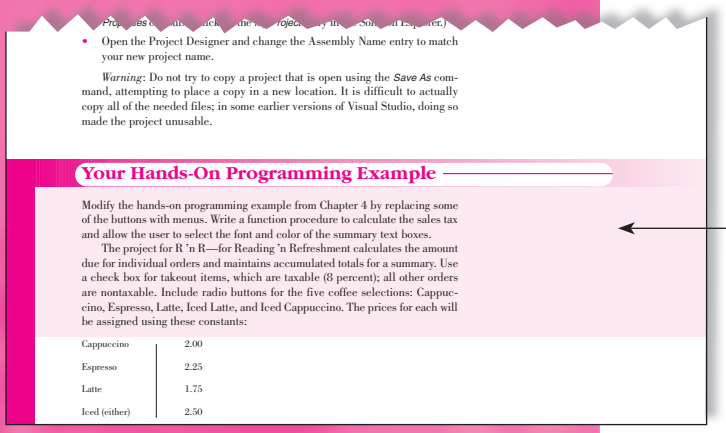
TIP

Indentation can help you catch errors. Visual Basic always matches an `Else` with the last unmatched `If` regardless of the indentation. ■

Using If Statements with Radio Buttons and Check Boxes

In Chapter 2 you used the CheckedChanged event for radio buttons and check boxes to carry out the desired action. Now that you can use `If` statements, you

Text Features



- Open the Project Designer and change the Assembly Name entry to match your new project name.

Warning: Do not try to copy a project that is open using the *Save As* command, attempting to place a copy in a new location. It is difficult to actually copy all of the needed files; in some earlier versions of Visual Studio, doing so made the project unusable.

Your Hands-On Programming Example

Modify the hands-on programming example from Chapter 4 by replacing some of the buttons with menus. Write a function procedure to calculate the sales tax and allow the user to select the font and color of the summary text boxes.

The project for R 'n R—for Reading 'n Refreshment calculates the amount due for individual orders and maintains accumulated totals for a summary. Use a check box for takeout items, which are taxable (8 percent); all other orders are nontaxable. Include radio buttons for the five coffee selections: Cappuccino, Espresso, Latte, Iced Latte, and Iced Cappuccino. The prices for each will be assigned using these constants:

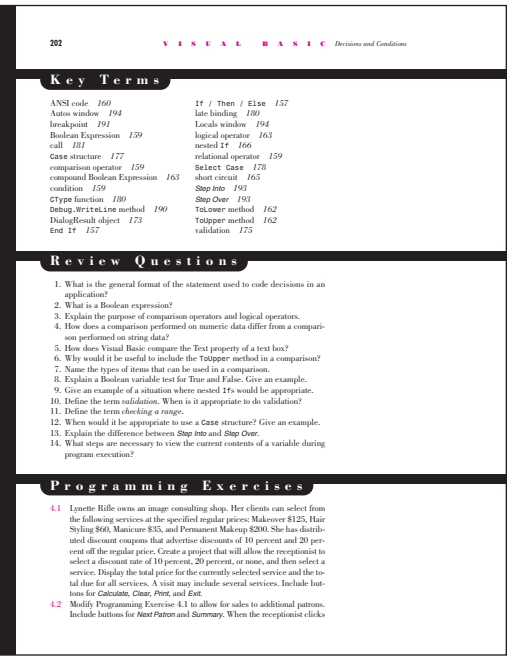
Cappuccino	2.00
Espresso	2.25
Latte	1.75
Iced (either)	2.50

Your Hands-On Programming Examples

guide students through the process of planning, writing, and executing Visual Basic programs.

Programming Exercises

test students' understanding of the specific programming skills covered in that chapter.



202 VISUAL BASIC *Decisions and Conditions*

Key Terms

ANSI code *160*
Autos window *194*
breakpoint *191*
Boolean Expression *159*
call *181*
`Case` structure *177*
comparison operator *159*
compound Boolean Expression *163*
condition *159*
`CType` function *180*
`Debug.WriteLine` method *190*
DialogResult object *173*
`End If` *157*
`If / Then / Else` *157*
late binding *180*
Locals window *194*
logical operator *163*
nested `If` *166*
relational operator *159*
`Select Case` *178*
short circuit *165*
Step Into *193*
Step Over *193*
`ToLower` method *162*
`ToUpper` method *162*
validation *175*

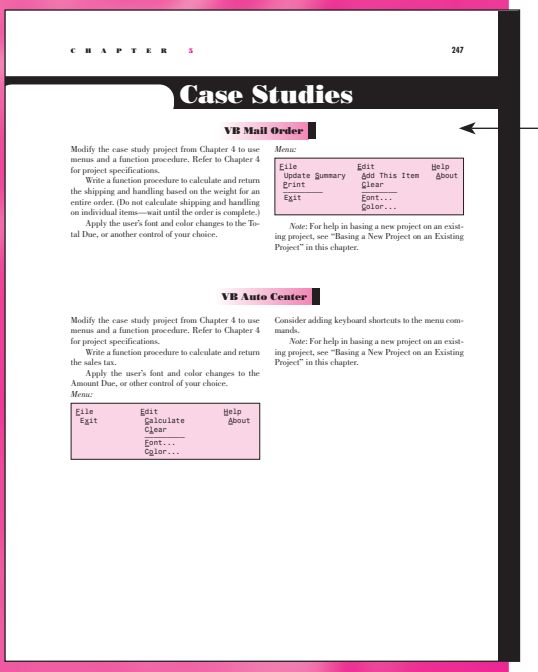
Review Questions

1. What is the general format of the statement used to code decisions in an application?
2. What is a Boolean expression?
3. Explain the purpose of comparison operators and logical operators.
4. How does a comparison performed on numeric data differ from a comparison performed on string data?
5. How does Visual Basic compare the Text property of a text box?
6. Why would it be useful to include the `ToUpper` method in a comparison?
7. Name the types of items that can be used in a comparison.
8. Explain a Boolean variable test for True and False. Give an example.
9. Give an example of a situation where nested `If`s would be appropriate.
10. Define the term *validation*. When is it appropriate to do validation?
11. Define the term *checking a range*.
12. When would it be appropriate to use a `Case` structure? Give an example.
13. Explain the difference between *Step Into* and *Step Over*.
14. What steps are necessary to view the current contents of a variable during program execution?

Programming Exercises

4.1 Lynette Rifle owns an image consulting shop. Her clients can select from the following services at the specified regular prices: Makeover $125, Hair Styling $60, Manicure $35, and Permanent Makeup $200. She has distributed discount coupons that advertise discounts of 10 percent and 20 percent off the regular price. Create a project that will allow the receptionist to select a discount rate of 10 percent, 20 percent, or none, and then select a service. Display the total price for the currently selected service and the total due for all services. A visit may include several services. Include buttons for *Calculate*, *Clear*, *Print*, and *Exit*.

4.2 Modify Programming Exercise 4.1 to allow for sales to additional patrons. Include buttons for *Next Patron* and *Summary*. When the receptionist clicks



CHAPTER 5 247

Case Studies

VB Mail Order

Modify the case study project from Chapter 4 to use menus and a function procedure. Refer to Chapter 4 for project specifications.

Write a function procedure to calculate and return the shipping and handling based on the weight for an entire order. (Do not calculate shipping and handling on individual items—wait until the order is complete.)

Apply the user's font and color changes to the Total Due, or another control of your choice.

Menu:

File: Update Summary, Print, Exit | Edit: Add This Item, Clear, Font..., Color... | Help: About

Note: For help in basing a new project on an existing project, see "Basing a New Project on an Existing Project" in this chapter.

VB Auto Center

Modify the case study project from Chapter 4 to use menus and a function procedure. Refer to Chapter 4 for project specifications.

Write a function procedure to calculate and return the sales tax.

Apply the user's font and color changes to the Amount Due, or other control of your choice.

Consider adding keyboard shortcuts to the menu commands.

Note: For help in basing a new project on an existing project, see "Basing a New Project on an Existing Project" in this chapter.

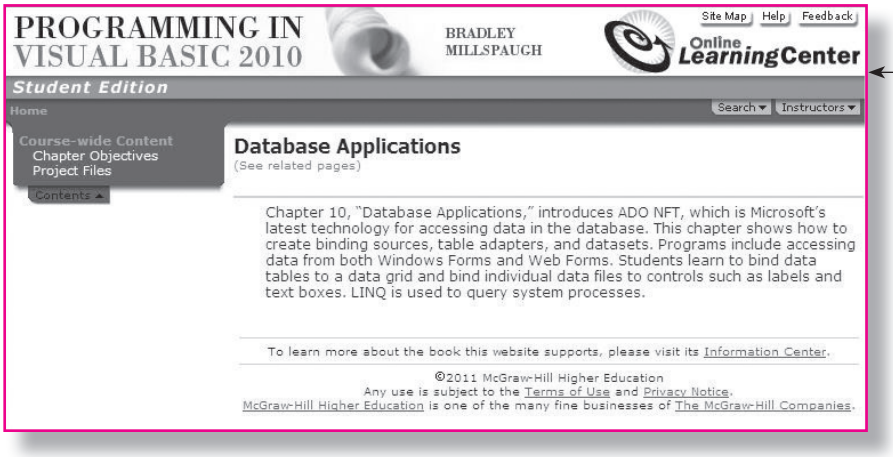
Menu:

File: Exit | Edit: Calculate, Clear, Font..., Color... | Help: About

Case Studies

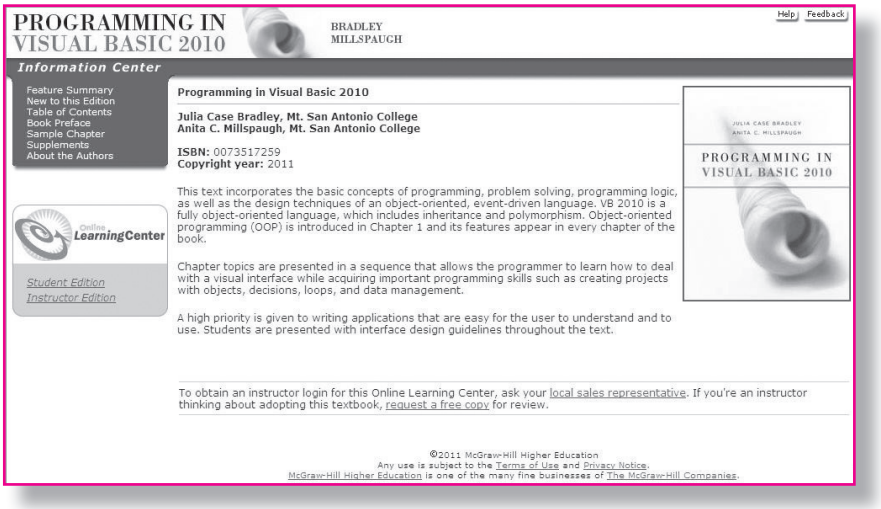
provide continuing-theme exercises that may be used throughout the course, providing many opportunities to expand on previous projects.

For the Student



Student Data

available on the text's Web site offers a debugging project, database files for the programming exercises and case studies, graphics, and sound files.

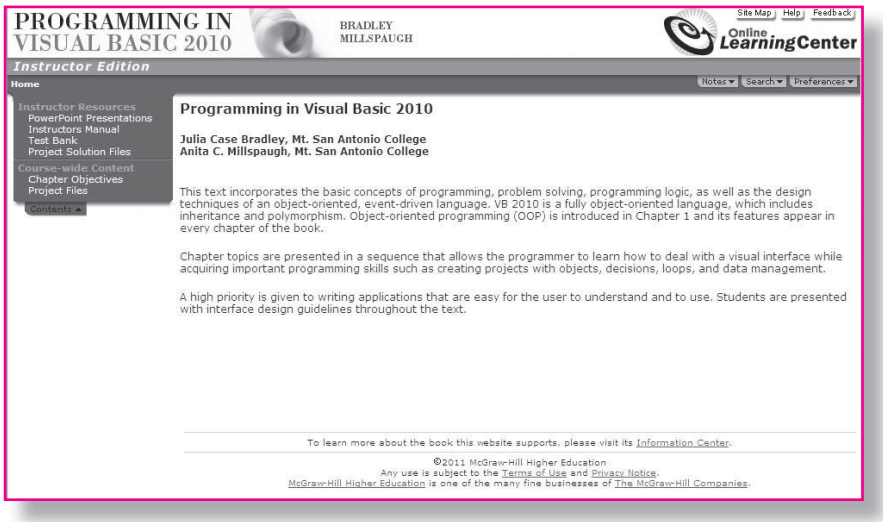


Visit the **VISUAL BASIC 2010** Web site at: http://www.mhhe.com/VB2010/ for instructor and student resources.

For the Instructor

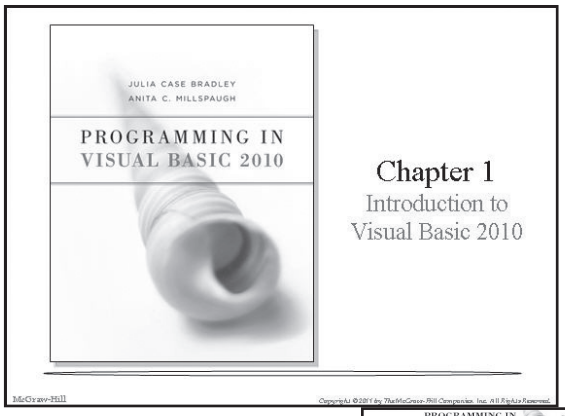
Instructor Web Site

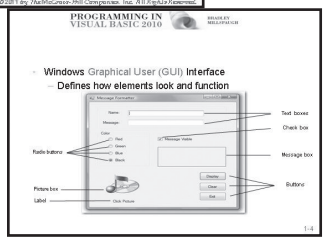
includes: Instructor's Manual with teaching hints, outlines, and a matrix of the chapter features required for each programming exercise; PowerPoint Slides; Testing Files (using EZ Test and in Word files); as well as Solutions to End-of-Chapter Exercises.

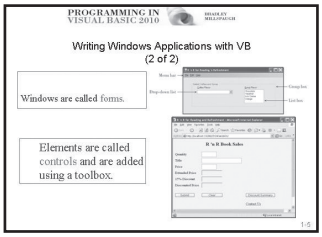


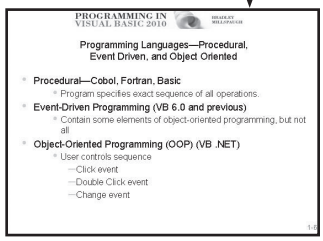
PowerPoint Presentations

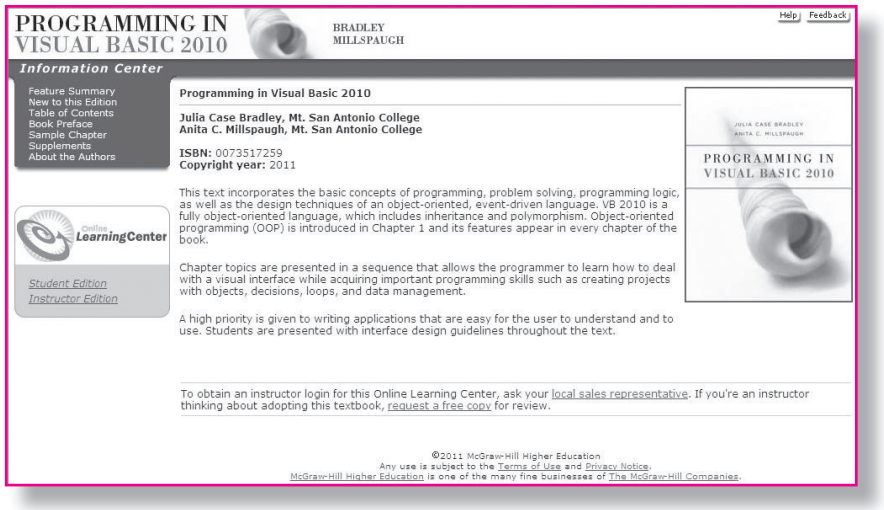
provide instructors with complete, detailed presentations that walk students through the important concepts covered in each chapter.











Visit the **VISUAL BASIC 2010** Web site at: http://www.mhhe.com/VB2010/ for instructor and student resources.

Changes in This Edition

This revision of the text is based on Visual Basic Professional 2010. VB 2010 provides for elimination of the line continuation character under most circumstances. The array and object initializers also have been improved.

The narrative, step-by-step exercises, screen captures, and appendices have all been updated to VB 2010. The screen captures are all based on Windows 7. A section covering collection objects has been added to the chapter on arrays.

Features of This Text

Each chapter begins with identifiable objectives and a brief overview. Numerous coding examples as well as hands-on projects with guidance for the planning and coding appear throughout. Thought-provoking feedback questions give students time to reflect on the current topic and to evaluate their understanding of the details. The end-of-chapter items include a chapter review, questions, programming exercises, and four case studies.

Chapter 1, "Introduction to Visual Basic 2010," introduces Microsoft's Visual Studio integrated development environment (IDE). The single environment is used for multiple programming languages. A step-by-step program gets students into programming very quickly (quicker than most books). The PrintForm control is included to allow students to easily submit screen captures of the form at run time. The chapter introduces the OOP concepts of objects, properties, methods, and events. The elements of debugging and using the Help system are also introduced.

Chapter 2, "User Interface Design," demonstrates techniques for good program design, including making the interface easy for users as well as guidelines for designing maintainable programs. Several controls are introduced, including text boxes, rich text boxes, masked text boxes, group boxes, check boxes, radio buttons, picture boxes, and the new Shape and Line controls.

Chapter 3, "Variables, Constants, and Calculations," presents the concepts of using data and declaring the data type. Students learn to follow standards to indicate the data type and scope of variables and constants and always to use Option Strict, which forces adherence to strong data typing.

Error handling is accomplished using structured exception handling. The `Try/Catch/Finally` structure is introduced in this chapter along with calculations. The student learns to display error messages using the MessageBox class and also learns about the OOP concept of overloaded constructors.

Chapter 4, "Decisions and Conditions," introduces taking alternate actions based on expressions formed with the relational and logical operators. This chapter uses the `If` statement to validate input data. Multiple decisions are handled with both nested `If` statements and the `Select Case` structure.

The debugging features of the IDE are covered, including a step-by-step exercise that covers stepping through program statements and checking intermediate values during execution.

Chapter 5, "Menus, Common Dialog Boxes, Sub Procedures, and Function Procedures," covers the concepts of writing and calling general sub procedures and function procedures. Students learn to include both menus and context menus in projects, display the Windows common dialog boxes, and use the input provided by the user.

Chapter 6, "Multiform Projects," adds splash forms and About forms to a project. Summary data are presented on a separate form. The `Friend` keyword is introduced.

Chapter 7, "Lists, Loops, and Printing," incorporates list boxes and combo boxes into projects, providing the opportunity to discuss looping procedures and printing lists of information. Printing is accomplished in .NET using a graphics object and a callback event. The printing controls also include a Print Preview, which allows students and instructors to view output without actually printing it.

Chapter 8, "Arrays and Collections," introduces arrays, which follow logically from the lists covered in Chapter 7. Students learn to use single- and multidimension arrays, table lookups, arrays of structures, and collections.

Chapter 9, "Web Applications," introduces Web applications using Web Forms. Students learn to design and develop simple Web applications that consist of Web pages that execute in a browser application. Multiple-page Web sites are covered along with validator controls and an introduction to state management.

Chapter 10, "Database Applications," introduces ADO.NET, which is Microsoft's latest technology for accessing data in a database. This chapter shows how to create binding sources, table adapters, and datasets. Programs include accessing data from both Windows Forms and Web Forms. Students learn to bind data tables to a data grid and bind individual data fields to controls such as labels and text boxes. LINQ is used to query system processes.

Chapter 11, "Data Files," presents the VB object-oriented techniques for data file handling. Students learn to save and read small amounts of data using the My object and using streams. The StreamWriter and StreamReader objects are used to store and reload the contents of a combo box.

Chapter 12, "OOP: Creating Object-Oriented Programs," explains more of the theory of object-oriented programming. Although we have been using OOP concepts since Chapter 1, in this chapter students learn the terminology and application of OOP. Inheritance is covered for visual objects (forms) and for extending existing classes. The samples are kept simple enough for an introductory class.

Chapter 13, "Graphics, Animation, Sound, and Drag-and-Drop," covers the classes and methods of GDI+. The chapter covers graphics objects, pens, and brushes for drawing shapes and lines. Animation is accomplished using the Timer control and the `SetBounds` method for moving controls. My.Computer.Audio.Play is used to provide sound, and drag-and-drop events are used to transfer the contents of a text box to a list box and to move images.

***Chapter 14*, "*Additional Topics in Visual Basic*,"** introduces some advanced VB topics. This final chapter covers validating user input using Error Providers and the Validating event of controls. Students learn to create applications using multiple document interfaces (MDI), create toolbars and status bars using ToolStrip and StatusStrip controls, and add Web content to a Windows Form using the WebBrowser control. The code snippet feature is introduced. Reading and writing XML text files is covered using the new XML literals and using LINQ.

An introduction to Windows Presentation Foundation (WPF) includes using WPF interoperability with a standard Windows Form and creating a WPF Form project.

The appendices offer important additional material. Appendix A holds the answers to all Feedback questions. Appendix B covers methods and functions for math, string handling, and date manipulation. In OOP style, most actions that were formerly done with functions are now accomplished with methods of the Math class and String class.

Appendix C, on mastering the Visual Studio environment, is based on the .NET IDE and includes instructions for using snap lines for form design. Appendix D discusses security issues for both Windows and Web programming.

Acknowledgments

Many people have worked very hard to design and produce this text. We would like to thank our editors, Scott Davidson and Alaina Grayson. Our thanks also to the many people who produced this text, including Marlena Pechan, David Shapiro, and Betsy Blumenthal.

We greatly appreciate Theresa Berry of Mt. San Antonio College and Peter van der Goes of Rose State College for their thorough technical reviews, constructive criticism, and many valuable suggestions. We would like to thank Brenda Nielsen of Mesa Community College for her work in creating the PowerPoint Presentations that accompany this text and to Theresa Berry for the Instructor's Manual and Test Bank. And most importantly, we are grateful to Dennis and Richard for their support and understanding through the long days and busy phone lines.

The Authors

We have had fun teaching and writing about Visual Basic. We hope that this feeling is evident as you read this book and that you will enjoy learning or teaching this outstanding programming language.

Julia Case Bradley
Anita C. Millspaugh