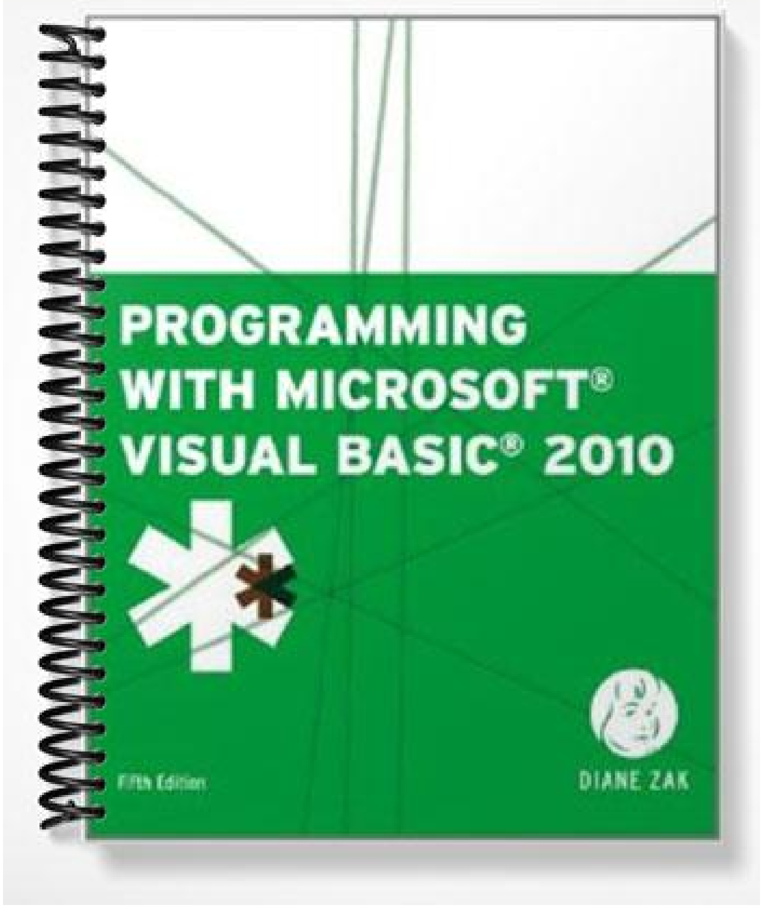
SOLUTIONS MANUAL



Chapter 2

Designing Applications

At a Glance

Instructor's Manual Table of Contents

- Overview
- Objectives
- Teaching Tips
- Quick Quizzes
- Class Discussion Topics
- Additional Projects
- Additional Resources
- Key Terms

Lecture Notes

Overview

Chapter 2 introduces students to designing applications in Visual Basic 2010. Students learn how to plan for an application, design the interface, and write the code. This chapter contains three lessons. Lesson A covers the planning of an object-oriented application using Task, Object, and Event charts. Lesson B introduces the design of the user interface. Lesson C covers the coding, testing, debugging, and documenting of the application.

Lesson A Objectives

After studying Lesson A, students should be able to:

- Plan an object-oriented application in Visual Basic 2010
- Complete a TOE (Task, Object, Event) chart
- Follow the Windows standards regarding the layout and labeling of controls

Teaching Tips

Previewing the Playtime Cellular Application

1. Describe the Playtime Cellular Order screen shown in Figure 2-1.

2. Describe the sample program output shown in Figure 2-2.

Teaching Tip	Explain that students will learn the benefits to be gained by planning the application before coding, using tools such as flowcharts and pseudocode.

Teaching Tip	Run the completed application to demonstrate how this program will work.

Creating an Object-Oriented Application

1. Use Figure 2-3 to compare the process of building a home and building an application.

2. Introduce the concept of program bugs.

Planning an Object-Oriented Application

1. Stress the importance of ensuring that a completed application meets the user's needs.

2. Point out that the user must be involved in the planning process for this to occur.

3. Use Figure 2-4 to list the four steps involved in planning an OO application.

4. Introduce the concept of a TOE chart.

Identifying the Application's Tasks

1. Describe the process of identifying required tasks for the application. Point out how the user's existing order form, shown in Figure 2-5, can be used as a starting point for identifying the required tasks.

2. Point out that both the necessary input information and the desired output information are identified during this phase of development, using Figure 2-6.

3. Use Figure 2-6 to describe the contents and format of a TOE chart, and point out that only the tasks have been identified during the consultation with the user.

Identifying the Objects

1. Describe the process of assigning objects in the user interface to the tasks that have been identified.

2. Point out that some objects will not have an association with a task, such as descriptive labels. Stress the importance of identifying any labels that are not descriptive labels, such as those used to display the results of calculations. The latter type will be associated with tasks.

3. Discuss the object assignments listed for the previously identified tasks shown in Figure 2-7.

Teaching Tip	Point out that the TOE chart provides a form of documentation for the entire project, as well as a plan for the programmer when writing code.

Identifying the Events

1. Remind students that the built-in behavior of text boxes allows them to accept and display information, so no events are required for these tasks.

2. Use Figure 2-8 to discuss the completed TOE chart. Point out that it is ordered by task.

Teaching Tip	Ensure that students understand why some labels have tasks and events associated with them while others do not.

3. Discuss the advantages of arranging the TOE chart by object for large applications.

Teaching Tip	Remind students that for very large applications, there will be many programmers involved.

4. Compare and contrast the completed TOE chart ordered by object in Figure 2-9 with the previous one ordered by task.

Drawing a Sketch of the User Interface

1. Point out that although the TOE chart describes the objects required in the visual interface, it does not describe the layout of the interface.

2. Describe the Windows standards for both vertical and horizontal layouts of a screen using Figures 2-10 and 2-11.

Teaching Tip	Studies have shown that applications that follow the Windows standards for layout and object behavior are perceived by users to be easier to learn.

3. Describe the use of white space or container objects to provide a visual grouping of related controls.

Teaching Tip	A good screen layout should “lead” the user by making it clear how items are related and how the screen should be used.

4. Introduce sentence capitalization and book title capitalization. Discuss when to use each in the user interface. Stress the importance of meaningful captions for buttons and descriptive labels.

5. Summarize the GUI Design tips that have been introduced, and point out that there is a complete list of GUI guidelines in Appendix A of the book.

Lesson A Summary

1. Summarize the six steps involved in creating an OO application.

2. Summarize the four steps involved in planning an OO application in Visual Basic 2010.

3. Describe the questions that should be asked when identifying the required tasks for an application.

Quick Quiz 1

1. Which of the three components of a TOE chart must be identified first?
   Answer: tasks

2. True or False: All objects in the GUI interface will be associated with events.
   Answer: False

3. True or False: TOE charts are only useful for developing small applications.
   Answer: False

Lesson B Objectives

After studying Lesson B, students should be able to:
- Build the user interface using their TOE chart and sketch
- Follow the Windows standards regarding the use of graphics, fonts, and color
- Set a control's BorderStyle property
- Add a text box to a form
- Lock the controls on the form
- Assign access keys to controls
- Set the TabIndex property

Teaching Tips

Building the User Interface

1. Describe the partially created user interface shown in Figure 2-12. Point out the resemblance to the sketch created in Lesson A.

2. Point out that the goal in designing a user interface is to keep the interface from distracting the user.

Including Graphics in the User Interface

1. Point out that graphics on a form have a tendency to draw the eyes to them.

Teaching Tip	Ask students to consider the effect of attention-grabbing screen elements such as graphics or animation when the user must interact with the application for several hours per day.

2. Point out that a small graphic in the upper-left corner helps to bring the user's attention to the area of the screen where you want the user to start.

Selecting Fonts for the Interface

1. Introduce the concept of a serif on a font. Describe when to use serif and sans serifs fonts.

Teaching Tip	It is important to also consider the needs of users with visual impairments when designing the user interface. Microsoft provides guidelines for meeting visual accessibility needs.

2. Point out that underlining and italics should be avoided on a user interface, and describe the standard font sizes that should be used.

3. Summarize the GUI design guidelines for font types, styles, and sizes shown in the GUI Design Tip box.

Teaching Tip	The user's standard screen resolution should also be considered to ensure that the entire screen will be viewable.

Adding Color to the Interface

1. Point out that color can also be distracting to the user.

2. Discuss the three points that should be considered when including color, and point out that color should never be the only means of identifying an important element on the screen.

Teaching Tip	Point out that Windows offers color schemes for visually impaired users that may remove most colors.

3. Summarize the guidelines for the use of color shown in the GUI Design Tip box.

The BorderStyle and AutoSize Properties

1. Introduce the BorderStyle property of GUI controls, such as buttons, text boxes, and labels.
2. Point out that descriptive labels should have no border, while labels used to display output should have the BorderStyle property set to FixedSingle.
3. Note that the Fixed3D BorderStyle is the default for text boxes.
4. Describe the use of a label control's AutoSize property for identifying labels and for labels used to display program output.
5. Summarize the guidelines for the use of the BorderStyle property shown in the GUI Design Tip box.
6. Summarize the guidelines for the use of the AutoSize property shown in the GUI Design Tip box.

Adding a Text Box Control to the Form

1. Introduce the text box control and the TextBox tool in the toolbox.
2. Describe the process of adding a text box to the form. Remind students to align controls according to guidelines, as shown in Figure 2-13.

Teaching Tip	Use Figure 2-13 to point out the careful use of graphics, colors, and fonts, and ask students if they feel that an improvement has been made to the user interface.

Locking the Controls on a Form

1. Introduce the concept of locking controls on a form to "freeze" the design when it has been completed.

Teaching Tip	Accidentally locking the controls on a form is an error occasionally made by students new to Visual Basic. Remind them to check this setting if they are unable to relocate controls in design mode.

2. Describe the process of locking controls on the form.

Teaching Tip	It is a good idea to frequently remind students of the importance of periodically saving their work.

Assigning Access Keys

1. Introduce the concept of an access key.

2. Discuss the three reasons for assigning access keys.

Teaching Tip	Remind students that access keys are another Windows standard that users expect to have.

3. Explain how to assign an access key to a button.

Teaching Tip	Alert students that depending on their system's settings, access keys may not appear underlined unless a Control Panel change is made.

4. Summarize the guidelines for the use of access keys shown in the GUI Design Tip box.

Controlling the Tab Order

1. Introduce the TabIndex property.

2. Introduce the concept of focus. Point out that not all types of controls can accept user input.

3. Point out that the TabIndex property is set by default based on the order in which controls were added to the form at design time.

Teaching Tip	Remind students that the user interface should support users who want to use only the keyboard. Such users should not have to use access keys to navigate through the screen.

4. Review the TabIndex settings required for the sample application shown in Figure 2-14.

5. Use Figure 2-15 to introduce the Tab Order option that can be used as an alternate method for setting the TabIndex property.

6. Summarize the guidelines for assigning access keys and controlling the focus shown in the GUI Design Tip box.

7. Describe how to test that the tab order has been set correctly.

Lesson B Summary

1. Summarize the use of the BorderStyle property.

2. Summarize the use of the AutoSize property.

3. Remind students of the ability to lock/unlock controls on the form.

4. Summarize the process of assigning access keys to controls.

5. Review the process of giving users keyboard access to a text box.

6. Remind students how to use an access key at run time.

7. Summarize the process of setting the tab order.

8. Summarize the process of displaying access keys in Windows 7.

Quick Quiz 2

1. Which property controls the appearance of a control's border?
   Answer: BorderStyle

2. True or False: Color should be generously used in the user interface.
   Answer: False

3. True or False: An access key is assigned by using the percent sign (%) in front of a character.
   Answer: False

Lesson C Objectives

After studying Lesson C, students should be able to:

- Code an application using its TOE chart
- Plan an object's code using pseudocode or a flowchart
- Write an assignment statement
- Send the focus to a control during run time
- Include internal documentation in the code
- Write arithmetic expressions
- Use the Val and Format functions
- Locate and correct syntax errors

Teaching Tips

Coding the Application

1. Remind students that the objects and events that need to be coded have already been identified in the application's TOE chart.

2. Use Figure 2-18 to review the sample application's user interface

3. Use Figure 2-19 to review the application's TOE chart.

Planning a Procedure Using Pseudocode

1. Introduce the concept of pseudocode.

2. Discuss the pseudocode for the sample application shown in Figure 2-20.

Teaching Tip	Point out that the pseudocode for a control describes how to accomplish the tasks that were associated with the control in the TOE chart.

Using a Flowchart to Plan a Procedure

1. Introduce the concept of a flowchart.

2. Introduce the elements of a flowchart: the start/stop symbol, the process symbol, the input/output symbol, and flowlines.

3. Use Figure 2-21 to describe the flowchart for the sample application.

Teaching Tip	Point out that it is not necessary to create both a flowchart and the pseudocode for an application. It is a matter of personal preference which one you use.

Coding the btnClear Control's Click Event Procedure

1. Review the requirements for the Clear Screen button in the TOE chart.

2. Introduce the concept of a string.

3. Introduce the concept of an empty string (or zero-length string).

4. Describe how to clear the contents of a text box using an empty string, the String.Empty value.

Assigning a Value to a Property During Run Time

1. Introduce the assignment statement for assigning a value to something.

2. Introduce the assignment operator.

Teaching Tip	Ensure that students understand that the value of the expression on the right side of the assignment operator will be stored in the variable or property shown on the left side of the assignment operator. Students new to programming often mistakenly believe that, like algebra, expressions can appear on both sides of an equal sign.

3. Use Figure 2-22 to describe the IntelliSense feature that can be used to speed the process of coding a program.

Teaching Tip	If possible, have students actually enter code as you develop the application to give them experience using IntelliSense. Urge them to type the first letter of the desired item and then use the arrow keys to make selections in the IntelliSense lists, as this will be quicker than using the mouse.

4. Point out that Visual Studio corrects capitalization when you use keywords and variable names that have already been declared.

5. Describe the rest of the statements that are required for the Clear button’s Click event.

Using the Focus Method

1. Introduce the Focus method.

Internally Documenting the Program Code

1. Introduce program comments as a type of internal documentation.

2. Describe the process of creating comments using Figure 2-23.

3. Describe the use of comments in the General Declarations section using Figure 2-24.

Teaching Tip	Remind students that as job responsibilities change, application maintenance may pass to another programmer. Providing documentation allows the other programmer to quickly understand your code.

Writing Arithmetic Expressions

1. Introduce the concept of precedence numbers. Point out that the use of parentheses will override the precedence of operators.

2. Use Figure 2-25 to describe the order of precedence of the commonly used arithmetic operators.

3. Explain how processing occurs when two operations have the same level of precedence.

4. Introduce unary and binary operators. Explain the difference between negation and the subtraction operation.

5. Introduce the integer division operator. Compare and contrast an example using both the integer division operator and the standard division operator.

6. Introduce the modulus arithmetic operator.

7. Describe the expressions and results shown in Figures 2-26 and 2-27.

Teaching Tip	Students new to programming will find the modulus operation to be very strange. Describe how to use this operator to determine if a number is odd or even to demonstrate its usefulness.

Coding the Calculate Order Button

1. Review the pseudocode written for the btnCalc control of the sample application, shown in Figure 2-28.

2. Use Figures 2-29 and 2-30 to discuss how the pseudocode directly relates to the code that is required.

3. Describe the process of adding the required code.

4. Use Figure 2-31 to demonstrate that string concatenation occurred instead of arithmetic. Explain that it is necessary to convert the values entered into the text boxes before performing arithmetic operations.

Teaching Tip	Not converting values entered into text boxes prior to performing arithmetic operations is a common error experienced by many new programming students. Time invested in addressing this error will be well spent.

The Val Function

1. Introduce the concept of a function.

2. Introduce the Val function and explain its purpose.

3. Point out that the string to be converted must contain only numeric characters, spaces, or periods and cannot contain any special characters such as dollar signs or percent signs.

4. Use Figure 2-32 to explain that the Val function stops converting as soon as an unacceptable character is found.

5. Describe how to use the Val function in the Calculate button's Click event code using Figure 2-33.

The Format Function

1. Introduce the Format function and describe its purpose.

Teaching Tip	Ensure that students understand that the actual value in the property or variable is not changed. The use of the Format function can be compared to the use of stage makeup to change appearance.

2. Discuss the commonly used format styles shown in Figure 2-34.

3. Describe the process of using the Format function in the sample application code shown in Figure 2-35.

Testing and Debugging the Application

1. Introduce the concept of valid and invalid data.

2. Discuss the importance of using both valid and invalid data when testing an application prior to its release to users.

3. Introduce the concepts of debugging, syntax errors, and logic errors.

Teaching Tip	Point out that logic errors involve syntactically correct statements. Remind students of the importance of flowcharts or pseudocode in helping to avoid logic errors.

4. Describe the process of testing and debugging the sample application.

5. Use Figure 2-37 to describe the correct output.

6. Use Figures 2-38 through 2-40 to describe how the IDE flags syntax errors and assists in the process of correcting them.

Assembling the Documentation

1. Explain the meaning of assembling the documentation.

2. Describe how to print the application's code.

3. Describe how to print the application's user interface.

Lesson C Summary

1. Review the use of pseudocode and flowcharts in planning an object's code.

2. Remind students how to clear a text box.

3. Summarize the structure of an assignment statement.

4. Review how to change the focus at run time.

5. Summarize the creation and use of comments in code.

6. Summarize the use of arithmetic operators, including integer division and the modulus operator.

7. Summarize how to convert string values to numbers and how to format numeric values.

8. Remind students how to print an application's code and interface.

Quick Quiz 3

1. What symbol is used for integer division?
   Answer: backslash (\)

2. The ____________________ function can be used to display a numeric value with currency formatting.
   Answer: Format

3. True or False: The SetFocus method will move the focus to a control at run time.
   Answer: False

4. True or False: In an arithmetic expression, multiplication will take place before addition.
   Answer: True

Class Discussion Topics

1. Why is it important to clear text boxes and move the focus at run time? Can you think of an application you have used recently that does not do this for you? What was your impression of that application?

2. Many companies use formal procedures for application development that include functional specifications and system specifications. Discuss what each type of specification entails. Ask students to determine when users should be involved and how detailed the flowcharts or pseudocode should be. What do you think will happen to those documents when the application is completed?

Additional Projects

1. Ask students to do some research and write a brief report summarizing the Microsoft guidelines for user interface design.

2. Ask students to design the GUI and code for a program that will allow a user to enter three test scores. The program should then calculate the test average and display it.

Additional Resources

1. Microsoft Windows GUI design guidelines:
   http://msdn.microsoft.com/en-us/library/aa511440.aspx

2. User interface design:
   www.ambysoft.com/essays/userInterfaceDesign.html

3. Object-oriented programming:
   www.aonaware.com/OOP1.htm

Key Terms

- **Access key**—the underlined character in an object’s identifying label or caption; allows the user to select the object using the Alt key in combination with the underlined character
- **Assignment operator**—the equal sign in an assignment statement
- **Assignment statement**—an instruction that assigns a value to something, such as to the property of an object

- **Book title capitalization—**the capitalization used for a button’s caption; refers to capitalizing the first letter in each word, except for articles, conjunctions, and prepositions that do not occur at either the beginning or end of the caption
- **Bugs—**the errors in a program
- **Debugging—**the process of locating and correcting the bugs (errors) in a program
- **Empty string—**a set of quotation marks with nothing between them (""); also called a zero-length string
- **Flowchart—**a planning tool that uses standardized symbols to show the steps a procedure must take to accomplish its goal
- **Flowlines—**the lines connecting the symbols in a flowchart
- **Focus method—**moves the focus to a specified control during run time
- **Focus—**indicates that a control is ready to accept user input
- **Format function—**used to improve the appearance of numbers in an interface
- **Function—**a procedure that processes a specific task and returns a value
- **General Declarations section—**the area above the Public Class clause in the Code Editor window
- **Input/output symbol—**the parallelogram in a flowchart; used to represent input and output tasks
- **Integer division operator—**represented by a backslash (\); divides two integers and then returns the quotient as an integer
- **Invalid data—**data that an application is not expecting the user to enter
- **Logic error—**occurs when you neglect to enter an instruction or enter the instructions in the wrong order; also occurs as a result of calculation statements that are correct syntactically but incorrect mathematically
- **Modulus operator—**represented by the keyword Mod; divides two numbers and returns the remainder of the division
- **Process symbols—**the rectangle symbols in a flowchart; used to represent assignment and calculation tasks
- **Pseudocode—**a planning tool that uses phrases to describe the steps a procedure must take to accomplish its goal
- **Sentence capitalization—**the capitalization used for identifying labels; refers to capitalizing only the first letter in the first word and in any words that are customarily capitalized
- **Start/stop symbol—**the oval symbol in a flowchart; used to indicate the beginning and end of the flowchart
- **String.Empty—**the value that represents the empty string in Visual Basic
- **String—**zero or more characters enclosed in quotation marks
- **Syntax error—**occurs when an instruction in an application’s code breaks one of a programming language’s rules
- **Text box—**a control that provides an area in the form for the user to enter data
- **Val function—**temporarily converts a string to a number and then returns the number
- **Valid data—**data that an application is expecting the user to enter
- **Zero-length string—**a set of quotation marks with nothing between them (""); also called an empty string