

**V**isual Basic .NET is used to create applications for Microsoft Windows. It includes tools that allow a programmer to create an application that has features similar to other Windows applications without having to write many lines of code. The concepts explained in this chapter include procedures, assignment, and expressions. The Form, Label, MainMenu, RadioButton, and Button controls are also introduced.

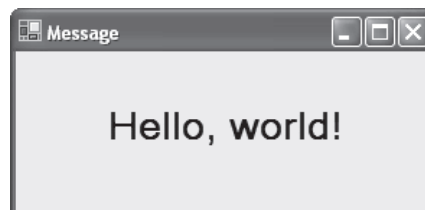
## 3.1 Visual Basic .NET Windows Programming

*application*  
*Windows Forms*

Visual Basic .NET is an object-oriented programming (OOP) language that is used to create Windows, Web, and command-line (console) programs, also called *applications*. A Windows application is based on *Windows Forms*, a framework that uses objects such as forms, buttons, text boxes, and so on to interact with a user.

*program code*  
*interface*

A Visual Basic .NET Windows application consists of a graphical interface and related program code. The *program code* is the instructions that determine how the application will behave. The *interface* is for the user to interact with an application. For example, the interface below is for a Visual Basic .NET application called Message:



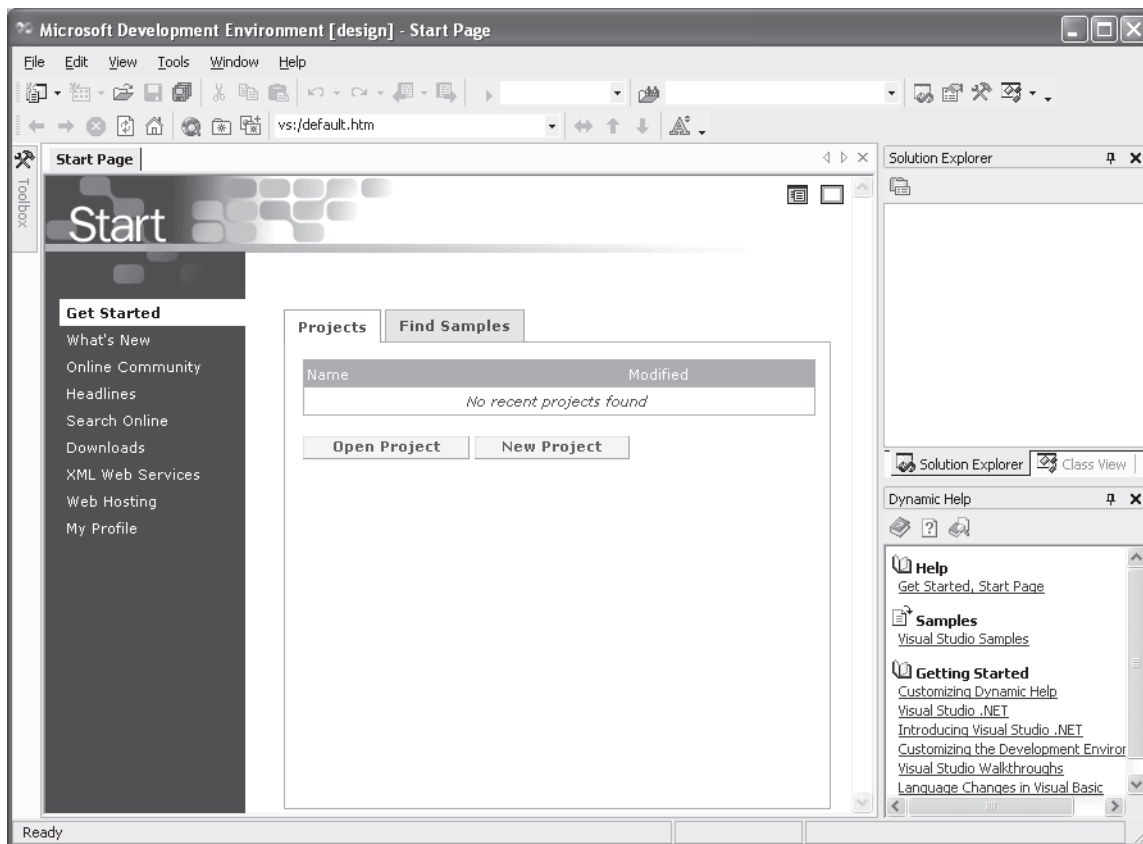
Message is a window that contains a label with the text "Hello, world!"

*event*  
*event-driven application*

A Visual Basic .NET application is event-driven like other Windows applications. An *event* is a way in which the user can interact with an object, such as clicking a button. An *event-driven application* waits for an event to occur before executing any code, then only code for the current event is executed. For example, the Message application above remains on the screen until the Close button in the upper-right corner of the window is clicked.

## 3.2 The Visual Basic .NET IDE

Visual Basic .NET applications can be created in an *integrated development environment*, or *IDE*. When the IDE is started, the Start Page is shown:




### Starting Visual Studio .NET

Visual Basic .NET is part of the Visual Studio package. Visual Studio can be started by using Programs in the Start menu.

- The **menu bar** contains the names of drop-down menus that contain commands.
- The **toolbars** contain buttons that represent different actions.
- The Open Project button is used to open an existing application.
- The New Project button is used to create a new application.

## 3.3 Creating a New Project

Developing an application requires several related files. In Visual Basic .NET these files are collectively maintained as a *project*. To create a new Windows project, select New Project on the Start Page or click the New Project button (  ) on the toolbar. The New Project dialog box is displayed:



New Project

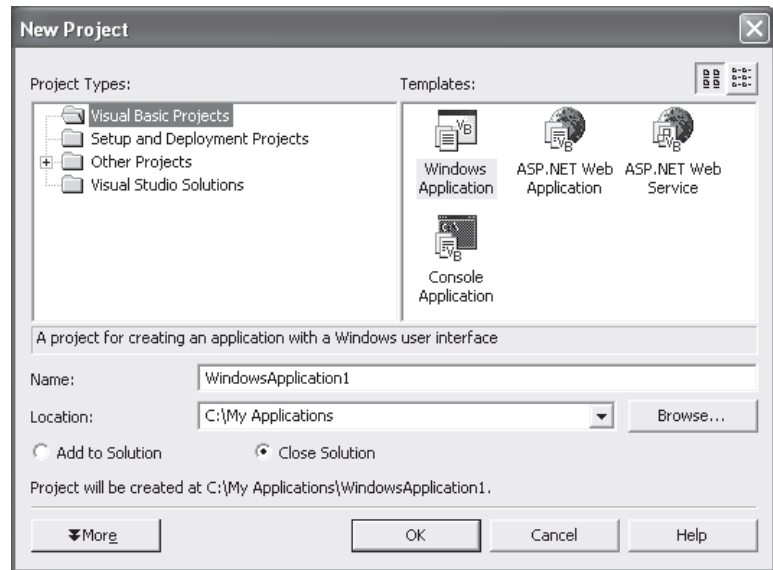


## The Project Command

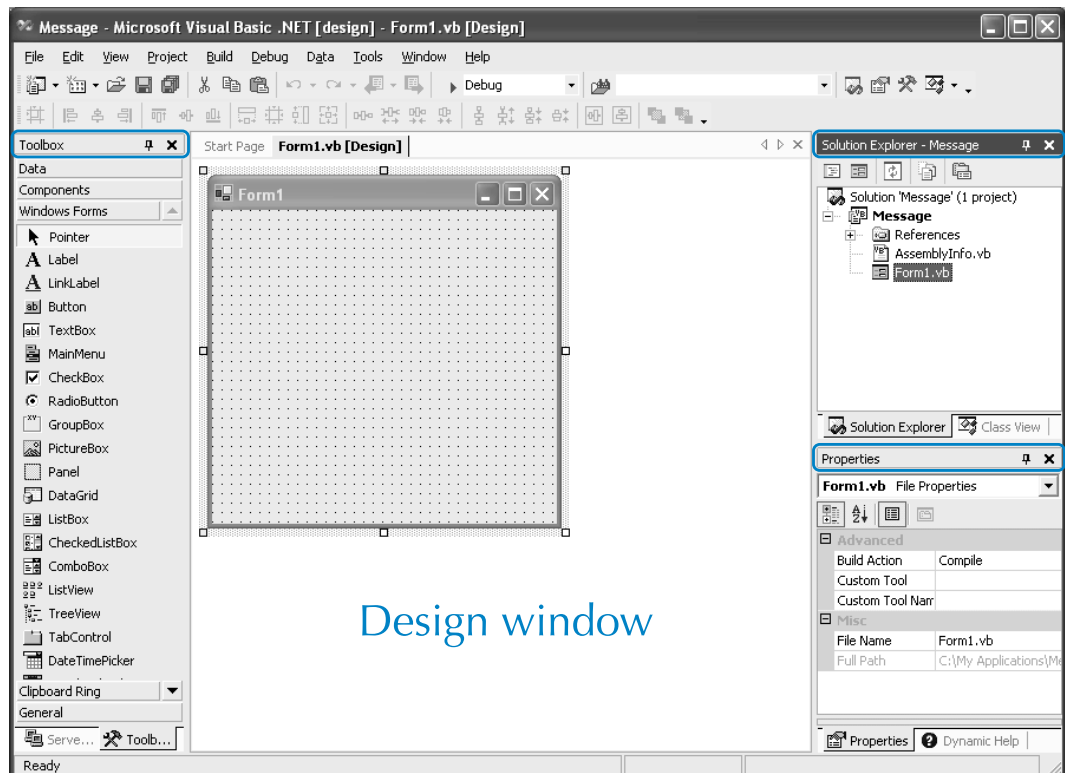
Select **File** → **New** → **Project** to display the New Project dialog box.

## Displaying the Toolbox

Select **View** → **Toolbox** or click the Toolbox button (  ) on the Toolbar to display the Toolbox. If the Toolbox is docked to the left of the Design window, point to the Toolbox and then click the AutoHide button (  ) to keep it displayed.



Select Visual Basic Projects in the Projects Types list and click the Windows Application icon in the Templates list. In the Name box, type a descriptive application name. Type the project location in the Location box or select Browse to choose the location of the project from a dialog box. Select OK. The new project is displayed in the *Design window*:



- The **Design window** displays the application interface where objects are added, deleted, and sized.
- The **Toolbox** contains controls that are used to create objects.
- The **Solution Explorer window** is used to switch between the Design and Code windows.
- The **Properties window** lists the properties of a selected object.

## Review 1

In this review you will create a project named Message.

### ① START VISUAL STUDIO .NET

### ② CREATE A NEW PROJECT

- a. On the toolbar, click the New Project button (📁). The New Project dialog box is displayed.
  1. In the Projects Type list, select Visual Basic Projects, if it is not already selected.
  2. In the Templates list, click the Windows Application icon if it is not already selected.
  3. In the Name box, replace the existing text with Message.
  4. In the Location box, type the appropriate project location path or click Browse to choose the appropriate folder.
  5. Select OK. A project is created and a form is displayed.
- b. Note the Toolbox with controls for creating objects. Note also the Solution Explorer window and the Properties window.

## 3.4 The Windows Form

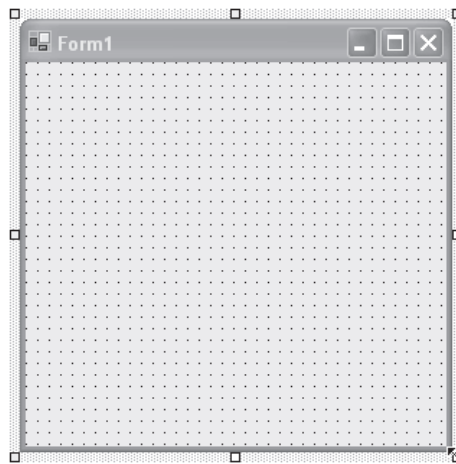
A Windows application interface uses at least one form. A form is a graphical object that contains a title bar, a system menu (☰), and Minimize (⏏), Maximize (⏏), and Close (✕) buttons.

### sizing a form

A form object is *selected* by clicking it, which displays handles. Once selected, dragging a handle sizes the form. Dragging a corner handle sizes both height and width together. Note the ↖ pointer:

### Designing an Interface

Interface design becomes an important step in developing an application of any complexity. Using pencil and paper and field testing different interfaces are usually part of the design process.

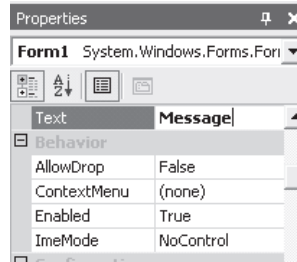


### properties

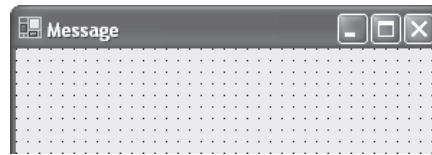
A form object has *properties* that define its appearance, behavior, position, and other attributes. Property values are displayed in the Properties window. For example, the Text property defines what is displayed in the title bar.

### Text property

A property value is changed by typing or selecting a new value. For example, selecting the form, clicking the Text property in the Properties window, and then typing Message displays the following:



Pressing Enter or clicking outside the Properties window applies the new Text property value to the form:

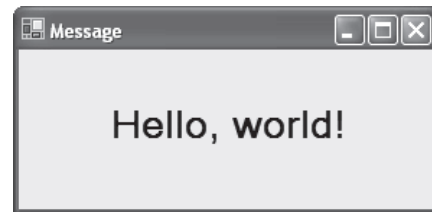


### 3.5 The Label Control

**Control Objects**

An *object* is a unit of code and data that is described by other code called a *class*. One class can be used to create many objects. Visual Basic .NET provides many built-in classes called *controls* for creating graphical objects, such as a Label, that can be used in an application interface.

A control is used to add objects to a form. *Control objects* display information or get user input. An application interface typically contains many control objects on a form. For example, the *Label control* displays text that cannot be changed by the user. The Message application interface uses a label to display Hello, world!:



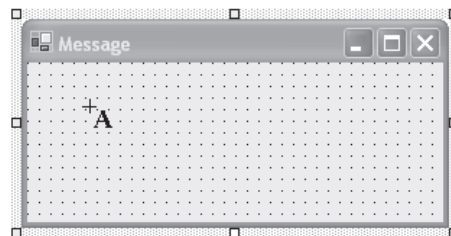
The Message form contains a Label object



An object is added to a form by clicking a control in the Toolbox and then pointing to the form, which changes the pointer shape to one that displays an icon similar to the selected control. For example, clicking the Label control in the Toolbox and then pointing to the form displays:

**Double-Clicking**

A control object can be added to a form by double-clicking the control in the Toolbox.



Clicking the form adds a new control object. The grid on the form can be used for precise placement of objects.

### Marquee Selection


Multiple control objects can be selected together using a technique called marquee selection. Clicking Pointer in the Toolbox and then dragging on the form displays a selection box called a marquee. Any objects within the boundaries of the marquee are selected.

### The Name Property

The Name property is displayed as (Name) in the Properties window.

A control object is moved by dragging it (not a handle) to a new location on the form. Multiple objects can be moved together by selecting them as a group and then dragging one of the selected objects. This technique is useful when multiple objects need to be moved while maintaining their relative position. Pressing the Ctrl key when clicking an object selects it in addition to already selected objects.

The Label control has the properties:

- **Name** identifies a control for the programmer. It is good programming style to begin Label object names with lbl.
- **Text** is the text displayed in the label.
- **Font** contains the  button that is clicked to display the Font dialog box where the font name, font style, and font size of label text is selected.
- **TextAlign** is the alignment of text in a label. Alignment can be set to TopLeft (the default), TopCenter, TopRight, MiddleLeft, MiddleCenter, MiddleRight, BottomLeft, BottomCenter, and BottomRight.

Properties are set using the Properties window, just as Form properties were set. For example, the properties of the Message label are set to lblMessage for Name, Hello, world! for Text, bold size 20 for Font, and MiddleCenter for TextAlign:




### Name property


For most objects, the Name property should always be set to a descriptive name. A proper object name begins with the appropriate prefix and then describes the purpose of the control. For example, the label in the Message application is lblMessage. This name begins with the appropriate prefix (lbl) and is descriptive of the label's purpose. Following proper naming conventions is good programming style.

### programming style

## 3.6 Saving and Running an Application



An application should be saved periodically to avoid losing changes. To save a project, select **S**ave from the **F**ile menu or click the Save Project button () on the toolbar.



To run an application, click the Start button () on the toolbar or select **S**tart (F5) from the **D**ebug menu. The IDE remains on the screen, but the project application becomes the active window. The form no longer displays a grid and controls cannot be modified. However, controls can be tested to see if events such as clicking a button, produce the desired effect. A Visual Basic .NET application can be run at any time during development. This allows testing of the application as it is being written.





To terminate a running application, click the Stop Debugging button (  ) on the toolbar or select Stop Debugging from the Debug menu. The Close button (  ) in the upper-right corner of the window can also be clicked to close the program.

### **compile**

Running an application means that program code is first converted to a language the computer understands in a process called *compiling*. During compilation, an output window is displayed with messages about the program. The output window can be closed by clicking the Close button in the upper-right corner of the window after the program has run.

## Review 2



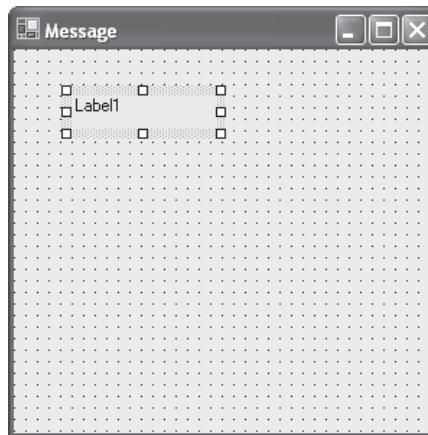
In this review you will add a Label object to the Message form and then run the application. The Message project should still be open from Review 1.

### ① SET THE FORM PROPERTIES

- Click the Form to select it.
- Scroll the Properties window to display the Text property in the Appearance section.
- Click Text, type Message, and press Enter. The title bar now displays “Message.”


### ② ADD A LABEL OBJECT TO THE FORM

- In the Toolbox, click the Label control.
- Click the grid of the form. A Label object is added to the form:



### ③ SET THE LABEL PROPERTIES

- Click the Label object to select it. Handles are displayed.
- Scroll the Properties window to display the Name property in the Design section.
- Click Name and enter lblMessage.
- Scroll the Properties window to display the Text property in the Appearance section.
- Click Text and enter Hello, world! The Label object now displays “Hello, world!” Note that the text is displayed in the upper-left corner of the object.
- Just below Text, click TextAlign and then click the TextAlign arrow. A group of icons is displayed.
  - Click the icon in the middle center. The TextAlign property is set to MiddleCenter. The label text is displayed in the center of the Label control.
- Scroll the Properties window to display the Font property in the Appearance section.

- h. Click Font and then click the  button. A dialog box is displayed.
  1. In the Font style list, select Bold.
  2. In the Size list, select 20.
  3. Select OK. The label text is displayed larger and may not be entirely shown because of the Label control size.


#### ④ SIZE THE OBJECTS

- a. Select the Label object.
- b. Drag the corner handle to size the label so that the text is completely displayed on one line.
- c. Select the form. Size the form so that it is a smaller rectangular shape.
- d. Drag the label so that it is in the center of the form.

Check – Your application interface should look similar to:



#### ⑥ SAVE AND RUN MESSAGE

- a. Select File → Save All. The Message application is saved using the location specified at the time the application was created.
- b. On the toolbar, click the Start button (). An Output window is displayed and Message is started. The Message application displays text.
- c. Click the Close button in the application window. The application is closed and the Visual Studio .NET IDE is again active.

## 3.7 The MainMenu Control

### Choosing Command Names

Command names should be short and descriptive. Grouping command names, creating smart menus, and other menu options are advanced topics that can be explored using online help.

A Windows application typically includes menus that contain commands. For example, the Message application can be modified to include a Program menu with an Exit command:



*Message with a Program menu and an Exit command*

 MainMenu

A menu is added to an interface by clicking the MainMenu control in the Toolbox and then clicking the form. A menu is automatically added to the upper-left corner of a form and the MainMenu component is shown in the component tray at the bottom of the Design window. Typing a menu name displays boxes for typing a command and another menu:

### The Menu Property

If a menu is not showing up on a form, the Menu property of a form can be used to assign the menu to the form.



The menu name Program has been typed

When a command or menu name is typed, another box is automatically added to allow for more commands and menus. Only the command and menu names actually typed will be included in the menu. Each menu and command name typed is a MenuItem object with the properties:

- **Name** identifies an object for the programmer. It is good programming style to begin MenuItem object names with mnu.
- **Text** is the menu or command name and is set by typing the appropriate name in a MainMenu box in the Design window.

### editing MenuItem objects

A menu is modified in the Design window by clicking a MenuItem object and typing a new name. A MenuItem is deleted by right-clicking the item and selecting Delete from the menu. A new MenuItem is inserted above an existing one by right-clicking the existing item and selecting Insert New from the menu.

## 3.8 Closing and Opening a Project and Quitting Visual Basic .NET

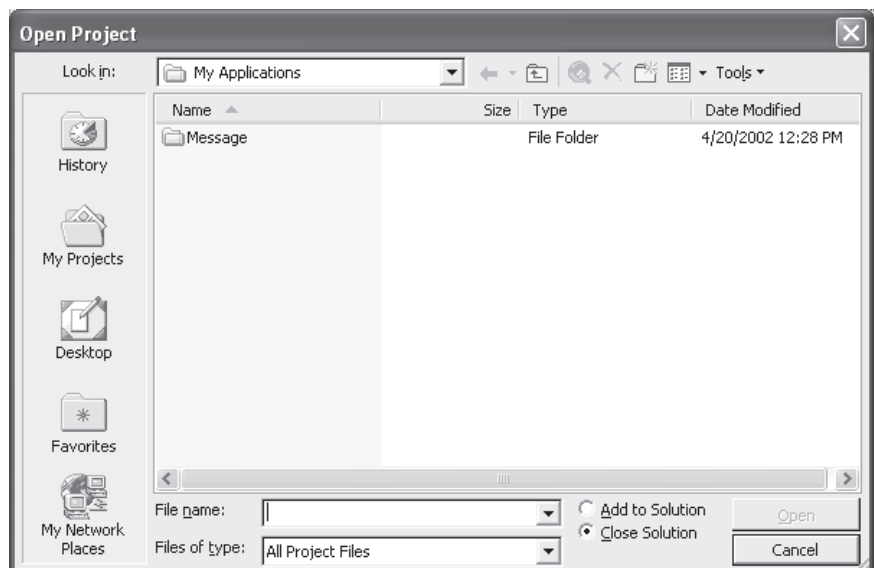
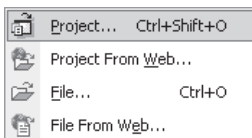
### closing a project

### quitting Visual Basic .NET

### opening a project

When finished working with a project, it should be saved and closed. To close a project, select Close Solution from the File menu. A warning dialog box is displayed if the project has been modified since it was last saved. Closing a project removes it from the IDE. To quit Visual Basic .NET, select Exit from the File menu.

To open a project, select Project from the Open submenu in the File menu. The Open Project dialog box is displayed:



The Look in list displays a location. The folders at that location are displayed in the contents box below the Look in list. Double-click a folder in the contents box to place that folder name in the Look in list and display that folder's contents. When the project folder is displayed in the Look in list, click the project name in the contents box and then select Open to open the project.

After opening a project, it may be necessary to double-click the form name in the Solution Explorer window to display the form in the Design window.

## Review 3



In this review you will add a Program menu with an Exit command to the Message application. Open the Message project in Visual Studio .NET and display the application interface if it is not already displayed.


### ① ADD A MENU TO THE FORM

- a. In the Toolbox, click the MainMenu control.
- b. Click the grid of the form. A menu is added to the upper-left corner of the form and a MainMenu component is displayed in the component tray.

### ② ADD A MENU NAME AND A COMMAND

- a. In the menu, click Type Here. The text is selected.
- b. Type Program and then press Enter. Additional MainMenu boxes are displayed.
- c. Click Program to select it.
- d. Scroll the Properties window to display the Name property and then enter mnuProgram.
- e. Click the MenuItem object below the Program menu and then type Exit and press Enter.
- f. Click Exit to select it and then change its Name property to mnuExit.

### ③ RUN MESSAGE

- a. Save the modified Message application.
- b. On the toolbar, click the Start button () . Note the Program menu.
- c. Click the Program menu and select Exit. Nothing happens because code has not been written for this part of the interface.
- d. Close the application.

## 3.9 Program Code

An application contains a set of instructions called *program code* that tells the computer how to perform a specific task. Each line of code is called a *statement*. Programs can contain tens to hundreds, even millions of lines of program code, or statements.

In a Visual Basic .NET project, some code is automatically generated for a form and the objects on the form. This code is added to initialize the objects and set property values.

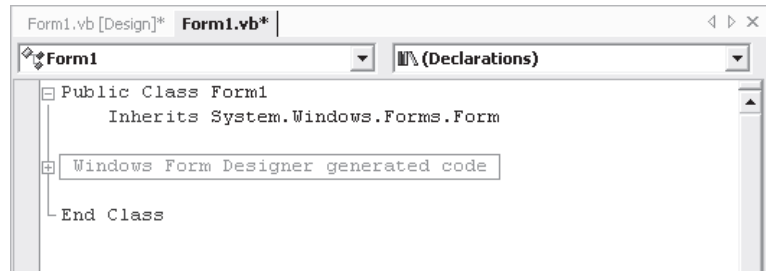


### Code window

#### Displaying the Solution Explorer Window

Click the Solution Explorer button (📁) on the toolbar or select **View** → **Solution Explorer** to display the Solution Explorer window.

The Code window is displayed by clicking the View Code button (📄) in the Solution Explorer window or selecting **C**ode from the **V**iew menu. The Code window for a new application looks similar to:



Icons are used in Code view to indicate if a block of code is expanded to show all its statements (-) or if the block is hidden (+). The statements for the Windows Form Designer generated code are hidden because this code should not be modified from the Code window.

### Inherits

The code that generates a form is a class. The **Inherits** statement indicates that the class contains, or has “inherited”, all the attributes of the Windows Form class. Statements can be added below the **Inherits** statement to extend the Windows Form class and add functionality to an application.

### switching between Design and Code windows



The buttons in the Solution Explorer window can be used to switch between Design and Code windows. The tabs in the IDE can also be used. The Design window is indicated by [Design] in the tab.

## 3.10 The Event Procedure

### procedure

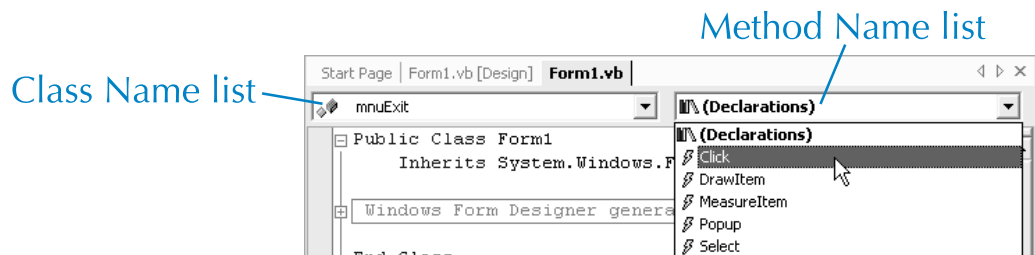
A *procedure* is a block of code written to perform specific tasks. Most control objects require an *event procedure*, also called an *event handler*, that responds to a user event. Event procedures add functionality to an application. For example, when the user clicks a menu command, specific actions should occur.

### event handler

### click event procedure

A *click event procedure* executes in response to a mouse click. For example, clicking a command executes the click event procedure for that MenuItem object. In the Message application, an Exit click event procedure should quit the application. An event procedure is added to the Code window by selecting the control name in the Class Name list and then the event from the Method Name list,

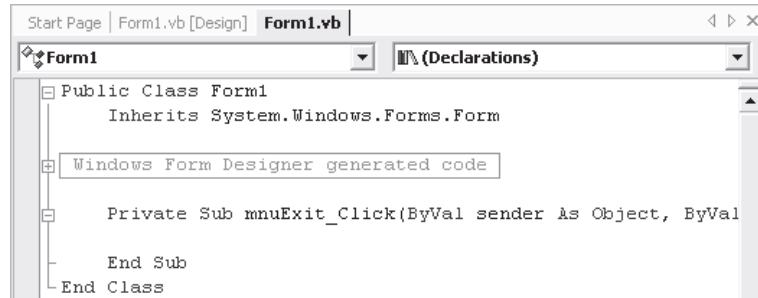
### adding an event procedure



## Naming Objects

Event procedures require object names as part of the procedure name. For example, `mnuExit_Click()` is the name of the click event procedure for the Exit command. Therefore, objects must be appropriately named before creating any event procedures. Visual Basic .NET will not automatically change an object's name in an event procedure if its Name property is set after the code is created.

which adds the event procedure to the class:



```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Windows Form Designer generated code
    Private Sub mnuExit_Click(ByVal sender As Object, ByVal
    End Sub
End Class
```

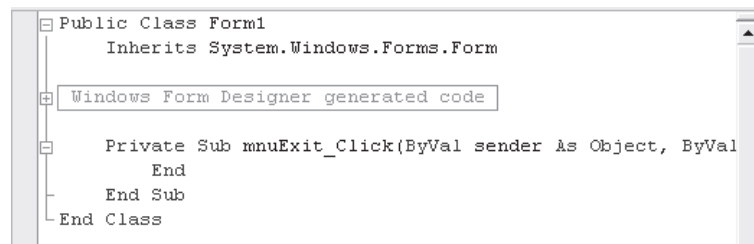
The procedure heading that is automatically added by Visual Basic .NET contains a set of parentheses with arguments after the procedure name. Note the arguments are not completely displayed in the screen above. Arguments are values that can be used by the procedure and will be discussed in a later chapter. It is important not to modify the procedure heading, including deleting any of the arguments.

**Private**  
**Sub**  
**End Sub**  
**body**

In an event procedure, **Private** indicates that the procedure cannot be accessed outside of the form class. **Sub** declares the procedure and **End Sub** is required to end the procedure. Between **Sub** and **End Sub** is the *body* with statements that execute when the event occur.

**End**

The **End** statement is used to stop program execution. The `mnuExit_Click` event procedure is completed by adding an **End** statement:



```
Public Class Form1
    Inherits System.Windows.Forms.Form
    Windows Form Designer generated code
    Private Sub mnuExit_Click(ByVal sender As Object, ByVal
        End
    End Sub
End Class
```

*good programming style*

Body statements are indented for better readability. The IDE automatically indents code, but the Tab key can be used as well. Properly indenting code is good programming style.

*printing code*

The program code of an application is printed by selecting **Print** from the **File** menu, which displays the Print dialog box. Selecting **OK** prints the application code.

## Review 4



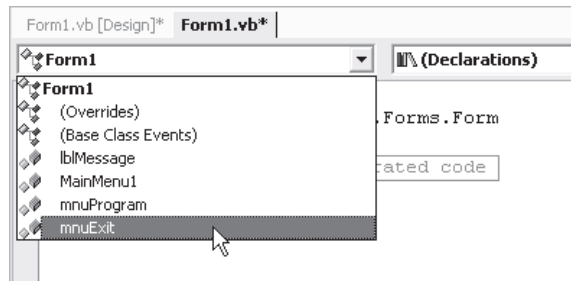
In this review you will add an event procedure to the Message application. The Message Design window should still be displayed.

### 1 DISPLAY THE CODE WINDOW

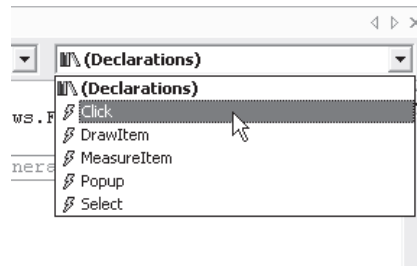
- In the Solution Explorer window, click the View Code button (📄). The Message Code window is displayed.
- In the Solution Explorer window, click the View Designer button (📐). The Design window is once again displayed. Note the tabs in the IDE for switching between the windows.
- In the IDE, click the `Form1.vb` tab. The Code window is again displayed.

## ② ADD THE EXIT COMMAND CLICK EVENT PROCEDURE

- From the Class Name list, select mnuExit:



- From the Method Name list, select Click:



The mnuExit\_Click event procedure has been added to the program code:

```
Private Sub mnuExit_Click(ByVal sender As Object, ByVal  
End Sub  
End Class
```

## ③ ADD A STATEMENT TO THE PROGRAM CODE

- Click the body of the mnuExit\_Click event procedure if the insertion point is not already there. Use the Tab key to indent the insertion point if it is not already properly indented.
- Type End. The event procedure looks like:

```
Private Sub mnuExit_Click(ByVal sender As Object, ByVal  
    End  
End Sub  
End Class
```

## ④ TEST THE EXIT COMMAND

- Save the modified Message application.
- On the toolbar, click the Start button (  ). Message is started.
- On the Message toolbar, select Program → Exit. Message is quit.

## ⑤ PRINT THE PROJECT

- Be sure the Code window is displayed and then select File → Print. A dialog box is displayed.
  - Select OK. The program code is printed.

## ⑥ CLOSE THE PROJECT

Select File → Close. The project is removed from the IDE and the Start page is displayed.

## Review 5

Create a My Name application that displays your name centered, bold, and size 14. Include a Program menu with an Exit command that terminates the application when clicked. Size the controls appropriately. Use the Message application as a guide. The application interface should look similar to that shown on the right after clicking the Program menu.



## 3.11 Assignment Statements

An *assignment statement* is used in a procedure to change a value at run time. One use of assignment is to set an object property at run time. In this case, an assignment statement takes the form:

*Object.Property = Value*

*dot (.)*

*equal sign (=)*

where *Object* is the control object name, *Property* is the name of a property for that object, and *Value* is a valid property setting. A dot (.) is required between *Object* and *Property* to access the property. The equal sign (=) assigns the property on the left of the equal sign the value on the right of the equal sign. For example, the statement

```
lblMessage.Text = "Smile!"
```

sets the Text property of the lblMessage Label control to Smile! Note that text must be enclosed in quotation marks in an assignment statement.

Assignment statements allow a single label to display different text at run time. The Message application can be modified to demonstrate this. First, two menu commands are added to the interface:



*Message with Hello World and Smile commands*

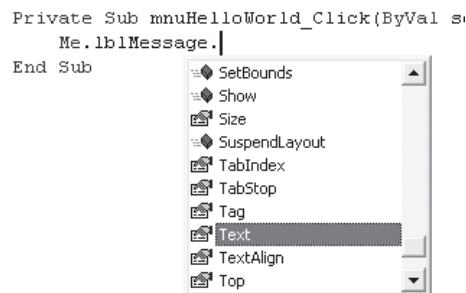
Next, event procedures are added for the two new commands. Note that assignment statements are used to change the label text:

```
Private Sub mnuExit_Click(ByVal sender As Object, ByVal  
    End  
End Sub  
  
Private Sub mnuHelloWorld_Click(ByVal sender As Object,  
    lblMessage.Text = "Hello, world!"  
End Sub  
  
Private Sub mnuSmile_Click(ByVal sender As Object, ByVa  
    lblMessage.Text = "Smile!"  
End Sub  
End Class
```

As shown by the code, when Smile is clicked the label is assigned “Smile!” and when Hello World is clicked the label is assigned “Hello, World!” Both procedures use the same label to display a different message depending on the user’s input.

## 13.12 Using AutoList

The Visual Studio .NET IDE makes coding easier with AutoList. When a dot (.) is typed in a statement, an AutoList of options is displayed:

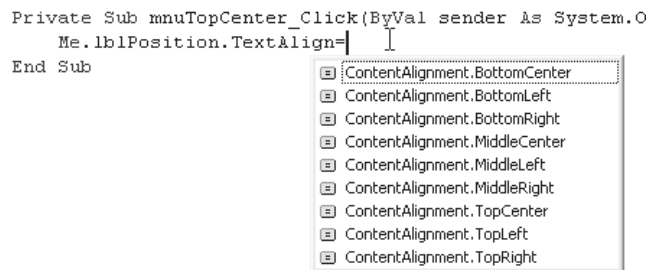


*AutoList includes properties for the current control*

The options displayed in an AutoList depend on what was typed just before the dot. For example, typing an object name and then a dot displays an AutoList with options that include control properties. In the example above, a label name was typed followed by a dot. The AutoList includes the properties of a Label control, such as the Text property selected above.

**Me** Also in the example above, **Me** was used in the assignment statement so that control names are displayed when a dot is typed. **Me** refers to the Form object.

**typing =** An AutoList is sometimes displayed when an equal sign (=) is typed. For example, typing the equal sign in the following assignment statement displays an AutoList with the possible assignment values:



Selecting from an AutoList can reduce typing errors in code. AutoList is also a good way to choose values in an assignment statement.

## Review 6



In this review you will add commands to the Message interface and write code for their corresponding event procedures. Message was last modified in Review 4.

### ① OPEN MESSAGE

### ② ADD COMMANDS

- a. In the upper-left corner of the Message form in the Design window, click the Program menu name. The Program menu is displayed.
- b. Right-click Exit. A menu is displayed.
- c. Select Insert New. A blank MenuItem is inserted above Exit and selected.
- d. Type Hello World and press Enter.
- e. Click Hello World and then scroll the Properties window to display the Name property. Enter mnuHelloWorld.
- f. Right-click Exit and select Insert New from the displayed menu. A blank MenuItem is inserted above Exit.
- g. Type Smile and press Enter.
- h. Click Smile and then scroll the Properties window to display the Name property. Enter mnuSmile.

### ③ ADD THE HELLO WORLD CLICK EVENT PROCEDURE

- a. Display the Code window. The program contains one event procedure.
- b. From the Class Name list, select mnuHelloWorld.
- c. From the Method Name list, select Click. The mnuHelloWorld\_Click event procedure is added to the program code.
- d. Click the body of the mnuHelloWorld\_Click event procedure if the insertion point is not already there. Use the Tab key to indent the insertion point if it is not already properly indented.
- e. Type Me. (Be sure to type the dot.) An AutoList is displayed.
- f. Type l (the letter L) to scroll to list items that begin with that letter.
- g. Press the down-arrow key until lblMessage is selected.
- h. Type a dot (period). lblMessage is added to the statement and a new AutoList is displayed.
- i. Type t (the letter T) to display list items that begin with that letter.
- j. Press the down-arrow key until Text is selected.
- k. Type = to add Text to the statement.
- l. Type "Hello, world!" The event procedure should look similar to:

```
Private Sub mnuHelloWorld_Click(ByVal sender As Object,
    Me.lblMessage.Text = "Hello, world!"
End Sub
```

### ④ ADD THE SMILE CLICK EVENT PROCEDURE

Use the techniques from step 2 above to create the mnuSmile\_Click event procedure, which should look similar to:

```
Private Sub mnuSmile_Click(ByVal sender As Object, ByVal
    Me.lblMessage.Text = "Smile!"
End Sub
```

## ⑤ SAVE THE PROJECT AND RUN THE APPLICATION

- Save the modified Message project.
- Run the application. Test the Smile and Hello World commands by selecting each one. Note how the assignment statements change the message displayed in the label.
- In the Message application, select Program → Exit. The application is closed.

## ⑥ PRINT THE CODE AND THEN CLOSE THE PROJECT

- Be sure the Code window is displayed and then print the code.
- Close the project.

### Formatting an Interface

The **Format** menu contains commands for aligning, sizing, and spacing objects. The objects to be arranged should first be selected as a group and the the appropriate command selected.

## 3.13 The RadioButton Control

A group of radio buttons is often used in an application to enable the user to choose from a set of choices. Only one radio button in a set can be selected at a time. For example, the Hello World International application provides radio buttons so that the user can choose the language of the message:



*Hello World International after clicking Spanish*

RadioButton

The RadioButton control has the properties:

- **Name** identifies a control for the programmer. It is good programming style to begin RadioButton object names with rad.
- **Text** is the text displayed next to the button.
- **Checked** can be set to either True or False to display the radio button as selected or not selected, respectively. Only one radio button in a group can be checked at any given time. Therefore, changing the Checked value of one button to True automatically changes the other buttons in the group to False.

### click event

A click event procedure is usually coded for each radio button. The click event procedure is executed when the user clicks a button. For example, the Spanish click event procedure includes an assignment statement that changes the Text property of the label to “Hola, mundo!”.

### group box

A GroupBox object is used to group related radio buttons. In the application above, the radio buttons for choosing a language were placed in the same group box. A GroupBox object must be added to a form before adding RadioButton objects. Radio buttons are then added to the group box by clicking the RadioButton control and then clicking the group box. Dragging a group box moves it and all the controls within it together.

### adding radio buttons

### moving a group box

The GroupBox control has the properties:

- **Name** identifies a control for the programmer. It is good programming style to begin GroupBox object names with `grp`.
- **Text** is the text displayed at the top of the group box.

## 3.14 Commenting Code

*Comments* are used to explain and clarify program code for other programmers. Comments have no effect on the way an application runs. The single quotation mark (') must begin a comment. Anything after the single quotation mark is considered a comment for that line of the program only. Multiline comments can be created by placing a quotation mark at the beginning of each line.

### programming style

Comments are good programming style and should be used wherever code may be ambiguous. Comments are also used to include information such as the programmer's name and date of modifications.

## Review 7

Follow the instructions below to create the Hello World International application.

### ① CREATE A NEW PROJECT

- Create a new Visual Basic .NET Windows project naming it Hello World International.
- Click the form to select it.
- Scroll the Properties window to display the Text property and then enter Hello World International. The form title bar now displays "Hello World International."

### ② COMPLETE THE INTERFACE

Refer to the form below to add a label, a group box, and three radio buttons. Be sure to add the group box before adding the radio buttons to the group box object. Size and move objects as necessary. Be sure to make the label large enough to display greetings of different lengths. Use the table below when setting object properties.



Object	Name	Text	Font	Text Align	Checked
Label1	lblGreeting	Hello, world!	Bold 20	MiddleCenter	
GroupBox1	grpLanguage	Select a language			
RadioButton1	radEnglish	English			True
RadioButton2	radSpanish	Spanish			False
RadioButton3	radFrench	French			False

### ③ WRITE THE APPLICATION CODE

- Display the Code window.
- Add comments that include your name and today's date, similar to:

```
Public Class Form1
    Inherits System.Windows.Forms.Form

    Windows Form Designer generated code

    'Student Name
    'Date
```

- From the Class Name list, select radEnglish.
- From the Method Name list, select Click. A radEnglish\_Click event procedure is added to the program code.
- Add the following assignment statement to the radEnglish\_Click event procedure:  
`Me.lblGreeting.Text = "Hello, world!"`
- Create a radSpanish\_Click event procedure and add an assignment statement to change the label text to *Hola, mundo!*
- Create a radFrench\_Click event procedure and add an assignment statement to change the label text to *Bonjour le monde!*

### ④ RUN THE APPLICATION

Save the modified Hello World International project and then run the application. Select each radio button to test the application. The label changes. Close the application.

### ⑤ PRINT THE CODE AND THEN CLOSE THE PROJECT

## 3.15 Operators and Expressions

### *arithmetic operators*

Visual Basic .NET includes a set of built-in arithmetic operators for exponentiation (^), multiplication (\*), division (/), addition (+), and subtraction (-). Arithmetic operators are used to form an *expression*. An expression can be used anywhere a numeric value is allowed. For example, the assignment statement below includes an expression:

```
Me.lblAnswer.Text = 2 + 6 * 3
```

An expression is not enclosed in quotation marks because quotation marks indicate text. When the above statement is executed at run time, the expression is evaluated by the computer and then the value assigned to the Text property of the Label control. Since a label displays text, the value is automatically converted to text.

### *operator precedence*

Visual Basic .NET evaluates an arithmetic expression using a specific order of operations, or *operator precedence*. Exponentiation is performed first, multiplication and division next, and then addition and subtraction. Two operators of the same precedence, for example + and -, are evaluated in order from left to right. For example, the expression above, `2 + 6 * 3`, evaluates to 20 because multiplication is performed first and then the addition.

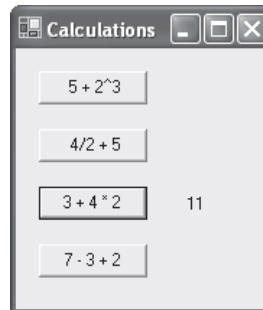
## parentheses

## programming style

Operator precedence can be changed by including parentheses in an expression. The operations within parentheses are evaluated first. For example, the result of  $(2 + 6) * 3$  is 24 because 2 and 6 were added before multiplication was performed. It is also good programming style to include parentheses when there is any ambiguity or question about the expression so a programmer reading the code will not have any doubts about what is intended.

## 3.16 The Button Control

A button is commonly found in Windows application interfaces. For example, the Calculations application below contains four Button controls and four Label controls. The labels initially display no text. When a button is clicked, the result of the expression displayed on the button is assigned to the Text property of the label next to the button. For example, when the  $3 + 4 * 2$  button is clicked, the application interface looks like:



ab Button

The Button control has the properties:

- **Name** identifies a control for the programmer. It is good programming style to begin Button object names with `btn`.
- **Text** is the text displayed on the button.

## click event

A click event procedure is usually coded for a Button object and is executed when the user clicks the button. For example, the click event procedure for the button that displays  $3 + 4 * 2$  above looks like:

```
Private Sub btnExpression3_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnExpression3.Click
    Me.lblExpression3.Text = 3 + 4 * 2
End Sub
```

When the button is clicked, the expression in the click event procedure is evaluated by the computer and the result is assigned to the Text property of the label. Note that the expression is not enclosed in quotation marks.

## Review 8



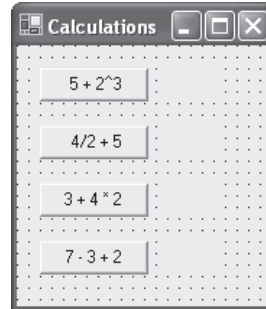
In this review, the Calculations application will be created.

### ① CREATE A NEW PROJECT

- a. Create a new Visual Basic .NET Windows project naming it Calculations.
- b. Set the Text property of the Form control to Calculations. The form title bar now displays "Calculations."

## 2 COMPLETE THE INTERFACE

Refer to the form below to add four buttons and four labels. Size and move objects as necessary. Use the table below when setting object properties.



Control	Name	Text	Text Align
Button1	btnExpression1	$5 + 2^3$	
Label1	lblExpression1	empty	MiddleCenter
Button2	btnExpression2	$4/2 + 5$	
Label2	lblExpression2	empty	MiddleCenter
Button3	btnExpression3	$3 + 4 * 2$	
Label3	lblExpression3	empty	MiddleCenter
Button4	btnExpression4	$7 - 3 + 2$	
Label4	lblExpression4	empty	MiddleCenter

## 3 WRITE THE APPLICATION CODE

- Display the Code window.
- Add comments that include your name and today's date.
- From the Class Name list, select btnExpression1.
- From the Method Name list, select Click. The btnExpression1\_Click event procedure is added.
- Add the following assignment statement to the btnExpression1\_Click event procedure:  
`Me.lblExpression1.Text = 5 + 2 ^ 3`
- Add the remaining event procedures and statements so that the program code looks like:

```
Private Sub btnExpression1_Click(ByVal sender As System
    Me.lblExpression1().Text = 5 + 2 ^ 3
End Sub

Private Sub btnExpression2_Click(ByVal sender As System
    Me.lblExpression2.Text = 4 / 2 + 5
End Sub

Private Sub btnExpression3_Click(ByVal sender As System
    Me.lblExpression3.Text = 3 + 4 * 2
End Sub

Private Sub btnExpression4_Click(ByVal sender As System
    Me.lblExpression4.Text = 7 - 3 + 2
End Sub
```

## 6 RUN THE APPLICATION

Save the modified Calculations project and then run the application. Test each of the buttons and then close the application.

## 7 PRINT THE CODE AND THEN CLOSE THE PROJECT

## 8 QUIT VISUAL BASIC .NET

## Chapter Summary

Visual Basic .NET is an object-oriented programming language that is used to create event-driven applications for Microsoft Windows. Event-driven programs wait for an event to occur and then respond to it by executing a corresponding event handler.

The Visual Basic .NET IDE (Integrated Development Environment) is used to create or modify an application. An application consists of several related files that are collectively maintained as a project. A project is created by selecting New Project on the Start page or clicking the New Project button on the toolbar.

New Project



An application interface is displayed in the Design window of the IDE. A form is a graphical object that contains a title bar, system menu, and Minimize, Maximize, and Close buttons. The Toolbox contains controls that are used to add objects to a form. An object can be sized by dragging a handle, and moved by dragging the object. The Properties window lists the properties of a selected object. A property value is set by clicking the property and then typing or selecting a new value. The Solution Explorer window is used to switch between the Design and Code windows.

A Label

A Label is an object that displays text. The Label control has the properties Name, Text, Font, and TextAlign. It is good programming style to change the Name property of an object to a descriptive name. A Label object name should begin with the prefix lbl.



An application is saved by selecting Save or clicking the Save Project button on the toolbar. An application is closed by selecting Close Solution, and the IDE is quit by selecting Exit. Selecting Open opens an existing project.



An application is run by clicking the Start button on the toolbar or selecting Start. A running application can be terminated by clicking the Stop Debugging button on the toolbar or selecting Stop Debugging. Clicking the Close button in the upper-right corner of the application also quits the application.

MainMenu

A MainMenu object displays a menu at run time. Each menu and command name typed is a MenuItem object with the properties Name and Text. A MenuItem object name should begin with the prefix mnu. New MenuItem objects can be added and existing items modified and deleted in the Design window.



A set of instructions for an application is called program code. A statement is a single line of code. Some code is automatically generated when controls are added to a form. Application code is displayed in the Code window by clicking the View Code button in the Solution Explorer window or selecting Code.

A procedure is a block of code written to perform specific tasks. The event procedure, also called an event handler, executes to a corresponding event. An event procedure is added to the Code window by selecting the control name in the Class Name list and then the event name from the Method Name list.

The **End** statement quits an application. **Private** indicates that a procedure cannot be accessed outside of the form class. **Sub** declares the procedure, and **End Sub** indicates the end of a procedure.

An assignment statement is used in a procedure to change a value at run time. Assignment can be used to change property values at run time. The IDE displays an AutoList of properties when a dot (.) is typed. Using an AutoList for selecting properties can reduce typing errors.


 RadioButton

A set of RadioButton objects allow the user to choose from a set of options. The RadioButton control has the properties Name, Text, and Checked. A RadioButton object name should begin with the prefix rad. A click event procedure is usually coded for each button. Related radio buttons must be grouped in a group box. The GroupBox control has the properties Name and Text.

 GroupBox

Comments are used to explain and clarify program code for other programmers. They can also be used to include the programmer's name and document modifications to a program. A comment is included in code by preceding text with a single quotation mark (').

Visual Basic .NET includes a set of built-in operators for exponentiation (^), multiplication (\*), division (/), addition (+), and subtraction (-). Arithmetic operators are used to form an expression. Expressions are evaluated using a specific operator precedence. Parentheses can be used to change operator precedence.

 Button

Button objects are commonly used in Windows applications. The Button control has the properties Name and Text. A Button object name should begin with the prefix btn. A click event procedure is usually coded for a button.

# Vocabulary

**Application** A program.

**Windows Forms** A framework for creating an application that uses forms and objects to interact with a user.

**Assignment statement** Uses the equal sign to give the control property on the left of the equal sign the value on the right of the equal sign.

**Body** The statements in a procedure.

**Button** An object the user clicks.

**Code window** The part of the IDE that displays the form module where program code is entered.

**Comment** Text that explains and clarifies program code for other programmers. Comments are preceded by a single quotation mark.

**Compiler** Converts a program to a language that the computer understands.

**Control** An object that interacts with the user.

**Design window** Displays the application interface and allows objects to be added, deleted, and sized.

**Event** A way in which the user can interact with an object.

**Event-driven application** Waits until an event occurs before executing code.

**Event handler** Block of code executed in response to an event.

**Event procedure** Event handler.

**Expression** Formed with arithmetic operators.

**Form** A graphical object that contains a title bar, system menu, and Minimize, Maximize, and Close buttons.

**IDE (Integrated Development Environment)** Used to create or modify a Visual Basic .NET application.

**Interface** What appears on the screen when an application is running.

**Label** An object that displays text that cannot be changed by the user.

**Menu bar** The part of the IDE that contains the names of menus that contain commands.

**MenuItem** A command name or menu name in a MainMenu object.

**Operator precedence** The order in which operators are evaluated in an expression.

**Procedure** See Event procedure.

**Program code** A set of instructions in an application.

**Project** The set of files that make up a Visual Basic .NET application.

**Project Explorer window** The part of the IDE that lists the files in the current project.

**Properties window** The part of the IDE that lists the properties values of an object.

**Property** The part of a control that defines its appearance, behavior, position, and other attributes.

**Radio button** A round object the user clicks. Appears filled in when selected.

**Run time** The time during which the application is being executed.

**Select** Clicking a control, which displays handles.

**Statement** A line of code.

**Solution Explorer window** Used to switch between the Design and Code windows.

**Toolbox** The part of the IDE that contains controls that are used to add objects to a form.

**Visual Basic .NET** Object-oriented programming environment used to create Windows applications.

**Windows Forms** A framework that uses forms and objects to interact with a user.

# Visual Basic .NET

- ^ Arithmetic operator for exponentiation.
- \* Arithmetic operator for multiplication.
- / Arithmetic operator for division.
- + Arithmetic operator for addition.
- Arithmetic operator for subtraction.

() Used to change operator precedence in an expression.

' Precedes a comment.

" Used to enclose text in an assignment statement.

= Used in an assignment statement to give the property on the left of the equal sign the value on the right of the equal sign.

**Close Solution command** Closes a project. Found in the File menu.

 **Code command** Displays the Code window. Found on the toolbar. The Code command from the View menu can be used instead of the button.

 **Button control** Used to add a button to an application interface.

**End** Programming statement used to stop program execution.

**End Sub** Programming statement used to end a procedure.


**Exit command** Closes the Visual Basic .NET IDE. Found in the File menu.

**Font** Label control property that sets the font name, font style, and font size of text in a label.

 **GroupBox control** Used to group a set of radio buttons on an application interface.

**Inherits** Programming statement used to indicate that a class contains all the attributes of another class.

 **Label control** Used to add a label to an application interface.

 **MainMenu control** Used to add a menu bar with menu names and commands to an application interface.

**Me** Used in code to refer to the current form.

**Name** Control property that sets the object name, which is used by the programmer.


 **New Project button** Clicked to create a new project. Found on the toolbar.


**Project command** Opens an existing project. Found in the Open submenu in the File menu.


**Print command** Prints the program code of an application. Found in the File menu.

**Private** Programming statement used to indicate that a procedure cannot be accessed outside of the form class.

 **RadioButton control** Used to add a radio button to an application interface.

 **Save command** Saves the current project. Found in the File menu. The Save Project button on the toolbar can be used instead of the command.

 **Start command** Runs an application. Found in the Debug menu. The button on the toolbar can be used instead of the command.


 **Stop Debugging command** Stops the current application. Found in the Debug menu. The button on the toolbar can be used instead of the command.

**Sub** Programming statement used to declare a procedure.

**Text** Control property that sets the text to be displayed in a title bar, button, or label.

**TextAlign** Label control property that sets the alignment of text.

 **View Code button** Clicked to view the Code window. Found in the Solution Explorer window.

 **View Designer button** Clicked to view the Design window. Found in the Solution Explorer window.

## Critical Thinking

1. For each of the following expressions, indicate what would be displayed if these expressions were assigned to a label.

- a)  $10 + 3 - 6$             g)  $2 + 4 / 2 + 1$   
b)  $2 ^ 3 + 5 * 4$         h)  $5 - 3 ^ 2 + 1$   
c)  $4 + 3 * 2$             i)  $15 / 3 + 2$   
d)  $2 + 9 ^ (1 / 2)$         j)  $"6 + 3 - 2"$   
e)  $15 * 2 + 4$             k)  $2 + 9 ^ 1 / 2$   
f)  $15 * (2 + 4)$         l)  $5 - 3 ^ (2 + 1)$

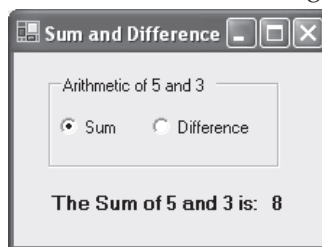
2. Explain the term "event-driven."  
3. List the controls presented in this chapter and explain how the user can interact with each one.  
4. List at least one difference in the way a form looks between the Design window and run time.  
5. List two ways a program can be terminated.  
6. Explain the difference between the Name and Text properties of a control.  
7. Make three sketches that show labels based on the statements below.

```
Me.lblOutput1.TextAlign = ContentAlignment.MiddleCenter  
Me.lblOutput1.Text = "Have a nice day!"
```

```
Me.lblOutput2.TextAlign = ContentAlignment.BottomLeft  
Me.lblOutput2.Text = "Thanks for Playing"
```

```
Me.lblOutput3.Text = "2-4^3"  
Me.lblOutput3.Text = 2-4^3
```

8. Why is it better to type the statement `Me.lblMessage.Text = "Hello World"` rather than just `lblMessage.Text = "Hello World"`  
9. Write the statements needed for the click event procedures of `radSum` and `radDifference` based on the following application. Assume the first label is named `lblMessage` and the second is named `lblResult`:



10. Explain the error(s) in the following comments:

```
"This is my first program  
9/9/02
```

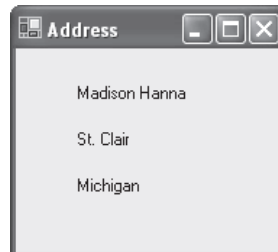
```
'Display greeting        Me.lblMessage.Text = "Hi There!"
```

11. Determine if each of the following statements is true or false. If false, explain why.  
a) The properties of an object can only be set in the property box.  
b) A MenuItem object can have a click event procedure.  
c) A MainMenu object can have only one MenuItem object.  
d) The only purpose of a GroupBox object is to display a title for a set of RadioButton objects.

## Exercises

### Exercise 1 Address

Create an Address application that displays your name, city, and state in three separate labels. The interface should look similar to:



### Exercise 2 School

- a) Create a School application that displays your school's name and mascot centered in two separate labels. Choose a different font, style, and size for the school name. The interface should look similar to:



- b) Modify the School application to include a Program menu with an Exit command.

### Exercise 3 Band Information

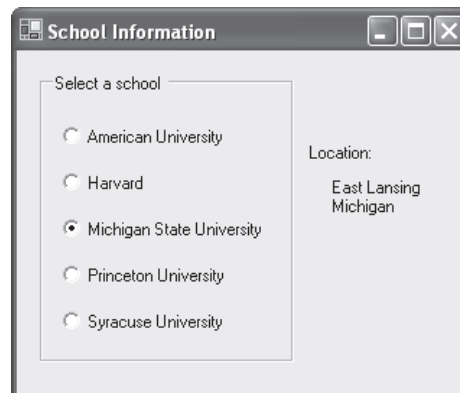
Create a Band Information application that displays the members of a selected band. Include at least three of your favorite bands. The interface should look similar to:



## Exercise 4

## School Information

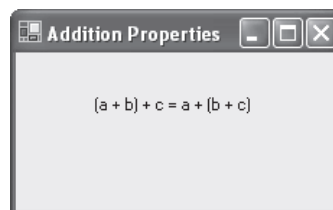
Create a School Information application that displays the city and state of a selected school. Include at least five of your favorite schools. The interface should look similar to:



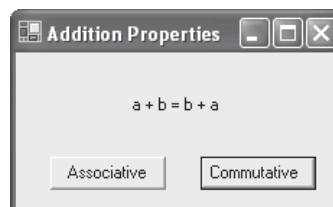
## Exercise 5

## Addition Properties

- a) Create an Addition Properties application that shows the associative property of addition,  $(a+b)+c = a+(b+c)$ , in a label. The interface should look similar to:



- b) Modify the Addition Properties application to display the associative property of addition when one button is clicked and the commutative property,  $a + b = b + a$ , when another button is clicked. The interface should look similar to the following after clicking Commutative:



- c) Modify the Addition Properties application to include a Program menu with Associative, Commutative, and Exit commands.

## Exercise 6

## Hello and Good-bye

- a) Create a Hello and Good-bye application that displays a label centered, bold, and size 18 that reads Hello! or Good-bye! depending on the button clicked. The interface should look similar to the following after clicking Hello:

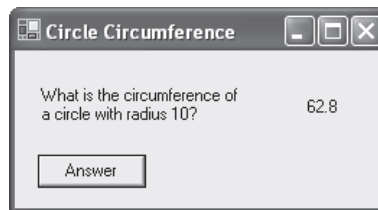


- b) Modify the Hello and Good-bye application to include a Program menu with Hello, Good-bye, and Exit commands.

## Exercise 7

## Circle Circumference

- a) Create a Circle Circumference application that displays in a label the circumference ( $2\pi r$ ) of a circle with radius 10. Use the value 3.14 for  $\pi$ . The interface should look similar to the following after clicking Answer:

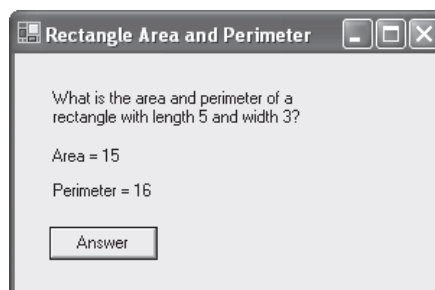


- b) Modify the Circle Circumference application to include a Program menu with an Exit command.

## Exercise 8

## Rectangle Area and Perimeter

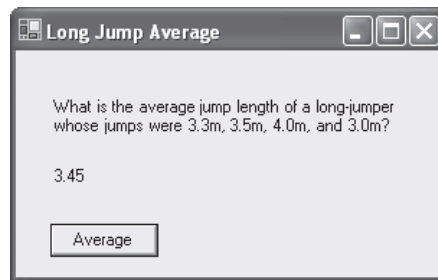
- a) Create a Rectangle Area and Perimeter application that displays the area (length \* width) and perimeter ( $2l + 2w$ ) of a rectangle of length 5 and width 3. The interface should look similar to the following after clicking Answer:



- b) Modify the Rectangle Area and Perimeter application to include a Program menu with an Exit command.

## Exercise 9 Long Jump Average

- a) Create a Long Jump Average application that calculates and displays the average jump length of an athlete whose jumps were 3.3m, 3.5m, 4.0m, and 3.0m. The interface should look similar to the following after clicking Average:



- b) Modify the Long Jump Average application to include a Program menu with an Exit command.

## Exercise 10 (advanced) Position

- a) Create a Position application that changes the position of text in a label according to the command selected by the user. The application should include a Program menu with an Exit command and a Position menu with TopLeft, TopCenter, TopRight, MiddleLeft, MiddleCenter, MiddleRight, BottomLeft, BottomCenter, and BottomRight commands. The interface should look similar to the following after selecting TopLeft from the Position menu:



Hints: Be sure the label is large enough to show changes in text position. Also, use the AutoList for selecting the assignment value for the TextAlign property.