PART I
Getting Started

- **CHAPTER 1:** Introducing C#
- **CHAPTER 2:** Understanding Objects

1

Introducing C#

WHAT YOU WILL LEARN IN THIS CHAPTER:

- How to download Visual Studio
- How to install Visual Studio Professional
- How to create a simple project
- How to develop a small program in Visual Studio
- Some features found in the Visual Studio IDE
- Running a Visual Studio program

Welcome to the world of object-oriented programming and C#! The primary goal of this book is to use the C# programming language from Microsoft to teach you object-oriented programming, or OOP. This book assumes that you have no prior programming experience in any language and that you know nothing about OOP. As such, this book must be viewed as a "beginning" text.

If you do have programming experience and some familiarity with OOP, that's fine. Having that experience makes things easier for you. However, I still encourage you to read the book from start to finish for a number of reasons. First, this book represents the distillation of 25 years of programming and teaching experience. I have a good idea of what works and what doesn't work to explaining complex topics so that they're easy to understand. Reading each chapter gives you the tools to understand the next chapter. Second, I may introduce topics in one chapter and then rely heavily on those topics in a much later chapter. In other words, the process used to learn OOP and C# is one that introduces new topics based upon ones that were introduced earlier. Obviously, you must master the earlier content before tackling the later content. Finally, the programming examples I use also build on concepts presented

in earlier program examples. It will be easier for you to understand the later program examples if you've experimented with those programs introduced earlier in the book.

One more thing: You cannot learn programming by just reading about it. You have to dig in and start programming. For that reason, there are exercises at the end of each chapter designed to help you hone your programming skills. The learning process is even more interesting if you try to create your own programs based upon some real problems you'd like to solve. Don't worry if things don't fall together instantly on the first try. You should plan to make a ton of "flat-forehead" mistakes…you know, the kind of mistake in which, upon discovering it, you pound the heel of your hand into your forehead and say, "How could I make such a stupid mistake!" Not to worry…we've all been there. Such mistakes are just part of the process to become a programmer, and you should expect to make your fair share. However, stick with it, read the book, and you'll be surprised at how fast things can come together. Indeed, I think you'll find programming to be a truly enjoyable pastime.

A SHORT HISTORY OF OBJECT-ORIENTED PROGRAMMING (OOP)

Many people believe that OOP is a product of the 1980s and the work done by Bjarne Stroustrup in moving the C language into the object-oriented world by creating the C++ language. Actually, SIMULA 1 (1962) and Simula 67 (1967) are the two earliest object-oriented languages. The work on the Simula languages was done by Ole-John Dahl and Kristen Nygaard at the Norwegian Computing Center in Oslo, Norway. Although most of the advantages of OOP were available in the earlier Simula languages, it wasn't until C++ became entrenched in the 1990s that OOP began to flourish.

C was the parent language of C++ and it was often said that C was powerful enough to shoot yourself in the foot multiple times. C++, on the other hand, not only was powerful enough to shoot yourself in the foot, but also you could blow your entire leg off without too much difficulty. Most programmers admit that C++ is a powerful language and it is still in widespread use today. However, with that power comes a lot of complexity. Language developers wanted a simpler and perhaps less complex language for OOP development.

The next step in the development of OOP started in January 1991 when James Gosling, Bill Joy, Patrick Naughton, Mike Sheradin, and several others met in Aspen, Colorado, to discuss ideas for the Stealth Project. The group wanted to develop intelligent electronic devices capable of being centrally controlled and programmed from a handheld device. They decided that OOP was the right direction to go with the development language but felt that C++ was not up to the job. The result was the Oak programming language (named after an oak tree outside Gosling's window), which eventually morphed into the Java programming language. (Oak had to be renamed because the team discovered that a language by that name already existed.)

Java quickly grew in popularity, spurred by the growth of the World Wide Web. In both cases this rapid growth was in part because the "guts" necessary to run Java programs on the web quickly

became an integral part of various web browsers. With the improved web functionality augmented by Java, the web hit light speed.

To many programmers, C# is Microsoft's answer to Java. Some would even say that C# is the result of Microsoft's stubbornness to refuse to promote a language it did not develop. That sentiment is a bit too harsh. Microsoft had good reasons for developing C#, not the least of which was that it wanted *type-safe programs* that run in a managed environment. You may not appreciate exactly what that means right now, but it will become clear as you learn C#.

C# provides you with a robust object-oriented programming language and an impressive set of tools to tackle almost any programming task. Whether you want to develop desktop, distributed, web, or mobile applications, C# can handle the task.

As you become familiar with C#, you can appreciate its relatively few keywords, its crisp syntax, and its easy-to-use development environment. You'll discover that pieces of programs you write in C# can be reused in other programs. Finally, you might appreciate that there are many job opportunities for programmers who know C#.

INSTALLING C#

If you have already purchased and installed Visual Studio 2012 and C#, you can skip this section. If you haven't installed C#, this section tells you how to download and install the Visual Studio version of Visual Studio. Visual Studio is a modified version of C# that is available from Microsoft at no charge. Although the Express version of C# is missing some features found in the commercial version of Visual Studio, you should compile and run all the sample programs in this book using Visual Studio.

At the present time, the Express version of Visual Studio requires the use of Windows 8. If you do not have Windows 8, you can download a trial version of Visual Studio Professional. By registering the trial version, you can use Visual Studio Professional for a period of 90 days. After you are convinced that you should do all your development work in C# (and you will be), you can purchase the full version of Visual Studio. Of course, if you later purchase Windows 8, you can always download the Express version of Visual Studio.

Due to the newness of Windows 8, you probably are not using Windows 8 and, hence, must use the trial version of Visual Studio Professional. The next section discusses how to download the trial version of Visual Studio Professional.

Downloading Visual Studio Professional

At the time that this book is written, you can go to `http://www.microsoft.com/visualstudio/11/en-us/downloads#professional` to download the Professional version of Visual Studio. The download page should look similar to what is shown in Figure 1-1. As you can see if you look closely at the figure, the download is for the 90-day trial version of Visual Studio Professional. Click the Install Now button to begin the installation process.

FIGURE 1-1: Download page for Visual Studio Professional

Depending upon the speed of your system and Internet connection, it can take more than 1 hour to download and install the 6+ gigabytes of files used during the installation process. You must first agree to the licensing terms, as shown in Figure 1-2.

After you agree to the licensing terms, the program displays a dialog box that tells you the items that are about to be installed. Unless you are constrained by disk space, you should leave all the options checked so that they are all installed. Otherwise, you can uncheck those options that you do not want installed. See Figure 1-3.

Having made your selections, click the Install button to initiate the installation process. Your display will look similar to Figure 1-4...for a *long* time.

FIGURE 1-2: License terms dialog

FIGURE 1-3: Installation items checklist

FIGURE 1-4: Installation process

> ***NOTE*** *While I was working through the installation process, the install software sensed that some of the Windows software on my system was outdated. I was asked if I wanted to install the updates and patches that had come out since I last updated my system. Because the list also included several patches to Visual Studio, I said yes. It took almost 2 hours to download and update my software. However, when the updates were finished, the Visual Studio installer picked up right where it had left off and the installation continued without a hitch. Although it does take some time, you should update your software if the installer offers you that option.*

Eventually, the installation process ends…honest! If you updated your system software as part of the installation process, the installer asks you at some point if you want to restart your computer. Answer yes if you are asked. After the system restarts and the software has been installed, you should see a new icon on your desktop labeled Visual Studio. You should double-click the new icon and launch Visual Studio.

A TEST PROGRAM USING C#

Although things may appear to have been installed properly, you can't be certain until you actually write a program and try to run it. That's the purpose of this section of the chapter. The program is about as simple as you can make a program while remaining confident that the installation was successful.

After you double-click the Visual Studio icon on your desktop, you should see a Visual Studio startup screen similar to the one shown in Figure 1-5.

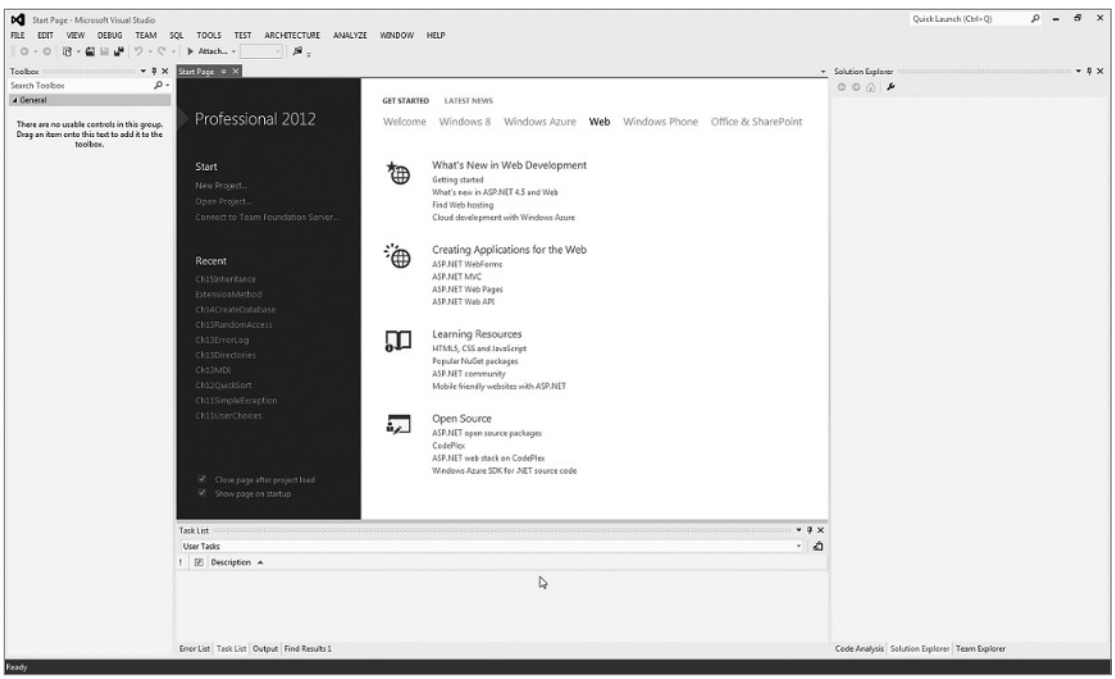

FIGURE 1-5: The Visual Studio IDE

Creating a Project

Move the cursor to the upper-left side of the screen, and select the File ➪ New ➪ Project menu option from the main program menu bar, as shown in Figure 1-6.

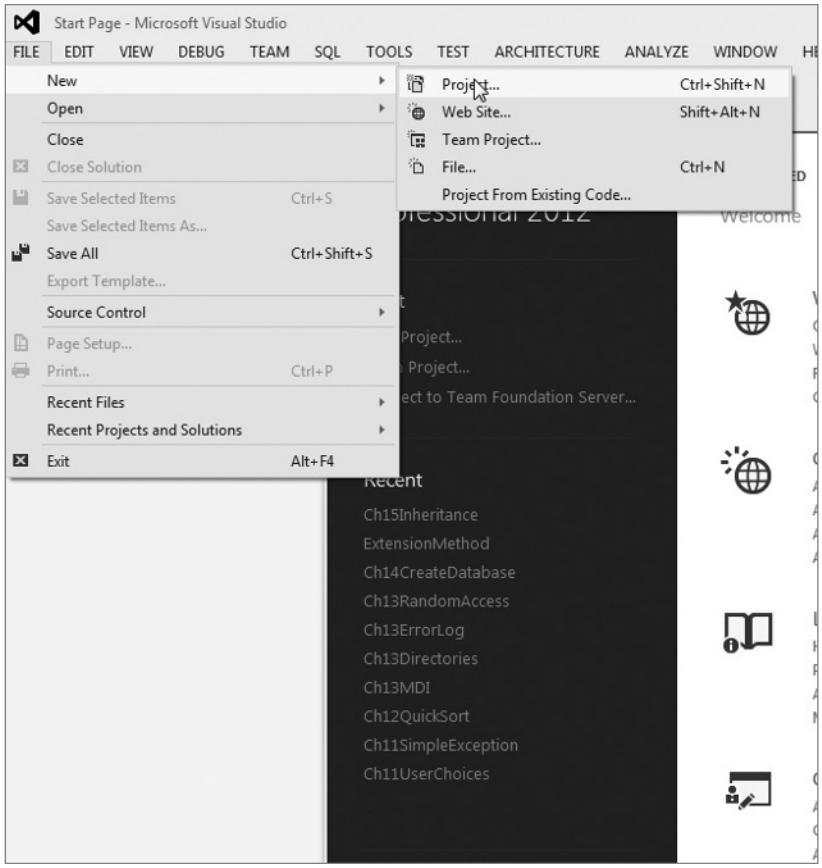

FIGURE 1-6: Creating a new project

Select New Project from the menu. Your program screen changes, as shown in Figure 1-7.

Figure 1-7 shows a number of predefined project templates from which to choose. Make sure you have selected the Templates Visual C# Windows menu option that appears on the left side of the screen. These templates define the types of programs that you can develop with C#. For your purposes, you want to select the Windows Forms Application template.

When you select one of the templates, Visual Studio creates that type of project for you as well as writing some stub code for you. *Stub codes* are predefined pieces of code that Visual Studio writes for you as a background process. From the templates shown in Figure 1-7, select the Windows Forms Application template. You should also type in the name you want the program to have. I have typed in TemplateProgram for the example and designated the project to be placed in the `C:\Temp\` directory. You can use the Browse button to select some other directory, or you can check the Create

Directory for solution and let Visual Studio create a directory for you. Click OK after you enter the program name you want to use.

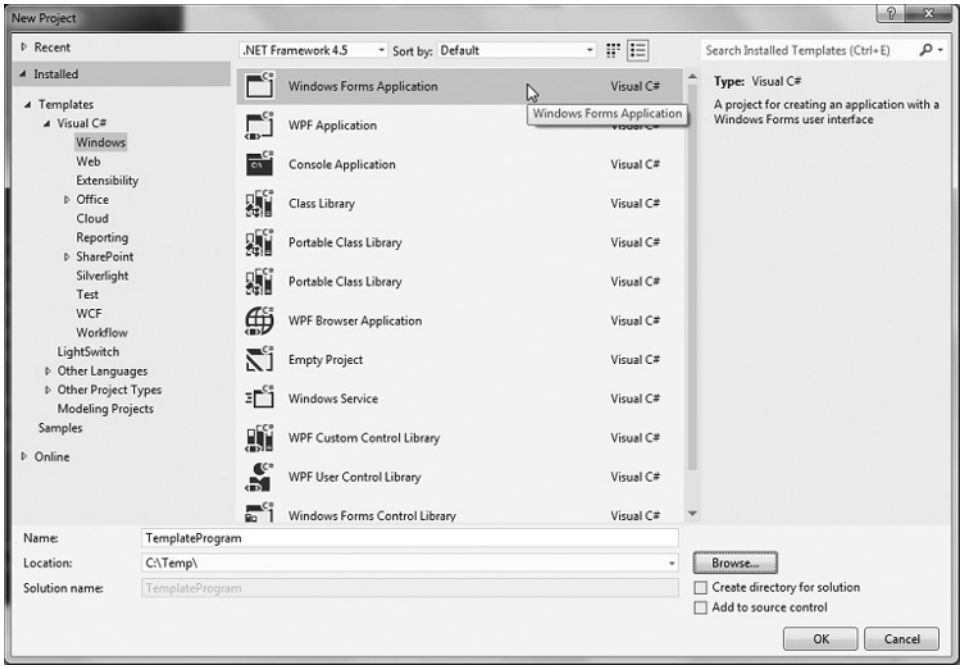

FIGURE 1-7: New Project dialog

The C# Integrated Development Environment

You should now see something like Figure 1-8 on your screen, which shows you where you'll be spending a lot of your programming time as you read this book. It's called the *Integrated Development Environment*, or IDE, because virtually every programming tool you need to write C# programs is available to you there.

Back in the Dark Ages of programming, you had to load and run a programming editor to write the program's source code. Next, you had to close the editor and load the language compiler program to check the program's statements for errors. Then you had to load an assembler program to convert the source code to assembly language. Finally, you had to load and run a linker program to combine all the elements of your program into an executable program. The Visual Studio IDE provides all these functions within a single program. This makes it much easier to write and test the programs you create.

The Major IDE Windows

The IDE shown in Figure 1-8 divides the screen into three *windows*. The left window shows the Visual Studio Toolbox, which, by default, shows some of the objects Visual Studio makes available to you. If you look closely you can see that the Toolbox presents a smorgasbord of objects you can

use in your programs, including textboxes, labels, buttons, and other controls you will find useful as you develop your programs.

The middle window is referred to as the Source window and currently shows an unadorned Visual Studio form object. (Microsoft refers to the window as the Forms Designer window.) As presented in Figure 1-8, the form has no other objects placed on it…yet. That is, you haven't added any other objects (such as textboxes or buttons) from the Toolbox onto the form. You change this in the section "Adding a Toolbox Object to a Windows Form" later in this chapter.

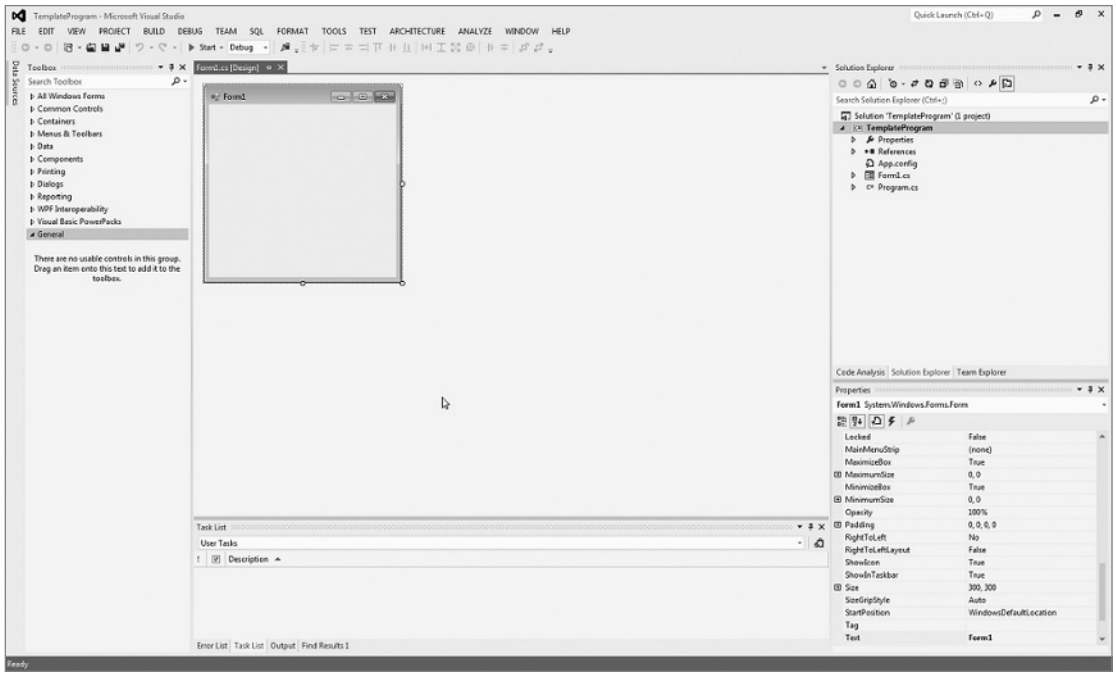

FIGURE 1-8: The Visual Studio IDE

The right side of the IDE currently shows two subwindows. The subwindow on top shows the Solution Explorer. Simply stated, the Solution Explorer shows the current development state of your project. It shows the forms the program has and its references, plus other information that is discussed in later chapters.

Below the Solution Explorer window is the Properties window. The purpose of the Properties window is to show you the properties associated with the object currently in focus in the Source window. If you look closely at the form in the Source window in Figure 1-8, you'll notice that it has a couple of small white boxes along its edge. These are called *sizing boxes* and their purpose is to enable you to alter the size of the objects to which they are attached. However, the sizing boxes also show which object in the Source window currently has the attention, or *focus*, of Visual Studio. Whatever object has the focus in the Source window is also the object that the Properties window displays. In Figure 1-8 the properties shown in the Properties window apply to the form shown in the Source window. Visual Studio always maintains this relationship between the object in focus in the Source window and the information displayed in the Properties window.

Using the Source Code Window

If you right-click `Form1.cs` in the Solution Explorer window, a small menu opens up from which you can select to view the source code for the file you just right-clicked. You can move to the Solution Explorer menu bar and click the source code icon to view the code that Visual Studio has written for you thus far. As you can see in Figure 1-9, you can also use the F7 key to view the source code for the file.

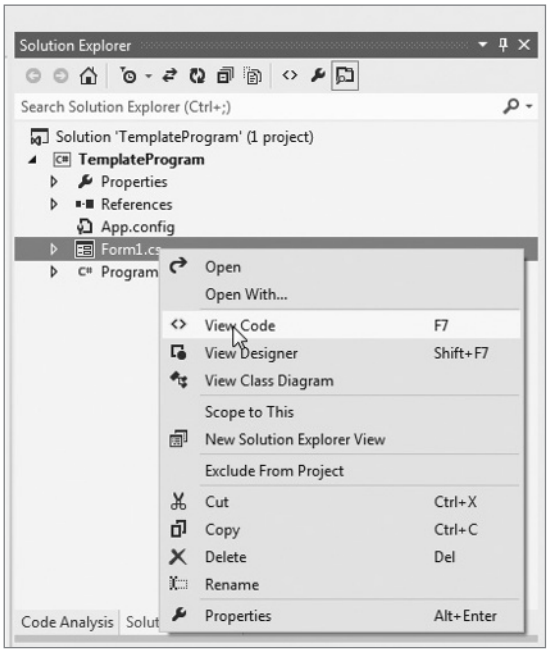

FIGURE 1-9: Right-clicking the form filename

The code associated with right-clicking the filename shown in Figure 1-9 is presented in Figure 1-10. Notice that another tab has been added at the top of the Source window. One tab has Form1.cs [Design] on it, whereas the other tab has Form1.cs. The first tab is the source code mode and shows you the code for the program under development. Even though you haven't written any code, Visual Studio has already written some program code for you behind your back! You can see the `TemplateProgram` C# source code in Figure 1-10. The second tab is for the view of the form in the design mode. (The design mode was shown earlier in Figure 1-8.)

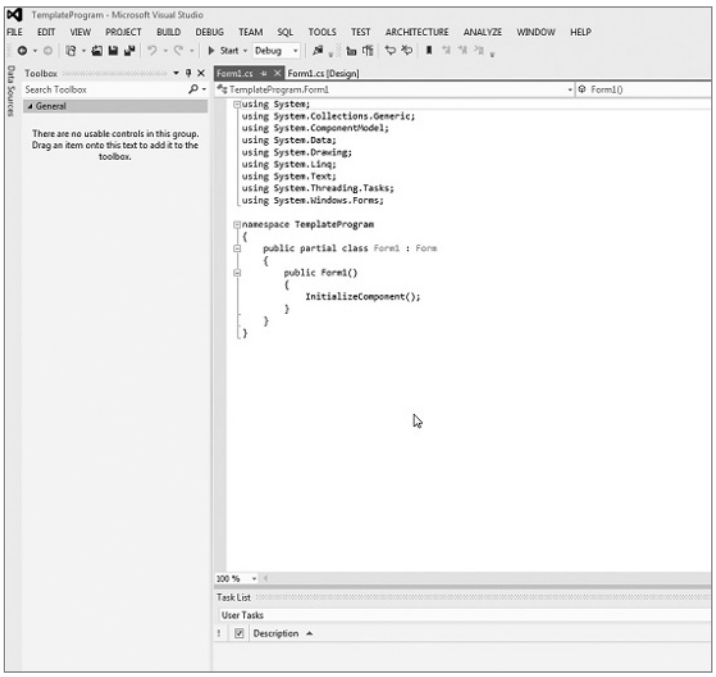

FIGURE 1-10: Source code for file

It's not necessary that you understand the code shown in Figure 1-10 at this time. All you're trying to do at this point is to write a short program to see if the installation was done correctly. However, you will be spending a lot of time in the source code window as you write your own programs.

Adding a Toolbox Object to a Windows Form

A form with nothing on it isn't interesting. Now change the form so that, when the program runs, it displays the message, My First Program. You need to add a label to the form object shown in the Source window in Figure 1-8 to hold your program message. There are two ways to add a label object to a form. First, as shown in Figure 1-11, you can click the label object in the Toolbox, and while holding the left mouse button down, drag the label over to the form and release the mouse button. Second, you can simply double-click the label object in the Toolbox. The mouse double-click causes Visual Studio to place a label object near the upper-left corner of the form shown in the Source window.

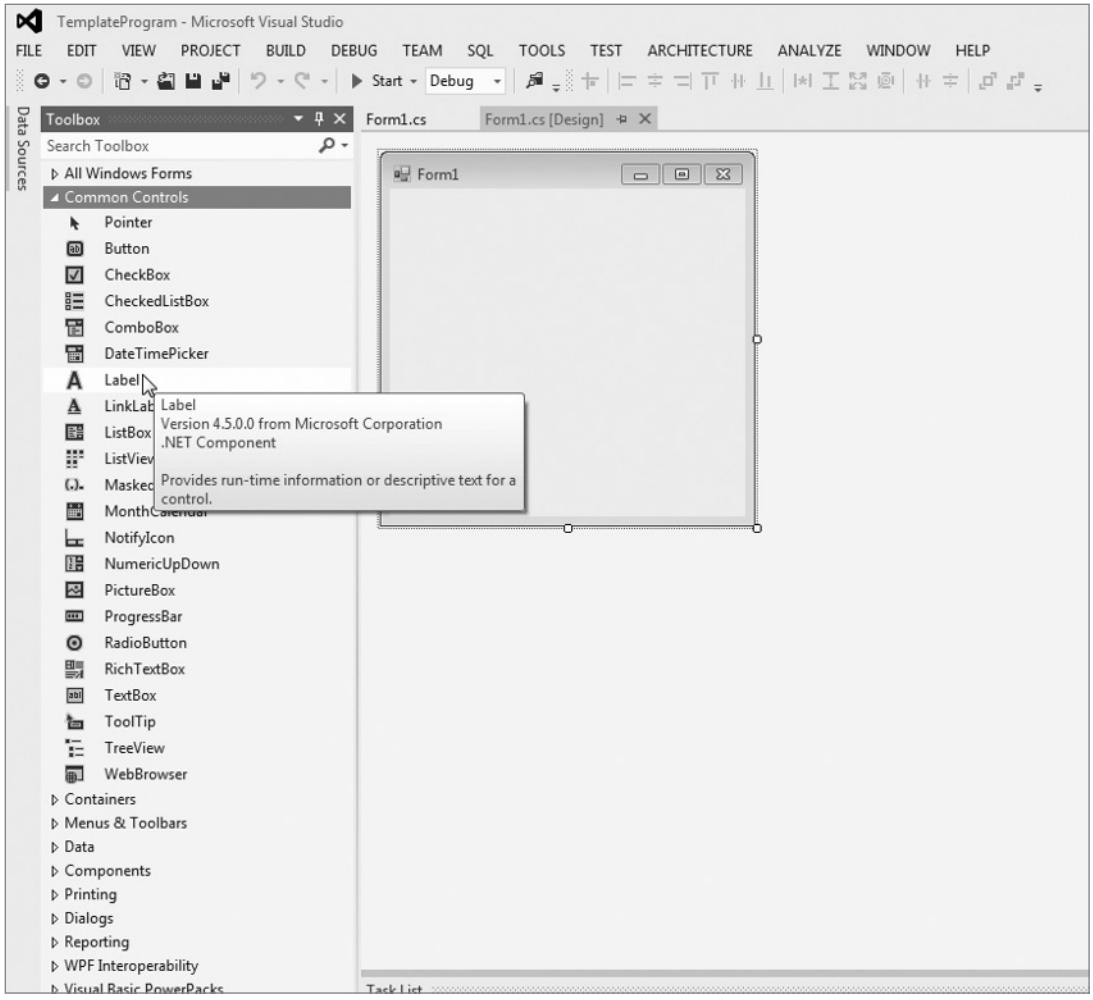

FIGURE 1-11: Adding a label object to a form

You should try both methods to place a label on the form. After you see that both actions do place a label on the form, click one of the labels to move Visual Studio's focus to the label you just clicked. You know this is true because you see the sizing boxes on the label you just clicked. Because you need only one label in this test program, press the Delete key on your keyboard. The label is then removed from the form.

You can click the label in the Source window while holding the left mouse button down and drag the label to wherever you want it to appear on the form object. When you have positioned the label object where you want it on the form, simply release the mouse button.

Changing the Text of a Label Object

After you position the label object on the form, you can set the text for the label. One of the things that Visual Studio does automatically is size the label to be just big enough to accommodate the label's text. Given the default font size used by Visual Studio, the default height for a label is 13 pixels. (A *pixel* is one dot, or point of light, on the screen of your display device.) If you change the font size of the label object, the height of the label is automatically adjusted for you.

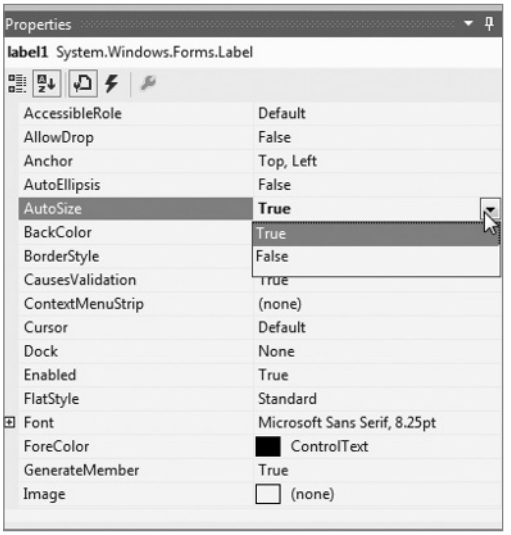

FIGURE 1-12: Changing the AutoSize property

Actually, I am not a big fan of automatic sizing of labels. For that reason, I turn off the auto-sizing feature. To turn off auto-sizing, go to the Properties Window on the lower-right side of the display and change the `AutoSize` property from `True`, as shown in Figure 1-12, to `False`. You may need to scroll the Property window to see the `AutoSize` property displayed in the Properties window. Click the down arrow at the right end of the `AutoSize` textbox to display the choices you have for the property. In this case you can choose only `True` or `False`. Select `False`. The auto-sizing of the label object is now turned off.

After you set `AutoSize` to `False`, you can resize the label object in the Source window by clicking the white sizing boxes that appear on the ends and corners of the label object. You can increase the width of the label by clicking the middle sizing box on the right edge of the label object and dragging the edge to the right until you get the wanted size. Then released the left mouse button to end up with the label object, as shown in Figure 1-13.

After you have set the size of the label object, you can change the `text` property of the label object to whatever you want to appear in the label. In your example, you want the text **My first program**. At the present time, the text property of the label object is `label1`, which is not what you want. Scroll the contents of the Property window down to the `Text` property for the label object, change it to `My first program`, and press the Enter key. (If the property window should ever "disappear" for some reason, press the F4 key to recall it.) Changing the `Text` property for the label object is shown in Figure 1-14. Notice how the first line below the Property window's title bar shows the name of the object currently in focus in the Source window (`label1` in Figure 1-13).

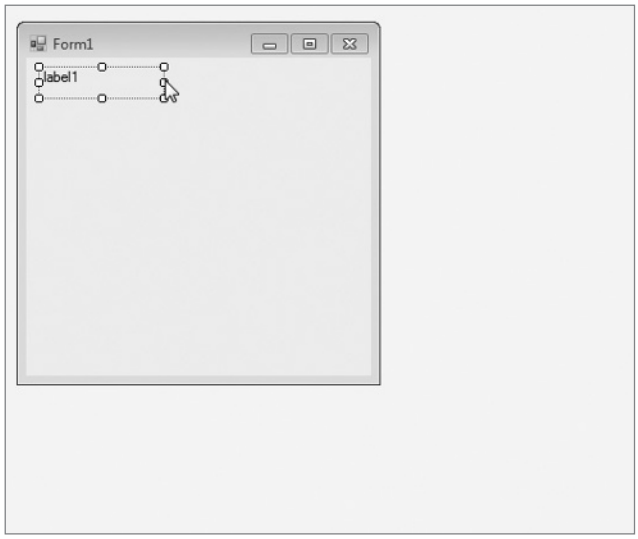

FIGURE 1-13: Resizing the label object

By default, the text of a label object is positioned so that it appears in the upper-left corner of the label. If `AutoSize` is set to `True`, you can't position the text with the label object; it's set for you automatically. However, because you turned off auto-sizing, you can position the text within the label wherever you want. Figure 1-15 shows how to do this using the `TextAlign` property in the Properties window. If you look closely at Figure 1-15, you can see that `TextAlign` is currently set to `TopLeft`. There are nine positions where you can place the text in a label that you can resize. You want to right-justify your text in the center of the label object. If you look where the cursor is positioned in Figure 1-15, clicking that box right-justifies the text in the label object.

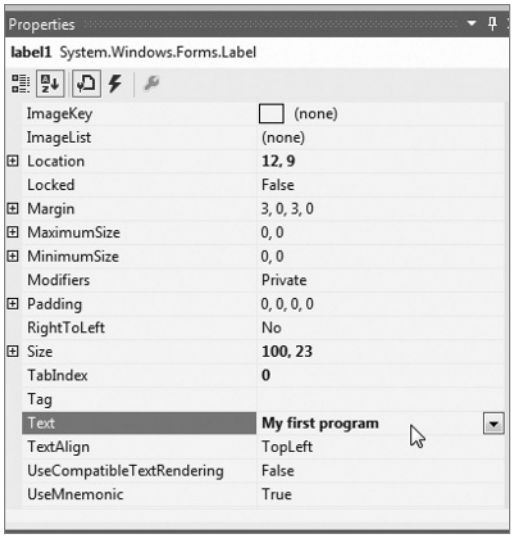

FIGURE 1-14: Changing the Text property of the label object

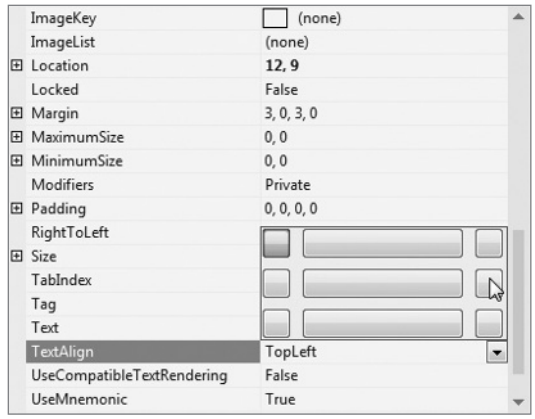

FIGURE 1-15: Changing the TextAlign property for the label object

Running the Program

This is all that you want to do at the present time to test your C# installation. If you can run the program at this point, you have probably installed C# correctly. There are two simple ways to run a program. The easiest is to press the F5 key. If you do that, the program should appear on the screen in a few moments with your message centered in the label object.

The second way is to click the Run Program icon. If you look carefully at Figure 1-11, below the DEBUG menu option near the top of the form you can see a small triangle followed by the word Start. (On your screen, the triangle appears green.) Click that green triangle and the program begins execution.

If your program does not run correctly and you're not getting an error message from within Visual Studio, go back to the start of the installation process and check to make sure you followed each step correctly. Before you start the reinstall, make sure that you have enough free disk space to hold Visual Studio. If you get an error message from Visual Studio, you performed some step in the coding process incorrectly. Just go back to the beginning of the "A Test Program Using Visual Studio" section of this chapter to check each step to ensure you followed the correct sequence.

SUMMARY

In this chapter you learned how object-oriented programming started more than four decades ago. This chapter also showed you how to download and install Visual Studio's Professional edition. Finally, you wrote a simple program to test the installation to make sure it was done correctly. So, now what?

You could immediately proceed to the next chapter and start reading. Not a good idea. Now that you have a simple program up and running, this is the perfect time to experiment a little. For example, your program doesn't have any text in the program's title bar. Surely C# provides a property that enables you to change the title bar. (Hint: It does provide such a property!) Play around with some of the other properties to see what they do. For example, change the foreground property and see what happens.

Looking forward, at the end of each chapter in this book are set of exercises that you should do before reading the subsequent chapter. You will probably be eager to move to the next chapter, but resist the temptation to skip the exercises—be sure to do them. They can help crystallize what you've learned in the current chapter and better prepare you for the content of the next chapter. You can find the solutions to all the exercises in Appendix A.

Programming should be fun, and some of that fun comes from discovering what happens if you change this to that. If you see smoke coming out of your computer, don't make that change again. (Just kidding…you can't hurt your computer if you make an incorrect change to a property.) Experiment and have fun!

▶ WHAT YOU LEARNED IN THIS CHAPTER

TOPIC	KEY POINTS
Short history of OOP	How OOP actually started with Simula in the 1960s but gained strength in the 1980s with C++, Java, and C#.
Download and install C#	The process of downloading and installing Visual Studio.
A sample program	Although very simple, the program does test your installation of Visual Studio.