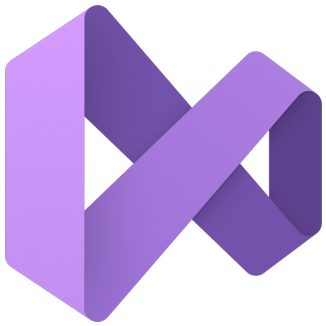


Introduction to Visual Programming with Windows Forms in C#

Introduction to Visual Programming

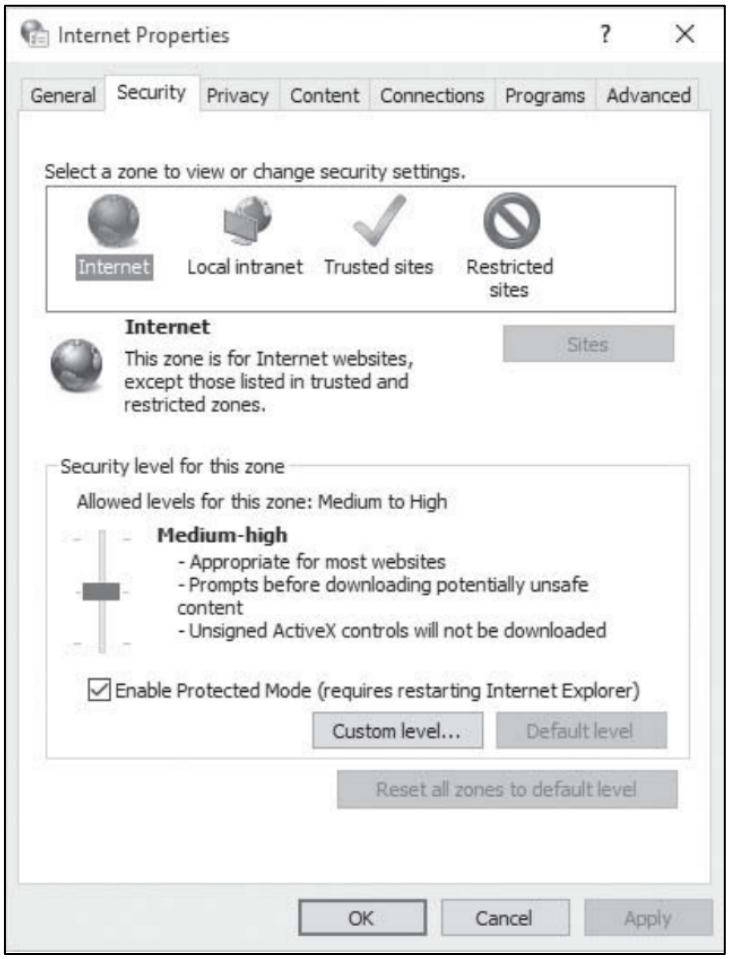
Visual Programming is a style of programming where developers design and build applications through visual elements such as windows, buttons, forms, and menus.
It enables developers to focus on the structure and flow of the application without needing to manually write all the code.
This is especially useful for building user interfaces (UIs) that allow end-users to interact with applications intuitively.

Graphical User Interfaces (GUI): A graphical user interface allows the user to interact with a program using graphical elements such as icons, buttons, and dialog boxes.

A graphical user interface, or GUI, allows the user to interact with the operating system and application programs through graphical elements on the screen.

Figure 1-1 shows an example of a window that allows the user to change the system's Internet settings. Instead of typing cryptic commands, the user interacts with graphical elements such as icons, buttons, and slider bars.

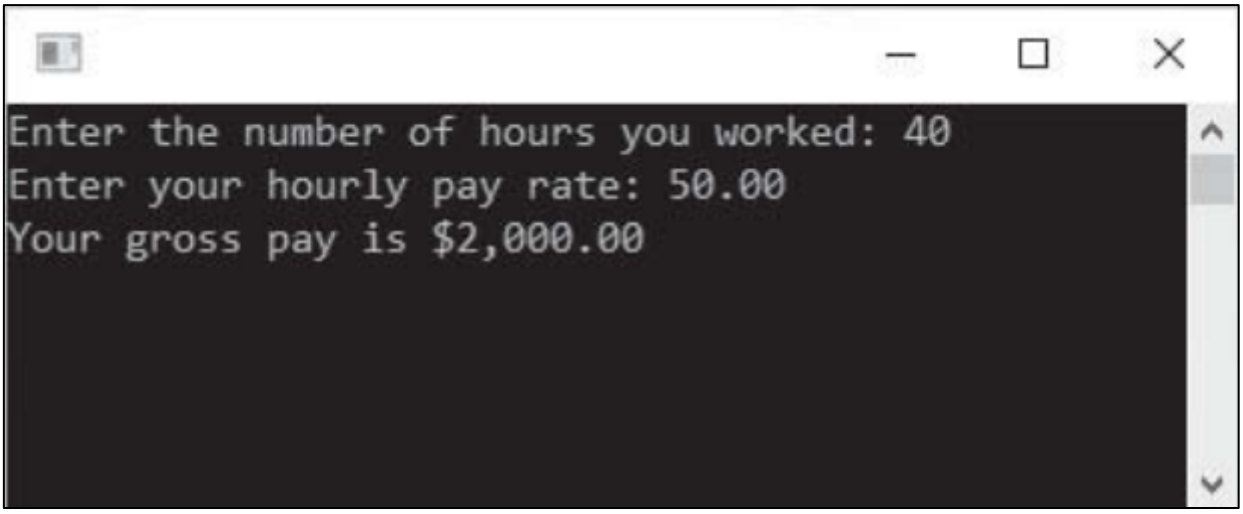
Figure 1-1 A window in a graphical user interface



Event-Driven GUI Programs

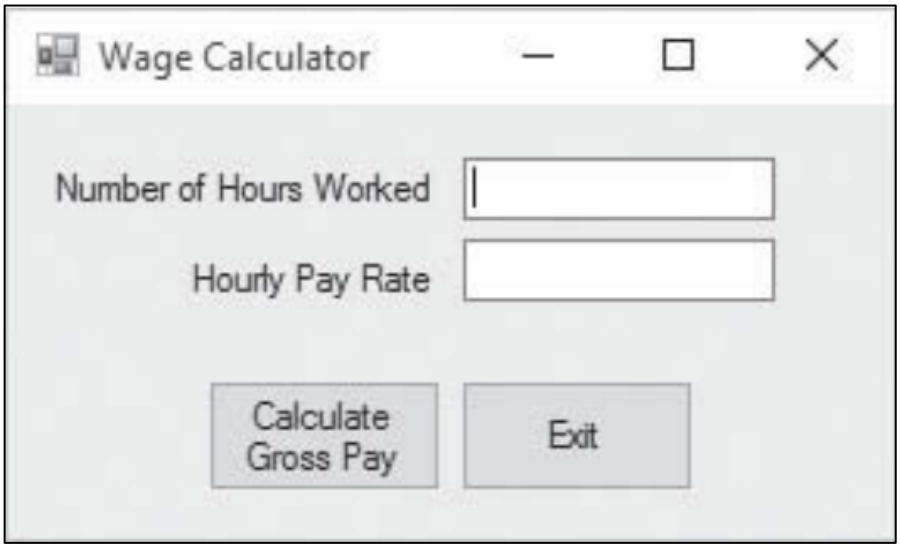
Visual Programming is often event-driven, meaning that the program responds to user actions such as mouse clicks, keystrokes, or other events triggered by the user.
In a text-based environment (such as the command line), the program controls the sequence of actions. For example, when calculating gross pay, the program first asks for the number of hours worked, then the hourly rate, and finally displays the result. The user must enter the data in the exact order requested by the program, As shown in Figure 1.2.

Figure 1-2 Interaction with a program in a text environment



In a GUI environment, the user controls the order of actions. For instance, As illustrated in Figure 1.3, in a gross pay calculator, the user may enter hours worked and hourly rate in any sequence. Mistakes can be corrected by retyping the data, and when ready, the user clicks the **Calculate Gross Pay** button to perform the calculation.

Figure 1-3 A GUI program



The user causes events, such as the clicking of a button, and the program responds to those events.

Objects: An object is a program component that contains data and performs operations. Programs use objects to perform specific tasks.

Example: Have you ever driven a car? If so, you know that a car is made of a lot of components. A car has a steering wheel, an accelerator pedal, a brake pedal, a gear shifter, a speedometer, and numerous other devices with which the driver interacts. There are also a lot of components under the hood, such as the engine, the battery, the radiator, and so forth. A car is not just one single object, but rather a collection of objects that work together

Most programming languages that are used today are object-oriented. When you use an object-oriented language, you create programs by putting together a collection of objects. In programming, an object is not a physical device, but rather like a steering wheel or a brake pedal. Instead, it is a software component that exists in the computer's memory. In software, an object has two general capabilities:

• **"An object can store data"**. The data stored in an object is commonly called fields or properties.

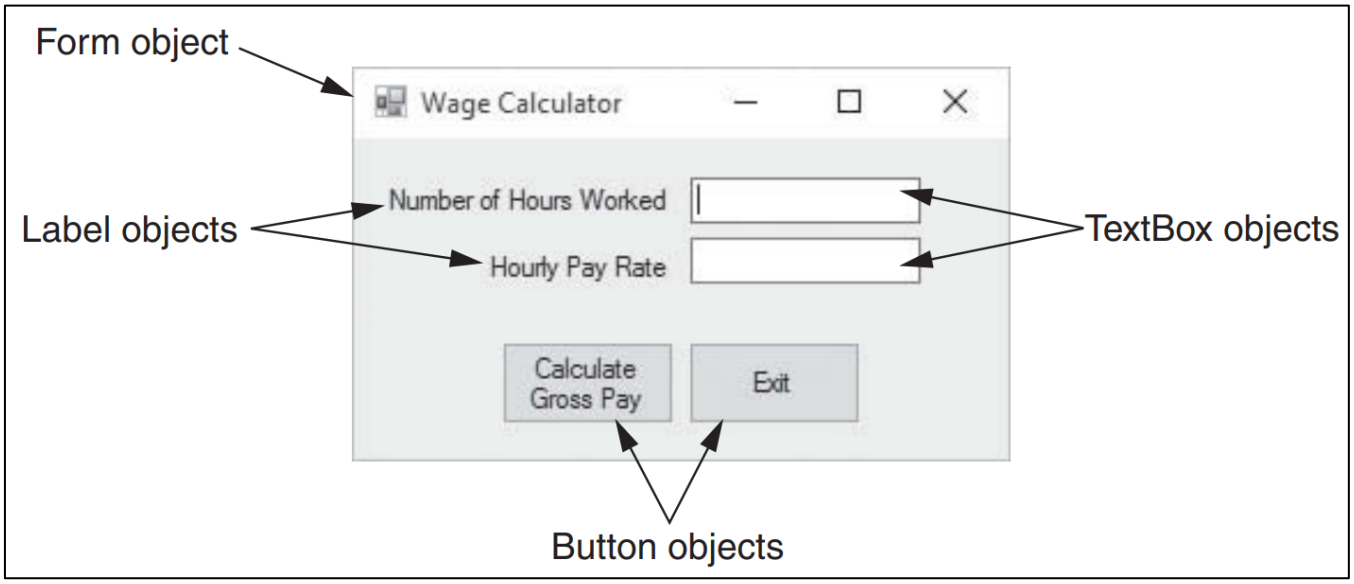
• **"An object can perform operations"**. The operations that an object can perform are called methods.

When you write a program using an object-oriented language, you use objects to accomplish specific tasks. Some objects have a visual part that can be seen on the screen. For example, Figure 1-4 shows the wage-calculator program that we discussed in the previous section. The graphical user interface is made of the few objects showed in the Table 1-1 below.

Table 1.1: Selected Objects and Their Definitions.

Objects	Definition
Form	That contains several other graphical objects.
Label	Displays text on a form.
TextBox	A rectangular region that can accept keyboard input from the user.
Button	A button with a caption written across its face. When the user clicks a Button object with the mouse, an action takes place.

Figure 1-4 Objects used in a GUI



Forms, Labels, TextBoxes, and Buttons are just a few of the objects that you will learn to use in C#. You will create applications that incorporate various types of objects.

Visible versus Invisible Objects

Objects that are visible in a program's graphical user interface are commonly referred to as **controls**. We could say that the form shown in Figure 1-3 contains two Label controls, two TextBox controls, and two Button controls. **When an object is referred to as a control, it simply means that the object plays a role in a program's graphical user interface**. Not all objects can be seen on the screen, however. Some objects exist only in memory for the purpose of helping your program perform some task. For example, there are objects that read data from files, objects that generate random numbers, objects that store and sort large datasets, and so forth. These types of objects help your program perform tasks, but they do not directly display anything on the screen. When you are writing a program, you will use objects that can help your program perform its tasks.

The .NET Framework

It is a collection of classes and other code that can be used, along with a programming language such as C#, to create programs for the Windows operating system. For example, the .NET Framework provides classes to create Forms, TextBoxes, Labels, Buttons, and many other types of objects.

When you use Visual C# to write programs, you are using a combination of the C# language and the .NET Framework.

The Visual Studio Environment

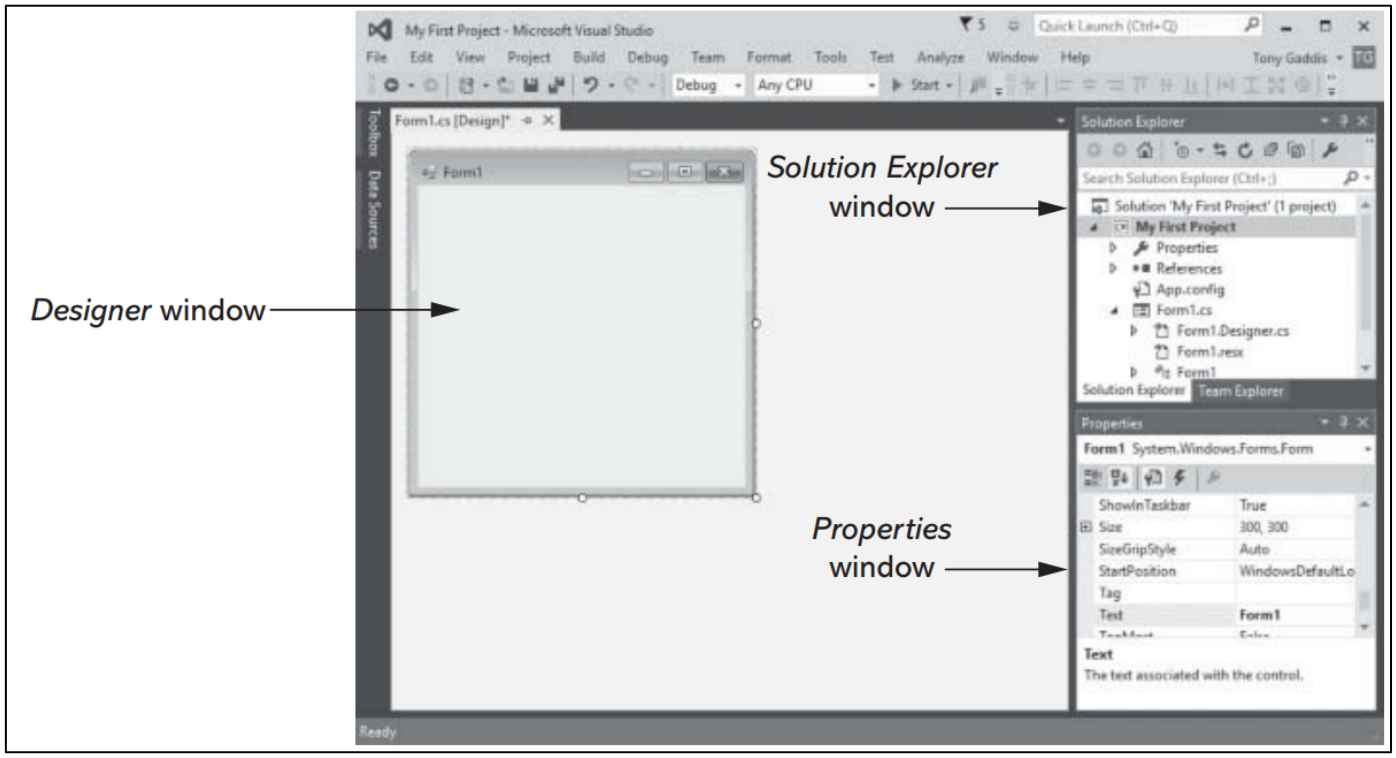
The Visual Studio environment consists of several windows that you will use on a regular basis. Figure 1-5 shows the locations of the following windows that appear within the Visual Studio environment: the Designer window, the Solution Explorer window, and the Properties window. Here is a brief summary of each window's purpose:

The Designer Window: You use the Designer window to create an application's graphical user interface. The Designer window shows the application's form and allows you to visually design its appearance by placing the desired controls that will appear on the form when the application executes.

The Solution Explorer Window: A solution is a container for holding Visual C# projects. The Solution Explorer window allows you to navigate among the files in a Visual C# project.

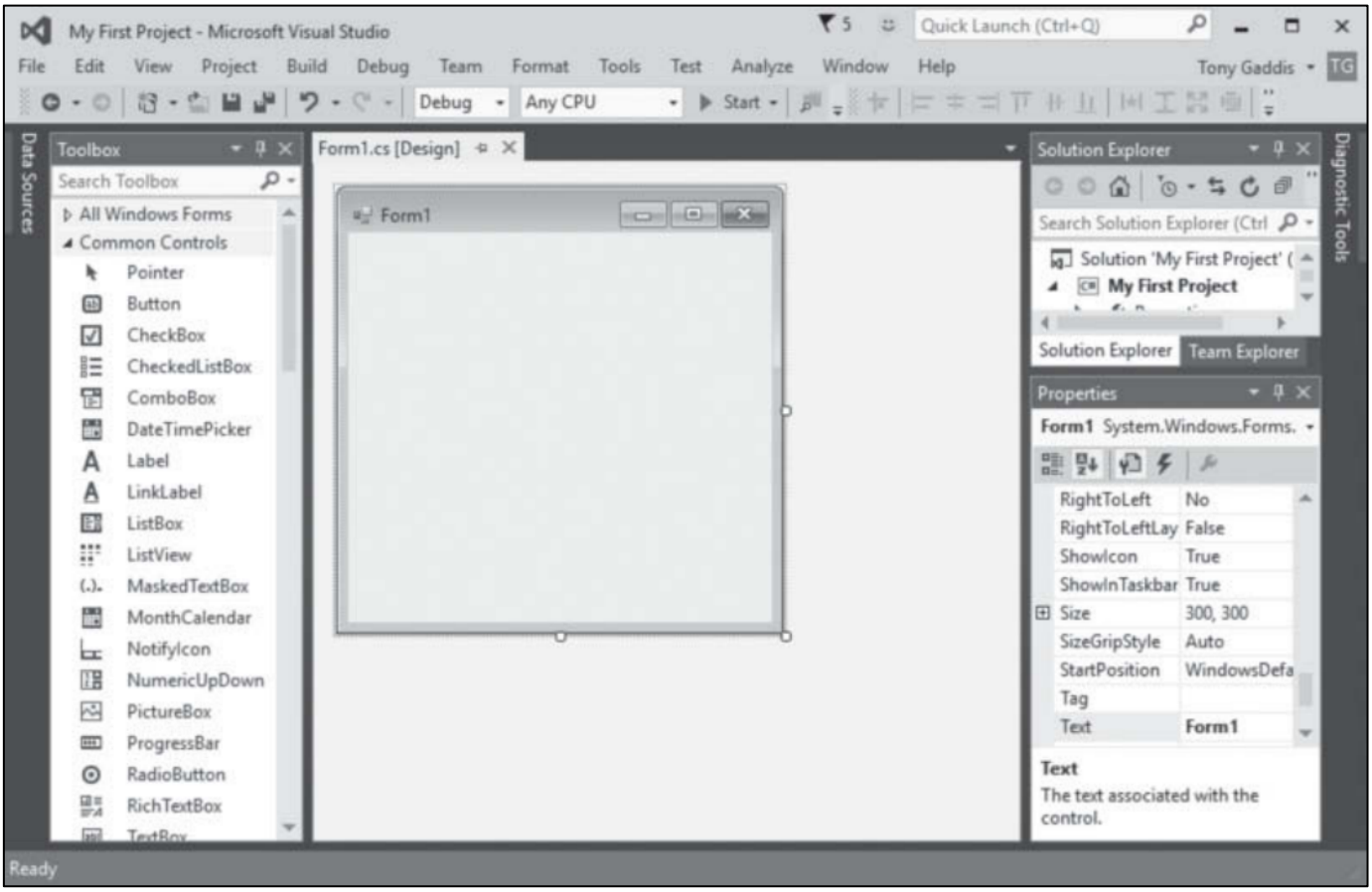
The Properties Window: A control's appearance and other characteristics are determined by the control's properties. When you are creating a Visual C# application, you use the Properties window to examine and change a control's properties.

Figure 1-5 The Designer window, Solution Explorer window, and Properties window



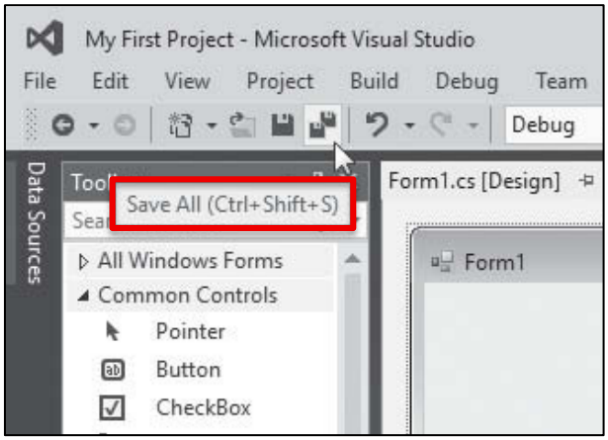
The Toolbox: is a window that allows you to select the controls that you want to use in an application's user interface, as shown in the Figure 1-6. When you want to place a Button, Label, TextBox, or other control on an application's form, you select it in the Toolbox.

Figure 1-6 The Toolbox



ToolTips: is a box contains a short description of the button's purpose, as Figure 1-7 showed.

Figure 1-7 Save All ToolTip



What is Windows Form?

Windows Form (WinForm) is a Graphical User Interface (GUI) framework in the .NET environment. It provides a wide set of controls and tools that make it easy to design desktop applications for Windows.

Forms and Controls

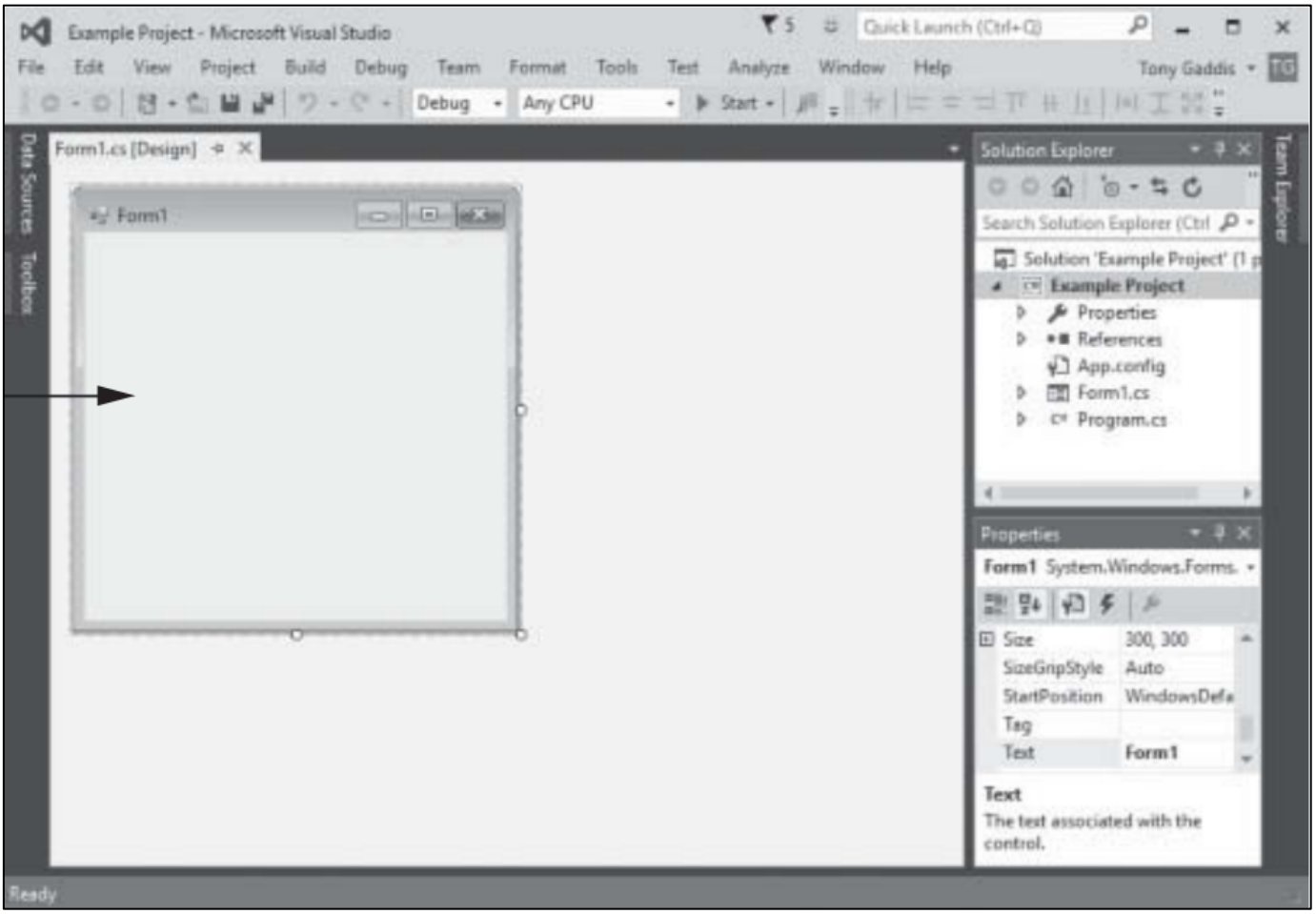
Instead of writing all interface code manually, developers can drag-and-drop controls from the Visual Studio Toolbox and then write event-handling code.

The first step in creating a Visual C# application is creating the application's GUI. You use the Visual Studio Designer, Toolbox, and Properties window to build the application's form with the desired controls and set each control's properties.

The Application's Form

When you start a new Visual C# project, Visual Studio automatically creates an empty form and displays it in the Designer. Figure 1-8 shows an example. Think of the empty form as a blank canvas that can be used to create the application's user interface. You can add controls to the form, change the form's size, and modify many of its characteristics. When the application runs, the form will be displayed on the screen.

Figure 1-8 A new project with a blank form displayed in the Designer



Identifying Forms and Controls by Their Names

An application's GUI is made of forms and various controls. Each form and control in an application's GUI must have a name that identifies it. The blank form that Visual Studio initially creates in a new project is named Form1.

The Properties Window

The appearance and other characteristics of a GUI object are determined by the object's properties. When you select an object in the Designer, that object's properties are displayed in the Properties window. For example, when the Form1 form is selected, its properties are displayed in the Properties window, as shown in Figure 1-9 and Figure 1-10.

Figure 1-9 The Properties window, showing the selected object's properties

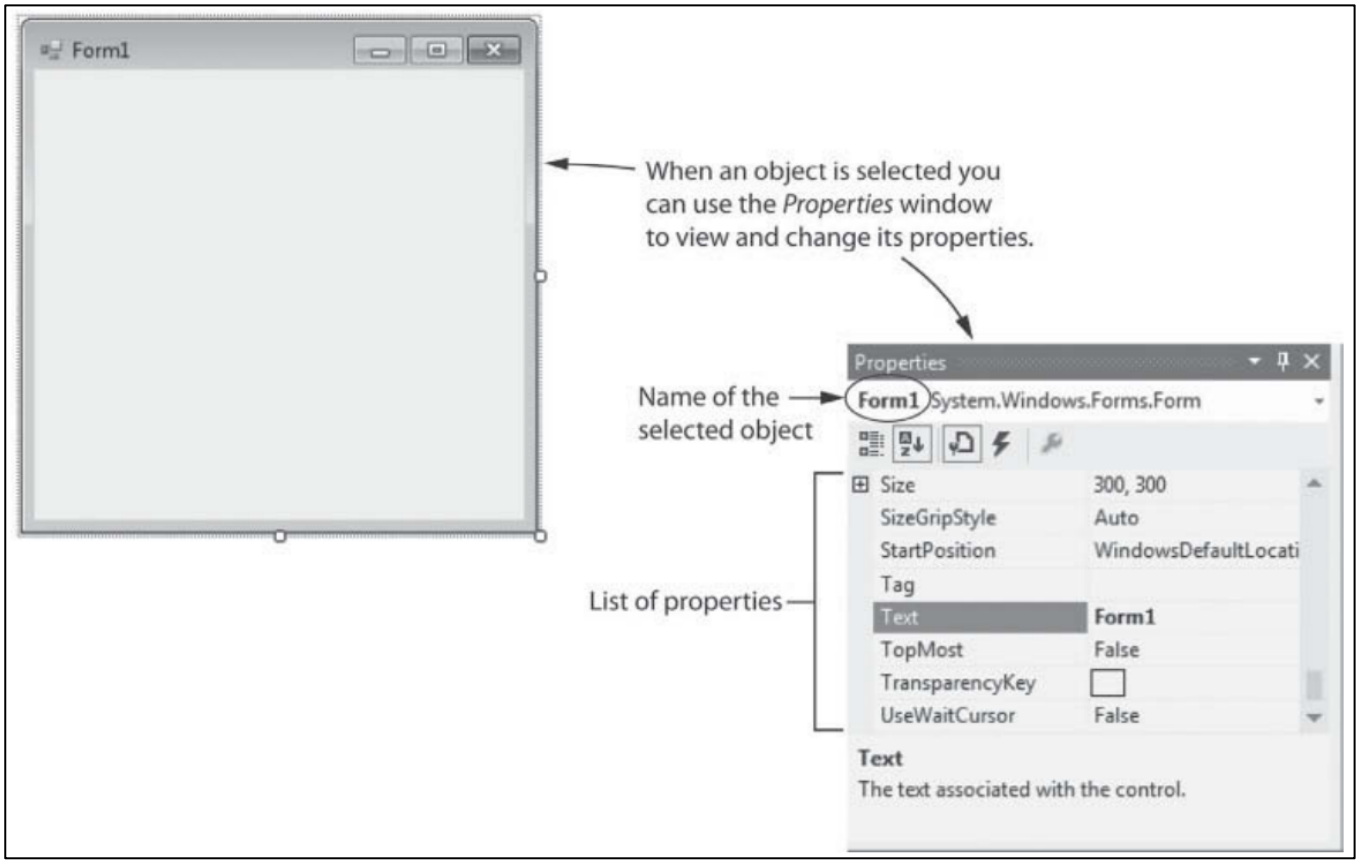
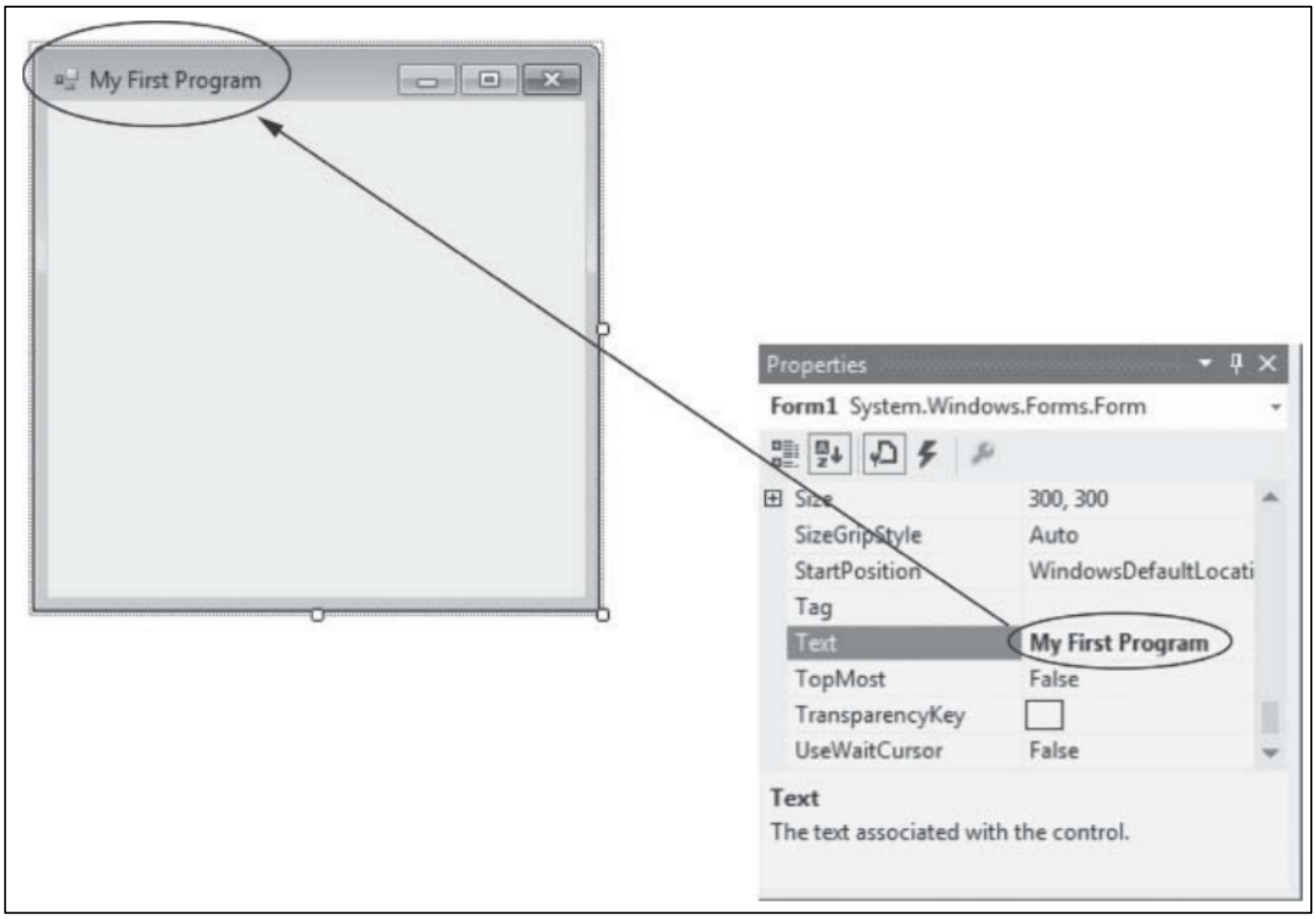


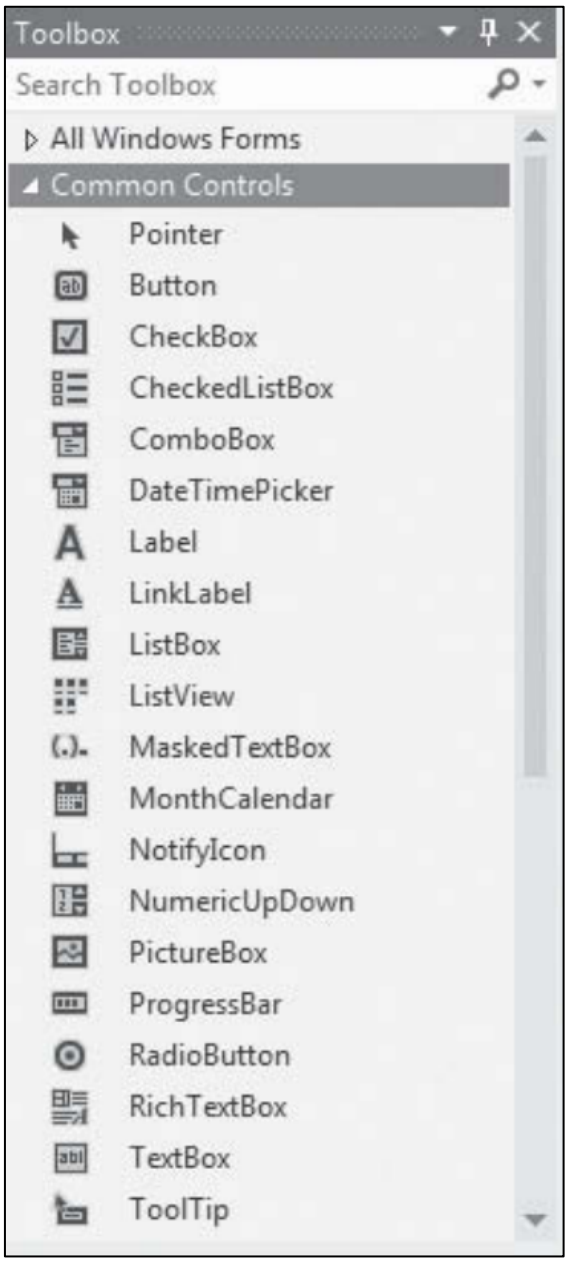
Figure 1-10 The form's Text property value displayed in the form's title bar



Adding Controls to a Form

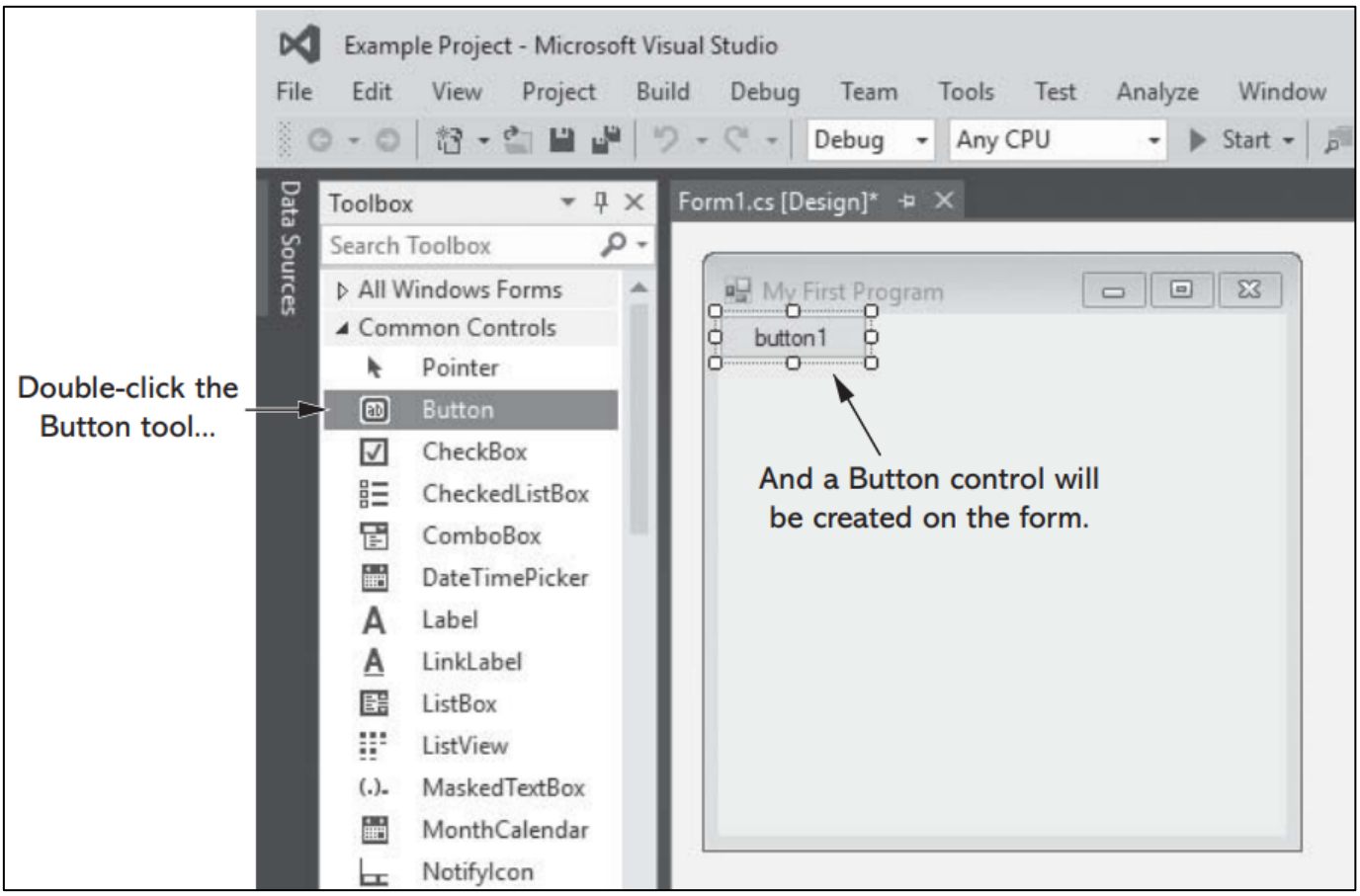
When you are ready to create controls on the application's form, you use the Toolbox, Figure 1-11 showing the common controls.

Figure 1-11 Common Controls in The Toolbox



The Toolbox shows a scrollable list of controls that you can add to a form. To add a control to a form, you find it in the Toolbox and then double-click it, as illustrated in the Figure 1-12. The control will be created on the form. Also, you can click and drag controls from the Toolbox onto the form.

Figure 1-12 Creating a Button control

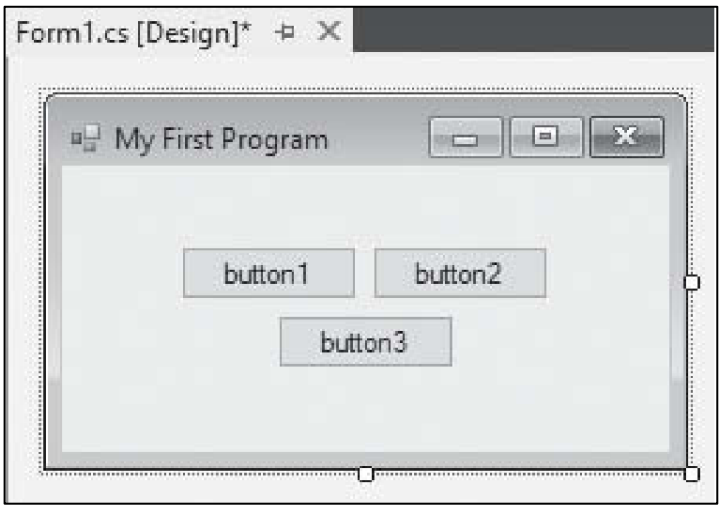


Button Controls

When you create Button controls, they are automatically given default names such as button1, button2, and so forth, as shown in the Figure 1-13.

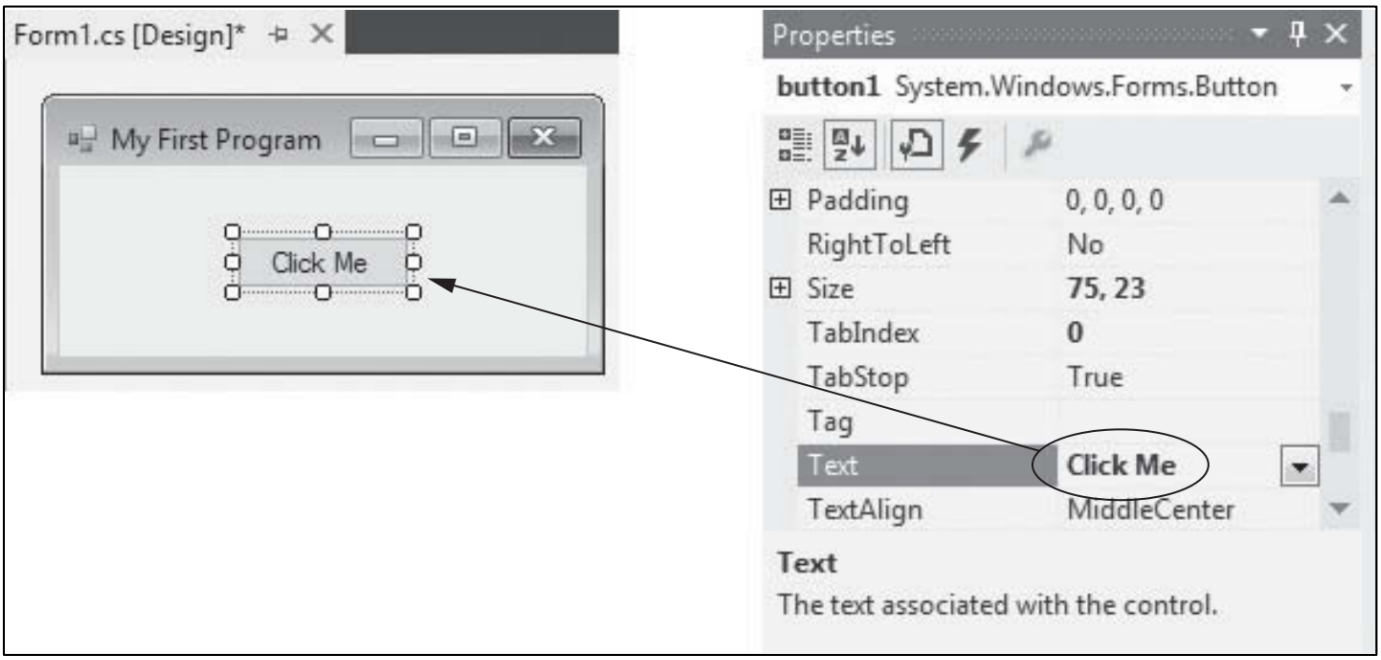
Button controls have a Text property, which holds the text that is displayed on the face of the button. When a Button control is created, its Text property is initially set to the same value as the Button control's name.

Figure 1-13 A form with three Button controls



After you create a Button control, you should always change its Text property. The text that is displayed on a button should indicate what the button will do when it is clicked, as Figure 1-14. For example, a button that calculates an average might have the text Calculate Average displayed on it.

Figure 1-14 A Button control's Text property changed



Difference between Names and Texts

The table below Table 1-2 shown the difference between each on Names and Texts in the windows Form applications. Figure 1-15 and Figure 1-16 display the place of existing of each Name and Text.

Table 1-2: The Difference between Names and Texts.

Aspect	Name	Text
Definition	Identifier of the control in code	Caption or content shown on the control
Visible to user?	No, hidden at runtime	Yes, shown at runtime (if the control supports text)
Purpose	Used by the developer to reference the control in C# code	Provides information, labels, or prompts to the user
Set by	Developer in the **Properties Window** or directly in code	Developer in the **Properties Window**, but the user can sometimes change it (e.g., TextBox)
Naming rules	Must follow C# identifier rules (no spaces, special chars, etc.)	Can be any string (spaces, symbols, etc.)
Example (Button)	Name = btnLogin	Text = "Login" (appears on the button)

Figure 1-15 The Name property

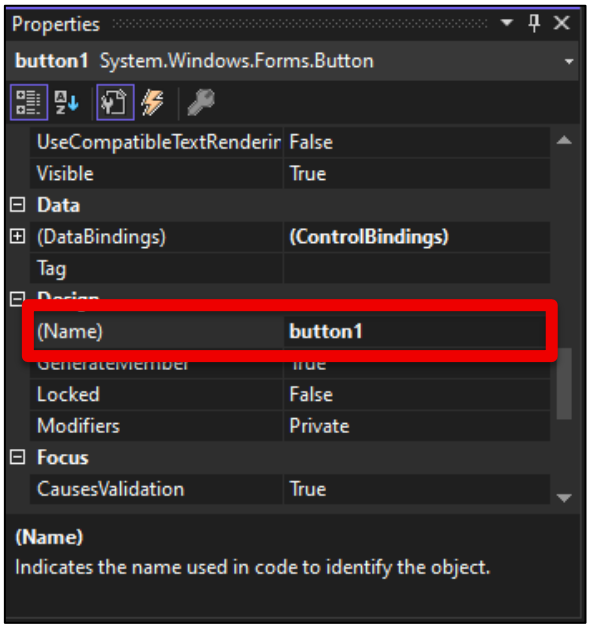
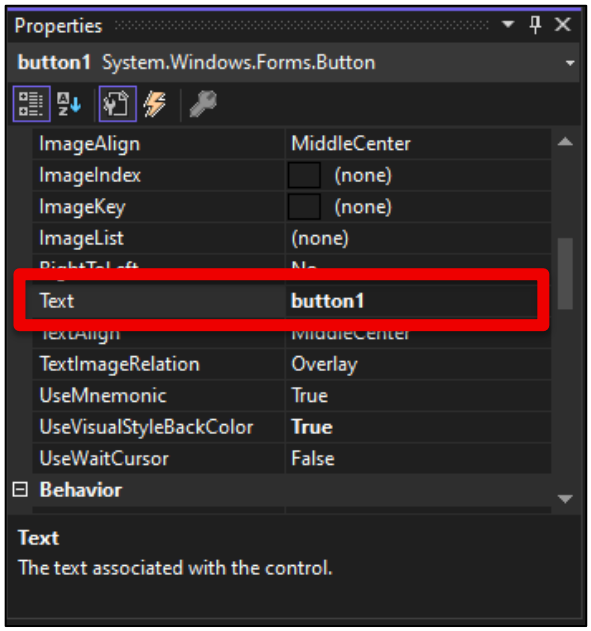


Figure 1-16 The Text property



Rules for Naming Controls

Control names are also known as identifiers. When naming a control, you must follow these rules for C# identifiers, Table 1-3 shown some examples:

• The first character must be one of the letters a through z or A through Z or an underscore character (_).

• After the first character, you may use the letters a through z or A through Z, the digits 0 through 9, or underscores.

• The name cannot contain spaces.

Table 2-1 Legal and illegal identifiers

Identifier	Legal or Illegal?
`showDayOfWeekButton`	Legal
`3rdQuarterButton`	Illegal because identifiers cannot begin with a digit.
`change*color*Button`	Illegal because the * character is not allowed.
`displayTotalButton`	Legal
`calculate Tax Button`	Illegal because identifiers cannot contain spaces.

References:

- Gaddis, T. (2017). *Starting out with Visual C#* (4th ed.). Pearson.
- Sharp, J. (2013). Microsoft Visual C# 2013 step by step. Pearson Education.