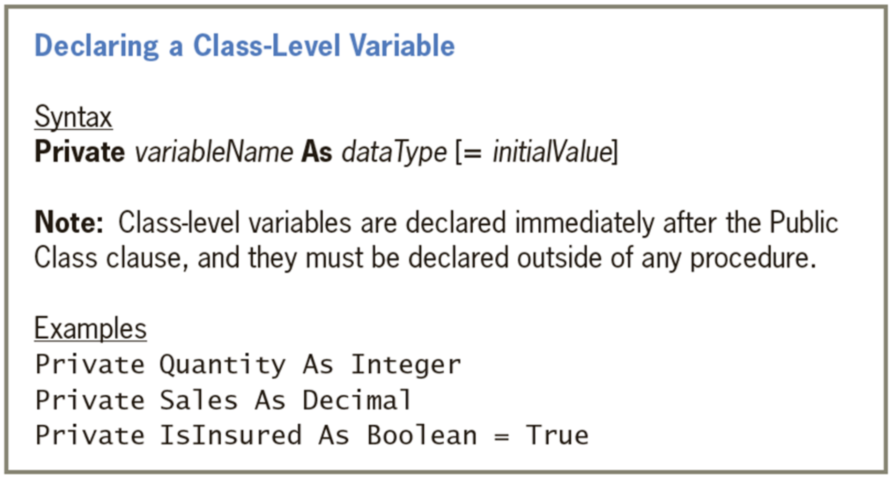
Determine a Memory Location’s Scope and Lifetime

Use a Class-Level Variable

Class-level variables are declared immediately after the Public Class clause in the Code Editor window, and they can be used by any of the procedures entered in the window. Class-level variables retain their values and remain in the computer’s main memory until the application ends. Figure 2-20 shows the syntax and examples of declaring a class-level variable. As the figure indicates, class-level variables are declared using the Private keyword.



Declaring a Class-Level Variable

Syntax
Private *variableName* **As** *dataType* [= *initialValue*]

Note: Class-level variables are declared immediately after the Public Class clause, and they must be declared outside of any procedure.

Examples
```
Private Quantity As Integer
Private Sales As Decimal
Private IsInsured As Boolean = True
```

Figure 2-20 Syntax and examples of declaring class-level variables

You can use a class-level variable when two or more procedures in the same form need access to the same variable. You can also use a class-level variable when a procedure needs to retain a variable’s value even after the procedure ends. This use of a class-level variable is illustrated in the Total Scores Accumulator application, which calculates and displays the total of the scores entered by the user. The application’s interface is shown in Figure 2-21.

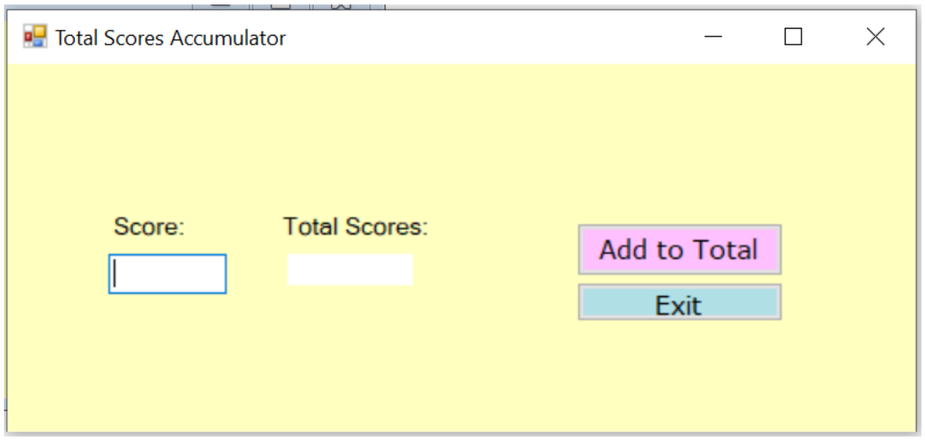
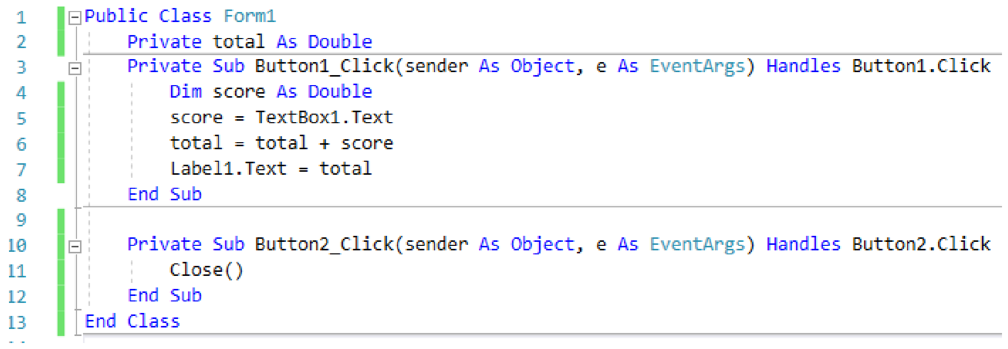


Figure 2-21 User interface for the Total Scores Accumulator application

Figure 2-22 shows most of the application’s code, which uses a class-level variable named

total to accumulate (add together) the *scores* entered by the user.



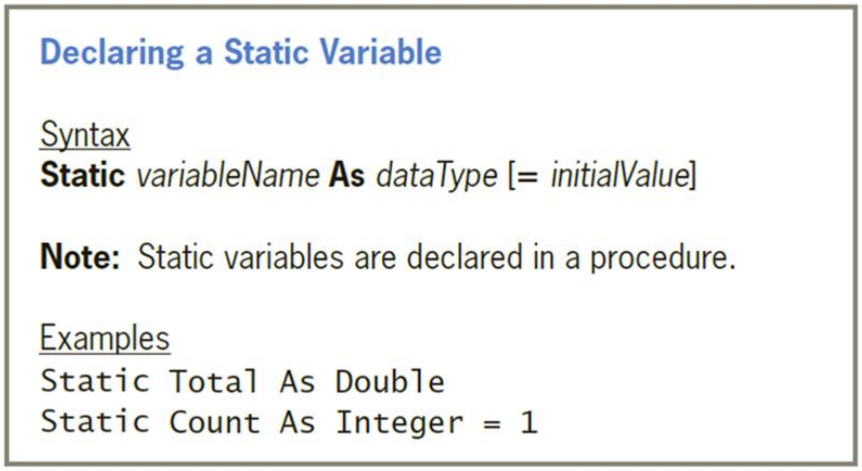
```
Public Class Form1
    Private total As Double
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim score As Double
        score = TextBox1.Text
        total = total + score
        Label1.Text = total
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        Close()
    End Sub
End Class
```

Figure 2-22 The application’s code using a class-level variable

When the user starts the application, the computer processes the ***Private total As Double*** statement first. The statement creates and initializes (to 0) the class-level total variable. The variable is created and initialized only once, when the application starts. It remains in the computer’s main memory until the application ends. Each time the user clicks the Add to total button, the Dim statement on Line 4 in the button1_Click procedure creates and initializes a procedure-level variable named score. The statement in Line 5 stores the content of TextBox1.Text property to the *score* variable. The assignment statement on Line 6 adds the contents of the procedure-level score variable to the contents of the class-level total variable. At this point, the total variable contains the sum of all of the scores entered so far. The assignment statement on Line 7 assigns the total variable's value to the Label1.Text property. When the procedure ends, the computer removes the procedure level score variable from its main memory. However, it does not remove the class-level total variable. The total variable is removed from the computer's memory only when the application ends. As mentioned earlier, a class-level variable can be accessed by any of the procedures entered in its Code Editor window. As a result, using class-level variables can lead to unexpected results when one of the procedures makes an inadvertent or incorrect change to the variable's value. Tracking down the errors in an application's code becomes more complicated as the number of procedures having access to the same variable increases. Therefore, the use of class-level variables should be minimized. Always keep in mind that fewer unintentional errors occur when an application's variables are declared using the minimum scope needed, which usually is procedure scope. Rather than using a class-level variable to accumulate values, you can use a static variable.

Use a Static Variable

A static variable is a procedure-level variable that remains in memory and also retains its value even when its declaring procedure ends. Like a class-level variable, a static variable is not removed from the computer's main memory until the application ends. However, unlike a class-level variable, a static variable can be used only by the procedure in which it is declared. In other words, a static variable has a narrower (or more restrictive) scope than does a class-level variable. As mentioned earlier, many unintentional errors in your code can be avoided by simply declaring the variables using the minimum scope needed. Figure 2-23 shows the syntax and examples of declaring static variables. Keep in mind that the Static keyword can be used only in a procedure.



Declaring a Static Variable

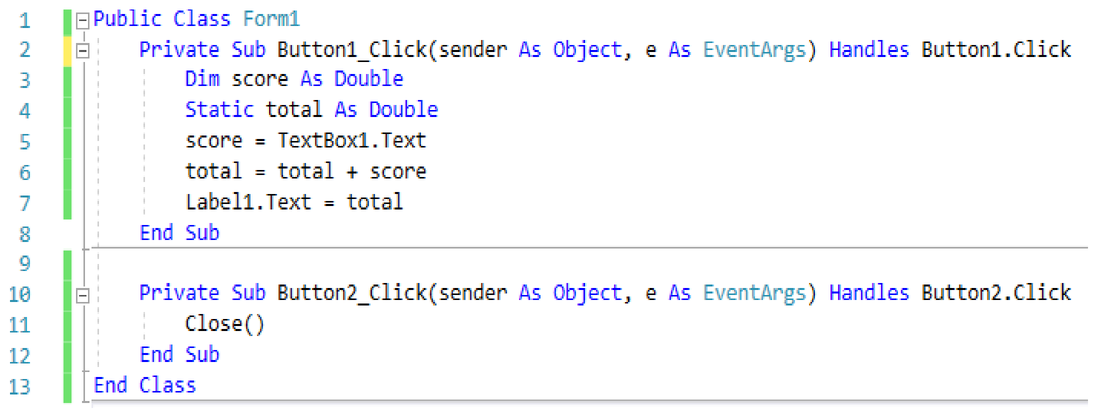
Syntax
Static *variableName* **As** *dataType* [= *initialValue*]

Note: Static variables are declared in a procedure.

Examples
```
Static Total As Double
Static Count As Integer = 1
```

Figure 2-23 Syntax and examples of declaring static variables

The Total Scores Accumulator application from the previous section used a class-level variable to accumulate the scores entered by the user. Rather than using a class-level variable for that purpose, you can also use a static variable, as shown in the code in Figure 2-24.



```
Public Class Form1
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click
        Dim score As Double
        Static total As Double
        score = TextBox1.Text
        total = total + score
        Label1.Text = total
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        Close()
    End Sub
End Class
```

Figure 2-24 The application’s code using a static variable

The first time the user clicks the Add to total button, the button's procedure creates and initializes (to 0) a procedure-level variable named score and a static variable named total. The assignment statement on Line 5 stores the content of TextBox1.Text property in the *score* variable. The assignment statement on Line 6 adds the contents of the score variable to the contents of the static total variable. The assignment statement on Line 7 assigns the total variable's value to the Label1.Text property. When the procedure ends, the computer removes the variable declared using the Dim keyword(score) from its main memory. But it does not remove the variable declared using the Static keyword (total). Each subsequent time the user clicks the Add to total button, the computer recreates and reinitializes the score variable. However, it does not recreate or reinitialize the static total variable because that variable, as well as its current value, is still in the computer's memory. After recreating and reinitializing the score variable, the computer processes the remaining instructions contained in the button's procedure. Here again, each time the procedure ends, the score variable is removed from the computer's main memory. The total variable is removed only when the application ends.