OBJECTIVES

In this chapter you’ll learn:

- The basics of the Visual Studio Integrated Development Environment (IDE) that assists you in writing, running and debugging your Visual Basic programs.
- Visual Studio’s help features.
- Key commands contained in the IDE’s menus and toolbars.
- The purpose of the various kinds of windows in the Visual Studio 2010 IDE.
- What visual programming is and how it simplifies and speeds program development.
- To create, compile and execute a simple Visual Basic program that displays text and an image using the Visual Studio IDE and the technique of visual programming.
2.1 Introduction
2.2 Overview of the Visual Studio 2010 IDE
2.3 Menu Bar and Toolbar
2.4 Navigating the Visual Studio IDE
   2.4.1 Solution Explorer
   2.4.2 Toolbox
   2.4.3 Properties Window
2.5 Using Help
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image
2.1 Introduction

- **Visual Studio 2010** is Microsoft’s Integrated Development Environment (IDE) for creating, running and debugging programs (also called **applications**) written in various .NET programming languages.

- This chapter provides an overview of the Visual Studio 2010 IDE and shows how to create a simple Visual Basic program by dragging and dropping predefined building blocks into place—a technique known as **visual programming**.
2.2 Overview of the Visual Studio 2010 IDE

- This book’s examples are based on the Microsoft Visual Basic 2010 Express Edition, which supports only the Visual Basic programming language.
- See the Before You Begin section that follows the Preface for information on installing the software.
- Microsoft also offers a full version of Visual Studio 2010, which includes support for other languages in addition to Visual Basic, such as Visual C# and Visual C++.
- Our screen captures and discussions focus on the IDE of the Visual Basic 2010 Express Edition.
- The examples will work on most other current versions of Visual Basic.
2.2 Overview of the Visual Studio 2010 IDE

- Introduction to Microsoft Visual Basic 2010 Express Edition
  - We use the > character to indicate the selection of a menu item from a menu.
  - For example, we use the notation File > Open File… to indicate that you should select the Open File… menu item from the File menu.
  - Once the Express Edition begins execution, the Start Page displays.
  - The Start Page contains a list of links to Visual Studio 2010 IDE resources and web-based resources.
  - At any time, you can return to the Start Page by selecting View > Start Page.
2.2 Overview of the Visual Studio 2010 IDE

Links on the Start Page

- The Start Page links are organized into sections—Recent Projects, Get Started and Latest News—that contain links to helpful programming resources.
- Clicking any link on the Start Page displays relevant information associated with the specific link.
- [Note: An Internet connection is required for the IDE to access some of this information.]
Fig. 2.1 | Start Page in Visual Basic 2010 Express Edition
The Recent Projects section contains information on projects you’ve recently created or modified.

You can also open existing projects or create new ones by clicking the links above this section.

The Get Started section focuses on using the IDE for creating programs and learning Visual Basic.

The Latest News tab provides links to the latest Visual Basic developments (such as updates and bug fixes) and to information on advanced programming topics.
2.2 Overview of the Visual Studio 2010 IDE

- The MSDN site contains articles, downloads and tutorials on technologies of interest to Visual Studio developers.

- You can also browse the web from the IDE by selecting View > Other Windows > Web Browser or by typing $<Ctrl> <Alt> R$.

- To request a web page, type its URL into the location bar and press the Enter key—your computer, of course, must be connected to the Internet.

- The web page that you wish to view appears as another tab in the IDE.
Fig. 2.2 | Displaying a web page in Visual Studio.
Customizing the IDE and Creating a New Project

- To begin programming in Visual Basic, you must create a new project or open an existing one.
- Select either File > New Project… to create a new project or File > Open Project… to open an existing project.
- From the Start Page, above the Recent Projects section, you can also click the links New Project… or Open Project….
- A project is a group of related files, such as the Visual Basic code and any images that might make up a program.
- Visual Studio 2010 organizes programs into projects and solutions, which contain one or more projects.
Multiple-project solutions are for large-scale programs. Each program we create consists of a single project. When you select File > New Project… or click the New Project… link on the Start Page, the New Project dialog displays.

Dialogs are windows that facilitate user–computer communication.

Visual Studio provides several templates. Templates are the project types users can create in Visual Basic—Windows Forms applications, console applications, WPF applications and others.
2.2 Overview of the Visual Studio 2010 IDE

- In this chapter, we build a Windows Forms Application.
- A Windows Forms application executes within a Windows operating system (such as Windows 7 or Windows Vista) and typically has a graphical user interface (GUI)—the visual part of the program with which the user interacts.
- Windows applications include Microsoft software products like Microsoft Word, Internet Explorer and Visual Studio; software products created by other vendors; and customized software that you and other programmers create.
2.2 Overview of the Visual Studio 2010 IDE

- By default, Visual Studio assigns the name WindowsApplication1 to a new Windows Forms Application project and solution.
- Select Windows Forms Application, then click OK to display the IDE in Design view, which contains the features that enable you to create programs.
- The IDE’s Design view is the Windows Forms Designer.
Fig. 2.3 | New Project dialog.
Fig. 2.4 | Design view of the IDE.
2.2 Overview of the Visual Studio 2010 IDE

- The rectangle in the Design area titled Form1 (called a Form) represents the main window of the Windows Forms application that you’re creating.
- Visual Basic applications can have multiple Forms (windows)—but we’ll use only one Form.
- You’ll learn how to customize the Form by adding GUI controls—in this example, you’ll add a Label and a PictureBox (as you’ll see in ).
- A Label typically contains descriptive text (for example, "Welcome to Visual Basic!"), and a PictureBox displays an image, such as the Deitel bug mascot.
Visual Basic Express has many preexisting controls and other components you can use to build and customize your programs.

In this chapter, you’ll work with preexisting controls from the .NET Framework Class Library.

As you place controls on the **Form**, you’ll be able to modify their properties (discussed in ).

For example, shows where the **Form**’s title can be modified and shows a dialog in which a control’s font properties can be modified.
**Fig. 2.5** Textbox control for modifying a property in the Visual Studio IDE.
Collectively, the Form and controls make up the program’s GUI.

Users enter data (inputs) into the program by typing at the keyboard, by clicking the mouse buttons and in a variety of other ways.

Programs use the GUI to display instructions and other information (outputs) for users to view.

For example, the New Project dialog in presents a GUI where the user clicks the mouse button to select a template type, then inputs a project name from the keyboard (the figure is still showing the default project name WindowsApplication1 supplied by Visual Studio).
2.2 Overview of the Visual Studio 2010 IDE

- Each open document’s name is listed on a tab.
- To view a document when multiple documents are open, click its tab.
- Tabs facilitate easy access to multiple open documents.
- The active tab (the tab of the currently displayed document) is highlighted in yellow (for example, Form1.vb [Design] in ).
Fig. 2.6 | Dialog for modifying a control’s font properties.
2.3 Menu Bar and Toolbar

- Commands for managing the IDE and for developing, maintaining and executing programs are contained in menus, which are located on the menu bar of the IDE.
- The set of menus displayed depends on what you’re currently doing in the IDE.
Fig. 2.7 | Visual Studio menu bar.
2.3 Menu Bar and Toolbar

- Menus contain groups of related commands (also called menu items) that, when selected, cause the IDE to perform specific actions (for example, open a window, save a file, print a file and execute a program).
- For example, new projects are created by selecting File > New Project….
- The menus depicted in  are summarized in .
<table>
<thead>
<tr>
<th>Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>File</td>
<td>Contains commands for opening, closing, adding and saving projects, as well as printing project data and exiting Visual Studio.</td>
</tr>
<tr>
<td>Edit</td>
<td>Contains commands for editing programs, such as cut, copy, paste, undo, redo, delete, find and select.</td>
</tr>
<tr>
<td>View</td>
<td>Contains commands for displaying IDE windows (for example, Solution Explorer, Toolbox, Properties window) and for adding toolbars to the IDE.</td>
</tr>
<tr>
<td>Project</td>
<td>Contains commands for managing projects and their files.</td>
</tr>
<tr>
<td>Debug</td>
<td>Contains commands for compiling, debugging (that is, identifying and correcting problems in programs) and running programs.</td>
</tr>
<tr>
<td>Data</td>
<td>Contains commands for interacting with databases (that is, organized collections of data stored on computers), which we discuss in Chapter 12, Databases and LINQ.</td>
</tr>
<tr>
<td>Format</td>
<td>Contains commands for arranging and modifying a Form’s controls. The Format menu appears only when a GUI component is selected in Design view.</td>
</tr>
<tr>
<td>Tools</td>
<td>Contains commands for accessing additional IDE tools and options for customizing the IDE.</td>
</tr>
</tbody>
</table>

**Fig. 2.8** | Summary of Visual Studio 2010 IDE menus. (Part 1 of 2.)
<table>
<thead>
<tr>
<th>Menu</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window</td>
<td>Contains commands for hiding, opening, closing and displaying IDE windows.</td>
</tr>
<tr>
<td>Help</td>
<td>Contains commands for accessing the IDE’s help features.</td>
</tr>
</tbody>
</table>

**Fig. 2.8** | Summary of Visual Studio 2010 IDE menus. (Part 2 of 2.)
2.3 Menu Bar and Toolbar

- You can access many of the more common menu commands from the toolbar, which contains graphics, called icons, that graphically represent commands.
- By default, the standard toolbar is displayed when you run Visual Studio for the first time—it contains icons for the most commonly used commands, such as opening a file, adding an item to a project, saving files and running applications.
- Some commands are initially disabled (grayed out or unavailable to use) and are enabled only when needed.
- For example, Visual Studio enables the command for saving a file once you begin editing a file.
Fig. 2.9 | Standard Visual Studio toolbar.
2.3 Menu Bar and Toolbar

- You can customize the IDE’s toolbars.
- Select View > Toolbars.
- Each toolbar you select is displayed with the other toolbars at the top of the Visual Studio window.
- To execute a command via the toolbar, click its icon.
- Some icons contain a down arrow that you can click to display related commands, as shown in .
Fig. 2.10 | Adding the Build toolbar to the IDE.
Fig. 2.11 | IDE toolbar icon showing additional commands.
2.3 Menu Bar and Toolbar

- It can be difficult to remember what each toolbar icon represents.
- Hovering the mouse pointer over an icon highlights it and, after a brief pause, displays a description of the icon called a tool tip.
- **Tool tips** help you become familiar with the IDE’s features and serve as useful reminders for each toolbar icon’s functionality.
**Fig. 2.12**  |  Tool tip demonstration.
2.4 Navigating the Visual Studio IDE

- The IDE provides windows for accessing project files and customizing controls.
- This section introduces several windows that you’ll use frequently when developing Visual Basic programs.
- These windows can be accessed via the toolbar icons or by selecting the desired window’s name in the View menu.
Fig. 2.13  |  Toolbar icons for four Visual Studio windows.
Visual Studio provides a space-saving feature called **auto-hide**.

When auto-hide is enabled, a tab appears along either the left, right or bottom edge of the IDE window.

This tab contains one or more icons, each of which identifies a hidden window.

Placing the mouse pointer over one of these icons displays that window.

Moving the mouse pointer outside the window’s area hides the window.

To “pin down” a window (that is, to disable auto-hide and keep the window open), click the pin icon.

When auto-hide is enabled, the pin icon is horizontal — when a window is “pinned down,” the pin icon is vertical.
Fig. 2.14  |  Auto-hide feature demonstration.
Fig. 2.15 | Displaying a hidden window when auto-hide is enabled.
Fig. 2.16 | Disabling auto-hide ("pinning down" a window).
2.4 Navigating the Visual Studio IDE

- The next few sections cover three of Visual Studio’s main windows—the Solution Explorer, the Properties window and the Toolbox.
- These windows display project information and include tools that help you build your programs.
2.4.1 Solution Explorer

- The Solution Explorer window provides access to all of a solution’s files.
- If it’s not shown in the IDE, click the Solution Explorer icon in the IDE, select View > Other Windows > Solution Explorer or type <Ctrl> <Alt> L.
- When you open a new or existing solution, the Solution Explorer displays the solution’s contents.
Fig. 2.17 | Solution Explorer window with an open project.
The solution’s **startup project** is the one that runs when you select **Debug > Start Debugging** (or press the **F5 key**).

For a single-project solution like the examples in this book, the startup project is the only project (in this case, **WindowsApplication1**) and the project name appears in bold text in the **Solution Explorer** window.

When you create an application for the first time, the **Solution Explorer** window lists only the project’s **My Project** and **Form1.vb** files.

The Visual Basic file that corresponds to the **Form** shown in  is named **Form1.vb** (selected in ).

Visual Basic files use the **.vb** file-name extension, which is short for “Visual Basic.”
2.4.1 Solution Explorer

- By default, the IDE displays only files that you may need to edit—other files that the IDE generates are hidden.

- The Solution Explorer window includes a toolbar that contains several icons.

- Clicking the Show All Files icon displays all the solution’s files, including those generated by the IDE.

- Clicking the arrows to the left of a file or folder expands or collapses the project tree’s nodes.
**Fig. 2.18** | Solution Explorer with the References node expanded.
2.4.1 Solution Explorer

- Try clicking the arrow to the left of References to display items grouped under that heading.
- Click the arrow again to collapse the tree.
- Other Visual Studio windows also use this convention.
2.4.2 Toolbox

- The Toolbox (View > Other Windows > Toolbox) contains icons representing controls used to customize Forms.
- With visual programming, you can “drag and drop” controls onto the Form and the IDE will write the code that creates the controls for you.
- Just as you don’t need to know how to build an engine to drive a car, you don’t need to know how to build controls to use them.
- Reusing preexisting controls saves time and money when you develop programs.
- You’ll use the Toolbox when you create your first program later in the chapter.
Fig. 2.19 | Toolbox window displaying controls for the Common Controls group.
2.4.2 Toolbox

- The Toolbox groups the prebuilt controls into categories—All Windows Forms, Common Controls, Containers, Menus & Toolbars, Data, Components, Printing, Dialogs, WPF Interoperability, Visual Basic Power Packs and General are listed in.
- Again, note the use of arrows, which can expand or collapse a group of controls.
To display the Properties window, select View > Other Windows > Properties Window, click the Properties window toolbar icon shown in , or press the F4 key.

The Properties window displays the properties for the currently selected Form control or file in Design view.

Properties specify information about the Form control, such as its size, color and position.

Each Form control has its own set of properties—a property’s description is displayed at the bottom of the Properties window whenever that property is selected.
Fig. 2.20 | Properties window.
2.4.3 Properties Window

- shows Form's Properties window.
- The left column lists the Form’s properties—the right column displays the current value of each property.
- You can sort the properties either alphabetically (by clicking the Alphabetical icon) or categorically (by clicking the Categorized icon).
- The properties can be sorted alphabetically from A–Z or Z–A—sorting by category groups the properties according to their use (that is, Appearance, Behavior, Design, etc.).
- Depending on the size of the Properties window, some of the properties may be hidden from view on the screen.
2.4.3 Properties Window

- Users can scroll through the list of properties by dragging the scrollbox up or down inside the scrollbar, or by clicking the arrows at the top and bottom of the scrollbar.
- We show how to set individual properties later in this chapter.
- The Properties window is crucial to visual programming—it allows you to modify a control’s properties visually, without writing code.
- You can see which properties are available for modification and, in many cases, can learn the range of acceptable values for a given property.
2.4.3 Properties Window

- The Properties window displays a brief description of the selected property, helping you understand its purpose.
- A property can be set quickly using this window, and no code needs to be written.
- At the top of the Properties window is the component selection drop-down list, which allows you to select the Form or control whose properties you wish to display in the Properties window.
- Using the component selection drop-down list is an alternative way to display a Form’s or control’s properties without clicking the actual Form or control in the GUI.
2.5 Using Help

- Microsoft provides extensive help documentation via the Help menu.
- Using Help is an excellent way to get information quickly about Visual Studio, Visual Basic and more.
- Before using Help the first time, you must configure it as follows:
  - Select Help > Manage Help Settings to display the Help Library Manager. The first time you do this, the dialog in will appear. Click OK to select the default location for help content that is stored on your local computer. If a dialog appears with the message Do you want to allow the following program to make changes to this computer?, click Yes.
**Fig. 2.21** | Help Library Manager window the first time you select Help > Manage Help Settings.
2.5 Using Help

- In the Help Library Manager window, click Choose online or local help. Accessing online help requires an Internet connection, but gives you access to the most up-to-date documentation, as well as tutorials, downloads, support, forums and more. Accessing local help requires that you first download the help files, which can take considerable time and use a significant amount of disk space. If possible, we recommend that you use the online help.
Fig. 2.22  |  Preparing to select online or local help.
2.5 Using Help

- Select I want to use online help, then click OK. Next, click Exit in the Help Library Manager window. Your IDE is now configured to use online help.
Fig. 2.23  |  Selecting online help.
2.5 Using Help

**Context-Sensitive Help**

- Visual Studio provides context-sensitive help pertaining to the “current content” (that is, the items around the location of the mouse cursor).
- To use context-sensitive help, click an item, such as the `Form` then press the `F1 key`.
- The help documentation is displayed in a web browser window.
- To return to the IDE, either close the browser window or select the icon for the IDE in your Windows task bar.
- Shows the help page for a `Form`'s `Text` property.
- You can view this help by selecting the `Form` clicking its `Text` property in the `Properties` window and pressing the `F1` key.
Fig. 2.24 | Using context-sensitive help.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- Next, we create a program that displays the text "Welcome to Visual Basic!" and an image of the Deitel & Associates bug mascot.
- The program consists of a single Form that uses a Label and a PictureBox.
- Shows the result of the program as it executes.
- The program and the bug image are available with this chapter’s examples.
- You can download the examples from www.deitel.com/books/vb2010http/.
- We assume the examples are located at C:\examples on your computer.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- You won’t write a single line of program code.
- Instead, you’ll use visual programming techniques.
- Visual Studio processes your actions (such as mouse clicking, dragging and dropping) to generate program code.
- Chapter 3 begins our discussion of writing program code.
- Throughout the book, you produce increasingly substantial and powerful programs that usually include a combination of code written by you and code generated by Visual Studio.
- The generated code can be difficult for novices to understand—but you’ll rarely need to look at it.
Fig. 2.25 | Simple program executing.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- Visual programming is useful for building GUI-intensive programs that require user interaction.
- To create, save, run and terminate this first program, perform the following steps:
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- **Closing the open project.** If a project is already open, close it by selecting File > Close Project. A dialog asking whether to save the current project might appear. Click Save to save your changes or Discard to ignore them.

- **Creating the new project.** To create a new Windows Forms application for the program, select File > New Project… to display the New Project dialog. Select Windows Forms Application. Name the project ASimpleProgram and click OK.
Fig. 2.26 | New Project dialog.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- **Saving the project.** We mentioned earlier in this chapter that you must set the directory in which the project is saved. To specify the directory in Visual Basic 2010 Express, select File > Save All to display the Save Project dialog.
By default, projects are saved to your user directory in the folder `Documents\Visual Studio 2010\Projects`. To change the project location, click the **Browse…** button, which opens the **Project Location dialog**. Navigate through the directories, select one in which to place the project (in our example, we use the directory `C:\MyVBProjects`) and click **Select Folder** to close the dialog. Click **Save** in the **Save Project** dialog to save the project and close the dialog.
Fig. 2.27  |  Save Project dialog.
Fig. 2.28  | Setting the project location in the Project Location dialog.
When you first begin working in the IDE, it is in design mode (that is, the program is being designed and is not executing).

This provides access to all the environment windows (for example, Toolbox, Properties), menus and toolbars, as you’ll see shortly.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- **Setting the text in the Form’s title bar.** The text in the Form’s title bar is determined by the Form’s Text property. If the Properties window is not open, click the properties icon in the toolbar or select View > Other Windows > Properties Window. Click anywhere in the Form to display the Form’s properties in the Properties window. In the textbox to the right of the Text property, type "A Simple Program", as in . Press the Enter key—the Form’s title bar is updated immediately.
Fig. 2.29 | Setting the Form’s Text property in the Properties window.
Fig. 2.30  |  Form with enabled sizing handles.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- *Resizing the Form.* Click and drag one of the Form’s enabled *sizing handles* (the small white squares that appear around the Form). Using the mouse, select the bottom-right sizing handle and drag it down and to the right to make the Form larger.
Fig. 2.31  |  Resized Form.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- **Changing the Form’s background color.** The BackColor property specifies a Form’s or control’s background color. Clicking BackColor in the Properties window causes a down-arrow button to appear next to the value of the property. When clicked, the down-arrow button displays other options, which vary depending on the property. In this case, the arrow displays tabs for Custom, Web and System (the default). Click the Custom tab to display the palette (a grid of colors). Select the box that represents light blue. Once you select the color, the palette closes and the Form’s background color changes to light blue.
Fig. 2.32  | Changing the Form’s BackColor property.
**Fig. 2.33**  |  Form with new BackColor property applied.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

Adding a **Label control to the Form.** If the Toolbox is not already open, select View > Other Windows > Toolbox to display the set of controls you’ll use for creating your programs. For the type of program we’re creating in this chapter, the typical controls we use are located in either the **All Windows Forms** group of the Toolbox or the **Common Controls** group. If either group name is collapsed, expand it by clicking the arrow to the left of the group name.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- Next, double click the **Label** control in the **Toolbox**. This action causes a **Label** to appear in the upper-left corner of the **Form**.
  - [Note: If the **Form** is behind the **Toolbox**, you may need to hide the **Toolbox** to see the **Label**.]

- Although double clicking any **Toolbox** control places the control on the **Form**, you also can “drag” controls from the **Toolbox** to the **Form**—you may prefer dragging the control because you can position it wherever you want.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- The Label displays the text Label1 by default. The Label’s background color is the same as the Form’s background color. When a control is added to the Form, its BackColor property is set to the Form’s BackColor or . You can change the Label’s background color by changing its BackColor or property.
Fig. 2.34 | Adding a Label to the Form.
Customizing the Label’s appearance. Select the Label by clicking it. Its properties now appear in the Properties window. The Label’s Text property determines the text (if any) that the Label displays. The Form and Label each have their own Text property—Forms and controls can have the same types of properties (such as BackColor, Text, etc.) without conflict. Set the Label’s Text property to Welcome to Visual Basic!. The Label resizes to fit all the typed text on one line.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- By default, the AutoSize property of the Label is set to True, which allows the Label to update its size to fit all of the text if necessary. Set the AutoSize property to False so that you can resize the Label on your own. Resize the Label (using the sizing handles) so that the text fits. Move the Label to the top center of the Form by dragging it or by using the keyboard’s left and right arrow keys to adjust its position. Alternatively, when the Label is selected, you can center the Label control horizontally by selecting Format > Center In Form > Horizontally.
Fig. 2.35 | Changing the Label's AutoSize property to False.
Fig. 2.36 | GUI after the Form and Label have been customized.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- Setting the Label’s font size. To change the font type and appearance of the Label’s text, select the value of the Font property, which causes an ellipsis button to appear next to the value. When the ellipsis button is clicked, a dialog that provides additional values—in this case, the Font dialog—is displayed. You can select the font name (the font options may be different, depending on your system-), font style (Regular, Italic, Bold, etc.) and font size (16, 18, 20, etc.) in this dialog.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- The Sample text shows the elected font settings. Under Font, select **Segoe UI**, Microsoft’s recommended font for user interfaces. Under Size, select **24** points and click **OK**. If the Label’s text does not fit on a single line, it wraps to the next line. Resize the Label vertically if it’s not large enough to hold the text. You may need to center the Label horizontally again after resizing.
**Fig. 2.37** Properties window displaying the Label’s Font property.
Fig. 2.38 | Font dialog for selecting fonts, styles and sizes.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- **Aligning the Label’s text.** Select the Label’s TextAlign property, which determines how the text is aligned within the Label. A three-by-three grid of buttons representing alignment choices is displayed. The position of each button corresponds to where the text appears in the Label.
For this program, set the **TextAlign** property to **MiddleCenter** in the three-by-three grid—this selection causes the text to appear centered in the middle of the **Label**, with equal spacing from the text to all sides of the **Label**. The other **TextAlign** values, such as **TopLeft**, **TopRight**, and **BottomCenter**, can be used to position the text anywhere within a **Label**. Certain alignment values may require that you resize the **Label** larger or smaller to better fit the text.
Fig. 2.39  |  Centering the Label’s text.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- Adding a **PictureBox** to the Form. The **PictureBox** control displays images. The process involved in this step is similar to that of Step 7, in which we added a **Label** to the **Form**. Locate the **PictureBox** in the **Toolbox** and double click it to add it to the **Form**. When the **PictureBox** appears, move it underneath the **Label**, either by dragging it or by using the arrow keys.
Fig. 2.40 | Inserting and aligning a PictureBox.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- **Inserting an image.** Click the **PictureBox** to display its properties in the **Properties** window. Locate the **Image property**, which displays a preview of the selected image or *(none)* if no image is selected. Click the ellipsis button to display the **Select Resource dialog**, which is used to import files, such as images, for use in a program. Click the **Import…** button to browse for an image to insert, select the image file and click **OK**.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- We used bug.png from this chapter’s examples folder. The image is previewed in the Select Resource dialog. Click OK to use the image. Supported image formats include PNG (Portable Network Graphics), GIF (Graphic Interchange Format), JPEG (Joint Photographic Experts Group) and BMP (Windows bitmap). To scale the image to the PictureBox’s size, change the SizeMode property to StretchImage. Resize the PictureBox, making it larger.
Fig. 2.41  |  Image property of the PictureBox.
Fig. 2.42  |  Select Resource dialog to select an image for the PictureBox.
Fig. 2.43 | Select Resource dialog displaying a preview of selected image.
Fig. 2.44 | Scaling an image to the size of the PictureBox.
Fig. 2.45 | PictureBox displaying an image.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- **Saving the project.** Select File > Save All to save the entire solution. The solution file (which has the file name extension `.sln`) contains the name and location of its project, and the project file (which has the file name extension `.vbproj`) contains the names and locations of all the files in the project. If you want to reopen your project at a later time, simply open its `.sln` file.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- **Running the project.** Recall that up to this point we have been working in the IDE design mode (that is, the program being created is not executing). In **run mode**, the program is executing, and you can interact with only a few IDE features—features that are not available are disabled (grayed out). The text **Form1.vb [Design]** in the project tab means that we’re designing the **Form** visually rather than programmatically. If we had been writing code, the tab would have contained in the tab, the file has been changed and should be saved.
Select Debug > Start Debugging- to execute the program (or you can press the *F5* key). shows the IDE in run mode (indicated by the title-bar text ASimpleProgram (Running) – Microsoft Visual Basic 2010 Express Edition). Many toolbar icons and menus are disabled, since they cannot be used while the program is running.
Fig. 2.46 | Debugging a solution.
IDE displays text **Running**, which signifies that the program is executing.

**Fig. 2.47** IDE in run mode, with the running program in the foreground.
2.6 Using Visual Programming to Create a Simple Program that Displays Text and an Image

- **Terminating execution.** Click the running program’s close box (the X in the top-right corner of the running program’s window). This action stops the program’s execution and returns the IDE to design mode. You can also select **Debug > Stop Debugging** to terminate the program.