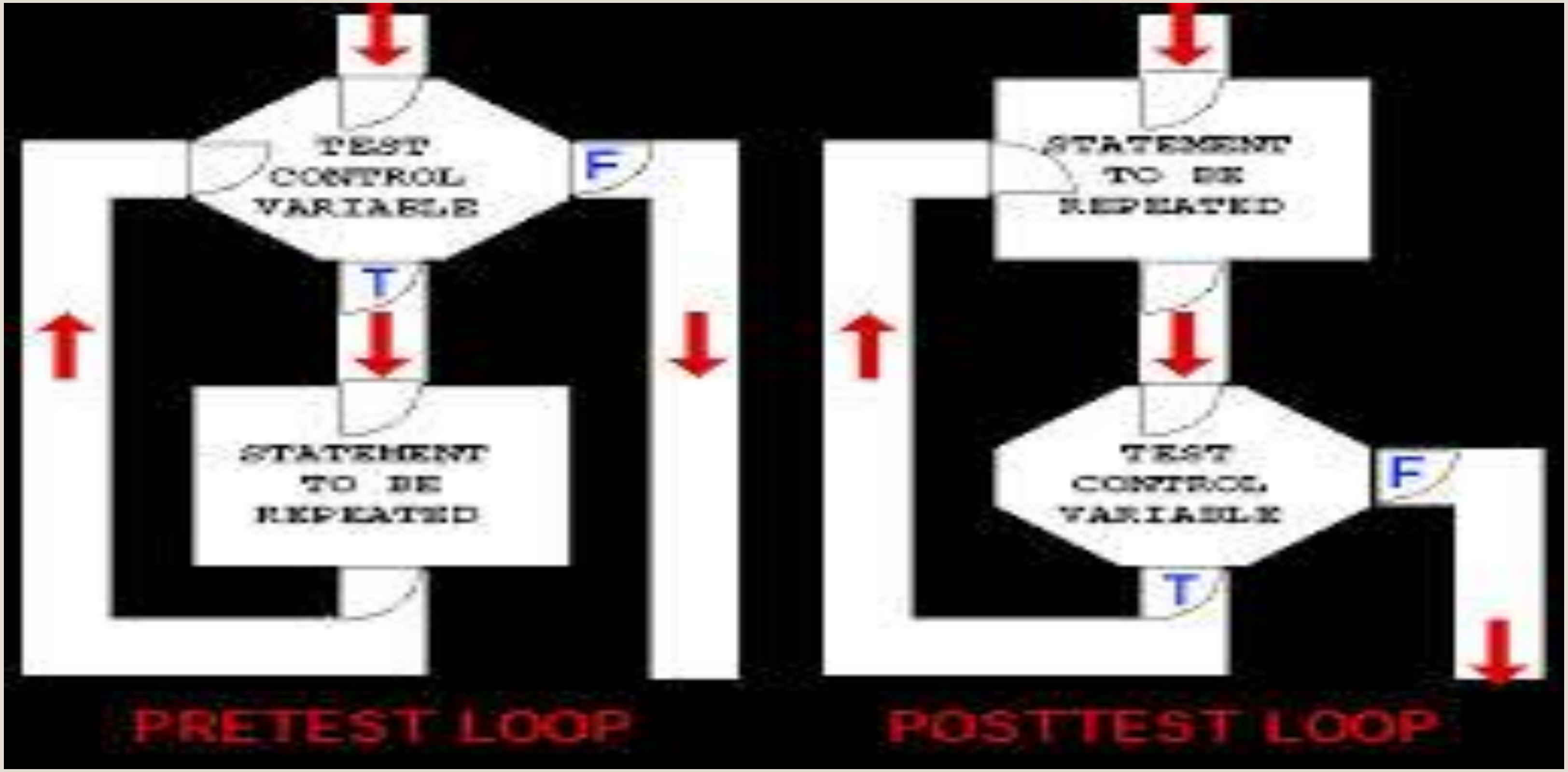
Introducing the Do…Loop While and Do…Loop Until Repetition Statements



App Requirements

A teacher regularly gives quizzes to a class of 10 students. The grades on these quizzes are integers in the range from 0 to 100 (0 and 100 are both valid grades). The teacher would like you to develop an app that computes the class average for one quiz.

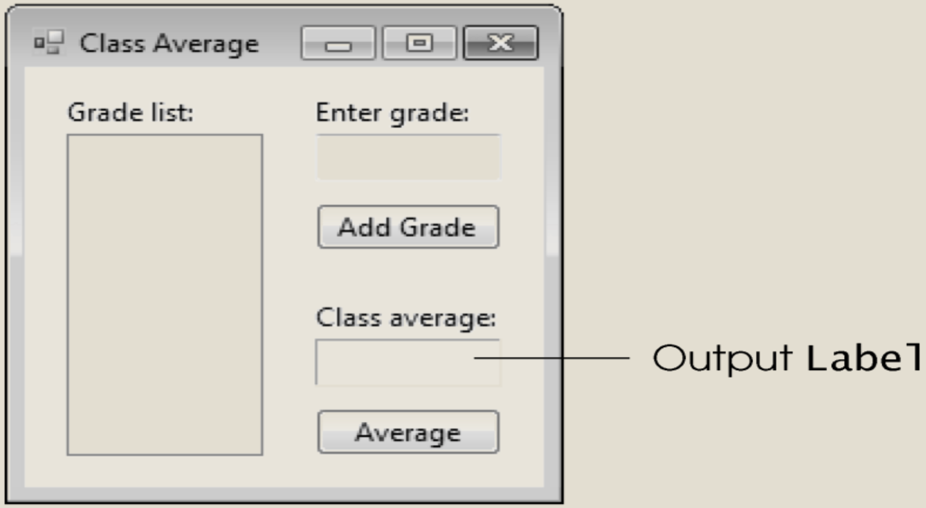


Figure 10.1 **Class Average** app's `Form` in run mode.

Test-Driving the Class Average App (Cont.)

- After you click the **Add Grade** `Button`, the cursor appears in the **Enter grade:** `TextBox` (Fig. 10.2).
  - When a control is selected, it is said to have the **focus** of the app.
  - Transferring the focus tells the user what information the app expects next.

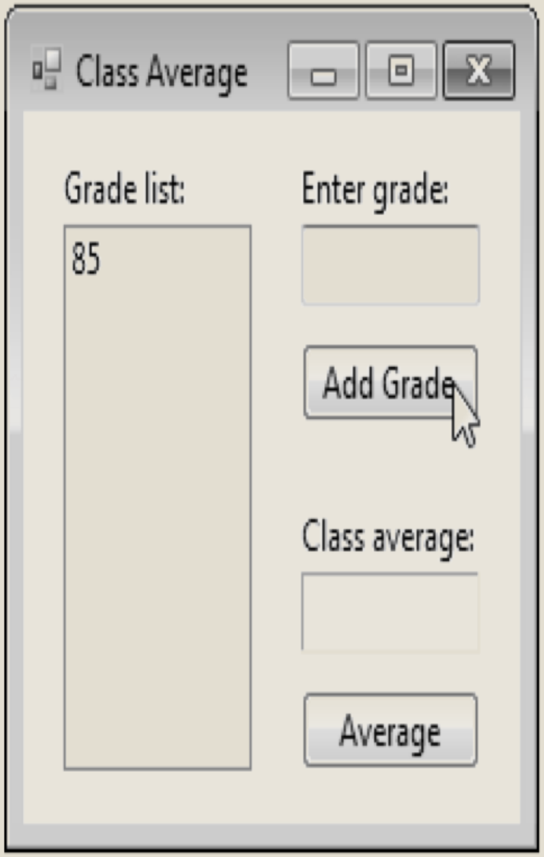
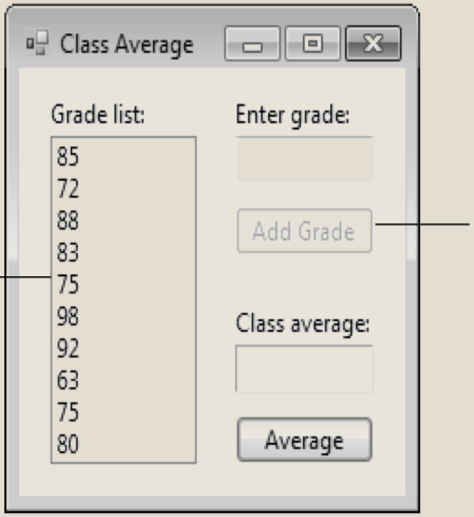


Figure 10.2 Entering grades in the **Class Average** app.

Test-Driving the Class Average App (Cont.)

- Enter nine other grades between 0 and 100.
- Note that the **Add Grade** `Button` is disabled once you have entered 10 grades (Fig. 10.3).



Grade list:

85
72
88
83
75
98
92
63
75
80

Enter grade:

Class average:

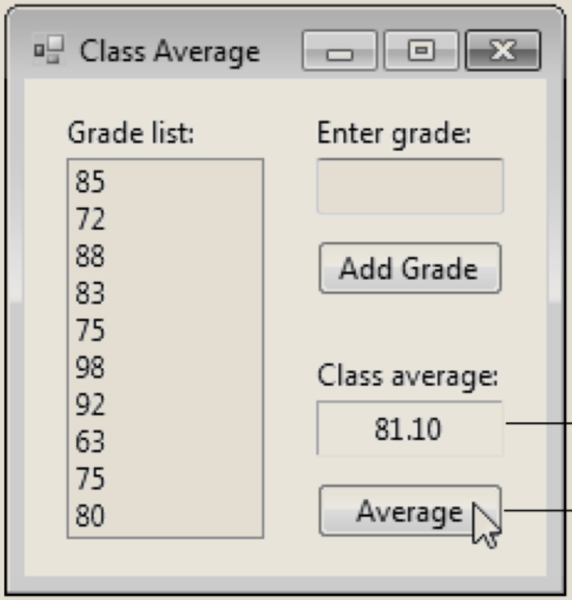
Ten quiz grades entered

Disabled **Add Grade** `Button`

Figure 10.3 **Class Average** app after 10 grades have been input.

Test-Driving the Class Average App (Cont.)

- Click the **Average** `Button` to calculate the average of the 10 quizzes (Fig. 10.4).



Grade list:

85
72
88
83
75
98
92
63
75
80

Enter grade:

Class average: 81.10

`Label` displaying average

Click to calculate class average

Figure 10.4 Displaying the class average.

Test-Driving the Class Average App (Cont.)

- You can calculate the class average for another set of 10 grades without restarting the app.
  - Enter a grade in the `TextBox`, and click the **Add Grade** `Button`.
  - Note that the **Grade list:** `ListBox` and the **Class average:** field are cleared when you start entering another set of grades (Fig. 10.5).

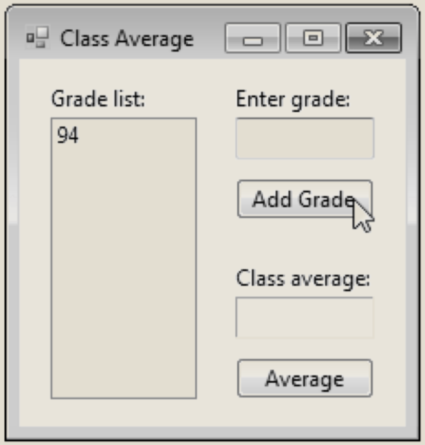
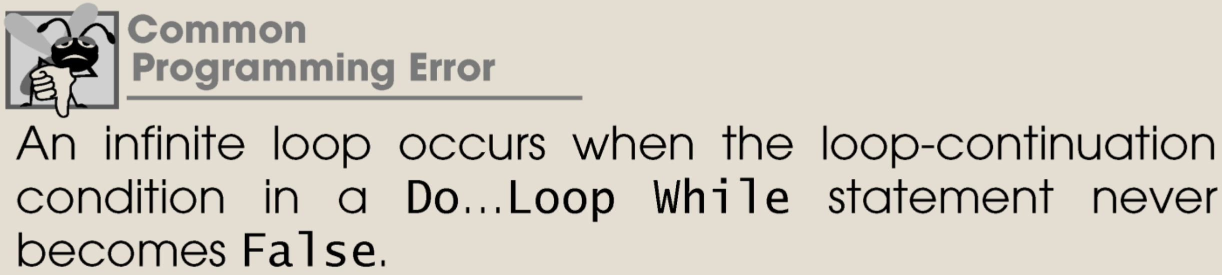


Figure 10.5 Entering a new set of grades.

10.2 Do...Loop While Repetition Statement

- **Do...Loop While** repetition statement is similar to the Do...While Loop statement, except that the loop-termination condition is tested *after* the loop body is performed.



Common Programming Error

An infinite loop occurs when the loop-continuation condition in a Do...Loop While statement never becomes False.

10.2 Do...Loop While Repetition Statement (Cont.)

- The following app segment displays the numbers 1 through 3 in a ListBox:

```
Dim counter As Integer = 1

Do
   displayListBox.Items.Add(counter)
   counter += 1
Loop While counter <= 3
```

10.2 Do...Loop While Repetition Statement (Cont.)

- Figure 10.6 illustrates the UML activity diagram for the general Do...Loop While statement.

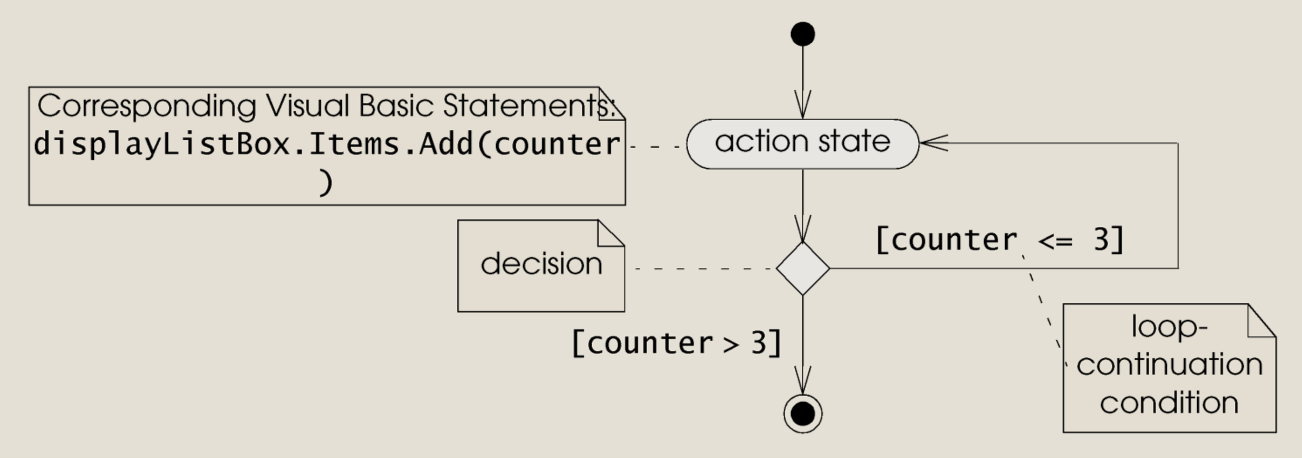
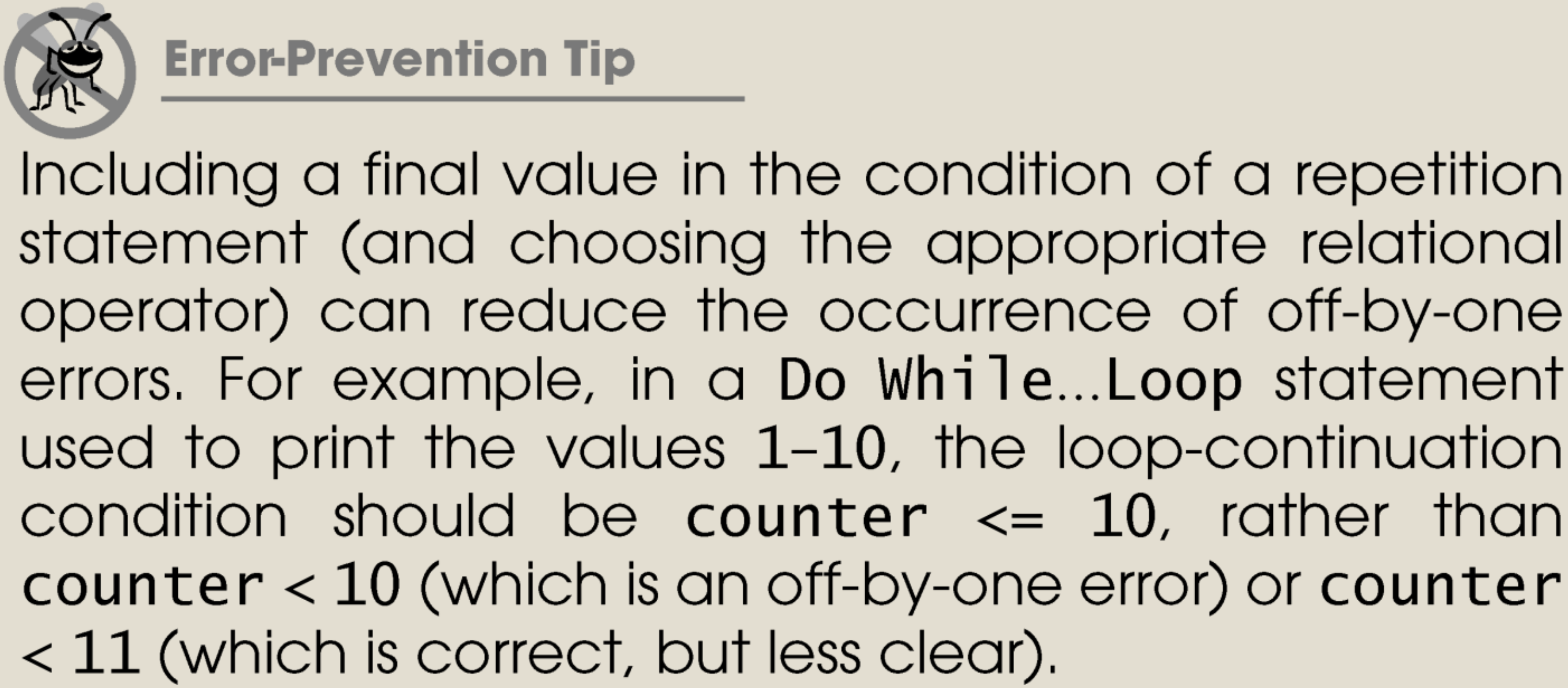

Corresponding Visual Basic Statements:
displayListBox.Items.Add(counter
)
action state
decision
[counter <= 3]
[counter > 3]
loop-continuation condition

Figure 10.6 Do...Loop While repetition statement UML activity diagram.

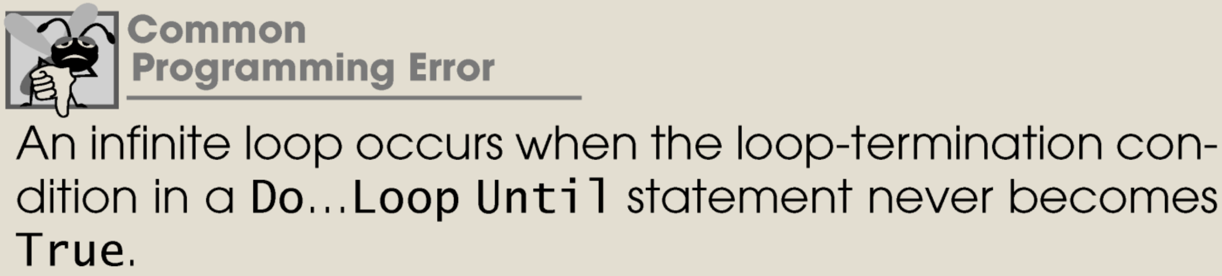


Error-Prevention Tip

Including a final value in the condition of a repetition statement (and choosing the appropriate relational operator) can reduce the occurrence of off-by-one errors. For example, in a **Do While…Loop** statement used to print the values 1–10, the loop-continuation condition should be **counter <= 10**, rather than **counter** < 10 (which is an off-by-one error) or **counter** < 11 (which is correct, but less clear).

10.3 `Do...Loop Until` Repetition Statement

- The **`Do...Loop Until`** statement is similar to the `Do...Until Loop` statement, except that in the `Do…Loop Until` statement the loop-termination condition is tested *after* the loop body executes, so the body executes *at least* once.
- Imagine that you place an item in the suitcase, then determine whether the suitcase is full. As long as the condition “the suitcase is full” is `False`, you continue to put items into the suitcase.



Common Programming Error

An infinite loop occurs when the loop-termination condition in a **`Do…Loop Until`** statement never becomes `True`.

10.3 Do...Loop Until Repetition Statement (Cont.)

- This app segment displays the numbers 1 through 3 in a ListBox:

```
Dim counter As Integer = 1

Do
   displayListBox.Items.Add(counter)
   counter += 1
Loop Until counter > 3
```

10.3 Do...Loop Until Repetition Statement (Cont.)

- This UML diagram (Fog. 10.7) indicates exactly the same guard conditions as detailed in Fig. 10.6.

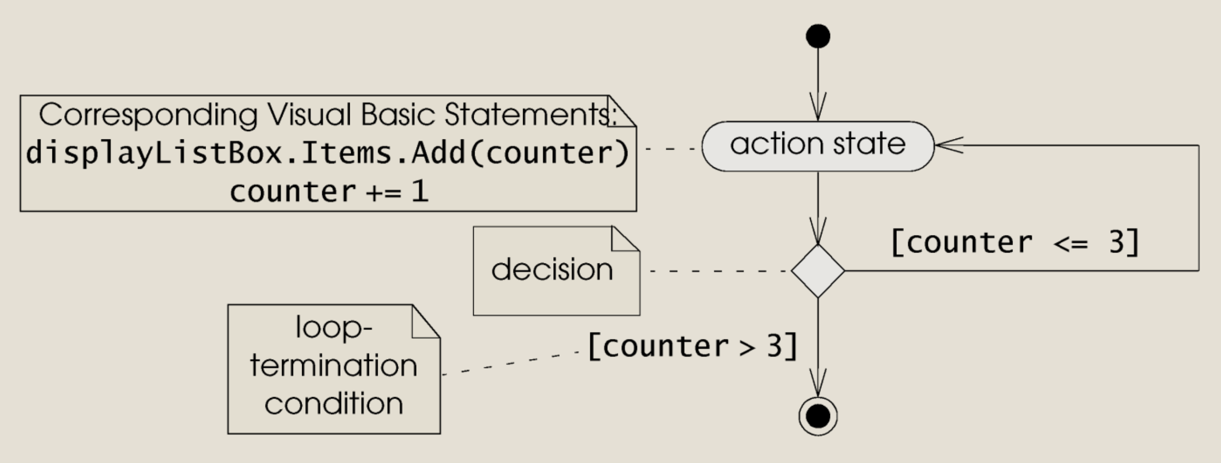

Corresponding Visual Basic Statements:
displayListBox.Items.Add(counter)
counter += 1
action state
[counter <= 3]
decision
loop-termination condition
[counter > 3]

Figure 10.7 Do...Loop Until repetition statement UML activity diagram.

10.4 Creating the Class Average App

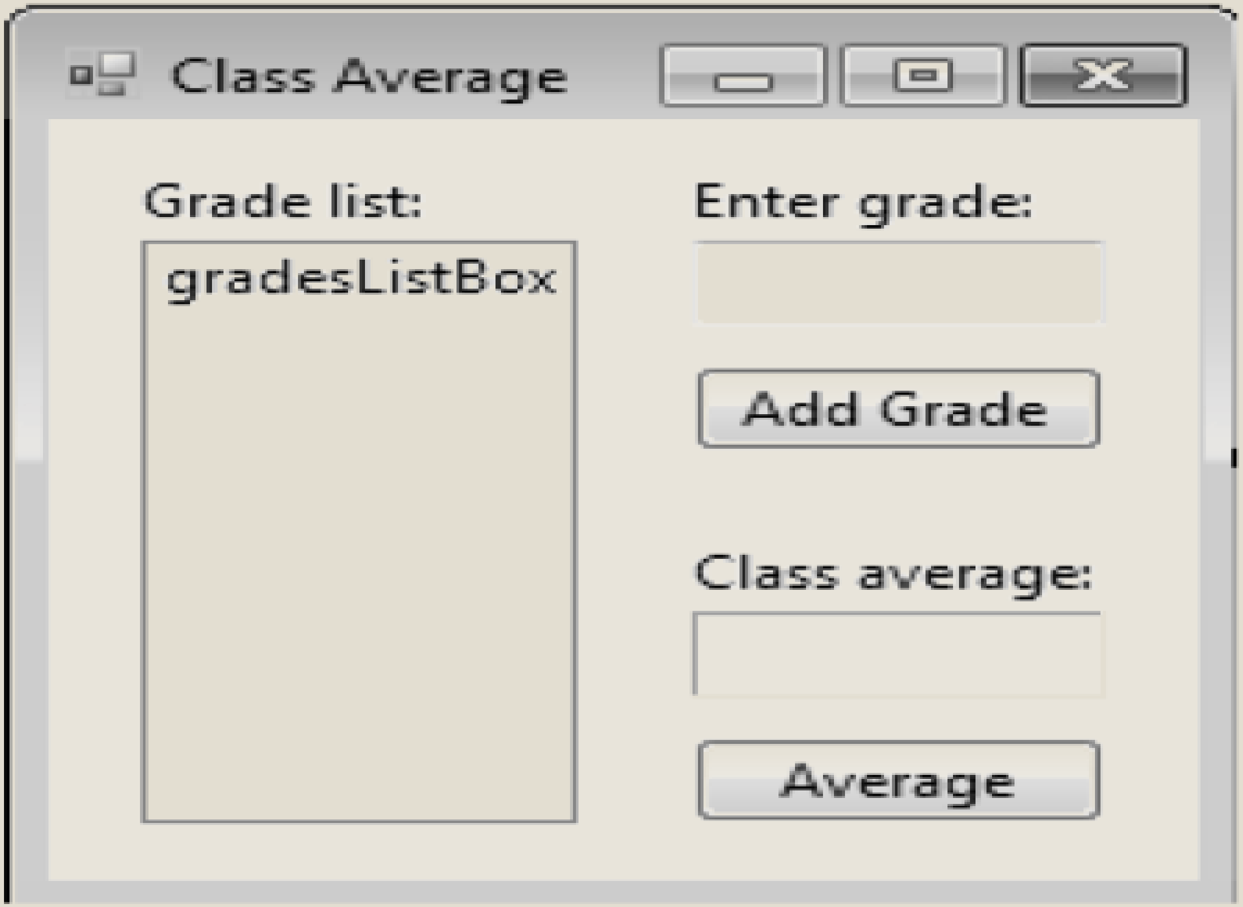
```
When the user clicks the Add Grade Button
    If an average has already been calculated for a set of grades
        Clear the output Label and the ListBox
    Retrieve grade entered by user in the Enter grade:TextBox
    Display the grade in the ListBox
    Clear the Enter grade: TextBox
    Transfer focus to the Enter grade: TextBox

    If the user has entered 10 grades
        Disable the Add Grade Button
        Transfer focus to the Average Button
```

10.4 Creating the Class Average App (Cont.)

```
When the user clicks the Average Button
    Set total to zero
    Set grade counter to zero
    Do
        Read the next grade in the ListBox
        Add the grade to the total
        Add one to the grade counter
    Loop While the grade counter is less than 10

    Calculate the class average by dividing the total by 10
    Display the class average
    Enable the Add Grade Button
    Transfer focus to the Enter grade: TextBox
```

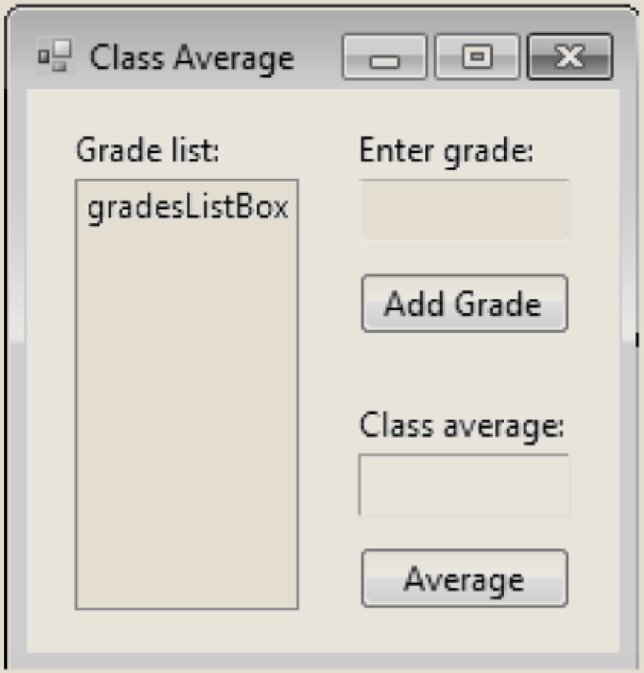
Entering Grades in the Class Average App

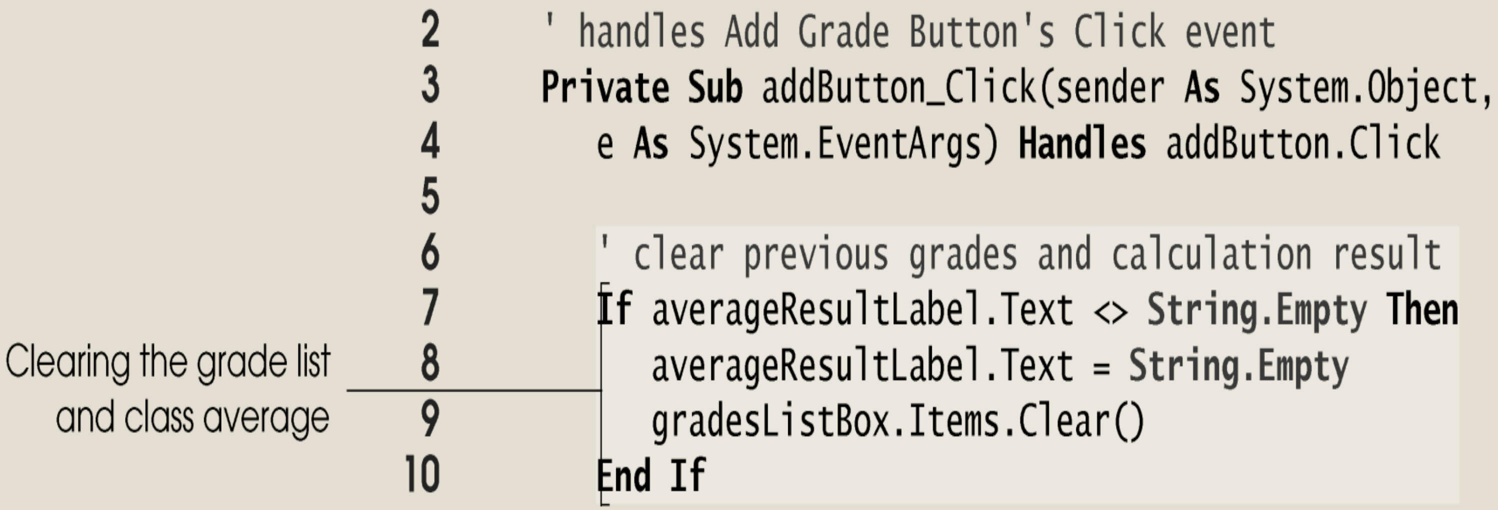
- Open the app (Fig. 10.9).



Entering Grades in the Class Average App (Cont.)

- Enter a grade in the Enter Grade box then Click the `Button` labeled **Add Grade** to create its event handler `addButton_Click` (Fig. 10.10).
- The program tests whether `averageResultLabel` displays any text by comparing the `Text` property's value to the empty string.

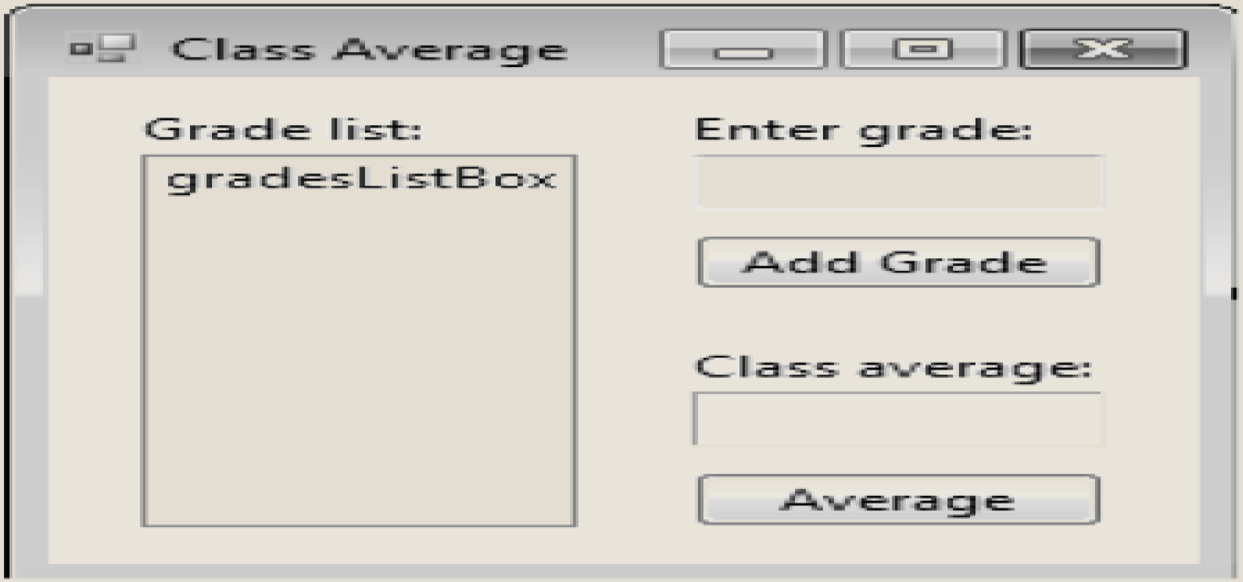




```
' handles Add Grade Button's Click event
Private Sub addButton_Click(sender As System.Object,
   e As System.EventArgs) Handles addButton.Click

   ' clear previous grades and calculation result
   If averageResultLabel.Text <> String.Empty Then
      averageResultLabel.Text = String.Empty
      gradesListBox.Items.Clear()
   End If
```

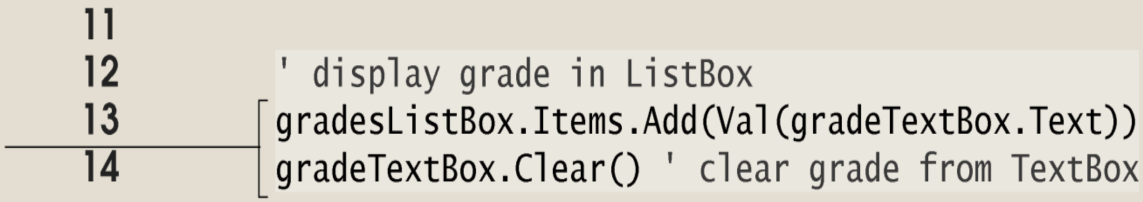
Clearing the grade list and class average

Figure 10.10 Clearing the output `Label` and `ListBox` after a calculation.



Entering Grades in the Class Average App (Cont.)

- Line 13 (Fig. 10.11) Adds the grade entered in gradeTextBox to gradesListBox's Items property. The grade is displayed in the ListBox.
- GradeTextBox.Clear deletes the grade from the TextBox so that the next grade can be entered.



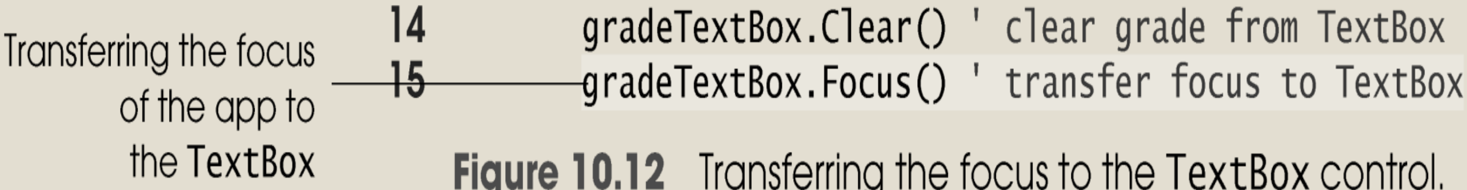
```
11
12         ' display grade in ListBox
13         gradesListBox.Items.Add(Val(gradeTextBox.Text))
14         gradeTextBox.Clear() ' clear grade from TextBox
```

Adding a numeric grade to the ListBox and clearing the user input from the TextBox

Figure 10.11 Adding the grade input to the ListBox and clearing the **Enter grade:** TextBox.

Transferring the Focus to a Control and Disabling a `Button`

- Calling `gradeTextBox`’s **`Focus`** method places the cursor in the `TextBox` for the next grade input (Fig. 10.12).
- This process is called **transferring the focus**.



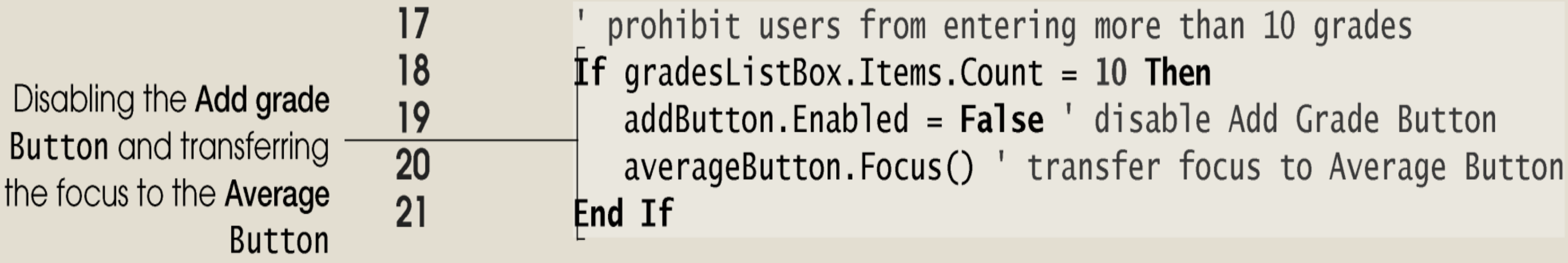
```
      gradeTextBox.Clear() ' clear grade from TextBox
      gradeTextBox.Focus() ' transfer focus to TextBox
```

Transferring the focus of the app to the TextBox

Figure 10.12 Transferring the focus to the `TextBox` control.

Transferring the Focus to a Control and Disabling a `Button` (Cont.)

- Your app should accept exactly 10 grades.
  - **`Items’s Count`** property returns the number of items displayed in the **Grade list:** `ListBox`.
  - If 10 grades have been entered, `addButton`’s **`Enabled`** property is set to `False` (Fig. 10.13).
  - After 10 grades have been entered, transfer the focus to the **Average** `Button`.



Disabling the **Add grade** `Button` and transferring the focus to the **Average** `Button`

```
' prohibit users from entering more than 10 grades
If gradesListBox.Items.Count = 10 Then
   addButton.Enabled = False ' disable Add Grade Button
   averageButton.Focus() ' transfer focus to Average Button
End If
```

Figure 10.13 App accepts only 10 grades.

GUI Design Tip

Disable `Buttons` when their function should not be available to users.

GUI Design Tip

Transfer the focus to the control that should be used next.

GUI Design Tip

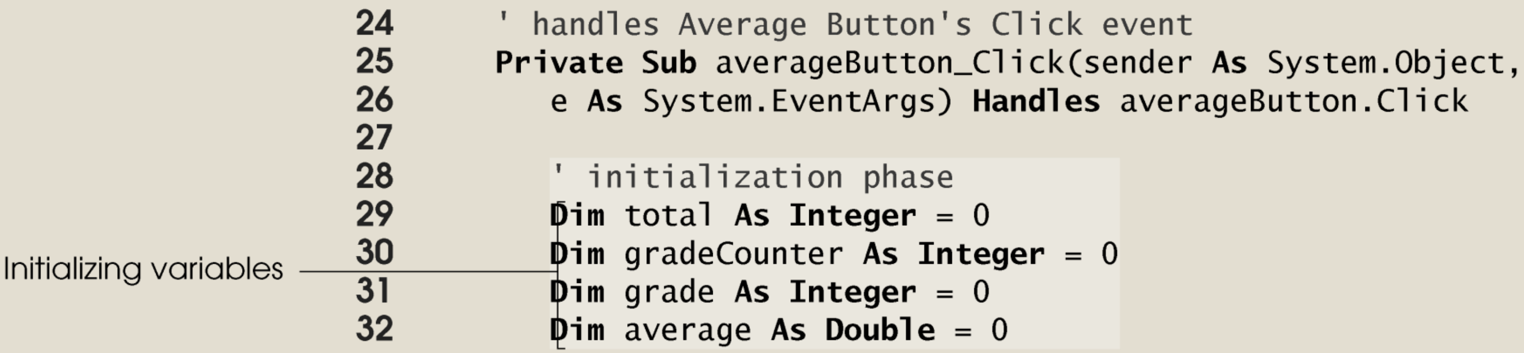
Enable a disabled `Button` when its function should be available to the user once again.

Common Programming Error

A control must be enabled in order to receive focus.

Calculating the Class Average

- Use the `Integer total` (Figure 10.14) to calculate the sum of the 10 grades.
- The result of the averaging calculation must be a floating-point value; therefore, you declare a `Double` variable to store the class average.



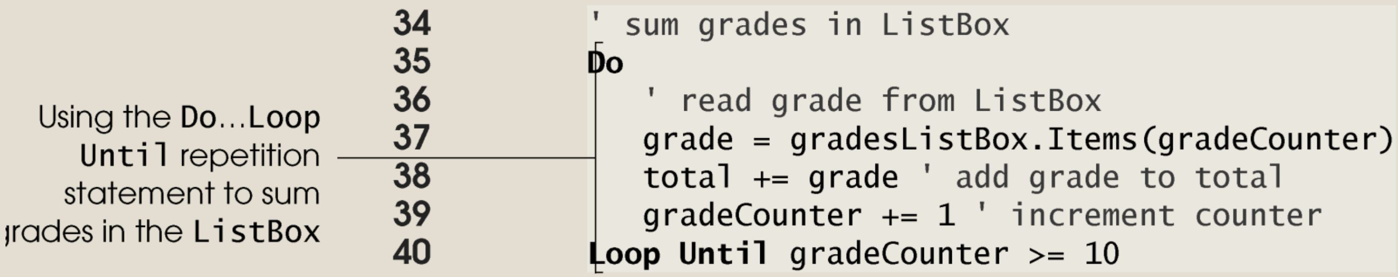
```
' handles Average Button's Click event
Private Sub averageButton_Click(sender As System.Object,
   e As System.EventArgs) Handles averageButton.Click

   ' initialization phase
   Dim total As Integer = 0
   Dim gradeCounter As Integer = 0
   Dim grade As Integer = 0
   Dim average As Double = 0
```

Initializing variables

Figure 10.14 Initialization phase of class average calculation.

Calculating the Class Average (Cont.)

- The Do...Loop Until statement in Figure 10.15 sums the grades that it reads from the ListBox.
- The statement should iterate until the value of gradeCounter is greater than or equal to 10.
- The items in a ListBox are accessed by their *position number*, starting from position number 0 (i.e., Items(0) is the first element).



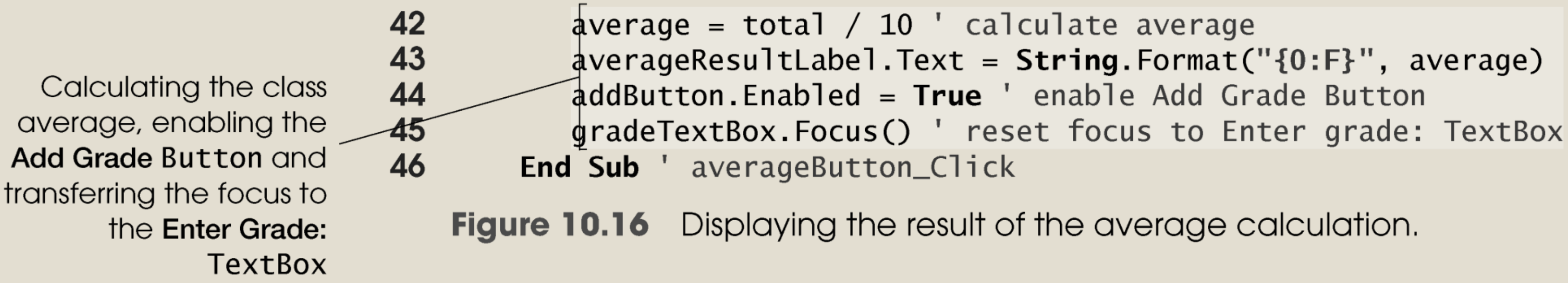
Using the Do...Loop Until repetition statement to sum grades in the ListBox

```
' sum grades in ListBox
Do
   ' read grade from ListBox
   grade = gradesListBox.Items(gradeCounter)
   total += grade ' add grade to total
   gradeCounter += 1 ' increment counter
Loop Until gradeCounter >= 10
```

Figure 10.15 Do...Loop Until summing grades.

Calculating the Class Average (Cont.)

- The F format specifier (Fig. 10.16) displays average in floating-point format.
- After the average is displayed, the app resets, and another list of grades can be entered.

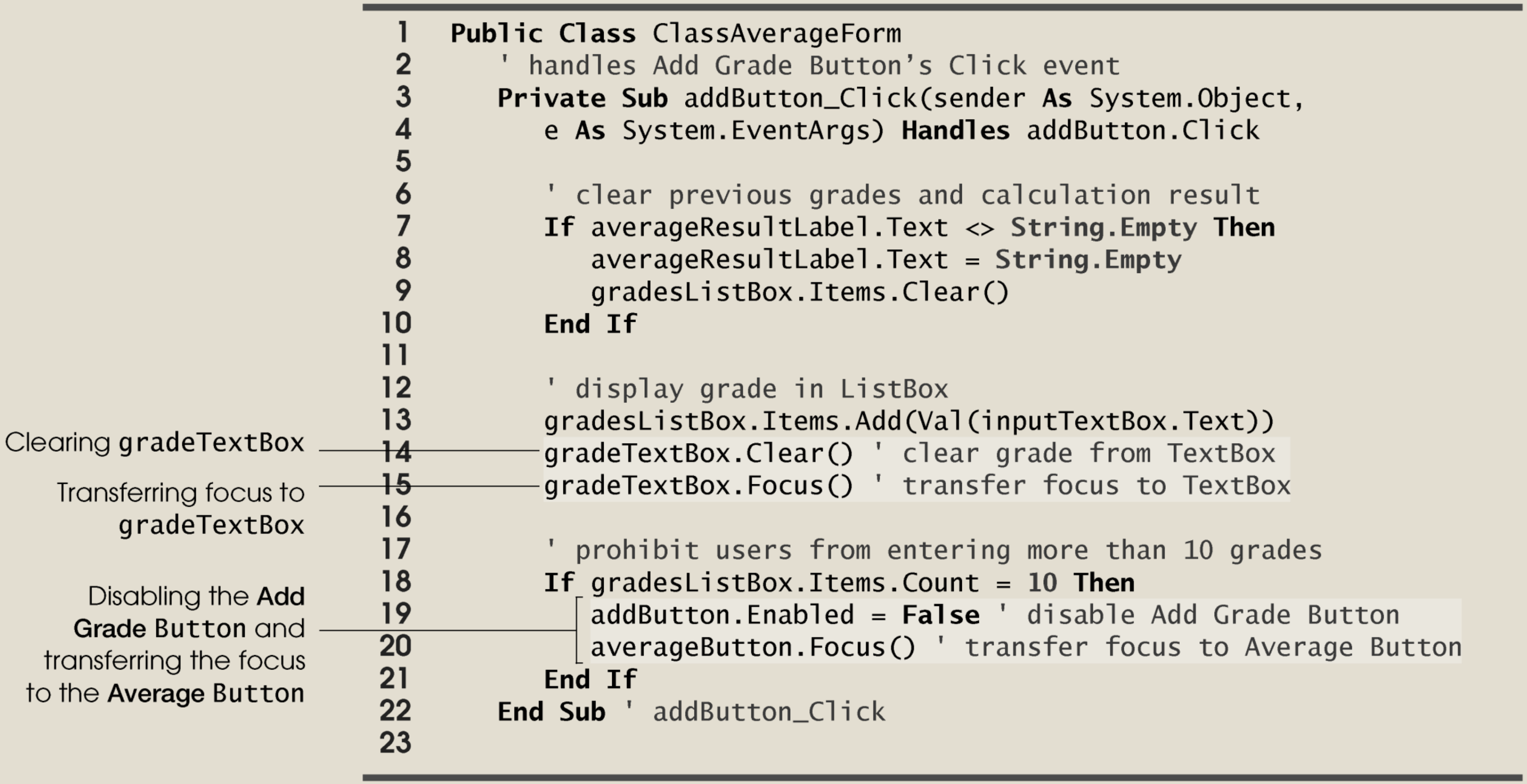


Calculating the class average, enabling the Add Grade Button and transferring the focus to the Enter Grade: TextBox

```
         average = total / 10 ' calculate average
         averageResultLabel.Text = String.Format("{0:F}", average)
         addButton.Enabled = True ' enable Add Grade Button
         gradeTextBox.Focus() ' reset focus to Enter grade: TextBox
      End Sub ' averageButton_Click
```

Figure 10.16 Displaying the result of the average calculation.

Figure 10.17 displays the source code for the app.



```
Public Class ClassAverageForm
   ' handles Add Grade Button's Click event
   Private Sub addButton_Click(sender As System.Object,
      e As System.EventArgs) Handles addButton.Click

      ' clear previous grades and calculation result
      If averageResultLabel.Text <> String.Empty Then
         averageResultLabel.Text = String.Empty
         gradesListBox.Items.Clear()
      End If

      ' display grade in ListBox
      gradesListBox.Items.Add(Val(inputTextBox.Text))
      gradeTextBox.Clear() ' clear grade from TextBox
      gradeTextBox.Focus() ' transfer focus to TextBox

      ' prohibit users from entering more than 10 grades
      If gradesListBox.Items.Count = 10 Then
         addButton.Enabled = False ' disable Add Grade Button
         averageButton.Focus() ' transfer focus to Average Button
      End If
   End Sub ' addButton_Click

```

Clearing **gradeTextBox**

Transferring focus to **gradeTextBox**

Disabling the **Add Grade Button** and transferring the focus to the **Average Button**

Figure 10.17 **Class Average** app code. (Part 1 of 2.)

```
   ' handles Average Button’s Click event
   Private Sub averageButton_Click(sender As System.Object,
      e As System.EventArgs) Handles averageButton.Click

      ' initialization phase
      Dim total As Integer = 0
      Dim gradeCounter As Integer = 0
      Dim grade As Integer = 0
      Dim average As Double = 0

      ' sum grades in ListBox
      Do
         ' read grade from ListBox
         grade = gradesListBox.Items(gradeCounter)
         total += grade ' add grade to total
         gradeCounter += 1 ' increment counter
      Loop Until gradeCounter >= 10

      average = total / 10 ' calculate average
      averageResultLabel.Text = String.Format(“{0:F}”, average)
      addButton.Enabled = True ' enable Add Grade Button
      gradeTextBox.Focus() ' reset focus to Enter grade: TextBox
   End Sub ' averageButton_Click
End Class ' ClassAverageForm
```

Accessing a grade in the ListBox via the Items property

Using a Do…Loop Until statement to total all the grades

Enabling the **Add Grade** Button and transferring the focus to the **Enter grade:** TextBox

Figure 10.17 **Class Average** app code. (Part 2 of 2.)