These steps cycle continuously until the program ends. Usually, an event must happen before Visual Basic will do anything. Event-driven programs are reactive more than active—and that makes them more user friendly.

The Different Versions of Visual Basic

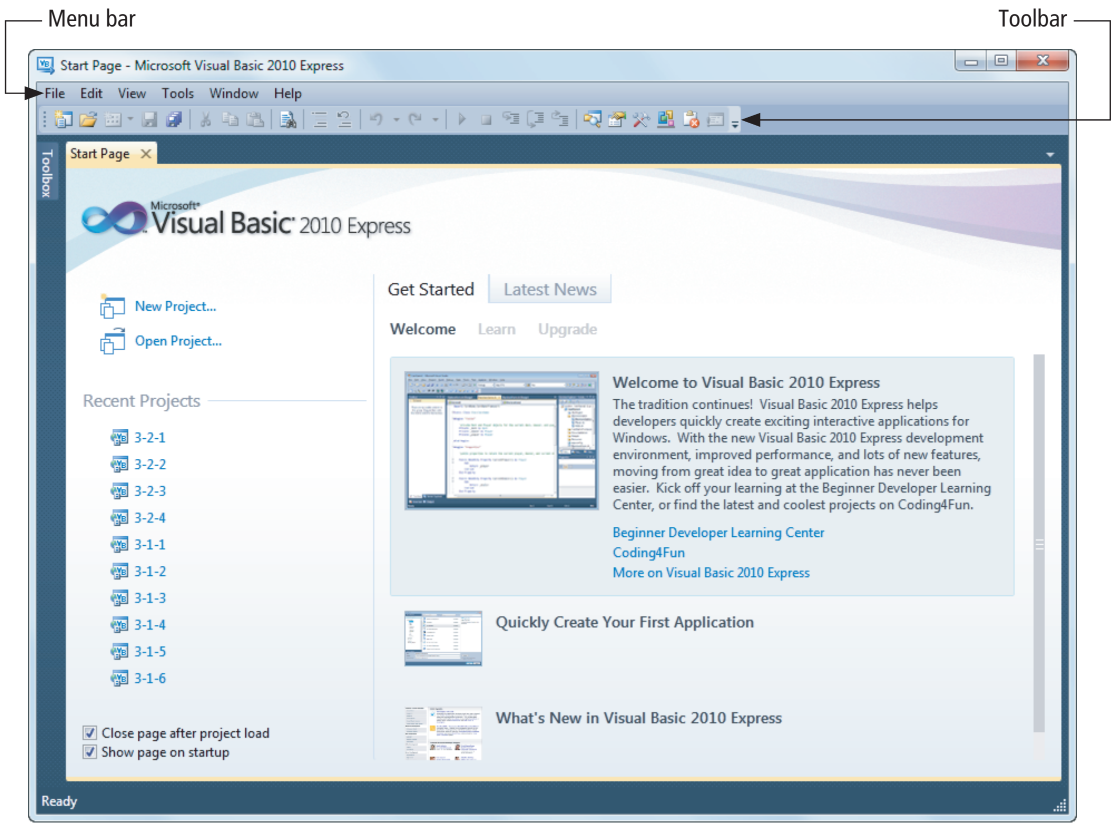
Visual Basic 1.0 first appeared in 1991. It was followed by version 2.0 in 1992, version 3.0 in 1993, version 4.0 in 1995, version 5.0 in 1997, and version 6.0 in 1998. VB.NET, initially released in February 2002, was not backward compatible with the earlier versions of Visual Basic. It incorporated many features requested by software developers, such as true inheritance. Visual Basic 2005, released in November 2005, Visual Basic 2008, released in November 2007, and Visual Basic 2010, released in April 2010 are significantly improved versions of VB.NET.

2.2 Visual Basic Controls

Visual Basic programs display a Windows-style screen (called a **form**) with boxes into which users type (and in which users edit) information and buttons that they click to initiate actions. The boxes and buttons are referred to as **controls**. In this section, we examine forms and four of the most useful Visual Basic controls.

Starting a New Visual Basic Program

Each program is saved (as several files and subfolders) in its own folder. Before writing your first program, you should use Windows Explorer to create a folder to hold your programs.


Menu bar
Toolbar
Start Page - Microsoft Visual Basic 2010 Express
File
Edit
View
Tools
Window
Help
Toolbox
Start Page
Microsoft® Visual Basic® 2010 Express
New Project...
Open Project...
Recent Projects
3-2-1
3-2-2
3-2-3
3-2-4
3-1-1
3-1-2
3-1-3
3-1-4
3-1-5
3-1-6
Close page after project load
Show page on startup
Get Started
Latest News
Welcome
Learn
Upgrade
Beginner Developer Learning Center
Coding4Fun
More on Visual Basic 2010 Express
Quickly Create Your First Application
What's New in Visual Basic 2010 Express
Ready

Welcome to Visual Basic 2010 Express

The tradition continues! Visual Basic 2010 Express helps developers quickly create exciting interactive applications for Windows. With the new Visual Basic 2010 Express development environment, improved performance, and lots of new features, moving from great idea to great application has never been easier. Kick off your learning at the Beginner Developer Learning Center, or find the latest and coolest projects on Coding4Fun.

FIGURE 2.3 **Visual Basic opening screen.**

The process for invoking Visual Basic varies slightly with the edition of Visual Basic installed on the computer. To invoke Visual Basic from a computer that has Visual Basic Express installed, click the Windows Start button, hover over All Programs, and then click on Microsoft Visual Basic 2010 Express. With the other editions of Visual Basic, hover over All Programs, hover over Microsoft Visual Studio 2010, and then click on Microsoft Visual Studio 2010 in the short list that is revealed.

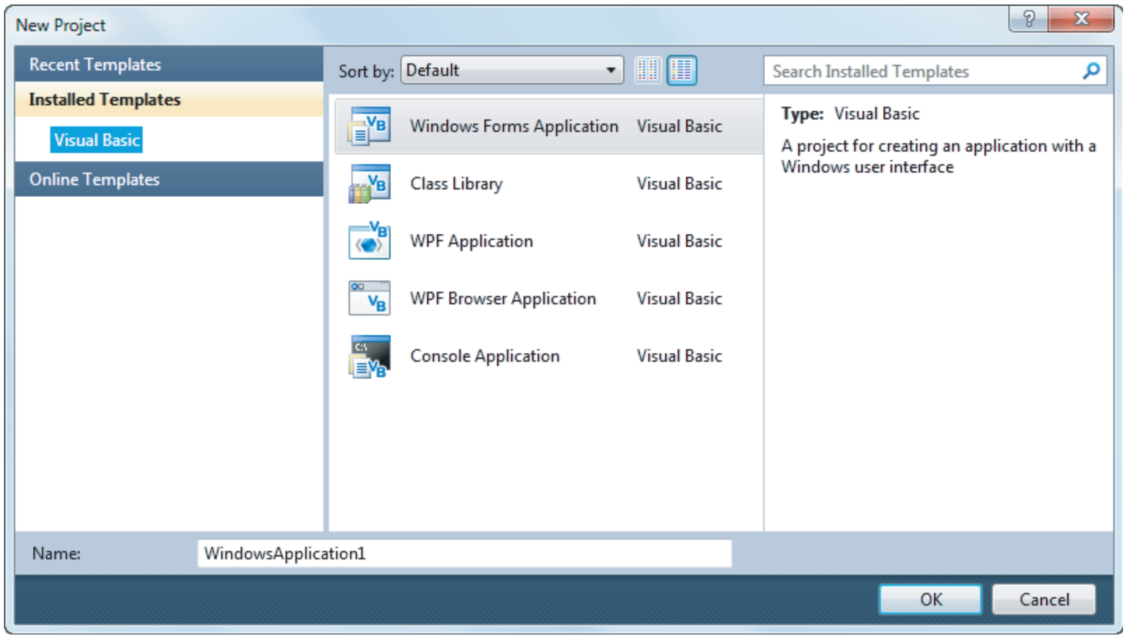
Figure 2.3 shows the top half of the screen after Visual Basic is invoked. A Menu bar and a Toolbar are at the very top of the screen. These two bars, with minor variations, are always present while you are working with Visual Basic. The remainder of the screen is called the **Start Page**. Some tasks can be initiated from the Menu bar, the Toolbar, and the Start Page. We will usually initiate them from the Menu bar or the Toolbar.

The first item on the Menu bar is *File*. Click on *File*, and then click on *New Project* to produce a New Project dialog box. Figure 2.4 shows the New Project dialog box produced by Visual Basic Express.

The Windows Forms Application item should be selected in the center list. If this is not the case, click on *Windows Forms Application* to select it. **Note:** The number of items in the list will vary depending on the edition of Visual Basic you are using.

The name of the program, initially set to WindowsApplication1, can be specified at this time. Since we will have a chance to change it later, let's just use the name WindowsApplication1 for now. Click on the *OK* button to invoke the Visual Basic programming environment. See Fig. 2.5. The Visual Basic programming environment is referred to as the **Integrated Development Environment** or **IDE**.

Very likely, your screen will look different than Fig. 2.5. The IDE is extremely configurable. Each window in Fig. 2.5 can have its location and size altered. New windows can be displayed in the IDE, and any window can be closed or hidden behind a tab. For instance, in Fig. 2.5 the Toolbox window is hidden behind a tab. The *View* menu is used to add additional windows to the IDE. If you would like your screen to look exactly like Fig. 2.5, click on *Reset Windows Layout* in the *Windows* menu, and then click on *Yes*.



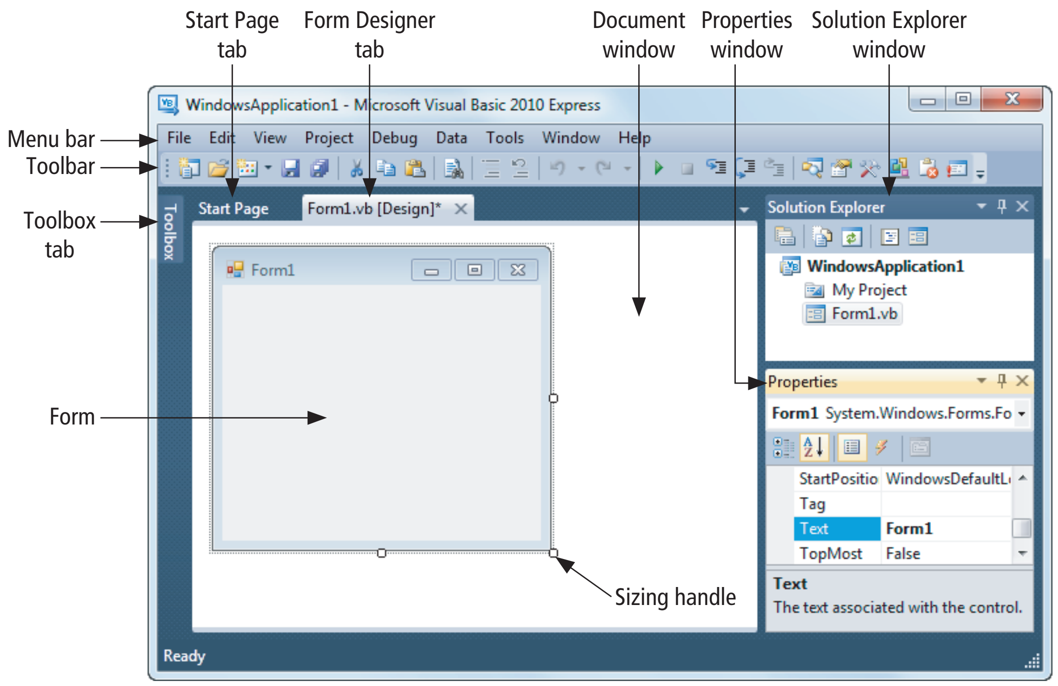
Windows Forms Application	Visual Basic
Class Library	Visual Basic
WPF Application	Visual Basic
WPF Browser Application	Visual Basic
Console Application	Visual Basic

Type: Visual Basic

A project for creating an application with a Windows user interface

Name: WindowsApplication1

FIGURE 2.4 The Visual Basic New Project dialog box.



Form1 System.Windows.Forms.Fo

StartPositio	WindowsDefaultL
Tag	
Text	Form1
TopMost	False

Text
The text associated with the control.

FIGURE 2.5 **The Visual Basic integrated development environment in Form Designer mode.**

The **Menu bar** of the IDE displays the menus of commands you use to work with Visual Basic. Some of the menus, like *File*, *Edit*, *View*, and *Window*, are common to most Windows applications. Others, such as *Project*, *Data*, and *Debug*, provide commands specific to programming in Visual Basic.

The **Toolbar** holds a collection of buttons that carry out standard operations when clicked. For example, you use the fifth button, which looks like a stack of three diskettes, to save the files associated with the current program. To reveal the purpose of a Toolbar button, hover the mouse pointer over it. The little information rectangle that pops up is called a **tooltip**.

The **Document window** currently holds the rectangular **Form window**, or **form** for short. The form becomes a Windows window when a program is executed. Most information displayed by the program appears on the form. The information usually is displayed in controls that the programmer has placed on the form. You can change the size of the form by dragging one of its sizing handles.

The **Properties window** is used to change how objects look and react.

The **Solution Explorer** window displays the files associated with the program and provides access to the commands that pertain to them. (***Note:*** If the Solution Explorer or the Properties window is not visible, click on it in the *View/Other Windows* menu.)

The **Toolbox** holds icons representing objects (called controls) that can be placed on the form. If your screen does not show the Toolbox, hover the mouse over the Toolbox tab at the left side of the screen. The Toolbox will slide into view. Then click on the pushpin icon in the title bar at the top of the Toolbox to keep the Toolbox permanently displayed in the IDE. (***Note:*** If there is no tab marked Toolbox, click on *Toolbox* in the *View/Other Windows* menu.)

The controls in the Toolbox are grouped into categories such as *All Windows Forms* and *Common Controls*. Figure 2.6 shows the Toolbox after the *Common Controls* group has been expanded. Most of the controls discussed in this text can be found in the list of common controls. (You can obtain a description of a control by hovering the mouse over the control.) The four controls discussed in this chapter are text boxes, labels, buttons, and list boxes. In order to see all the group names, collapse each of the groups.

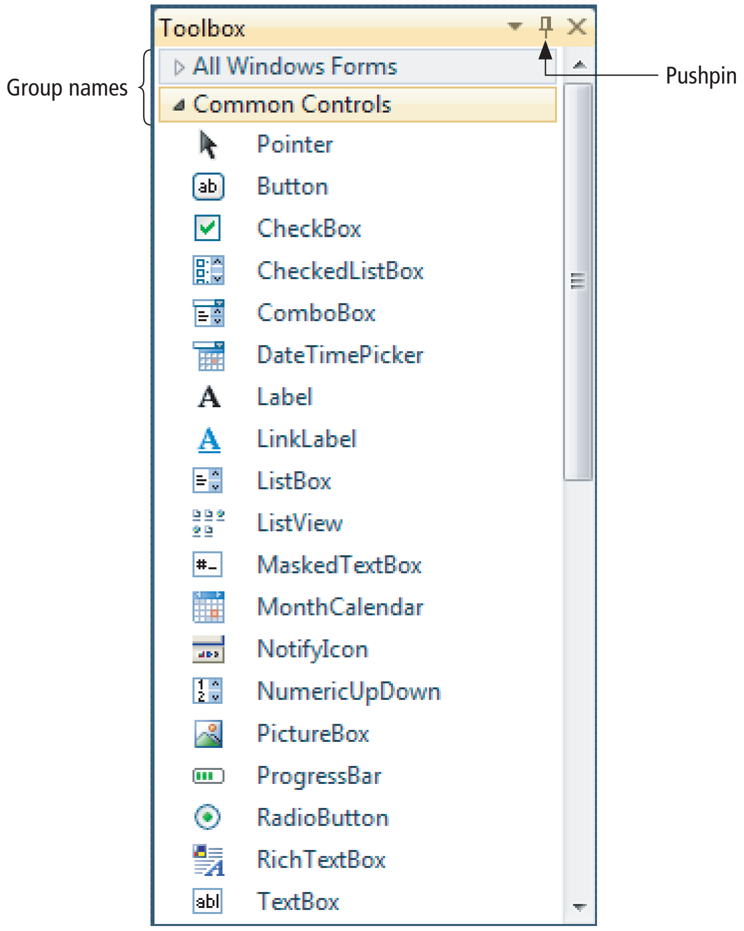


FIGURE 2.6 The Toolbox's common controls.

VideoNote Visual Basic controls

Text boxes: You use a text box to get information from the user, referred to as **input**, or to display information produced by the program, referred to as **output**.

Labels: You place a label near a text box to tell the user what type of information is displayed in the text box.

Buttons: The user clicks a button to initiate an action.

List boxes: In the first part of the book, we use list boxes to display output. Later, we use list boxes to make selections.

A Text Box Walkthrough

1. Double-click on the text box icon in the *Common Controls* group of the Toolbox. A rectangle with two small squares, called **sizing handles**, appears at the upper left corner of the form. (You can alter the width of the text box by dragging one of its sizing handles.) Move the mouse arrow to any point of the text box other than a sizing handle, hold down the left mouse button, and drag the text box to the center of the form. See Fig. 2.7. (***Note:*** Tasks buttons are used to set certain properties of controls.)

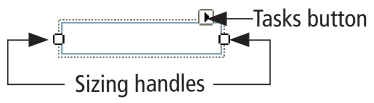
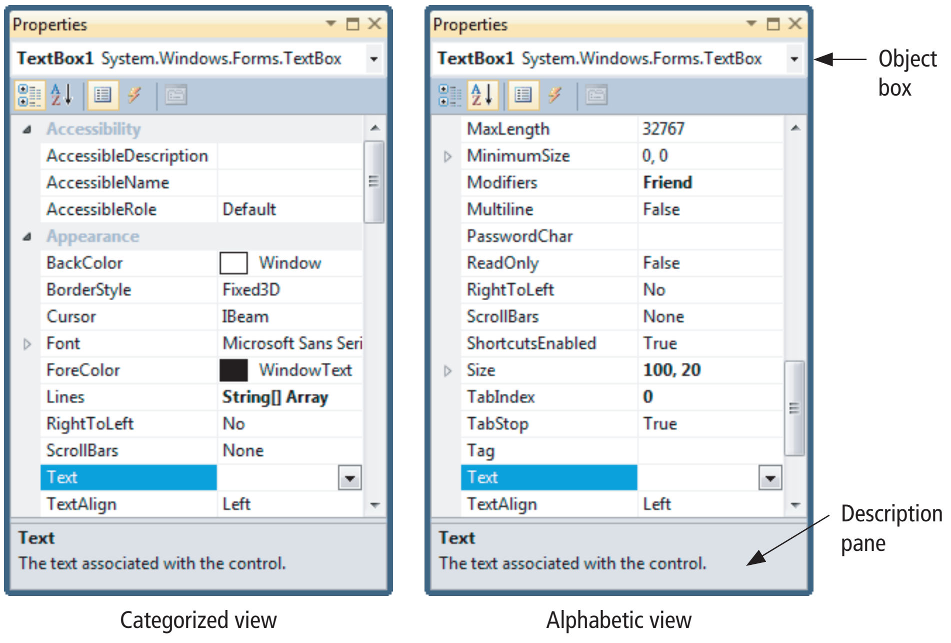


FIGURE 2.7 A text box with sizing handles.

2. Click anywhere on the form outside the rectangle to deselect the text box.
3. Click on the rectangle to restore the handles. An object showing its handles is said to be **selected**. A selected text box can have its width altered, location changed, and other properties modified.
4. Move the mouse arrow to the handle in the center of the right side of the text box. The cursor should change to a double arrow (↔). Hold down the left mouse button, and move the mouse to the right. The text box is stretched to the right. Similarly, grabbing the text box on the left side and moving the mouse to the left stretches the text box to the left. You also can use the handles to make the text box smaller. Steps 1 and 4 allow you to place a text box of any width anywhere on the form. ***Note:*** The text box should now be selected; that is, its sizing handles should be showing. If not, click anywhere inside the text box to select it.
5. Press the delete key, DEL, to remove the text box from the form. Step 6 gives an alternative way to place a text box of any width at any location on the form.
6. Click on the text box icon in the Toolbox. Then move the mouse pointer to any place on the form. (When over the form, the mouse pointer becomes a pair of crossed thin lines.) Hold down the left mouse button, and drag the mouse on a diagonal to generate a rectangle. Release the mouse button to obtain a selected text box. You can now alter the width and location as before. ***Note:*** The text box should now be selected. If not, click anywhere inside the text box to select it.
7. Press F4 to activate the Properties window. [You also can activate the Properties window by clicking on it, clicking on the *Properties Window* button ( ) on the Toolbar, selecting *Properties Window* from the *View* menu, or clicking on the text box with the right mouse button and selecting *Properties*.] See Fig. 2.8. The first line of the Properties window (called the **Object box**) reads "TextBox1 etc." TextBox1 is the current name of the text box. The first two buttons below the Object box permit you to view the list of properties either grouped into categories or alphabetically. Use the up- and down-arrow keys (or the up- and



TextBox1 System.Windows.Forms.TextBox

Categorized view:

Accessibility	
AccessibleDescription	
AccessibleName	
AccessibleRole	Default
Appearance	
BackColor	Window
BorderStyle	Fixed3D
Cursor	IBeam
Font	Microsoft Sans Seri
ForeColor	WindowText
Lines	**String[] Array**
RightToLeft	No
ScrollBars	None
Text	
TextAlign	Left

Text
The text associated with the control.

Alphabetic view:

MaxLength	32767
MinimumSize	0, 0
Modifiers	**Friend**
Multiline	False
PasswordChar	
ReadOnly	False
RightToLeft	No
ScrollBars	None
ShortcutsEnabled	True
Size	**100, 20**
TabIndex	**0**
TabStop	True
Tag	
Text	
TextAlign	Left

Text
The text associated with the control.

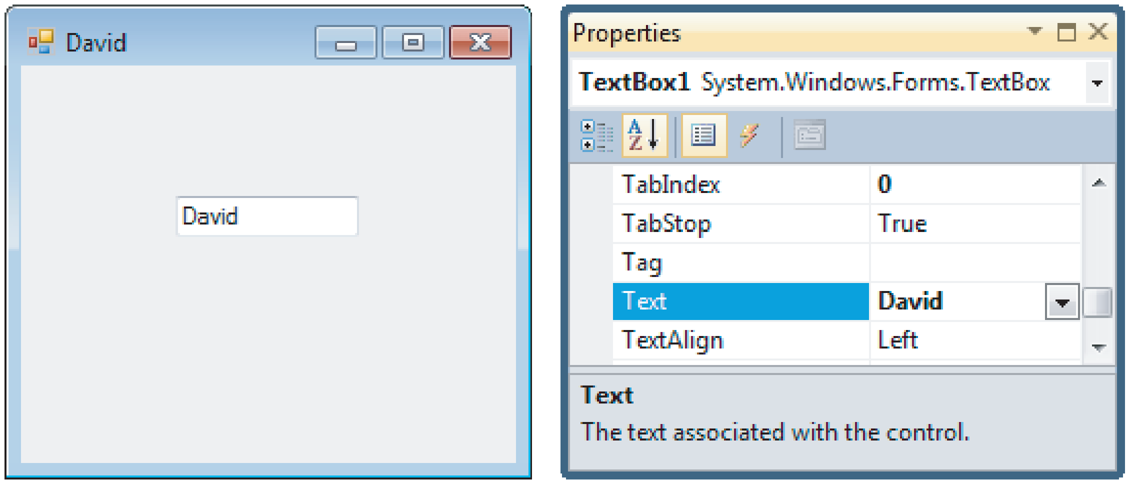
FIGURE 2.8 Text box Properties window.

down-scroll arrows) to move through the list. The left column gives the property names, and the right column gives the current settings of the properties. We discuss four properties in this walkthrough.

Note 1: The third and fourth buttons below the Object box, the Properties button and the Events button, determine whether properties or events are displayed in the Properties window. Normally the Properties button is highlighted. If not, click on it.

Note 2: If the Description pane is not visible, right-click on the Properties window, then click on *Description*. The Description pane describes the currently highlighted property.

8. Move to the Text property with the up- and down-arrow keys. (Alternatively, scroll until the property is visible, and click on the property.) The Text property, which determines the words displayed in the text box, is now highlighted. Currently, there is no text displayed in the **Settings box** on the right.
9. Type your first name. Then press the Enter key, or click on another property. Your name now appears in both the Settings box and the text box. See Fig. 2.9.



TextBox1 System.Windows.Forms.TextBox

TabIndex	0
TabStop	True
Tag	
Text	David
TextAlign	Left

Text
The text associated with the control.

FIGURE 2.9 **Setting the Text property to David.**

10. Click at the beginning of your name in the Text Settings box, and add your title, such as Mr., Ms., or The Honorable. (If you mistyped your name, you can easily correct it now.) Then, press Enter.
11. Use the up-arrow key or the mouse to move to the ForeColor property. This property determines the color of the information displayed in the text box.
12. Click on the down arrow in the right part of the Settings box, and then click on the Custom tab to display a selection of colors. See Fig. 2.10. Click on one of the colors, such as *blue* or *red*. Notice the change in the color of your name. (***Note:*** The sixteen white boxes at the bottom of the grid are used to create custom colors. See item L under "Manage Visual Basic Controls" in Appendix B for details.)
13. Select the Font property with a single click of the mouse. The current font is named Microsoft Sans Serif.
14. Click on the ellipsis (. . .) box in the right part of the Settings box to display a dialog box. See Fig. 2.11. The three lists give the current name (Microsoft Sans Serif), current style

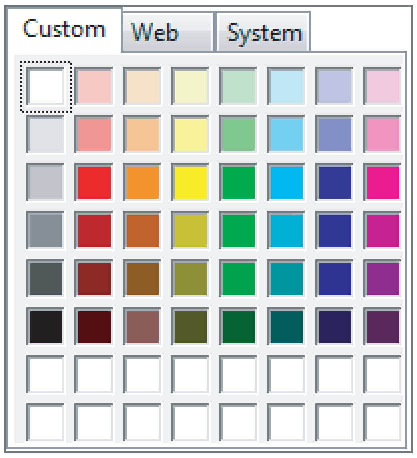
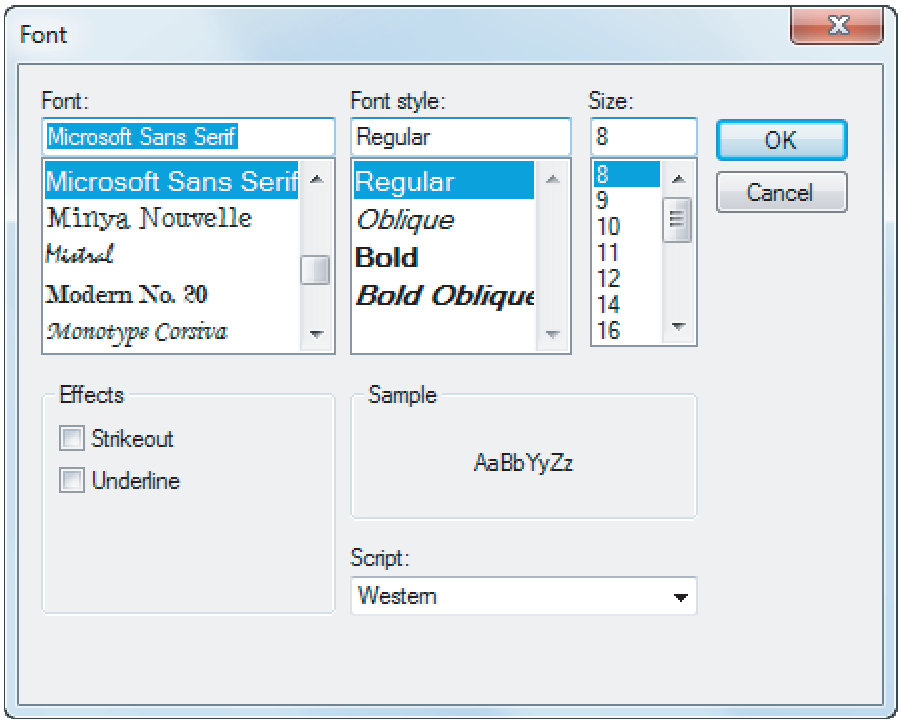


FIGURE 2.10 Setting the ForeColor property.



Font: Microsoft Sans Serif

Microsoft Sans Serif
Minya Nouvelle
Mistral
Modern No. 20
Monotype Corsiva

Font style: Regular

Regular
Oblique
Bold
Bold Oblique

Size: 8

8
9
10
11
12
14
16

Effects

Strikeout

Underline

Sample

AaBbYyZz

Script:

Western

FIGURE 2.11 The Font dialog box.

(Regular), and current size (8) of the font. You can change any of these attributes by clicking on an item in its list or by typing into the box at the top of the list. Click on Bold in the style list, and click on 12 in the size list. Now click on the *OK* button to see your name displayed in a larger bold font. The text box will be longer so that it can accommodate the larger font.

15. Click on the text box and resize it to be about 3 inches wide.

Visual Basic programs consist of three parts: interface, values of properties, and code. Our interface consists of a form with a single object, a text box. We have set a few properties for the text box—the text (namely, your name), the foreground color, the font style, and the font size. In Section 2.3, we discuss how to place code into a program. Visual Basic endows certain capabilities to programs that are independent of any code we will write. We will now run the existing program without adding any code in order to experience these capabilities.

16. Click on the *Start Debugging* button (▶) on the Toolbar to run the program. [Alternatively, you can press F5 to run the program or can click on *Start Debugging* in the *Debug* menu.] After a brief delay, a copy of the form appears that has neither the form nor the text box selected.

17. Your name is highlighted. Press the End key to move the cursor to the end of your name. Now type in your last name, and then keep typing. Eventually, the words will scroll to the left.

18. Press Home to return to the beginning of the text. You have a miniature word processor at your disposal. You can place the cursor anywhere you like in order to add or delete text. You can drag the cursor across text to select a block, place a copy of the block in the Clipboard with Ctrl+C, and then duplicate it elsewhere with Ctrl+V.

19. Click on the *Stop Debugging* button (■) on the Toolbar to end the program. [Alternatively, you can end the program by clicking on the form's *Close* button (✕) or pressing Alt + F4.]

20. Select the text box, activate the Properties window, select the ReadOnly property, click on the down-arrow button, and finally click on True. Notice that the background color is now gray.

21. Run the program, and try typing into the text box. You can't. Such a text box is used for output. Only code can display information in the text box.

(***Note:*** In this textbook, whenever a text box will be used only for the purpose of displaying output, we will always set the ReadOnly property to True.)

22. End the program.

23. Let's now save the program on a disk. Click on the Toolbar's *Save All* button ( ) to save the work done so far. (Alternatively, you can click on *Save All* in the *File* menu.) You will be prompted for the name of the program and the path to the folder where you want the program to be saved. Type a name, such as "VBdemo". You can either type a path or use Browse to locate a folder. (This folder will automatically be used the next time you click on the *Save All* button.) The files for the program will be saved in a subfolder of the selected folder.

Important: If the "Create directory for solution" check box is checked, then click on the check box to uncheck it. Finally, click on the *Save* button.

24. Create a new program as before by clicking on *New Project* in the *File* menu. [Alternatively, you can click on the *New Project* button ( ), the first button on the Toolbar, or you can click on *New Project* in the Start Page.] A New Project dialog box will appear.

25. Give a name to the project, such as MyProgram, and then click on the *OK* button.

26. Place three text boxes on the form. (If you use the double-click technique, move the text boxes so that they do not overlap.) Notice that they have the names TextBox1, TextBox2, and TextBox3.

27. Run the program. Notice that the cursor is in TextBox1. We say that TextBox1 has the **focus**. (This means that TextBox1 is the currently selected object and any keyboard actions will be sent directly to this object.) Any text typed will display in that text box.

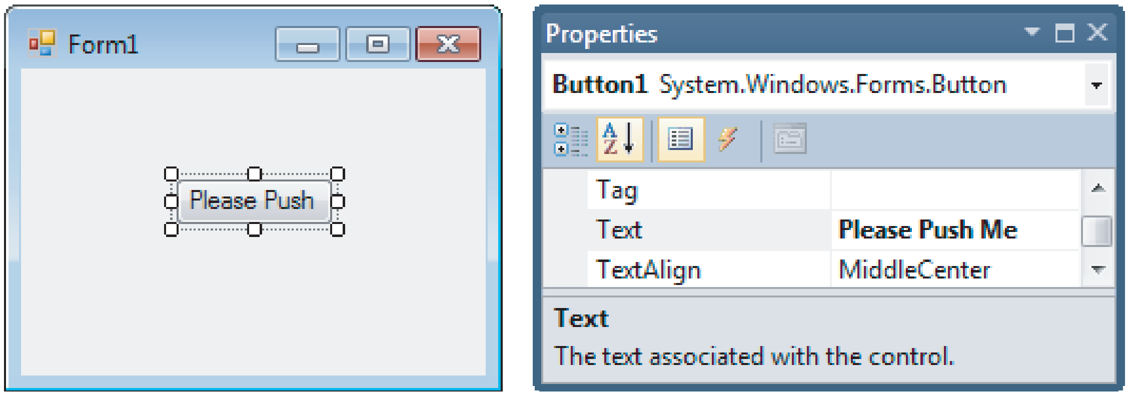
28. Press Tab once. Now, TextBox2 has the focus. When you type, the characters appear in TextBox2.

29. Press Tab several times, and then press Shift+Tab a few times. With Tab, the focus cycles through the objects on the form in the order they were created. With Shift+Tab, the focus cycles in the reverse order.
30. End the program you created.
31. We would now like to return to the first program. Click on *Open Project* from the *File* menu. An Open Project dialog box will appear stating that "You must choose to either save or discard changes in the current project before opening a project." There is no need to save this program, so click on the *Discard* button. Then a second Open Project dialog box will appear.
32. Navigate to the folder corresponding to the program you saved earlier, double-click on the folder, and double-click on the file with extension *sln*. You have now reloaded the first program.

    Note: As an alternative to using the Open Project dialog box in Steps 31 and 32 to return to the first program, click on the Start Page tab at the top of the Document window, and click on the program in the Recent Projects part of the Start Page.
33. If you do not see the Form Designer for the program, double-click on Form1.vb in the Solution Explorer.
34. Click on *Close Project* in the *File* menu to close the program.

A Button Walkthrough

1. Click on the *New Project* button to start a new program. (Give a name, such as ButtonProg, to the program, and click on the *OK* button.)
2. Double-click on the Button icon in the Toolbox to place a button on the form. (The Button icon is the second icon in the Common Controls group of the Toolbox.)
3. Move the button to the center of the form.
4. Activate the Properties window, highlight the Text property, type "Please Push Me", and press Enter. See Fig. 2.12. The button is too small.



Button1 System.Windows.Forms.Button

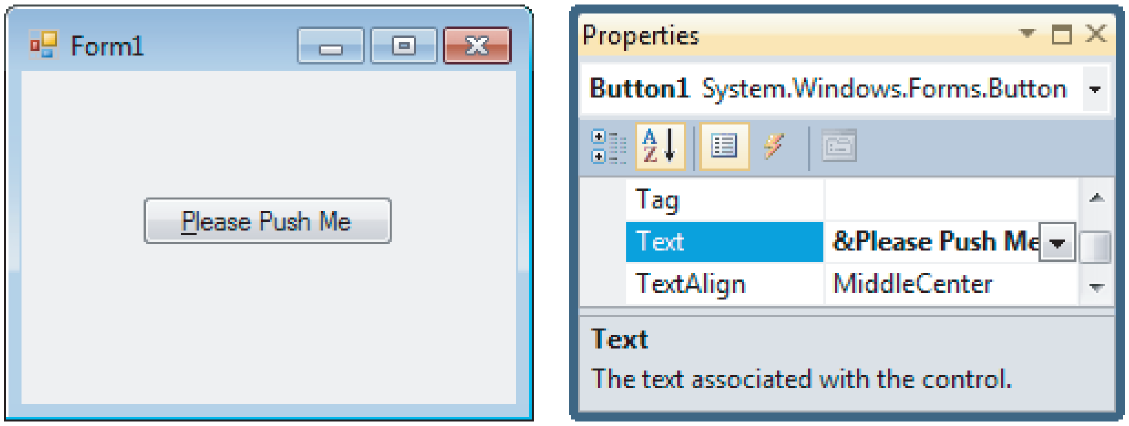
Tag	
Text	Please Push Me
TextAlign	MiddleCenter

Text
The text associated with the control.

FIGURE 2.12 **Setting the Text property.**

5. Click on the button to select it, and then widen it to accommodate the phrase "Please Push Me" on one line.
6. Run the program, and click on the button. The button appears to move in and then out. In Section 2.3, we will write code that is executed when a button is clicked on.
7. End the program and select the button.

8. From the Properties window, edit the Text setting by inserting an ampersand (&) before the first letter, P. Press the Enter key, and notice that the first letter P on the button is now underlined. See Fig. 2.13. Pressing Alt+P while the program is running causes the same event to occur as does clicking the button. However, the button will not appear to move in and out. Here, P is referred to as the **access key** for the button. (The access key is always specified as the character following the ampersand.)
9. Click on *Close Project* in the *File* menu to close the program.



Button1 System.Windows.Forms.Button

Tag	
Text	&Please Push Me
TextAlign	MiddleCenter

Text
The text associated with the control.

FIGURE 2.13 **Designating P as an access key.**

A Label Walkthrough

1. Click on the *New Project* button to begin a new program. Feel free to keep the default name, such as WindowsApplication1.
2. Double-click on the label icon to place a label on the form. (The label icon is a large letter A.) Move the label to the center of the form.
3. Activate the Properties window, highlight the Text property, type "Enter Your Phone Number:", and press Enter. (Such a label is placed next to a text box into which the user will type a phone number.) Notice that the label widened to accommodate the text. This happened because the AutoSize property of the label is set to True by default.
4. Change the AutoSize property to False. Press Enter. Notice that the label now has eight sizing handles when selected.
5. Make the label narrower and longer until the words occupy two lines.
6. Activate the Properties window, and click on the down arrow to the right of the setting for the TextAlign property. Experiment by clicking on the various rectangles and observing their effects. The combination of sizing and alignment permits you to design a label easily.
7. Run the program. Nothing happens, even if you click on the label. Labels just sit there. The user cannot change what a label displays unless you write code to make the change.
8. End the program.
9. Click on *Close Project* in the *File* menu to close the program.

A List Box Walkthrough

1. Click on the New Project button to begin a new program. Feel free to keep the default name, such as WindowsApplication1.

2. Place a list box on the form. (The list box icon is the ninth icon in the *Common Controls* group of the Toolbox.)
3. Press F4 to activate the Properties window and notice that the list box does not have a Text property. The word ListBox1 is actually the setting for the Name property.
4. Also place a text box, a button, and a label on the form.
5. Click on the Object box just below the title bar of the Properties window. The name of the form and the names of the four controls are displayed. If you click on one of the names, that object will become selected and its properties displayed in the Properties window.
6. Run the program. Notice that the word ListBox1 has disappeared, but the words Button1 and Label1 are still visible. The list box is completely blank. In subsequent sections, we will write code to place information into the list box.
7. End the program and then click on *Close Project* in the *File* menu.

The Name Property

The form and each control on it has a Name property. By default, the form is given the name Form1 and controls are given names such as TextBox1 and TextBox2. These names can (and should) be changed to descriptive ones that reflect the purpose of the form or control. Also, it is good programming practice to have each name begin with a three-letter prefix that identifies the type of the object. See Table 2.1.

TABLE 2.1 Some three-letter prefixes.

Object	Prefix	Example
form	frm	frmPayroll
button	btn	btnComputeTotal
label	lbl	lblAddress
list box	lst	lstOutput
text box	txt	txtCity

To change the name of the form, change the base name of the file Form1.vb appearing in the Solution Explorer. To make the change, right-click on Form1.vb in the Solution Explorer window, click on *Rename* in the context menu that appears, type in a new name (such as frm-Payroll.vb), and press the Enter key. ***Important:*** Make sure that the new filename keeps the extension "vb".

The name of a control is changed from the control's Properties window. (The Name property is always the third property in the alphabetized list of properties.) Names of controls and forms must begin with a letter and can include numbers and underscore (_) characters but cannot include punctuation marks or spaces.

The Name and Text properties of a button are both initially set to something like Button1. However, changing one of these properties does not affect the setting of the other property, and similarly for the Name and Text properties of forms, text boxes, and labels. The Text property of a form specifies the words appearing in the form's title bar.

Fonts

The default font for controls is Microsoft Sans Serif. Two other useful fonts are Courier New and Wingdings.

Courier New is a fixed-width font; that is, each character has the same width. With such a font, the letter i occupies the same space as the letter m. Fixed-width fonts are used with tables when information is to be aligned in columns.

The Wingdings font consists of assorted small pictures and symbols, each corresponding to a character on the keyboard. For instance if one of the characters %, (, 1, or J is typed into the Text setting of a control whose Font is Wingdings, the control will display a bell, phone, open folder, or smiling face, respectively.

To view the character set for a Windows font, click on the Windows Start button in the Windows task bar and successively click on All Programs, Accessories, System Tools, and Character Map. A rectangular array of characters will appear. After selecting the font, click on any item to enlarge it. You can insert the keyboard character for the item into the Clipboard by pressing the *Select* button and then the *Copy* button. To place the character into the Text property of a control having that font, just move the cursor to the Settings box for the Text property for that control and press Ctrl+V.

Auto Hide

The Auto Hide feature allows you to make more room for the Document window of the screen by hiding windows (such as the Toolbox, Solution Explorer, or Properties window). Let's illustrate the feature with a walkthrough using the Toolbox window. We start by discussing the situation where the feature is *disabled*.

1. If the Toolbox window is currently visible and the pushpin icon in the window title is vertical, then the Auto Hide feature is disabled. (If the Toolbox window is not visible, click on *Toolbox* in the menu bar's *View* menu. If the pushpin icon is horizontal, then click on the icon to make it vertical.) When the Auto Hide feature is disabled, the Toolbox window stays fixed and is always ready for use.
2. Click the mouse cursor somewhere outside the Toolbox window and note that the Toolbox window stays fixed.
3. Click on the pushpin icon to make it horizontal. The Auto Hide feature is now *enabled*.
4. Move the mouse cursor somewhere outside the Toolbox window and note that the window slides into a tab on the left side of the screen. The name and icon of the Toolbox window appear on the tab.
5. Hover the mouse cursor over the tab. The window slides into view and is ready for use.

   Note: We recommend that you keep the Toolbox, Solution Explorer, and Properties windows displayed unless you are creating a program with a very large form and need extra space.

Positioning and Aligning Controls

VideoNote
Positioning and aligning controls

Visual Basic provides several tools for positioning and aligning controls on a form. **Proximity lines** are short line segments that help you place controls a comfortable distance from each other and from the sides of the form. **Snap lines** are horizontal and vertical line segments that help you align controls. The ***Format* menu** is used to align controls, center controls horizontally and vertically in a form, and make a group of selected controls the same size.

A Positioning and Aligning Walkthrough

1. Begin a new program.
2. Place a button near the center of the form.

3. Drag the button toward the upper-right corner of the form until two short line segments appear. See Fig. 2.14(a). The button is now a comfortable distance from the two sides of the form.
4. Place a second button below the first button and drag it upward until a proximity line appears between the two buttons. The buttons are now a comfortable distance apart.
5. Resize and position the two buttons as shown in Fig. 2.14(b).
6. Drag Button2 upward until a blue line appears along the bottoms of the two buttons. See Fig. 2.14(c). This blue line is called a **snap line**. The bottoms of the two buttons are now aligned.
7. Continue dragging Button2 upward until a purple snap line appears just underneath the words Button1 and Button2. See Fig. 2.14(d). The middles of the two buttons are now aligned. If we were to continue dragging Button2 upward, a blue snap line would tell us when the tops were aligned. Step 10 shows another way to align the controls.

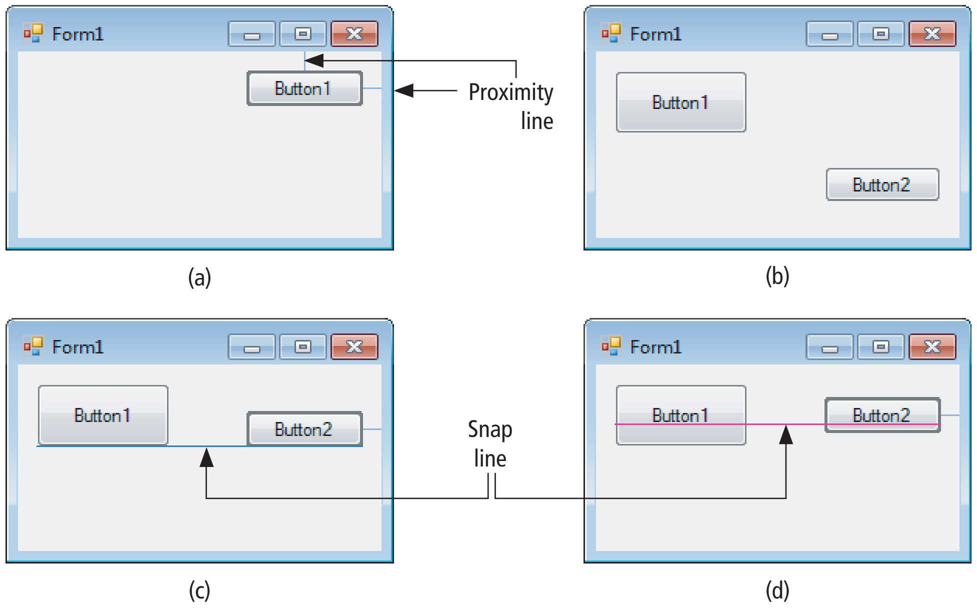


FIGURE 2.14 **Positioning controls.**

8. Click on Button1 and then hold down the Ctrl key and click on Button2. After the mouse button is released, both buttons will be selected.

   Note: This process (called **selection of multiple controls**) can be repeated to select a group of any number of controls.
9. With the two buttons still selected, press F4 to open the Properties window. Then set the ForeColor property to Blue. Notice that the ForeColor property has been altered for both buttons at the same time. Actually, any property that is common to every control in selected multiple controls can be set simultaneously for the entire group.
10. With the two buttons still selected, open the *Format* menu in the Menu bar, hover over *Align*, and click on *Tops*. The tops of the two buttons are now aligned. Precisely, Button1 (the first button selected) will stay fixed, and Button2 will move up so that its top is aligned with the top of Button1.

The most common uses of the submenus of the *Format* menu are as follows:

Align: Align middles or corresponding sides of a group of selected controls.

Make Same Size: Make the width and/or height of a group of selected controls the same.

Horizontal Spacing: Equalize the horizontal spacing between a group of three or more selected controls arranged in a row.

Vertical Spacing: Equalize the vertical spacing between a group of three or more selected controls arranged in a column.

Center in Form: Center a selected control either horizontally or vertically in a form.

When multiple controls are selected with the Ctrl key, the first control selected (called the **primary control** of the group) will have white sizing handles, while the other controls will have black sizing handles. All alignment and sizing statements initiated from the *Format* menu will keep the primary control fixed and will align (or size) the other controls with respect to the primary control.

After multiple controls have been selected, they can be dragged as a group and deleted as a group. Exercises 35 and 36 show how the arrow keys can be used to move and size a control. The arrow keys also can be used to move and size multiple controls as a group.

A group of controls also can be selected by clicking the mouse outside the controls, dragging it across the controls, and releasing it. The *Select All* command from the *Edit* menu (or the key combination Ctrl + A) causes all the controls on the form to be selected. Although these methods are easy to apply, they do not allow the programmer to designate the primary control.

Setting Tab Order

Each time the Tab key is pressed while a program is running, the focus moves from one control to another. The tab order for the controls is determined by the settings of their TabIndex properties. Initially, controls receive the focus in the order they were placed on the form. The first control placed on the form has a TabIndex setting of 0, the second control has a setting of 1, and so on. The tab order can be changed by renumbering the controls' TabIndex settings.

Whether or not a control can receive the focus by tabbing is determined by the setting of its TabStop property. By default, this setting is True for buttons, text boxes, and list boxes, and False for labels. In this book we always use these default settings. ***Note:*** Even though labels do not receive the focus while tabbing, they are still assigned a tab index.

Comments

1. While you are working on a program, the program resides in memory. Removing a program from memory is referred to as **closing** the program. A program is automatically closed when you begin a new program. Also, it can be closed directly with the *Close Project* command from the *File* menu.
2. Three useful properties that have not been discussed are the following:
   (a) BackColor: This property specifies the background color for the form or a control.
   (b) Visible: Setting the Visible property to False causes an object to disappear when the program is run. The object can be made to reappear with code.
   (c) Enabled: Setting the Enabled property of a control to False restricts its use. It appears grayed and cannot receive the focus. Controls sometimes are disabled temporarily if they do not apply to the current state of the program.
3. Most properties can be set or altered with code as the program is running instead of being preset from the Properties window. For instance, a button can be made to disappear with a line such as `Button1.Visible = False`. The details are presented in Section 2.3.
4. If you inadvertently double-click on a form, a window containing text will appear. (The first line is Public Class Form1.) This is the Code Editor, which is discussed in the next

section. Press Ctrl+Z to undo the addition of this new code. To return to the Form Designer, click on the tab at the top of the Document window labeled "Form1.vb [Design]."

5. We have seen two ways to place a control onto a form. Another way is to just click on the control in the Toolbox and then click on the location in the form where you would like to place the control. Alternatively, you can just drag the control from the Toolbox to the location in the form.
6. Figure 2.9 shows a small down-arrow button on the right side of the Text property setting box. When you click on that button, a rectangular box appears. The setting for the Text property can be typed into this box instead of into the Settings box. This method of specifying the setting is especially useful when you want the button to have a multiline caption.

Practice Problems 2.2

1. What is the difference between the Text and the Name properties of a button?
2. The first two group names in the Toolbox are *All Windows Forms* and *Common Controls*. How many groups are there?

EXERCISES 2.2

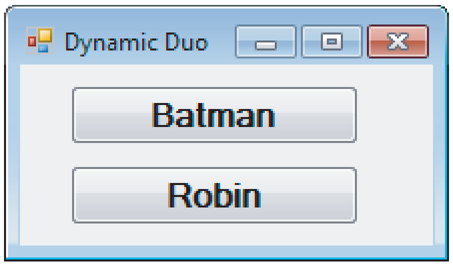
1. Create a form with two buttons, run the program, and click on each button. Do you notice anything different about a button after it has been clicked?
2. While a program is running, a control is said to lose focus when the focus moves from that control to another control. Give three ways the user can cause a control to lose focus.

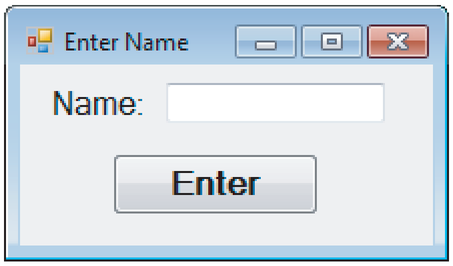
In Exercises 3 through 24, carry out the task.

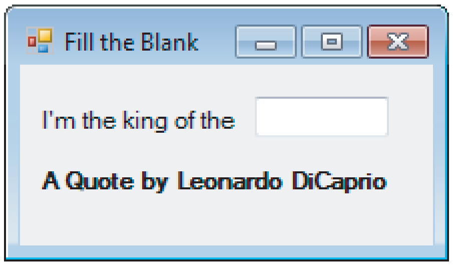
3. Place "CHECKING ACCOUNT" in the title bar of a form.
4. Create a text box containing the words "PLAY IT, SAM" in blue letters.
5. Create a text box with a yellow background.
6. Create a text box named txtGreeting and containing the word "HELLO" in large italic letters.
7. Create a label containing the sentence "After all is said and done, more is said than done." The sentence should occupy three lines, and each line should be centered horizontally in the label.
8. Create a read-only text box containing the words "Visual Basic" in bold white letters on a red background.
9. Create a text box named txtLanguage and containing the words "Visual Basic 2010" in Courier New font.
10. Create a yellow button named btnPush and containing the word "PUSH".
11. Create a white button containing the word "PUSH" in large italic letters.
12. Create a button containing the word "PUSH" in bold letters with the letter P underlined.
13. Create a button containing the word "PUSH" with the letter H as the access key.
14. Create a label containing the word "ALIAS" in white on a blue background.
15. Create a label named lblAKA and containing the centered italicized word "ALIAS".
16. Place "BALANCE SHEET" in the title bar of a form having a yellow background.
17. Create a label containing "VISUAL" on the first line and "BASIC" on the second line. Each word should be right-justified.

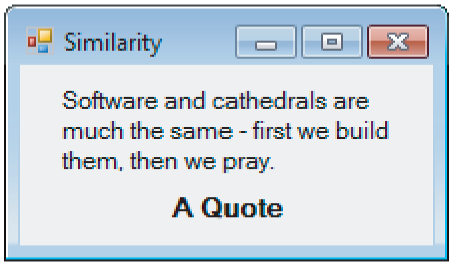
18. Create a form named frmHello whose title bar reads "Hello World".
19. Create a label containing a picture of a diskette. (***Hint:*** Use the Wingdings character <.) Make the diskette as large as possible.
20. Create a label containing the bold word "ALIAS" in the Courier New font.
21. Create a list box with a yellow background.
22. Create a list box that will be invisible when the program is run.
23. Create a form named frmYellow with a yellow background.
24. Create a button containing a picture of a red bell. (***Hint:*** Use the Wingdings character %.) Make the bell as large as possible.

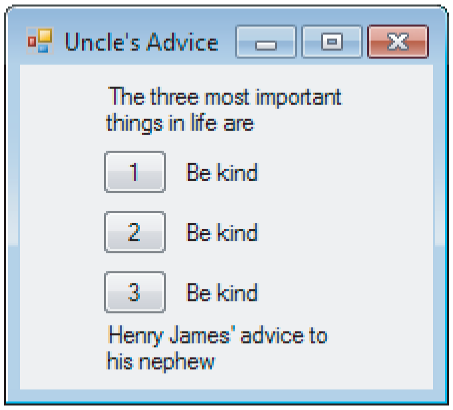
In Exercises 25 through 30, create the interface shown in that figure. (These exercises give you practice creating controls and assigning properties. The interfaces do not necessarily correspond to actual programs.)

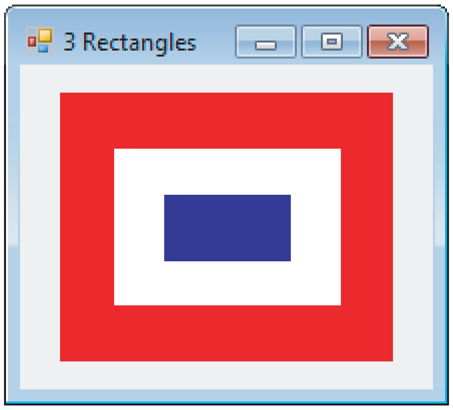
25.

Dynamic Duo
Batman
Robin

26.

Enter Name
Name:
Enter

27.

Fill the Blank
I'm the king of the
A Quote by Leonardo DiCaprio

28.

Similarity
Software and cathedrals are much the same - first we build them, then we pray.
A Quote

29.

Uncle's Advice
The three most important things in life are
1 Be kind
2 Be kind
3 Be kind
Henry James' advice to his nephew

30.

3 Rectangles

31. Create a replica of your bank check on a form. Words common to all checks, such as "PAY TO THE ORDER OF", should be contained in labels. Items specific to your checks, such as your name at the top left, should be contained in text boxes. Make the check on the screen resemble your personal check as much as possible. ***Note:*** Omit the account number.
32. Create a replica of your campus ID on a form. Words that are on all student IDs, such as the name of the college, should be contained in labels. Information specific to your ID, such as

your name and student ID number, should be contained in text boxes. Simulate your picture with a text box containing a smiling face—a size 24 Wingdings J.

33. Consider the form shown in Exercise 25. Assume the *Batman* button was added to the form before the *Robin* button. What is the tab index of the *Robin* button?

34. Consider the form shown in Exercise 26. Assume the first control added to the form was the label. What is the tab index of the label?

The following hands-on exercises develop additional techniques for manipulating and accessing controls placed on a form.

VideoNote
Moving a textbox (Homework)

35. Place a text box on a form and select the text box. What is the effect of pressing the various arrow keys?

36. Place a text box on a form and select the text box. What is the effect of pressing the various arrow keys while holding down the Shift key?

37. Repeat Exercise 36 for selected multiple controls.

38. Repeat Exercise 35 for selected multiple controls.

39. Place a label and a list box on a form and change their font sizes to 12 at the same time.

40. Place a button in the center of a form and select it. Hold down the Ctrl key and press an arrow key. Repeat this process for each of the other arrow keys. Describe what happens.

41. Place a label and a text box on a form with the label to the left of and above the text box. Select the label. Hold down the Ctrl key and press the down-arrow key twice. With the Ctrl key still pressed, press the right-arrow key. Describe what happens.

42. Place two buttons on a form with one button to the right of and below the other button. Select the lower button, hold down the Ctrl key, and press the left-arrow key. With the Ctrl key still pressed, press the up-arrow key. Describe the effect of pressing the two arrow keys.

43. Experiment with the *Align* command on the *Format* menu to determine the difference between the *center* and the *middle* of a control.

44. Place four large buttons vertically on a form. Use the *Format* menu to make them the same size and to make the spacing between them uniform.

45. Place a label and a text box on a form as in Exercise 26, and then lower the label slightly and lower the text box until it is about one inch lower than the label. Use the mouse to slowly raise the text box to the top of the form. Three snap lines will appear along the way: a blue snap line, a purple snap line, and finally another blue snap line. What is the significance of each snap line?

46. Place a text box on a form, select the text box, and open its Properties window. Double-click on the name (not the Settings box) of the ReadOnly property. Double-click again. What is the effect of double-clicking on a property whose possible settings are True and False?

47. Place a button on a form, select the button, and open its Properties window. Double-click on the name (not the Settings box) of the ForeColor property. Double-click repeatedly. Describe what is happening.

Solutions to Practice Problems 2.2

1. The text is the words appearing on the button, whereas the name is the designation used to refer to the button in code. Initially, they have the same value, such as Button1. However, each can be changed independently of the other.
2. The Toolbox in the Express Edition of Visual Basic contains 11 groups. Figure 2.15 shows the Toolbox after each group has been collapsed. ***Note:*** In the other editions of Visual Basic the Toolbox contains 12 groups.

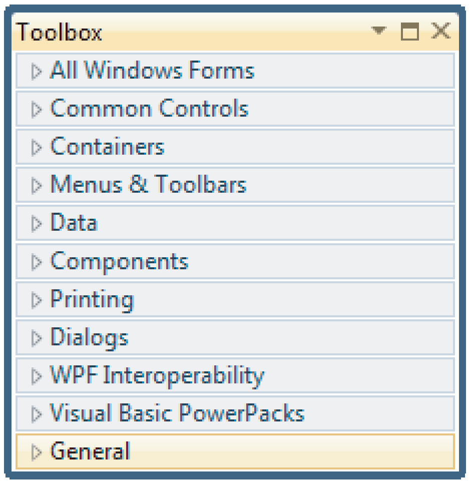


FIGURE 2.15 Toolbox group names.

2.3 Visual Basic Events

VideoNote
Event procedures

When a Visual Basic program runs, the form and its controls appear on the screen. Normally, nothing happens until the user takes an action, such as clicking a control or pressing a key. We call such an action an **event**. The programmer writes code that reacts to an event by performing some functionality.

The three steps in creating a Visual Basic program are as follows:

1. Create the interface; that is, generate, position, and size the objects.
2. Set properties; that is, configure the appearance of the objects.
3. Write the code that executes when events occur.

Section 2.2 covered Steps 1 and 2; this section is devoted to Step 3. Code consists of statements that carry out tasks. Writing code in Visual Basic is assisted by an autocompletion system called **IntelliSense** that reduces the amount of memorization needed and helps prevent errors. In this section, we limit ourselves to statements that change properties of a control or the form while a program is running.

Properties of controls are changed in code with statements of the form

```
controlName.property = setting
```

where *controlName* is the name of the control, *property* is one of the properties of the control, and *setting* is a valid setting for that property. Such statements are called **assignment statements**. They assign values to properties. Here are three examples of assignment statements:

1. The statement

   ```
   txtBox.Text = "Hello"
   ```

   displays the word *Hello* in the text box.
2. The statement

   ```
   btnButton.Visible = True
   ```

   makes the button visible.