An Introduction to Visual Studio 2017 and Visual Basic

In this chapter’s Focus on the Concepts lesson, you will learn the definitions for many of the terms used by programmers. The lesson also introduces you to Microsoft’s newest integrated development environment (IDE): Visual Studio 2017. The IDE contains the latest version of the Visual Basic programming language. You will use the IDE and language to create the applications in this book. As stated in the Read This Before You Begin section of this book, the steps and figures in this book assume you are using the Community edition of Visual Studio 2017. Your steps and screen might differ slightly in some instances if you are using a different edition of Visual Studio 2017.

In the Apply the Concepts lesson, you will apply the concepts covered in the Focus lesson. The Apply lesson is designed to help you get comfortable with both the Visual Studio IDE and the Visual Basic programming language.
FOCUS ON THE CONCEPTS LESSON

Concepts covered in this lesson:

- F-1 Computer programming terminology
- F-2 The programmer's job
- F-3 The Visual Basic programming language
- F-4 The Visual Studio IDE
- F-5 Assigning names to objects

F-1 Computer Programming Terminology

In essence, the word **programming** means *giving a mechanism the directions to accomplish a task*. When the mechanism is a computer, the directions are typically referred to as instructions. A set of instructions that tells a computer how to accomplish a task is called a **computer program** or, more simply, a **program**.

Programs are written by **programmers** using a variety of special languages called **programming languages**. Some popular programming languages are Visual Basic, C#, C++, and Java. In this book, you will write your programs using the Visual Basic programming language, which is built into Microsoft's newest integrated development environment: Visual Studio 2017. An **integrated development environment (IDE)** is an environment that contains all of the tools and features you need to create, run, and test your programs. You also will use the IDE to create graphical user interfaces for your programs. A **graphical user interface (GUI)** is what the person using your program (referred to as the user) sees and interacts with while your program is running. The user interface and its program instructions are referred to as an **application**.

Figure 1-1 shows the user interface and program instructions for the Good Morning application.
To run the Good Morning application:

1. Use Windows to locate and then open the VB2017\Chap01 folder on your computer’s hard disk or on the device designated by your instructor. Double-click Good Morning.exe in the list of filenames. (Depending on how Windows is set up on your computer, you might not see the .exe extension on the filename. Refer to the Read This Before You Begin section to learn how to show filename extensions.) The application’s user interface appears on the screen with a blinking “It’s time to get up!!!” message.

2. Click the Exit button to close the application.

F-2 The Programmer’s Job

When a company has a problem that requires a computer solution, typically it is a programmer who comes to the rescue. The programmer might be an employee of the company; or he or she might be a freelance programmer, who works on temporary contracts rather than for a long-term employer.

First, the programmer meets with the person (or people) responsible for describing the problem. This person might be the one who will eventually use the solution; or he or she might be a software developer, who serves as an intermediary between the user and the programmer. The software developer will meet with the user and then outline the problem specification for the programmer. After the programmer understands the problem, he or she will begin planning an appropriate solution. After the planning is complete, the programmer will translate the solution into computer instructions—a process called coding. The programmer then will test the program rigorously with sample data to make sure it works both correctly and to the user’s satisfaction. Depending on the complexity of the problem, multiple programmers might be involved in the planning and coding phases. Programming teams often contain subject matter experts, who might or might not be programmers. For example, an accountant might be part of a team working on a program that requires accounting expertise.

F-3 The Visual Basic Programming Language

Visual Basic is an object-oriented programming language, which is a language that allows the programmer to use objects to accomplish a program’s goal. In object-oriented programming, or OOP, an object is anything that can be seen, touched, or used. In other words, an object is nearly any thing. Programs written for the Windows environment typically use objects such as check boxes, list boxes, and buttons.

Every object in an object-oriented program is created from a class, which is a pattern that the computer uses to create the object. The class contains the instructions that tell the computer how the object should look and behave. An object created from a class is called an instance of the class and is said to be instantiated from the class. An analogy involving a cookie cutter and cookies is often used to describe a class and its objects: The class is the cookie cutter, and the objects instantiated from the class are the cookies. You will learn more about classes and objects throughout this book.

In the following set of steps, you will run two Visual Basic applications that employ many of the objects you will learn about in this book.
CHAPTER 1
An Introduction to Visual Studio 2017 and Visual Basic

To run the applications:

1. If necessary, open the VB2017\Chap01 folder. Double-click Payment.exe. After a few moments, the Monthly Payment Calculator application shown in Figure 1-2 appears on the screen. (If some of the letters on your screen are not underlined, press the Alt key.) The interface contains a text box, a list box, buttons, radio buttons, and labels. You can use the application to calculate the monthly payment for a car loan.

![Figure 1-2 Monthly Payment Calculator application](image)

2. First, you will use the application to calculate the monthly payment for a $15,000 loan at 3.5% interest for five years. Type 15000 in the Principal text box. Scroll down the Interest list box and then click 3.50%. Finally, click the Calculate button. The application indicates that your monthly payment would be $272.88. See Figure 1-3.

![Figure 1-3 Computed monthly payment](image)

3. Next, you will determine what your monthly payment would be if you borrowed $4,500 at 2.75% interest for four years. Type 4500 in the Principal text box, click 2.75% in the Interest list box, click the 4 years radio button, and then click the Calculate button. The Monthly payment box shows $99.11.
4. Click the Exit button to close the application.

5. Now double-click Einstein.exe. (The file is located in the VB2017\Chap01 folder.)
   Click the Show equation button to display Einstein’s famous equation. See Figure 1-4.
   The interface contains picture boxes and buttons. (If some of the letters on your screen
   are not underlined, press the Alt key.)

![Einstein’s Famous Equation]

Figure 1-4 Equation shown in the interface

6. Click the Hide equation button to hide the equation, and then click the Exit button to
   close the application.

Mini-Quiz 1-1

1. What is a computer program?
2. What is a GUI?
3. What is the process of translating a solution into a computer program called?
4. In object-oriented programming, what is a class?
F-4 The Visual Studio IDE

The Visual Studio IDE contains many different windows, each with its own special purpose. The four windows you will use most often when designing your user interfaces are shown in Figure 1-5.

The **designer window** is where you create (or design) your application’s GUI. A Windows Form object, or form, appears in the designer window shown in Figure 1-5. A **form** is the foundation for the user interface in an application created for the Windows environment. As you learned earlier, all objects in an object-oriented program are instantiated (created) from a class. A form, for example, is an instance of the Windows Form class. The form (an object) is automatically instantiated for you when you create a Windows Forms application in Visual Basic.

You use the **Toolbox window** to add other objects, called **controls**, to the form. Each tool listed in the Toolbox window represents a class. You add an object by clicking its corresponding tool (class) in the toolbox and then dragging it with your mouse pointer to the form. When you drag the tool to the form, Visual Basic creates (instantiates) an instance of the class (an object) and places it on the form. For example, the two picture box objects shown in Figure 1-5 were instantiated (created) by dragging the PictureBox tool from the toolbox to the form. Similarly, the three button objects were instantiated using the Button tool.

Each object has a set of attributes that determine its appearance and behavior. The attributes, called **properties**, are listed in the **Properties window** when the object is selected in the designer window. In Figure 1-5, the form is selected, and the names of its properties (such as StartPosition and Text), along with their values (CenterScreen and Einstein’s Famous Equation), appear in the Properties window. You can use the Properties window to change the value of an object’s property. For example, you can use it to change the form’s Text property, which appears in the form’s title bar, from Einstein’s Famous Equation to Guess My Equation.

Windows applications in Visual Basic are composed of solutions, projects, and files. A solution is a container that stores the projects and files for an entire application. A project is also a container, but it stores only the files associated with that particular project. The **Solution Explorer** window displays a list of the projects contained in the current solution and the items contained in each project. The Solution Explorer window shown in Figure 1-5 indicates that the Einstein Solution contains the Einstein Project, which contains several items. The Einstein.png
and Equation.png items are the names of files on your disk. These files contain the images that appear in the picture boxes on the form. The Main Form.vb item is also the name of a file on your disk. The Main Form.vb file stores the program instructions (code) that tell the three buttons how to respond when the user clicks them. You enter the code in the Code Editor window, which is shown in Figure 1-6.

![Code Editor window](image.png)

**Figure 1-6  Code Editor window**

At this point, you are not expected to understand the contents of the Code Editor window in Figure 1-6; you will learn about the contents in this chapter’s Apply the Concepts lesson. However, briefly, the green lines of text are comments and are not executed by the computer when the application is run; they serve simply to internally document the program. The code on Lines 6 through 9 tell the computer to close (end) the application when the Exit button is clicked. The code on Lines 11 through 16 indicate that the computer should hide the equation picture box when the Hide equation button is clicked. The code on Lines 18 through 23 tell the computer to show the equation picture box when the Show equation button is clicked.

### F-5 Assigning Names to Objects

As mentioned earlier, each object has a set of properties attached to it. One of the most important of these properties is the Name property. This is because you use the **Name property** to refer to the object in code. The code in Figure 1-6, for example, refers to objects named frmMain, btnExit, btnHide, btnShow, and picEquation. Figure 1-7 lists the rules and conventions this book will follow when naming objects. Typically, you assign names to only objects that are either coded or referred to in code.
CHAPTER 1
An Introduction to Visual Studio 2017 and Visual Basic

Camel case refers to the fact that the uppercase letters appear as “humps” in the name because they are taller than the lowercase letters.

Naming rules (these are required by Visual Basic)
1. Each object must have a unique name.
2. Each name must begin with a letter and contain only letters, numbers, and the underscore character.

Naming conventions used in this book
1. Each name will begin with an ID of three (or more) characters that represents the object’s type—for example, frm for a form, btn for a button, and txt for a text box.
2. The remaining characters after the ID will indicate the object’s purpose.
3. Each name will be entered using camel case: the ID will be in lowercase, and the first letter in each subsequent word in the name will be capitalized.

Examples
frmMain the main form in a project
btnExit a button that ends the application when clicked
txtFirstName a text box for entering a customer’s first name
picEquation a picture box that displays an image of an equation
lblTotalDue a label that displays the total amount a customer owes
chkDiscount a check box for specifying whether a customer gets a discount

Figure 1-7 Rules and conventions for naming objects

Mini-Quiz 1-2

1. Which window in the IDE lists the tools you can use to add objects to a form?
2. While designing an interface, which window in the IDE allows you to change the default value of an object’s property?
3. Using the naming rules and conventions listed in Figure 1-7, which of the following are valid names and which are not? Explain why the names are not valid.
   a. lblTotal
   b. txtFirstName
   c. lblCity&State
   d. btnCalc Total
   e. txtFirstQuarter

Answers a and e are valid. Answer b is invalid because it contains a period. Answer c is invalid because it contains an ampersand. Answer d is invalid because it contains a space.

1) Toolbox 2) Properties 3) Answer 4) Answer 5) Answer Answer 6) Answer
After studying this lesson, you should be able to:

• A-1 Start and configure Visual Studio Community 2017
• A-2 Create a Windows Forms application
• A-3 Manage the windows in the IDE
• A-4 Change a form file’s name
• A-5 Change the properties of a form
• A-6 Save a solution
• A-7 Close and open a solution
• A-8 Add a control to a form
• A-9 Use the Format menu
• A-10 Lock the controls on the form
• A-11 Start and end an application
• A-12 Enter code and comments in the Code Editor window
• A-13 Print an application’s code and interface
• A-14 Exit Visual Studio and run an executable file

Videos:

• Ch01-Adding a Control
• Ch01-Format Menu
• Ch01-Snipping Tool

A-1 Start and Configure Visual Studio Community 2017

In this Apply lesson, you will create the Einstein’s Famous Equation application that you viewed in the Focus lesson. First, you need to start and configure Visual Studio Community 2017, which contains the Visual Basic language. (Keep in mind that your steps might differ slightly if you are using a different edition of Visual Studio 2017.)

To start Visual Studio Community 2017:

1. Click the Start button on the Windows 10 taskbar. Locate and then click Visual Studio 2017 in the Start menu.

2. Click Tools on the menu bar, click Import and Export Settings, select the Reset all settings radio button, click the Next button, select the No, just reset settings, overwriting my current settings radio button, click the Next button, click Visual Basic, and then click the Finish button. Click the Close button to close the Import and Export Settings Wizard dialog box.

3. Click Window on the menu bar, click Reset Window Layout, and then click the Yes button. When you start Visual Studio 2017, your screen will appear similar to Figure 1-8. However, your menu bar might not contain the underlined letters, called access keys. You can show/hide the access keys by pressing the Alt key on your keyboard.
Next, you will configure Visual Studio Community 2017 so that your screen and tutorial steps agree with the figures and tutorial steps in this book. As mentioned in the Read This Before You Begin section of this book, the figures in this book reflect how your screen will look if you are using Visual Studio Community 2017 and a Microsoft Windows 10 system. Your screen might vary in some instances if you are using a different edition of Visual Studio or another version of Microsoft Windows. Do not worry if your screen display differs slightly from the figures.

To configure Visual Studio 2017:

1. Click **Tools** on the menu bar and then click **Options** to open the Options dialog box. Click the **Projects and Solutions** node. Use the information shown in Figure 1-9 to select and deselect the appropriate check boxes.
A-2 Create a Windows Forms Application

The Einstein's Famous Equation application will be a Windows Forms application, which is an application that has a Windows user interface and runs on a laptop (or desktop) computer.

To create a Windows Forms application:

1. Click File on the menu bar and then click New Project to open the New Project dialog box. If necessary, click the Visual Basic node in the Installed Templates list, and then click Windows Forms App (.NET Framework) in the middle column of the dialog box.

2. Change the name entered in the Name box to Einstein Project.

3. Click the Browse button to open the Project Location dialog box. Locate and then click the VB2017\Chap01 folder. Click the Select Folder button to close the Project Location dialog box.

2. Expand the Text Editor node, expand the Basic node, expand the Code Style node, and then click Naming. Locate the Types row in the dialog box and then click the list arrow in its Severity column. See Figure 1-10.

3. Click None in the list. Then locate the Non-Field Members row and change its Severity column entry to None.

4. Click the Code Style node. Locate the Qualify property access with 'Me' row, click the list arrow in its Preference column, and then click Prefer 'Me.' Now locate the Qualify method access with 'Me' row and change its Preference column entry to Prefer 'Me.'

5. Scroll down the left column of the dialog box until you locate the Debugging node. Click the Debugging node and then deselect the Step over properties and operators (Managed only) check box. Also deselect the Enable Diagnostic Tools while debugging check box (if necessary) and the Show elapsed time PerfTip while debugging check box.

6. Click the OK button to close the Options dialog box.

Note: If you change your default environment settings after performing the previous six steps, you will need to perform the steps again.
4. If necessary, select the **Create directory for solution** check box in the New Project dialog box. Change the name entered in the Solution name box to **Einstein Solution**. Figure 1-11 shows the completed New Project dialog box in Visual Studio Community 2017. (Your dialog box might look slightly different if you are using another edition of Visual Studio. Do not be concerned if your dialog box shows a different version of the .NET Framework.)

![Figure 1-11 Completed New Project dialog box](image)

5. Click the **OK** button to close the New Project dialog box. The computer creates a solution and adds a Visual Basic project to the solution. The names of the solution and project, along with other information pertaining to the project, appear in the Solution Explorer window. Visual Basic also automatically instantiates (creates) a form object, which appears in the designer window. See Figure 1-12.

![Figure 1-12 Solution and Visual Basic project](image)
A-3 Manage the Windows in the IDE

In most cases, you will find it easier to work in the IDE if you either close or auto-hide the windows you are not currently using. You close an open window by clicking the Close button on its title bar. You will find the options for opening a closed window on the View menu. You auto-hide a window by using its Auto Hide button (refer to Figure 1-12), which is located on the window's title bar. The Auto Hide button is a toggle button: Clicking it once activates it, and clicking it again deactivates it. The Toolbox and Data Sources windows in Figure 1-12 are auto-hidden windows.

To close, open, auto-hide, and display windows in the IDE:

1. Click the Close button on the Properties window’s title bar to close the window. Then click View on the menu bar and click Properties Window to open the window.
2. Click the Team Explorer tab. When the Team Explorer window appears, click the Close button on its title bar.
3. Click the Auto Hide (vertical pushpin) button on the Solution Explorer window. The Solution Explorer window now appears as a tab on the edge of the IDE.
4. To temporarily display the Solution Explorer window, click the Solution Explorer tab. Notice that the Auto Hide button is now a horizontal pushpin rather than a vertical pushpin. To return the Solution Explorer window to its auto-hidden state, click the Solution Explorer tab again.
5. To permanently display the Solution Explorer window, click the Solution Explorer tab and then click the Auto Hide (horizontal pushpin) button on the window's title bar. The vertical pushpin replaces the horizontal pushpin on the button.
6. On your own, close the Data Sources window.
7. If necessary, click Form1.vb in the Solution Explorer window. The name of the selected object (Form1.vb) appears in the Properties window’s Object box. The Properties window also contains two columns of information. The left column, called the Properties list, displays the names of the selected object’s properties. The right column contains the Settings boxes; each box displays the current value (or setting) of its associated property.
8. If the names listed in the Properties list do not appear in alphabetical order, click the Alphabetical button, which is the second button on the Properties window’s toolbar. Figure 1-13 shows the current status of the windows in the IDE. Only the designer, Solution Explorer, and Properties windows are open; the Toolbox window is auto-hidden. (If necessary, close any other open or auto-hidden windows in the IDE.)
A-4 Change a Form File’s Name

The code associated with the first form included in a project is automatically stored in a file, referred to as a form file, named Form1.vb. It’s called a form file because it contains the code associated with the form. All files with a .vb filename extension are also referred to as source files because they contain Visual Basic code. The code associated with the second form in the same project is stored in a file named Form2.vb, and so on. To help you keep track of the various form files in a project, you should give each a unique and meaningful name.

To use the Properties window to change the form file’s name:

1. Click File Name in the Properties list for the Form1.vb file. Type Main Form.vb in the Settings box and press Enter. (Be sure to include the .vb extension on the filename; otherwise, the computer will not recognize the file as a source file.) Main Form.vb appears in the Solution Explorer and Properties windows and on the designer window’s tab, as shown in Figure 1-14.
A-5 Change the Properties of a Form

In order to display the properties of a form in the Properties window, the form must be selected in the designer window. (It is easy to confuse a form file, whose File Name property you changed in the previous section, with a form. The form file is an actual file that resides on your disk and contains code, like the code shown earlier in Figure 1-6. The form is the object that appears in the designer window.)
Chapter 1
An Introduction to Visual Studio 2017 and Visual Basic

To display the form's properties:

1. Click the form in the designer window. Scroll to the top of the Properties list and then click (Name). See Figure 1-15.

The Object box in Figure 1-15 shows the text Form1 System.Windows.Forms.Form. Form1 is the name of the form that appears in the designer window. The name is automatically assigned to the form when the form is instantiated (created). In System.Windows.Forms.Form, Form is the name of the class used to instantiate the form. System.Windows.Forms is the namespace that contains the Form class definition. A class definition is a block of code that specifies (or defines) an object’s appearance and behavior. All class definitions in Visual Basic are contained in namespaces, which you can picture as blocks of memory cells inside the computer. Each namespace contains the code that defines a group of related classes. The System.Windows.Forms namespace contains the definition of the Windows Form class. It also contains the class definitions for objects you add to a form, such as buttons and text boxes.

The period that separates each word in System.Windows.Forms.Form is called the dot member access operator. Similar to the backslash (\) in a folder path, the dot member access operator indicates a hierarchy, but of namespaces rather than folders. In other words, the backslash in the path E:\VB2017\Chap01\Einstein Solution\Einstein Project\Main Form.vb indicates that the Main Form.vb file is contained in (or is a member of) the Einstein Project folder, which is a member of the Einstein Solution folder, which is a member of the Chap01 folder, which is a member of the VB2017 folder, which is a member of the E: drive. Likewise, the name System.Windows.Forms indicates that the Form class is a member of the Forms namespace, which is a member of the Windows namespace, which is a member of the System namespace. The dot member access operator allows the computer to locate the Form class in the computer’s main memory, similar to the way the backslash (\) allows the computer to locate the Main Form.vb file on your computer’s disk.

Figure 1-16 lists the names and purposes of the most commonly used properties of a form. In the next several sections, you will change the default values assigned to many of the properties listed in the figure.
A-5 Change the Properties of a Form

The Name Property

Following the naming rules and conventions shown earlier in Figure 1-7, you will name the current form frmMain. The *frm* identifies the object as a form, and *Main* reminds you of the form's purpose, which is to be the main form in the application.

To change the form's name:

1. The form's Name property should be selected in the Properties window. Type *frmMain* and press Enter. The asterisk (*) on the designer window's tab indicates that the form has been changed since the last time it was saved.

The Font Property

A form's Font property determines the type, style, and size of the font used to display the text on the form. A font is the general shape of the characters in the text. Font sizes are typically measured in points, with one point (pt) equaling 1/72 of an inch. The recommended font for applications created for systems running Windows 10 is Segoe UI because it offers improved readability. Segoe is pronounced "SEE-go," and UI stands for user interface. For most of the elements in the interface, you will use a 9pt font size. However, to make the figures in the book more readable, some of the interfaces created in this book will use a slightly larger font size. In this case, for example, you will set the font size to 10.

To change the form's Font property:

1. Click Font in the Properties list and then click the ... (ellipsis) button in the Settings box to open the Font dialog box.

2. Locate and then click Segoe UI in the Font box. Click 10 in the Size box and then click the OK button. (Do not be concerned if the size of the form changes. Also do not be concerned if the Font property shows 9.75pt rather than 10pt.)

The MaximizeBox, StartPosition, and Text Properties

You can use a form's MaximizeBox property to disable the Maximize button, which appears in the form's title bar; doing this prevents the user from maximizing the user interface while the
application is running. The form's StartPosition property determines the form's initial position on the screen when an application is started. To center the form on the screen, you set the property to CenterScreen. A form's Text property specifies the text that appears in the form's title bar. Form1 is the default value assigned to the Text property of the first form in a project. For the current form, "Einstein's Famous Equation” would be a more descriptive value.

To change the three properties:

1. Click MaximizeBox in the Properties list, click the list arrow button in the Settings box, and then click False.
2. Change the StartPosition property to CenterScreen.
3. Click Text in the Properties list, type Einstein's Famous Equation and then press Enter.

Notice that Einstein's Famous Equation, rather than Form1, appears in the form's title bar.

Mini-Quiz 1-4

1. What character is the dot member access operator?
2. What is the recommended font type for Windows 10 applications?
3. Which of a form's properties determines the location of the form when the application is started?
4. To display the words ABC Company in a form's title bar, you need to set which of the form's properties?

A-6 Save a Solution

As mentioned earlier, an asterisk (*) on the designer window's tab indicates that a change was made to the form since the last time it was saved. It is a good idea to save the current solution every 10 or 15 minutes so that you will not lose a lot of your work if a problem occurs with your computer. You can save the solution by clicking File on the menu bar and then clicking Save All. You can also click the Save All button on the Standard toolbar. When you save the solution, the computer saves any changes made to the files included in the solution. It also removes the asterisk that appears on the designer window’s tab.

To save the solution:

1. Click File on the menu bar and then click Save All. The asterisk is removed from the designer window’s tab, indicating that all changes made to the form have been saved.

A-7 Close and Open a Solution

Before learning how to add a control to the form, you will close the current Einstein Solution and open a partially completed one. You should always close a solution when you are finished working on it because doing so ensures that all of the projects and files contained in the solution are closed.
To close the current solution and open a partially completed one:

1. Click **File** on the menu bar. Notice that the menu contains a Close option and a Close Solution option. The Close option closes the designer window in the IDE; however, it does not close the solution itself. Only the Close Solution option closes the solution.

2. Click **Close Solution**. The Solution Explorer window indicates that no solution is currently open in the IDE.

3. You can also use the File menu to open an existing solution. Click **File** and then click **Open Project** to open the Open Project dialog box. Locate and then open the VB2017\Chap01\Partial Einstein Solution folder.

4. The names of solution files end with .sln. Click **Einstein Solution.sln** in the list of filenames and then click the **Open** button. (Depending on how Windows is set up on your computer, you might not see the .sln extension on the filename. Refer to the Read This Before You Begin section to learn how to show/hide filename extensions.)

5. The Solution Explorer window indicates that the solution is open. If the designer window is not open, right-click **Main Form.vb** in the Solution Explorer window and then click **View Designer**.

6. Expand the **Resources** node in the Solution Explorer window. See Figure 1-17. The interface contains a picture box and two buttons.

### A-8 Add a Control to a Form

Two controls are missing from the interface shown in Figure 1-17: a picture box that displays Einstein’s famous equation \( E = MC^2 \) and an Exit button. You will add the picture box to the form first. You use a **picture box** to display an image on the form. In this case, the picture box will display the image stored in the VB2017\Chap01\Equation.png file. Figure 1-18 lists the most commonly used properties of a picture box.
To add a picture box to the form:

1. Click the Toolbox tab and then click the Toolbox window’s Auto Hide button. If necessary, expand the Common Controls node.

2. Click the PictureBox tool, but do not release the mouse button. Hold down the mouse button as you drag the tool to the form. See Figure 1-19.

3. Release the mouse button. The PictureBox tool (class) instantiates a picture box control (object) and places it on the form. The picture box’s properties appear in the Properties list, and a box containing a triangle appears in the upper-right corner of the control. The box is referred to as the task box because when you click it, it displays a list of the tasks associated with the control. Each task in the list is associated with one or more properties. You can set the properties using the task list or the Properties window.
4. Click the **task box** on the PictureBox1 control. See Figure 1-20.

![Figure 1-20 Open task list for a picture box](image)

5. Click **Choose Image** to open the Select Resource dialog box. The Choose Image task is associated with the Image property in the Properties window.

6. To include the image file within the project itself, the Project resource file radio button must be selected in the dialog box. Verify that the radio button is selected, and then click the **Import** button to open the Open dialog box.

7. Open the VB2017\Chap01 folder. Click **Equation.png** in the list of filenames and then click the **Open** button. See Figure 1-21. (The Einstein.png file was already added to the project.)

![Figure 1-21 Completed Select Resource dialog box](image)
8. Click the **OK** button. A small portion of the image appears in the picture box control on the form, and `Einstein_Project.My.Resources.Resources.Equation` (which indicates that the Equation.png file is stored in the project’s Resources folder) appears in the control’s Image property in the Properties window. In addition, Equation.png appears in the Resources folder in the Solution Explorer window.

9. If necessary, click the **task box** on the control to open the task list. Click the **list arrow** in the Size Mode box and then click **StretchImage** in the list. (The Size Mode task is associated with the SizeMode property in the Properties window.) Click the **picture box** control to close the task list.

10. The picture box will be referred to in code, so you will give it a more meaningful name. The three-character ID for picture box names is pic. Change the picture box’s name to `picEquation`. (The Name property is near the top of the Properties list.)

11. Make the `picEquation` control slightly wider by placing your mouse pointer on the control’s middle-right sizing handle and then dragging the sizing handle to the right. Stop dragging when a thin blue line appears between the control’s border and the form’s border, as shown in Figure 1-22. The designer provides a blue margin line to assist you in spacing the controls properly on the form.

   ![Figure 1-22 Result of dragging the sizing handle](image)

12. Release the mouse button.

13. The `picEquation` control should not be visible when the interface appears on the screen after the application is started. Set the control’s Visible property to **False**.

Now you will add the missing Exit button to the form. You use a button to perform an immediate action when clicked; in this case, the button will exit (end) the application. Figure 1-23 lists the most commonly used properties of a button.

<table>
<thead>
<tr>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enabled</td>
<td>indicate whether the button can respond to the user’s action</td>
</tr>
<tr>
<td>Font</td>
<td>specify the font to use for text</td>
</tr>
<tr>
<td>Image</td>
<td>specify the image to display on the button’s face</td>
</tr>
<tr>
<td>ImageAlign</td>
<td>indicate the alignment of the image on the button’s face</td>
</tr>
<tr>
<td>Name</td>
<td>give the button a meaningful name (use btn as the ID)</td>
</tr>
<tr>
<td>Text</td>
<td>specify the text that appears on the button’s face; the text should include an access key</td>
</tr>
</tbody>
</table>

![Figure 1-23 Most commonly used properties of a button](image)
To add a button to the form:

1. Click the Button tool in the toolbox, and then drag the tool to the form. Position the tool to the right of the Hide equation button, using the blue margin and snap lines, as shown in Figure 1-24.

![Figure 1-24 Correct position for the button control](image)

2. Release the mouse button. The Button tool (class) instantiates a button control (object) and places it on the form. The button will be coded, so you will give it a more meaningful name. The three-character ID for button names is btn. Change the button’s name to btnExit.

3. A button’s Text property determines the text that appears on the button’s face. As indicated earlier in Figure 1-23, the Text property should include an access key. You will learn about access keys in Chapter 2. For now, you just need to know that a button’s access key allows the user to select the button by pressing the Alt key in combination with a character that appears in the Text property. You designate the character by preceding it with an ampersand. For example, to designate the letter x as the access key for an Exit button, you enter E&xit in the button’s Text property. The access key will appear underlined on the button’s face. Set the button’s Text property to E&xit.

4. Now drag the Exit button’s bottom sizing handle down slightly until the underline below the letter x is visible, and then release the mouse button.

5. Auto-hide the Toolbox and Properties windows.

6. Click File on the menu bar and then click Save All. (Or you can click the Save All button on the Standard toolbar.)

**Mini-Quiz 1-5**

1. What is the three-character ID used when naming picture boxes?
2. What is the three-character ID used when naming buttons?
3. What is the purpose of an access key?
4. What character is used to designate an access key?
A-9 Use the Format Menu

Visual Basic’s Format menu provides many options that you can use when designing your user interface. Figure 1-25 lists each option and explains its purpose.

<table>
<thead>
<tr>
<th>Option</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Align</td>
<td>align two or more controls by their left, right, top, or bottom borders</td>
</tr>
<tr>
<td>Make Same Size</td>
<td>make two or more controls the same width and/or height</td>
</tr>
<tr>
<td>Horizontal Spacing</td>
<td>adjust the horizontal spacing between two or more controls</td>
</tr>
<tr>
<td>Vertical Spacing</td>
<td>adjust the vertical spacing between two or more controls</td>
</tr>
<tr>
<td>Center in Form</td>
<td>center one or more controls either horizontally or vertically on the form</td>
</tr>
<tr>
<td>Order</td>
<td>specify the layering of one or more controls on the form</td>
</tr>
<tr>
<td>Lock Controls</td>
<td>lock the controls in place on the form</td>
</tr>
</tbody>
</table>

Figure 1-25  Format menu options

Before you can use the Format menu to change the alignment or size of two or more controls, you first must select the controls. You should always select the reference control first. The reference control is the one whose size and/or location you want to match. The reference control will have white sizing handles, whereas the other selected controls will have black sizing handles. In the next set of steps, you will use the Format menu to adjust the Exit button’s height (if necessary) to match the height of the Hide equation button. You will also use it to align the top border of the Exit button (if necessary) with the top border of the Hide equation button.

To adjust the Exit button’s height and top border:

1. Click the Hide equation button (the reference control) and then Ctrl+click the Exit button. Click Format, point to Make Same Size, and then click Height.
2. Next, click Format, point to Align, and then click Tops.
3. Click the form’s title bar to deselect the selected controls.

A-10 Lock the Controls on the Form

After placing all of the controls in their appropriate locations, you should lock them on the form, which prevents them from being moved inadvertently as you work in the IDE. You can lock the controls by clicking the form (or any control on the form) and then clicking the Lock Controls option on the Format menu; you can follow the same procedure to unlock the controls. You can also lock and unlock the controls by right-clicking the form (or any control on the form) and then clicking Lock Controls on the context menu. When a control is locked, a small lock appears in the upper-left corner of the control.

To lock the controls on the form and then save the solution:

1. Right-click the form and then click Lock Controls. A small lock appears in the upper-left corner of the form. See Figure 1-26.
A-11 Start and End an Application

Before you start an application for the first time, you should open the Project Designer window and verify the name of the startup form, which is the form that the computer automatically displays each time the application is started. You can open the Project Designer window by right-clicking My Project in the Solution Explorer window and then clicking Open on the context menu. Or, you can click Project on the menu bar and then click <project name> Properties on the menu.

To verify the startup form:

1. Right-click My Project in the Solution Explorer window, and then click Open to open the Project Designer window.

2. Auto-hide the Solution Explorer window.

3. If necessary, click the Application tab to display the Application pane, which is shown in Figure 1-27. If frmMain does not appear in the Startup form list box, click the Startup form list arrow and then click frmMain in the list. (Do not be concerned if your Target framework list box shows a different value.)

Figure 1-26  Completed interface with the controls locked on the form

2. Save the solution. Try dragging one of the controls to a different location on the form. You will not be able to do so.
You can start an application by clicking Debug on the menu bar and then clicking Start Debugging. You can also press the F5 key on your keyboard or click the Start button on the Standard toolbar.

When you start a Windows Forms application from within the IDE, the computer automatically creates a file that can be run outside of the IDE, like the application files you ran in this chapter's Focus lesson. The file is referred to as an **executable file**. The executable file's name is the same as the project's name, except it ends with .exe. The name of the executable file for the Einstein Project, for example, is Einstein Project.exe. However, you can use the Project Designer window to change the executable file's name.

The computer stores the executable file in the project's bin\Debug folder. In this case, the Einstein Project.exe file is stored in the VB2017\Chap01\Einstein Solution\Einstein Project\bin\Debug folder. When you are finished with an application, you typically give the user only the executable file because it does not allow the user to modify the application's code. To allow someone to modify the code, you need to provide the entire solution.

**To change the name of the executable file, and then start and end the application:**

1. The Project Designer window should still be open. Change the filename in the Assembly name box to **My Einstein**. Save the solution and then close the Project Designer window by clicking its **Close** button. (If necessary, refer to Figure 1-27 for the location of the Close button.)

2. Click **Debug** on the menu bar and then click **Start Debugging** to start the application. (Or you can press the F5 key or use the Start button on the Standard toolbar.) See Figure 1-28. (Do not be concerned about any windows that appear at the bottom of the screen.)
A-11 Start and End an Application

Mini-Quiz 1-6

1. What menu provides options for centering two or more controls on the form?

2. To use the menu from Question 1 to make the btnCalc control the same size as the btnExit control, which of the two controls should you select first?

3. Why is it a good idea to lock the controls on the form?

4. What filename extension indicates that the file is an executable file that can be run outside of the IDE?

3. Recall that the purpose of the Exit button is to allow the user to end the application. Click the Exit button. Nothing happens because you have not yet entered the instructions that tell the button how to respond when clicked.

4. Click the Close button on the form’s title bar to stop the application. (You can also click the designer window to make it the active window, then click Debug on the menu bar, and then click Stop Debugging. Or you can click the Stop Debugging button on the Standard toolbar.)
A-12 Enter Code and Comments in the Code Editor Window

After creating your application's interface, you can begin entering the Visual Basic instructions (code) that tell the controls how to respond to the user's actions. Those actions—such as clicking, double-clicking, and scrolling—are called events. You tell an object how to respond to an event by writing an event procedure, which is a set of Visual Basic instructions that are processed only when the event occurs. Instructions that are processed (executed) by the computer are also called statements. You enter the procedure's code in the Code Editor window.

To open the Code Editor window:

1. Right-click the form and then click View Code on the context menu. The Code Editor window opens in the IDE, as shown in Figure 1-29. If the line numbers do not appear in your Code Editor window, click Tools on the menu bar, click Options, expand the Text Editor node, click Basic, select the Line numbers check box, and then click the OK button.

The Code Editor window in Figure 1-29 contains the Class statement, which is used to define a class in Visual Basic. In this case, the class is the frmMain form. The Class statement begins with the Public Class frmMain clause and ends with the End Class clause. Within the Class statement, you enter the code to tell the form and its objects how to react to the user's actions.

If the Code Editor window contains many lines of code, you might want to hide the sections of code that you are not currently working with. You hide a section (or region) of code by clicking the minus box that appears next to it. To unhide a region of code, you click the plus box that appears next to the code. Hiding and unhideing the code is also referred to as collapsing and expanding the code, respectively.

To collapse and expand a region of code:

1. Click the minus box that appears next to the Public Class frmMain clause. Doing this collapses the Class statement, as shown in Figure 1-30.

2. Click the plus box to expand the code.
As Figure 1-30 indicates, the Code Editor window contains three dropdown list boxes named Project, Object, and Event. The Project box contains the name of the current project, Einstein Project. The Object box lists the names of the objects included in the user interface, and the Event box lists the events to which the selected object is capable of responding. To code the Exit button so that it ends the application when it is clicked by the user, you select btnExit in the Object list box and select Click in the Event list box.

To select the btnExit control's Click event:

1. Click the **Object** list arrow and then click **btnExit** in the list. Click the **Event** list arrow and then click **Click** in the list. A code template for the btnExit control's Click event procedure appears in the Code Editor window. See Figure 1-31.

![Figure 1-31 btnExit control's Click event procedure](image)

The Code Editor provides the code template to help you follow the rules of the Visual Basic language. The rules of a programming language are called its **syntax**. The first line in the code template is called the **procedure header**, and the last line is called the **procedure footer**. The procedure header begins with the keywords **Private Sub**. A **keyword** is a word that has a special meaning in a programming language, and it appears in a different color from the rest of the code. The **Private** keyword in Figure 1-31 indicates that the button's Click event procedure can be used only within the class defined in the current Code Editor window. The **Sub** keyword is an abbreviation of the term **sub procedure**, which is a block of code that performs a specific task.

Following the **Sub** keyword is the name of the object, an underscore, the name of the event, and parentheses containing some text. For now, you do not have to be concerned with the text that appears between the parentheses. After the closing parenthesis is the following Handles clause: **Handles btnExit.Click**. This clause indicates that the procedure handles (or is associated with) the btnExit control's Click event. It tells the computer to process the procedure only when the btnExit control is clicked.

The code template ends with the procedure footer, which contains the keywords **End Sub**. You enter your Visual Basic instructions at the location of the insertion point, which appears between the Private Sub and End Sub clauses in Figure 1-31. The Code Editor automatically indents the lines between the procedure header and footer. Indenting the lines within a procedure makes the instructions easier to read and is a common programming practice. In the next section, you will enter an instruction that tells the btnExit object to end the application when it is clicked.

**The Me.Close() Statement**

The **Me.Close() statement** tells the computer to close the current form. If the current form is the only form in the application, closing it terminates the entire application. In the instruction, **Me** is a keyword that refers to the current form, and **Close** is one of the methods available in Visual Basic. A **method** is a predefined procedure that you can call (or invoke) when needed. To have the computer close the current form when the user clicks the Exit button, you enter the **Me.Close()** statement in the button's Click event procedure. Notice the empty set of parentheses after the method's name in the statement. The parentheses are required when calling some Visual Basic methods; however, depending on the method, the parentheses might or might not be empty. If you forget to enter the empty set of parentheses, the Code Editor will enter them for you when you move the insertion point to another line in the Code Editor window.
To code the btnExit_Click procedure:

1. You can type the `Me.Close()` statement on your own or use the Code Editor window’s IntelliSense feature. In this set of steps, you will use the IntelliSense feature. Type `me.` (be sure to type the period, but don’t press Enter). When you type the period, the IntelliSense feature displays a list of properties, methods, and so on from which you can select.

   **Note:** If the list of choices does not appear, the IntelliSense feature might have been turned off on your computer system. To turn it on, click Tools on the menu bar, click Options, expand the Text Editor node, click Basic, select the Auto list members check box, and then click the OK button.

2. Type `clo` (but don’t press Enter). The IntelliSense feature highlights the Close method in the list. See Figure 1-32. For now, don’t be concerned with the LightBulb indicator or the red jagged line (called a squiggle) below `Me.clo`.

   ![Figure 1-32 List displayed by the IntelliSense feature](image)

3. Press **Tab** to include the Close method in the statement and then press **Enter**. See Figure 1-33.

   ![Figure 1-33 Completed btnExit_Click procedure](image)

It is a good idea to test a procedure after you have coded it so you will know where to look if an error occurs. You can test the Exit button’s Click event procedure by starting the application and then clicking the button. When the button is clicked, the computer will process the `Me.Close()` statement contained in the procedure.
To test the btnExit_Click procedure:

1. Save the solution and then click the Start button on the Standard toolbar (or press the F5 key). The user interface appears on the screen.
2. Click the Exit button to end the application.

Assignment Statements and Comments

Earlier in this chapter, you learned how to use the Properties window to set an object’s properties during design time, which is when you are building the interface. You can also set an object’s properties during run time, which occurs while the application is running; you do this by using an assignment statement. An assignment statement is one of many different types of Visual Basic instructions. Its purpose is to assign a value to something, such as to the property of an object.

The syntax of an assignment statement is shown in Figure 1-34 along with examples of using the syntax. In the syntax, object and property are the names of the object and property, respectively, to which you want the value of the expression assigned. The expression can be a keyword, a number, or a string literal, which is defined as zero or more characters enclosed in quotation marks. The expression can also be a calculation; you will learn how to assign calculations in Chapter 3. You use the dot member access operator (a period) to separate the object name from the property name. The operator indicates that the property is a member of the object. You use an equal sign between the object.property information and the expression. The equal sign in an assignment statement is called the assignment operator. When the computer processes an assignment statement, it assigns the value of the expression that appears on the right side of the assignment operator to the object and property that appear on the left side of the assignment operator.

<table>
<thead>
<tr>
<th>Assignment Statement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax</td>
</tr>
<tr>
<td>object.property = expression</td>
</tr>
</tbody>
</table>

Examples

- picEquation.Visible = False assigns the keyword False to the picEquation's Visible property
- lblDue.Width = 120 assigns the number 120 to the lblDue's Width property
- txtState.Text = "Ohio" assigns the string literal "Ohio" to the txtState's Text property

Figure 1-34  Syntax and examples of assigning a value to a property during run time

You will use assignment statements to code the btnShow_Click and btnHide_Click procedures. In each procedure, you will enter a comment that indicates the procedure's purpose. A comment is a line of text that serves to internally document a program. You create a comment in Visual Basic by placing an apostrophe (') before the text that represents the comment. The computer ignores everything that appears after the apostrophe on that line. Although it is not required, some programmers use a space to separate the apostrophe from the comment text; they also begin the comment text with a capital letter and end it with a period. You will use these conventions in this book.

To enter a comment and code in both procedures:

1. Use the Object and Event boxes to open the code template for the btnShow object’s Click event. Type 'Show Einstein's equation' (be sure to type the apostrophe followed by a space) and press Enter twice.

START HERE

The Start button looks like this:
2. Type `picEq` to highlight `picEquation` in the list and then press `Tab`. Type `.V` (a period followed by the letter `V`) to highlight `Visible` in the list and then press `Tab`. Type `=t` to highlight `True` in the list and then press `Enter`. The `picEquation.Visible = True` statement appears in the procedure.

3. Save the solution and then start the application. Click the `Show equation` button to display the image stored in the `picEquation` control, and then click the `Exit` button.

4. On your own, open the code template for the `btnHide` object’s `Click` event. Type `'Hide Einstein's equation. and press `Enter` twice. Now enter the `picEquation.Visible = False` statement in the procedure.

5. Save the solution and then start the application. Click the `Show equation` button to display the image stored in the `picEquation` control, and then click the `Hide equation` button to hide the image.

6. Now use the `Show equation` button’s access key to show the image again. Press and hold down the `Alt` key as you tap the letter `s`, and then release the `Alt` key. The `Alt+s` combination tells the computer to process (execute) the statements contained in the button’s `Click` event procedure.

7. Use the `Hide equation` button’s access key (`Alt+h`) to hide the image, and then use the `Exit` button’s access key (`Alt+x`) to end the application.

8. Click immediately before the letter `P` in Line 1 of the Code Editor window and then press `Enter`. `Enter` the additional comments shown in Figure 1-35. Replace `<your name>` and `<current date>` with your name and the current date, respectively.

9. Save the solution.

**A-13 Print an Application’s Code and Interface**

It is a good idea to print a copy of your application’s code for future reference. To print the code, the Code Editor window must be the active (current) window. You might also want to print a copy of the user interface; you can do this using the Window’s Snipping Tool, as long as the designer window is the active window.
To print the code and interface:

1. The Code Editor window is currently the active window. Click File on the menu bar, and then click Print to open the Print dialog box. If your computer is connected to a printer, select the appropriate printer (if necessary) and then click the Print button; otherwise, click the Cancel button.

2. Close the Code Editor window. Click the Start button on the Windows 10 taskbar. Click Windows Accessories on the Start menu and then click Snipping Tool. Click the New button. Hold down your left mouse button as you drag your mouse pointer around the form, and then release the mouse button. Click File on the Snipping Tool's menu bar and then click Print. If your computer is connected to a printer, select the appropriate printer (if necessary) and then click the Print button; otherwise, click the Cancel button. Close the Snipping Tool window without saving the snip.

3. Close the solution.

A-14 Exit Visual Studio and Run an Executable File

You can exit Visual Studio using either the Close button on its title bar or the Exit option on its File menu. You can run an application's executable (.exe) file by locating the file in the project's bin\Debug folder and then double-clicking it.

To exit Visual Studio and then run the My Einstein.exe file:

1. Click File and then click Exit.

2. Open the VB2017\Chap01\Partial Einstein Solution\Einstein Project\bin\Debug folder, and then double-click the My Einstein.exe file. Test the application to verify that it works correctly.

Mini-Quiz 1-7

1. What is an event procedure?
2. What is a keyword?
3. If an application contains only one form, what Visual Basic statement tells the computer to end the application?
4. Write an assignment statement that assigns the keyword False to the btnPrint control's Enabled property.
5. In the Code Editor window, what character designates that the text that follows it is a comment?

© 2018 Cengage Learning. All Rights Reserved. This content is not yet final and Cengage Learning does not guarantee this page will contain current material or match the published product.
Summary

- Programs are the step-by-step instructions that tell a computer how to perform a task.
- Programmers use various programming languages to communicate with the computer.
- Programmers are responsible for translating a problem’s solution into instructions that the computer can understand.
- Programmers rigorously test a program before releasing the program to the user.
- An object-oriented programming language, such as Visual Basic, enables programmers to use objects (for example, check boxes and buttons) to accomplish a program’s goal. An object is anything that can be seen, touched, or used.
- Every object in an object-oriented program is instantiated (created) from a class, which is a pattern that tells the computer how the object should look and behave. An object is referred to as an instance of the class.
- The four windows you use most often when designing your application’s GUI are the designer window, Toolbox window, Solution Explorer window, and Properties window.
- Each tool in the toolbox represents a class.
- Each object has a set of properties that determines its appearance and behavior.
- Windows applications in Visual Basic are composed of solutions, projects, and files.
- You enter your application’s program instructions in the Code Editor window.
- An object’s name, which is entered in its Name property, can be used to refer to the object in code.
- To start Visual Studio 2017:
  Click the Start button on the Windows 10 taskbar. Locate and then click Visual Studio 2017.
- To change the default environment settings:
  Click Tools, click Import and Export Settings, select the Reset all settings radio button, click the Next button, select the appropriate radio button, click the Next button, click the settings collection you want to use, click the Finish button, and then click the Close button to close the Import and Export Settings Wizard dialog box.
- To reset the window layout in the IDE:
  Click Window, click Reset Window Layout, and then click the Yes button.
- To configure Visual Studio:
  Click Tools, click Options, click the Projects and Solutions node, and then use the information shown earlier in Figure 1-9 to select and deselect the appropriate check boxes. Next, expand the Text Editor node, expand the Basic node, expand the Code Style node, click Naming, and then change the Severity column in the Types and Non-Field Members rows to None. Click the Code Style node, and then change the Preference column in the Qualify property access with ’Me’ and Qualify method access with ’Me’ rows to Prefer ’Me’. Finally, click the Debugging node, deselect the Step over properties and operators (Managed only) check box, deselect the Enable Diagnostic tools while debugging check box, deselect the Show elapsed time PerfTip while debugging check box, and then click the OK button to close the Options dialog box.
• To create a Visual Basic 2017 Windows Forms application:
  Start Visual Studio 2017. Click File, click New Project, click the Visual Basic node, and then
  click Windows Forms App (.NET Framework). Enter an appropriate name and location in the
  Name and Location boxes, respectively. If necessary, select the Create directory for solution
  check box. Enter an appropriate name in the Solution name box and then click the OK button.

• To size a window:
  Position your mouse pointer on one of the window’s borders and then drag the border.

• To close and open a window in the IDE:
  Close the window by clicking the Close button on its title bar. Use the appropriate option on
  the View menu to open the window.

• To auto-hide a window in the IDE:
  Click the Auto Hide (vertical pushpin) button on the window’s title bar.

• To temporarily display an auto-hidden window in the IDE:
  Click the window’s tab.

• To permanently display an auto-hidden window in the IDE:
  Click the window’s tab to display the window, and then click the Auto Hide (horizontal
  pushpin) button on the window’s title bar.

• The code associated with a form is stored in a form file whose filename ends with .vb.
  All files with a .vb filename extension are referred to as source files because they contain
  Visual Basic code.

• To set the value of a property:
  Select the object whose property you want to set and then click the appropriate property in
  the Properties list. Type the new property value in the selected property’s Settings box, or
  choose the value from the list, color palette, or dialog box.

• To change the name of a form file:
  Set the form file’s File Name property in the Properties window.

• To change an object’s name:
  Set the object’s Name property.

• To specify the type, style, and size of the font used to display text on the form:
  Set the form’s Font property.

• To disable/enable the form’s MaximizeBox button:
  Set the form’s MaximizeBox property.

• To specify the starting location of the form:
  Set the form’s StartLocation property.

• To control the text appearing in the form’s title bar:
  Set the form’s Text property.

• To save a solution:
  Click File on the menu bar and then click Save All. You can also click the Save All button on
  the Standard toolbar.
• To close a solution:
  Click File on the menu bar and then click Close Solution.

• To open an existing solution:
  Click File on the menu bar and then click Open Project. Locate and then open the
  application's solution folder. Click the solution filename, which ends with .sln. Click the Open
  button. If the designer window is not open, right-click the form file's name in the Solution
  Explorer window and then click View Designer.

• To add a control to a form:
  Click a tool in the toolbox, but do not release the mouse button. Hold down the mouse
  button as you drag the tool to the form, and then release the mouse button. As demonstrated
  in the Ch01-Adding a Control video, you can also click a tool and then click the form. In
  addition, you can click a tool, place the mouse pointer on the form, and then press the left
  mouse button and drag the mouse pointer until the control is the desired size. You can also
  double-click a tool in the toolbox.

• To display a graphic in a control in the user interface:
  Use the PictureBox tool to instantiate a picture box control. Use the task box or Properties
  window to set the control's Image and SizeMode properties.

• To display a standard button that performs an action when clicked:
  Use the Button tool to instantiate a button control.

• An access key allows the user to select a control by pressing the Alt key in combination with
  the access key.

• To assign an access key to a button:
  In the button's Text property, precede the access key character with an ampersand (&).

• To select multiple controls on a form:
  Click the first control you want to select, and then Ctrl-click each of the other controls
  you want to select. As demonstrated in the Ch01-Format Menu video, you can also select a
  group of controls on the form by placing the mouse pointer slightly above and to the left of
  the first control you want to select, and then pressing the left mouse button and dragging. A
dotted rectangle appears as you drag. When all of the controls you want to select are within
(or at least touched by) the dotted rectangle, release the mouse button. All of the controls
surrounded or touched by the dotted rectangle will be selected.

• To cancel the selection of one or more controls:
  You cancel the selection of one control by pressing and holding down the Ctrl key as you click
  the control. You cancel the selection of all of the selected controls by releasing the Ctrl key
  and then clicking the form or any unselected control on the form.

• To make two or more controls on the form the same size:
  Select the reference control and then select the other controls you want to size. Click Format
  on the menu bar, point to Make Same Size, and then click the appropriate option.

• To align the borders of two or more controls on the form:
  Select the reference control and then select the other controls you want to align. Click Format
  on the menu bar, point to Align, and then click the appropriate option.
• To lock/unlock the controls on a form:
  Click the form (or any control on the form), click Format, and then click Lock Controls. Or
  you can right-click the form (or any control on the form) and then click Lock Controls on the
  context menu.

• To verify or change the names of the startup form and executable file:
  Use the Application pane in the Project Designer window. You can open the Project Designer
  window by right-clicking My Project in the Solution Explorer window, and then clicking
  Open on the context menu. Or, you can click Project on the menu bar and then click
  `<project name>` Properties on the menu.

• To start and stop an application:
  You can start an application by clicking Debug on the menu bar and then clicking Start
  Debugging. You can also press the F5 key on your keyboard or click the Start button on the
  Standard toolbar. You can stop an application by clicking the form's Close button. You can
  also first make the designer window the active window, then click Debug, and then click Stop
  Debugging.

• When you start a Visual Basic application, the computer automatically creates an executable
  file that can be run outside of the IDE. The file's name ends with .exe.

• You tell an object how to respond to the user's actions by writing an event procedure, which
  can contain comments and Visual Basic statements.

• To open the Code Editor window:
  Right-click the form and then click View Code on the context menu.

• To show/hide the line numbers in the Code Editor window:
  Click Tools, click Options, expand the Text Editor node, click Basic, select/deselect the Line
  numbers check box, and then click OK.

• To collapse or expand a region of code in the Code Editor window:
  Click the minus box next to the region of code to collapse the code; click the plus box to
  expand the code.

• To display an object's event procedure in the Code Editor window:
  Open the Code Editor window. Use the Object list box to select the object’s name, and then
  use the Event list box to select the event.

• To allow the user to close the current form while an application is running:
  Enter the `Me.Close()` statement in an event procedure.

• To turn on/off the IntelliSense feature in the Code Editor window:
  Click Tools, click Options, expand the Text Editor node, click Basic, select/deselect the Auto
  list members check box, and then click the OK button.

• During run time, you can use an assignment statement to assign a value to an object's
  property.

• All comments in the Code Editor window begin with an apostrophe.

• To print an application's code:
  With the Code Editor window the active window, click File on the menu bar and then click
  Print. Select the printer (if necessary) and then click Print.
• To print an application’s interface:

As demonstrated in the Ch01-Snipping Tool video, make the designer the active window. Click the Start button on the Windows 10 taskbar, click Windows Accessories, and then click Snipping Tool. Click the New button. Drag your mouse pointer around the form and then release the mouse button. Click File, select the printer (if necessary), and then click Print.

• To exit Visual Studio:

Click the Close button on the Visual Studio title bar. You can also click File on the menu bar and then click Exit.

• To run an application’s executable (.exe) file outside of the IDE:

Locate the file in the project’s bin\Debug folder and then double-click it.

Key Terms

Application—a GUI along with its program instructions
Assignment operator—the equal sign in an assignment statement
Assignment statement—an instruction that assigns a value to something, such as to the property of an object
Camel case—used when entering object names; the practice of entering the object’s ID characters in lowercase and then capitalizing the first letter of each subsequent word in the name
Class—a pattern that the computer uses to create (Instantiate) an object
Class definition—a block of code that specifies (or defines) an object’s appearance and behavior
Code—program instructions
Code Editor window—where you enter the program instructions (code) for your application
Coding—the process of translating a solution into a language that the computer can understand
Comment—a line of text that serves to internally document a program; begins with an apostrophe
Computer program—the directions given to computers; also called a program
Controls—objects (such as a picture box or a button) added to a form
Design time—occurs when you are building an interface
Designer window—used to create an application’s GUI
Dot member access operator—the period used to indicate a hierarchy
Event procedure—a set of Visual Basic instructions that tell an object how to respond to an event
Events—actions to which an object can respond; examples include clicking and double-clicking
Executable file—a file that can be run outside of the IDE; the filename ends with the .exe extension
Form—the foundation for the user interface in a Windows Forms application; also called a Windows Form object
Form file—a file that contains the code associated with a Windows form
Graphical user interface—what the user sees and interacts with while your program is running; also called a GUI
GUI—graphical user interface
IDE—integrated development environment
Instance—an object created (instantiated) from a class
Instantiated—the process of creating an object from a class
Integrated development environment—an environment that contains all of the tools and features you need to create, run, and test your programs; also called an IDE
Keyword—a word that has a special meaning in a programming language
Me.Close() statement—tells the computer to close the current form
Method—a predefined procedure that you can call (invoke) when needed
Name property—assigns a name to an object; the name can be used to refer to the object in code
Namespace—a block of memory cells inside the computer; contains the code that defines a group of related classes
Object—anything that can be seen, touched, or used
Object-oriented programming language—a programming language that allows the programmer to use objects to accomplish a program's goal
OOP—object-oriented programming
Picture box—used to display an image
Procedure footer—the last line in a procedure
Procedure header—the first line in a procedure
Program—the directions given to computers; also called a computer program
Programmers—the people who write computer programs
Programming—the process of giving a mechanism the directions to accomplish a task
Programming languages—languages used to communicate with a computer
Properties—the attributes that determine an object's appearance and behavior
Properties window—lists an object's attributes (properties)
Reference control—the first control selected in a group of controls; this is the control whose size and/or location you want the other selected controls to match
Run time—occurs while an application is running
Solution Explorer—displays a list of the projects contained in the current solution and the items contained in each project
Source files—files that contain program instructions; in Visual Basic, the names of source files end with .vb
Startup form—the form that appears automatically when an application is started
Statements—Visual Basic instructions that are processed (executed) by the computer
String literal—zero or more characters enclosed in quotation marks
Sub procedure—a block of code that performs a specific task
Syntax—the rules of a programming language
Toolbox window—contains the tools used when creating an interface (each tool represents a class); referred to more simply as the toolbox
CHAPTER 1  An Introduction to Visual Studio 2017 and Visual Basic

Review Questions

1. A(n) _____________________ is an environment that contains all of the tools and features you need to create, run, and test your programs.
   a. GUI
   b. IDE
   c. UDE
   d. user interface

2. When using an analogy involving a blueprint and a tree house, the _____________________ is the class, and the _____________________ is an object created from it.
   a. blueprint, tree house
   b. tree house, blueprint

3. You create your application’s user interface in the _________________ window in the IDE.
   a. designer
   b. form
   c. GUI
   d. interface

4. Each tool in the toolbox represents a class from which an object can be instantiated.
   a. True
   b. False

5. Which window is used to set the characteristics that control an object’s appearance and behavior?
   a. Characteristics
   b. Object
   c. Properties
   d. Toolbox

6. Which window lists the projects and files included in a solution?
   a. Object
   b. Project
   c. Properties
   d. Solution Explorer

7. Which of the following is an invalid name for an object?
   a. picMy.Dog
   b. btnCalcSalesTax
   c. lbI2017
   d. All of the above are invalid names.

8. Which property controls the text displayed in a form’s title bar?
   a. Caption
   b. Text
   c. Title
   d. TitleBar

9. Which property is used to give a form file a more meaningful name?
   a. File
   b. File Name
   c. Form Name
   d. Name

10. Which property determines the initial position of a form when the application is started?
    a. InitialLocation
    b. Location
    c. StartLocation
    d. StartPosition
11. Which property is used to disable the Maximize button on a form's title bar?
   a. ButtonMaximize   c. MaximizeBox
   b. Maximize          d. MaximizeButton

12. Which property is used to give a button control a more meaningful name?
   a. Application       c. Name
   b. Caption           d. Text

13. The text displayed on a button's face is stored in which property?
   a. Caption           c. Name
   b. Label             d. Text

14. When a form has been modified since the last time it was saved, what appears on its tab in the designer window?
   a. an ampersand (&)   c. a percent sign (%)
   b. an asterisk (*)     d. a plus sign (+)

15. Which option on the File menu closes the current solution?
   a. Close              c. Close Solution
   b. Close All          d. Exit Solution

16. What is the three-character extension appended to solution filenames in Visual Basic?
   a. .prg               c. .src
   b. .sln               d. .vbs

17. Which of the following can be accomplished using the Format menu?
   a. aligning the borders of two or more controls
   b. centering one or more controls horizontally on the form
   c. making two or more controls the same size
   d. All of the above.

18. When two or more controls are selected, how can you tell which one is the reference control?
   a. The reference control has white sizing handles.
   b. The reference control has black sizing handles.
   c. The reference control displays the number 1 in the left corner.
   d. You can't tell. You just need to remember which control you selected first.

19. Which statement terminates an application that contains only one form?
   a. Me.Close()         c. Me.Finish()
   b. Me.Done()          d. Me.Stop()
20. When you start a Windows Forms application in the IDE, the computer saves the application’s executable (.exe) file in the _____________________ folder.
   a. project
   b. project’s bin
   c. project’s bin\Debug
   d. project’s Debug\bin

21. Explain the difference between a form’s Text property and its Name property.

22. Explain the difference between a form file and a form.

23. What does the dot member access operator indicate in the text System.Windows.Forms.Label?

24. Define the term “syntax.”

25. How do you verify the name of the startup form?

Exercises

1. Create a Windows Forms application. Use the following names for the project and solution, respectively: Florist Project and Florist Solution. Save the application in the VB2017\Chap01 folder.
   a. Change the form file’s name to Main Form.vb.
   b. Change the form’s Name property to frmMain.
   c. Change the form’s Font property to Segoe UI, 9pt.
   d. Change the form’s MaximizeBox property to False.
   e. Change the form’s StartPosition property to CenterScreen.
   f. Change the form’s Text property to Florist Haven.
   g. Add a button to the form. Change the button’s Name and Text properties to btnExit and Exit, respectively. Position the button in the lower-right corner of the form.
   h. Add three picture boxes to the form. PictureBox1 should display the image stored in the Florist.png file, which is located in the VB2017\Chap01 folder. PictureBox2 and PictureBox3 should display the images stored in the Message.png and Hours.png files, respectively. Use the interface shown in Figure 1-36 as a guide when sizing and positioning the picture boxes.
   i. Lock the controls on the form.
   j. Code the Exit button and then add comments to the Code Editor window, as shown in Figure 1-36.
   k. Open the Project Designer window and verify the name of the startup form, which should be frmMain. Also change the name of the application’s executable file to MyFlorist. Save the solution and then close the Project Designer window.
   l. Start the application and then test the Exit button.
   m. Print the application’s code and interface, and then close the solution.
   n. Verify that the MyFlorist.exe file works correctly. (The file is contained in the VB2017\Chap01\Florist Solution\Florist Project\bin\Debug folder.)
2. If necessary, complete Exercise 1. Use Windows to rename the Florist Solution folder to Florist Solution-Modified. Open the Florist Solution.sln file contained in the VB2017\Chap01\Florist Solution-Modified folder. If necessary, open the designer window.
   a. Change the name of the picture box that displays the hours to picHours.
   b. The picHours control should not be visible when the application is started and the interface appears on the screen; set the appropriate property.
   c. Unlock the controls and then modify the interface as shown in Figure 1-37. Change the buttons’ names to btnShow and btnHide, and then lock the controls.
   d. Code the Show hours and Hide hours buttons. The Show hours button should display the picHours control when the button is clicked; the Hide hours button should hide the picHours control.
   e. Save the solution and then start the application. Test the buttons and then close the solution.

3. Create a Windows Forms application. Use the following names for the project and solution, respectively: Jackets Project and Jackets Solution. Save the application in the VB2017\Chap01 folder.
   a. Change the form file’s name to Main Form.vb.
   b. Change the form’s Name property to frmMain.
   c. Change the form’s Font property to Segoe UI, 9pt.
   d. Change the form’s MaximizeBox property to False.
   e. Change the form’s StartPosition property to CenterScreen.
   f. Create the interface shown in Figure 1-38. The images in the picture boxes are stored in the BlackJacket.png and SizeChart.png files; both files are contained in the VB2017\Chap01 folder.
CHAPTER 1 An Introduction to Visual Studio 2017 and Visual Basic

g. Lock the controls on the form. The picChart control should not be visible when the application is started and the interface appears on the screen; set the appropriate property.

h. Code the Exit button. Also code the Size chart button so that it displays the picChart control when the button is clicked. Include comments in the Code Editor window.

i. Open the Project Designer window and verify the name of the startup form, which should be frmMain. Also change the name of the application's executable file to JacketsAndMore. Save the solution and then close the Project Designer window.

j. Start and test the application. Close the solution and then verify that the JacketsAndMore.exe file works correctly.

4. Open the Colors Solution.sln file contained in the VB2017\Chap01\Colors Solution folder. If necessary, open the designer window. The interface contains one label control and two buttons. You can use a label control to display information that you do not want the user to change during run time. You will learn more about label controls in Chapter 2.

a. Add two buttons to the form, as shown in Figure 1-39, and then lock the controls on the form. Change the names of the buttons to btnGreen and btnRed; also change their Text properties.

b. Open the Code Editor window, which contains comments and the code for the btnBlue_Click and btnExit_Click procedures. The btnBlue_Click procedure assigns the string literal “Azul” to the Text property of the lblSpanish control. Start the application. Click the Blue button; the word Azul appears in the label control on the form. Click the Exit button.

c. Using the btnBlue_Click procedure as a guide, code the Click event procedures for the btnGreen and btnRed controls. (Hint: The Spanish words for Green and Red are Verde and Rojo, respectively.)

d. Save the solution and then start and test the application.
5. If necessary, complete Exercise 4. Use Windows to rename the Colors Solution folder to Colors Solution-Modified. Open the Colors Solution.sln file contained in the VB2017\Chap01\Colors Solution-Modified folder. If necessary, open the designer window. Locate the btnBlue_Click procedure in the Code Editor window. In the line above the procedure footer, enter the following assignment statement: `lblSpanish.BackColor = Color.Blue`. Save the solution and then start the application. Click the Blue button; the word Azul appears in the label control and the control's background color changes to blue. Click the Exit button. Modify the code in the btnGreen_Click and btnRed_Click procedures so that each changes the label control's background color appropriately. Save the solution and then start and test the application.

6. Open the Emoji Solution.sln file contained in the VB2017\Chap01\Emoji Solution folder. If necessary, open the designer window. The interface contains one label control and one button. You can use a label control to display information that you do not want the user to change during run time. You will learn more about label controls in Chapter 2.
   a. Add five picture boxes to the form, as shown in Figure 1-40. Name the picture boxes picCrying, picHappy, picLove, picSad, and picTired. The picture boxes should display the images stored in the following files, which are contained in the VB2017\Chap01 folder: Crying.png, Happy.png, Love.png, Sad.png, and Tired.png.
   b. Lock the controls on the form. Open the Code Editor window, which contains comments and the code for the btnExit_Click procedure. When the user clicks the picCrying control, its Click event procedure should assign the string literal "I am crying." to the lblMessage control's Text property; code the procedure. (Hint: Refer to the examples shown earlier in Figure 1-34.)
   c. Using the messages included in Figure 1-40, code the Click event procedures for the remaining picture boxes. Save the solution and then start and test the application.

7. If necessary, complete Exercise 3. Use Windows to rename the Jacket Solution folder to Jackets Solution-Modified. Open the Jackets Solution.sln file contained in the VB2017\Chap01\Jackets Solution-Modified folder. If necessary, open the designer window.
   a. Unlock the controls and then modify the interface, as shown in Figure 1-41.
   b. Change the name of the picture box that displays the jacket to picJacket. Name the four new buttons btnBlack, btnBlue, btnBrown, and btnRed.
   c. Each of the four color buttons will display an image of an appropriately colored jacket in the picJacket control. The different colored images are stored in the
BlackJacket.png, BlueJacket.png, BrownJacket.png, and RedJacket.png files, which are contained in the VB2017\Chap01 folder. In Exercise 3, you added the BlackJacket.png file to the project; now you need to add the other three files. Use the task box on the picJacket control to add the BlueJacket.png, BrownJacket.png, and RedJacket.png files to the project, and then close the Select Resource dialog box. (Do not be concerned that the jacket in the picJacket control is no longer the black one.) If necessary, expand the Resources node in the Solution Explorer window. Notice that the folder contains five .png files.

d. Lock the controls on the form. When the user clicks the Black button, the black jacket should appear in the picJacket control. Open the Code Editor window and then open the code template for the btnBlack_Click procedure. Enter the comment and assignment statement shown in Figure 1-41. The My in the statement refers to the current application, Resources refers to the Resources folder, and BlackJacket refers to the BlackJacket.png file within the Resources folder. The dot member access operator indicates that BlackJacket is a member of the Resources folder, which is a member of the current application.

e. Code the btnBlue_Click, btnBrown_Click, and btnRed_Click procedures. Save the solution and then start and test the application.

8. Create a Windows Forms application. Use the following names for the project and solution, respectively: Texting Project and Texting Solution. Save the application in the VB2017\Chap01 folder.

a. Change the form file's name to MainForm.vb. Change the form's Name property to frmMain, its Font property to Segoe UI 9pt, its MaximizeBox property to False, itsStartPosition property to CenterScreen, and its Text property to Text Message Symbols.
b. Create the interface shown in Figure 1-42. Use the Label tool to create the label control. However, you will not be able to size the label control until you complete Step c. Set each picture box’s BorderStyle property to FixedSingle. The images in the picture boxes represent text message symbols. The images are stored in the BFF.png, BRB.png, IDK.png, LOL.png, SRY.png, and XO.png files, which are contained in the VB2017\Chap01 folder.

c. Change the label control’s name to lblMeaning. Set its AutoSize property to False, its BorderStyle property to FixedSingle, its Font size to 18pt, and its TextAlign property to MiddleCenter. Delete the contents of its Text property. Now size the label control as shown in Figure 1-42.

d. Lock the controls on the form. Give the picture boxes and the Exit button meaningful names.

e. Code the Exit button.

f. When clicked, each picture box should assign the meaning of its associated symbol to the lblMeaning control’s Text property. For example, the picture box that displays BFF should assign the message “Best friends forever” to the Text property. Code each picture box’s Click event procedure. (If necessary, use the Internet to research the meaning of any symbols you do not recognize.)

g. Open the Project Designer window and verify the name of the startup form. Save the solution and then close the Project Designer window. Start and test the application.

9. Create a Windows Forms application. Use the following names for the project and solution, respectively: OnYourOwn Project and OnYourOwn Solution. Save the application in the VB2017\Chap01 folder. Design and code an application that adheres to the guidelines listed in Figure 1-43. Open the Project Designer window and verify the name of the startup form. Save the solution and then close the Project Designer window. Start and test the application. Print the application’s code and interface.

1. The interface must contain at least two picture boxes. You can use your own image files, or you can download image files from openclipart.org. When downloading from openclipart.org, be sure to use the SMALL IMAGE (.PNG) button.
2. The interface must contain at least one button.
3. The interface can contain label controls; however, this is not a requirement.
4. Objects that are either coded or referred to in code should be named appropriately.
5. The Code Editor window must contain comments, the Me.Close() statement, and at least one assignment statement.
10. Open the VB2017\Chap01\FixIt Solution\FixIt Solution.sln file. If necessary, open the designer window. The interface contains a picture box, two buttons, and a label control that displays the price of the parrot.

   a. Start the application. Click the Show price button, which should display the label control. Notice that the button is not working. Click the Exit button to end the application.

   b. Open the Code Editor window and then fix the code contained in the btnShow_Click procedure. Save the solution and then start the application. Click the Show price button to verify that it now displays the price of the parrot. Click the Exit button to end the application.

   c. Start the application again. This time, use the Exit button's access key (Alt+x) to end the application. Notice that the access key is not working. Click the Exit button to end the application.

   d. Which event procedure in the Code Editor window contains the `Me.Close()` statement? In which event procedure should the statement be entered? Delete the entire incorrect event procedure and then enter the `Me.Close()` statement in the correct one. Save the solution and then start the application. Test the Exit button by clicking it and also by using its access key.