Tutorial 9 – **Class Average** Application
Introducing the `do…while` Repetition Statement

Objectives

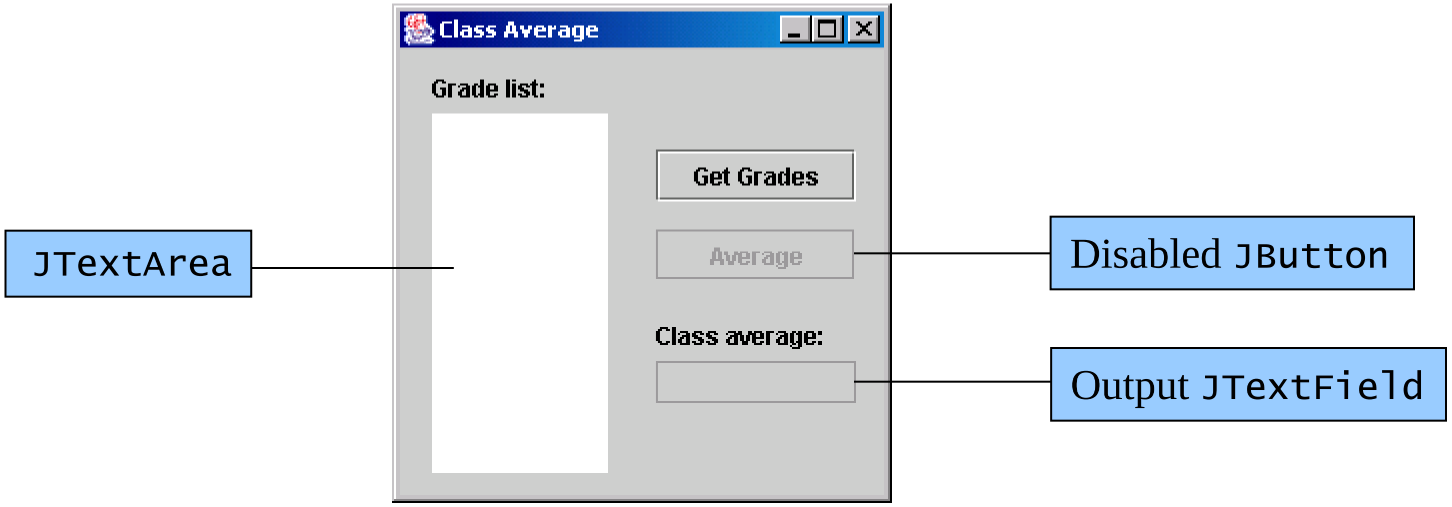
- In this tutorial, you will learn to:
  - Use the `do…while` statement.
  - Understand counter-controlled repetition.
  - Display an input dialog.
  - Enable and disable `JButton`s.

9.1 Test Driving the **Class Average** Application

Application Requirements
A teacher gives quizzes to a class of 10 students. The grades on these quizzes are integers in the range from 0 to 100, inclusive (0 and 100 are each valid grades). The teacher would like you to develop an application that computes the class average for a quiz. Your application should use an input dialog to enable the teacher to enter the grades.

9.1 Test Driving the **Class Average** Application (Cont.)

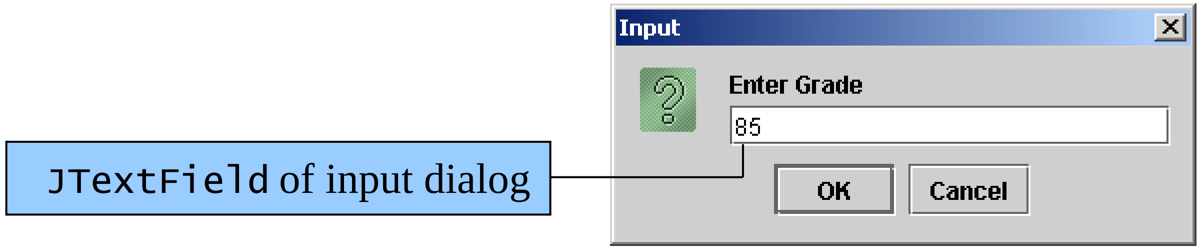
Figure 9.1 Running the completed **Class Average** application.



- Display input dialog
  - Click **Get Grades** `JButton`

9.1 Test Driving the **Class Average** Application (Cont.)

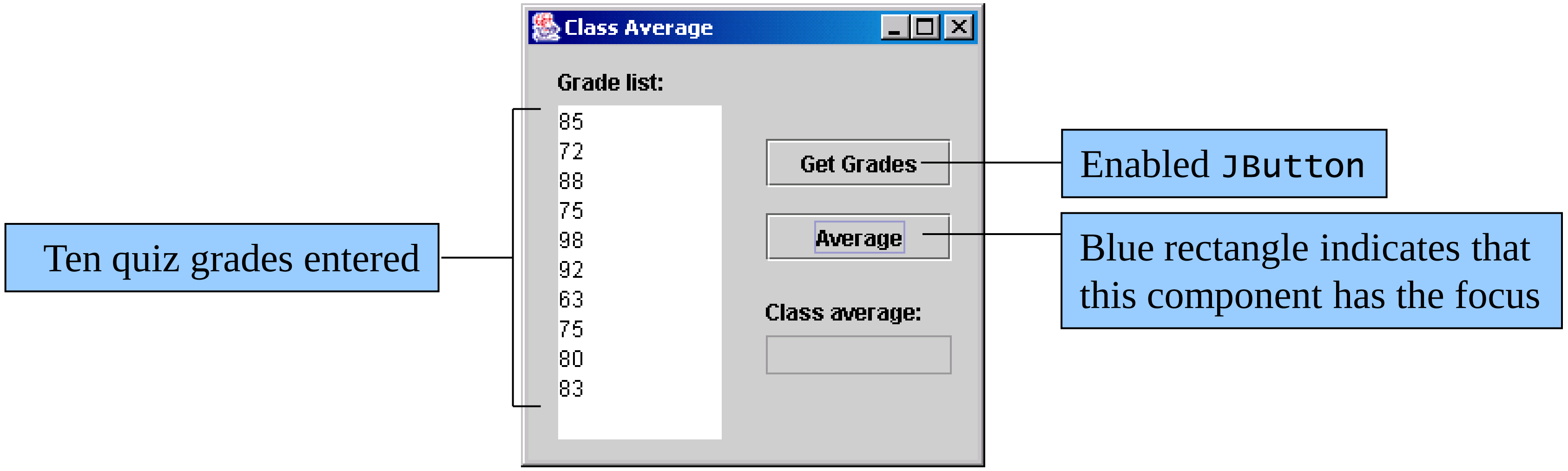
Figure 9.2 Entering grades in the **Class Average** application.



- Entering grades
  - Enter 85 as the first grade
  - Click the **OK** `JButton`

9.1 Test Driving the **Class Average** Application (Cont.)

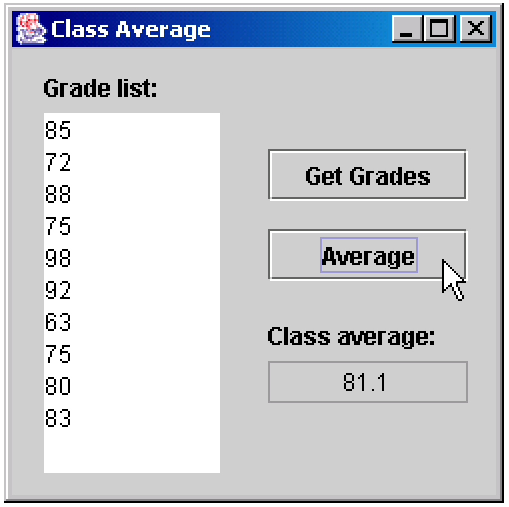
Figure 9.3 **Class Average** application after 10 grades have been entered.



- Entering more grades
  - Complete entering the students' grades
- Focus is set on **Average** `JButton`

9.1 Test Driving the **Class Average** Application (Cont.)

Figure 9.4 **Class Average** application after calculating the average.



Grade list:

```
85
72
88
75
98
92
63
75
80
83
```

Class average: 81.1

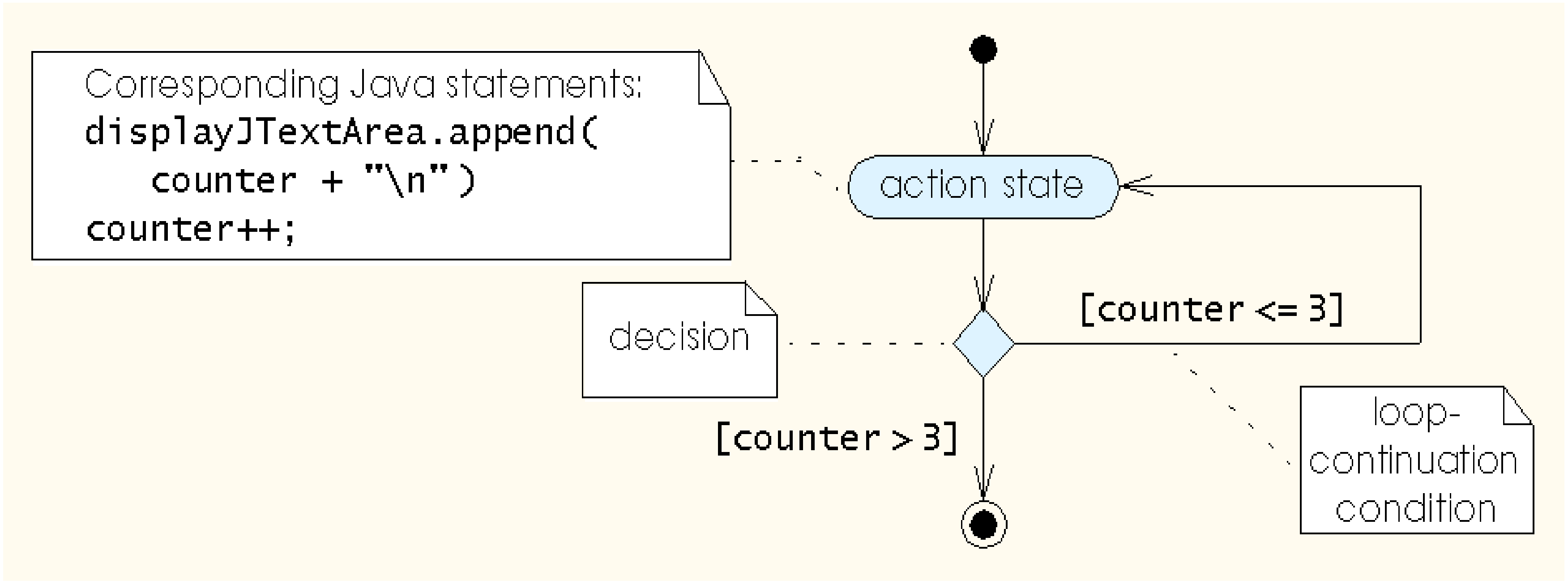
- Calculating average
  - Press the *Space Bar* to calculate average

9.2 do…while Repetition Statement

- `do…while` Repetition statement
  - Similar to the `while` statement
  - Iterates while loop-continuation condition is `true`
  - Loop-continuation condition checked after the body of the loop is performed
  - Body always executes at least once
- Off-by-one errors
  - Occur when the loop executes one too many or one less iteration than is necessary

9.2 do…while Repetition Statement (Cont.)

Figure 9.5 `do…while` repetition statement UML activity diagram.



9.3 Creating the **Class Average** Application

```
Retrieve grades from user when the user clicks the Get Grades JButton:
    Set total to zero
    Set grade counter to one

    Clear the JTextArea
    Clear the output JTextField

    Do
        Get the next grade from the input dialog
        Append the grade to the JTextArea
        Add the grade to the total
        Add one to the grade counter
    While the grade counter is less than or equal to 10

    Enable Average JButton
    Give focus to Average JButton

Calculate average when the user clicks the Average JButton:
    Calculate the class average by dividing the total by 10
    Display the class average in the output JTextField
    Disable Average JButton
    Give focus to Get Grades JButton
```

9.3 Creating the **Class Average** Application (Cont.)

Action	Component	Event
Label the application's components	`gradeListJLabel`, `classAverageJLabel`	
Retrieve grades from user	`getGradesJButton`	User clicks **Get Grades** `JButton`
Clear the JTextArea	`gradeListJTextArea`	
Clear the output JTextField	`classAverageJTextField`	
Get the next grade from the input dialog	`JOptionPane`	
Append the grade to the JTextArea	`gradeListJTextArea`	
Enable Average JButton	`averageJButton`	
Give focus to Average JButton	`averageJButton`	
Calculate average	`averageJButton`	User clicks **Average** `JButton`
Display the class average	`classAverageJTextField`	
Disable Average JButton	`averageJButton`	
Give focus to Get Grades JButton	`getGradesJButton`	

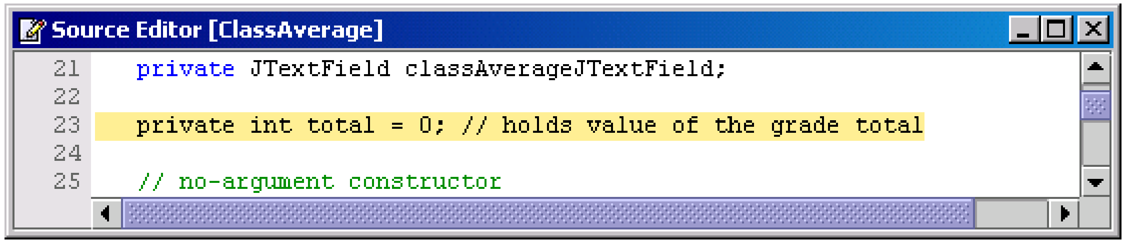
Figure 9.6 ACE table for the **Class Average** application.

9.3 Creating the **Class Average** Application (Cont.)

- Local variables
  - Can only be used in the body of the method in which they are declared
- Instance variables
  - Defined within a class, but outside any methods
  - Can be accessed from any method

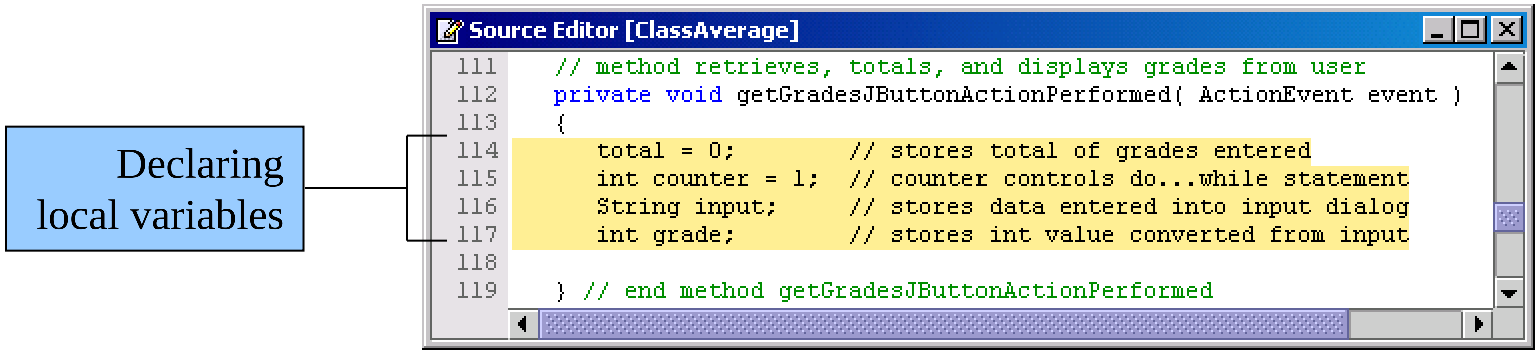
9.3 Creating the **Class Average** Application (Cont.)

Figure 9.7 Variable `total` declared outside a method.



```
   private JTextField classAverageJTextField;

   private int total = 0; // holds value of the grade total

   // no-argument constructor
```

Figure 9.8 Initializing your application's variables.



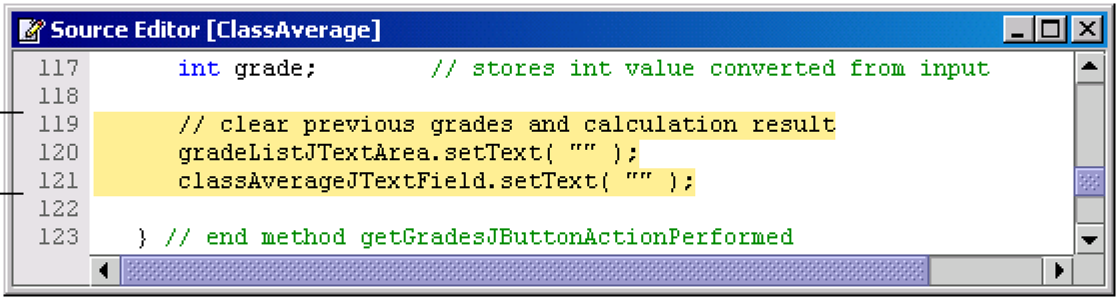
```
   // method retrieves, totals, and displays grades from user
   private void getGradesJButtonActionPerformed( ActionEvent event )
   {
      total = 0;           // stores total of grades entered
      int counter = 1;     // counter controls do...while statement
      String input;        // stores data entered into input dialog
      int grade;           // stores int value converted from input

   } // end method getGradesJButtonActionPerformed
```

9.3 Creating the **Class Average** Application (Cont.)

- Retrieving data from an input dialog
  - Call the `JOptionPane.showInputDialog` method to display an input dialog
  - Returns the input entered by the user as a `String` when the user clicks the **OK** `JButton`
  - Use `Integer.parseInt` to convert the `String` to an `int`

9.3 Creating the **Class Average** Application (Cont.)

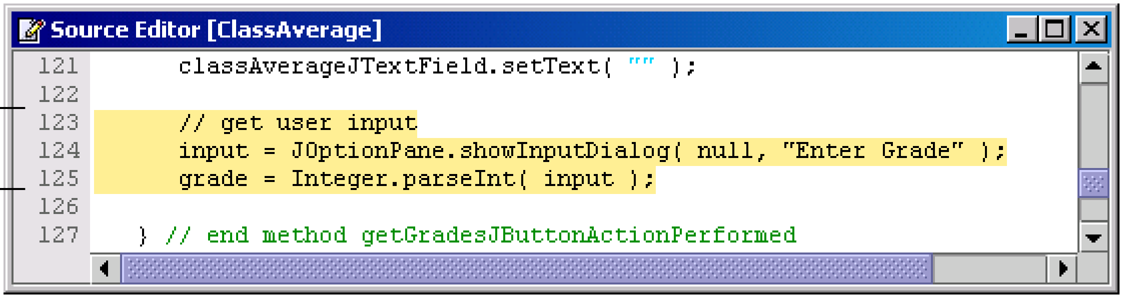
Figure 9.9 Clearing the output components.



Clearing the grade list and class average

```
      int grade;          // stores int value converted from input

      // clear previous grades and calculation result
      gradeListJTextArea.setText( "" );
      classAverageJTextField.setText( "" );

   } // end method getGradesJButtonActionPerformed
```

Figure 9.10 Getting the grade input using an input dialog.

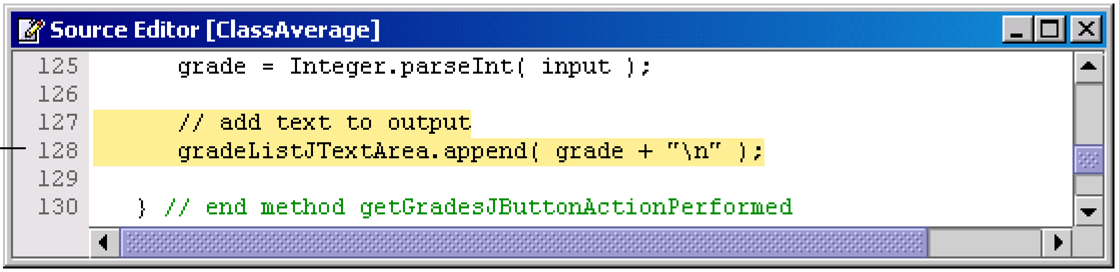


Retrieving data from an input dialog

```
      classAverageJTextField.setText( "" );

      // get user input
      input = JOptionPane.showInputDialog( null, "Enter Grade" );
      grade = Integer.parseInt( input );

   } // end method getGradesJButtonActionPerformed
```

9.3 Creating the **Class Average** Application (Cont.)

Figure 9.11 Adding the grade input to the `gradeListJTextArea`.

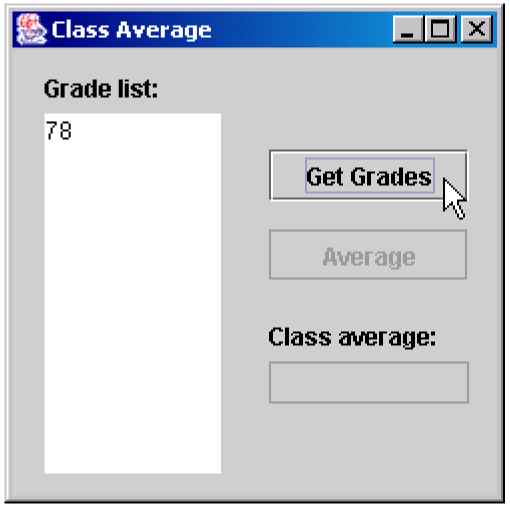


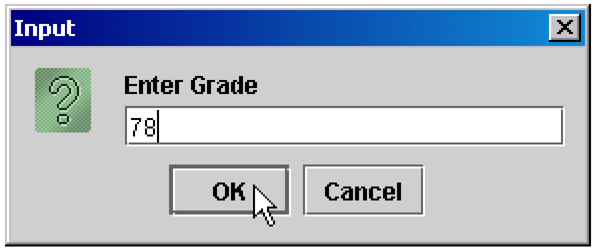
```
         grade = Integer.parseInt( input );

         // add text to output
         gradeListJTextArea.append( grade + "\n" );

      } // end method getGradesJButtonActionPerformed
```

Appending the input value to the `gradeListJTextArea`

9.3 Creating the **Class Average** Application (Cont.)

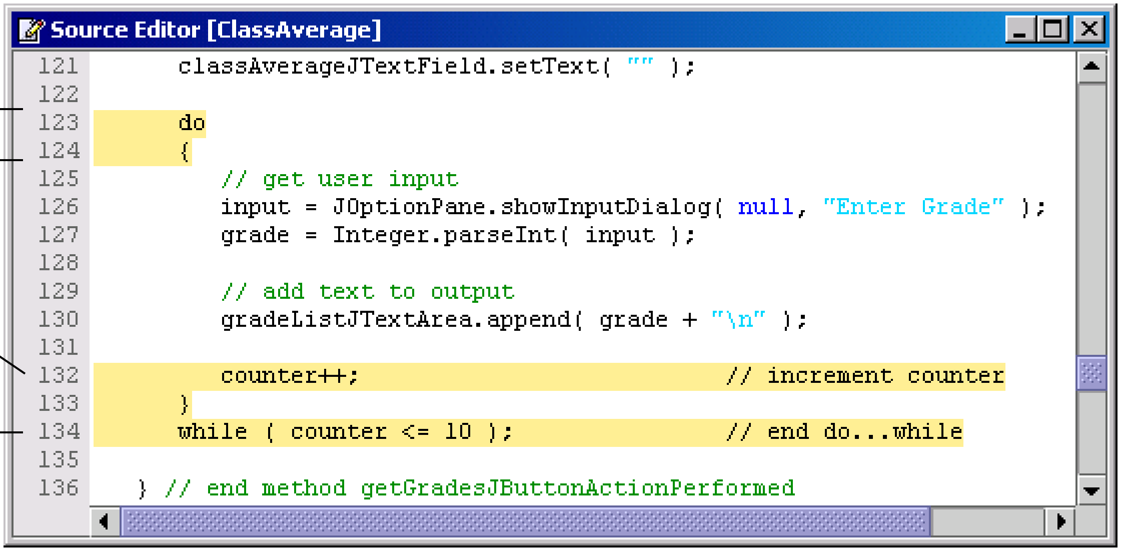
Figure 9.12 Running the updated application.





9.3 Creating the **Class Average** Application (Cont.)

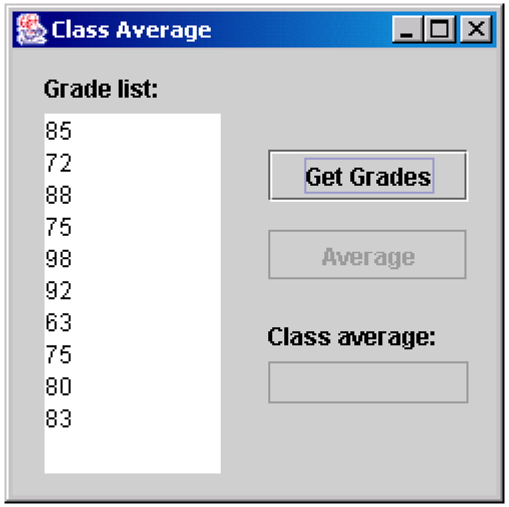
Figure 9.13 Defining the do…while loop.


Source Editor [ClassAverage]
Start of do…while statement
Incrementing counter for next iteration
Condition of do…while statement

```
      classAverageJTextField.setText( "" );

      do
      {
         // get user input
         input = JOptionPane.showInputDialog( null, "Enter Grade" );
         grade = Integer.parseInt( input );

         // add text to output
         gradeListJTextArea.append( grade + "\n" );

         counter++;                                    // increment counter
      }
      while ( counter <= 10 );                         // end do...while

   } // end method getGradesJButtonActionPerformed
```

9.3 Creating the **Class Average** Application (Cont.)

Figure 9.14 Running the updated application.

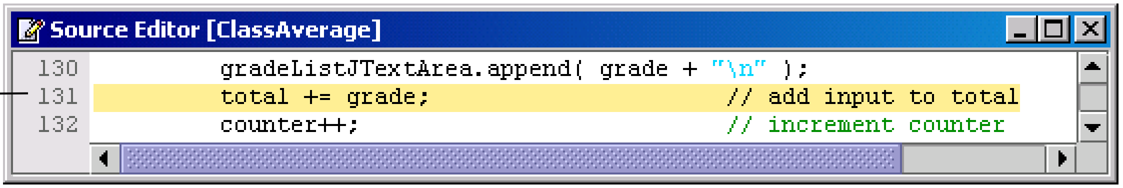


```
85
72
88
75
98
92
63
75
80
83
```

9.3 Creating the **Class Average** Application (Cont.)

Figure 9.15 Summing the grades.

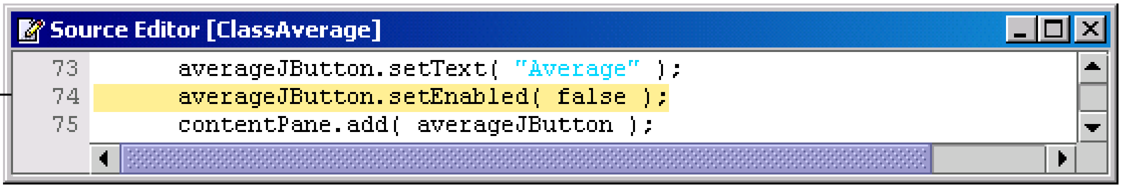
Add `input` to `total`



```
        gradeListJTextArea.append( grade + "\n" );
        total += grade;                          // add input to total
        counter++;                               // increment counter
```

Figure 9.16 Disabling a `JButton`.

Use method `setEnabled` to enable or disable a `JButton`



```
     averageJButton.setText( "Average" );
     averageJButton.setEnabled( false );
     contentPane.add( averageJButton );
```

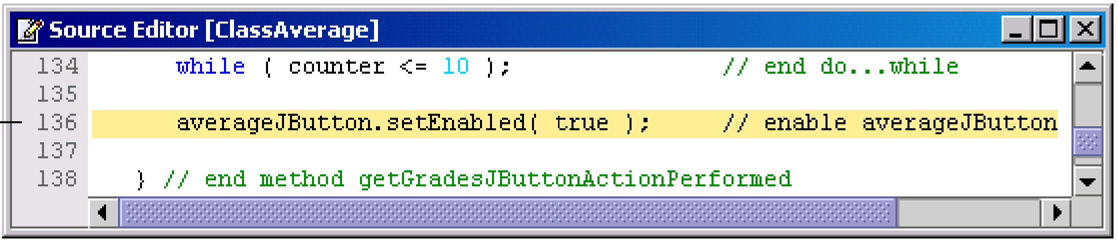
9.3 Creating the **Class Average** Application (Cont.)

- `setEnabled` method
  - Pass the argument `true` to allow the user to press the `JButton` (enable the `JButton`)
  - Pass the argument `false` to prevent the user from pressing the `JButton` (disable the `JButton`)

9.3 Creating the **Class Average** Application (Cont.)

Figure 9.17 Enabling a `JButton`.

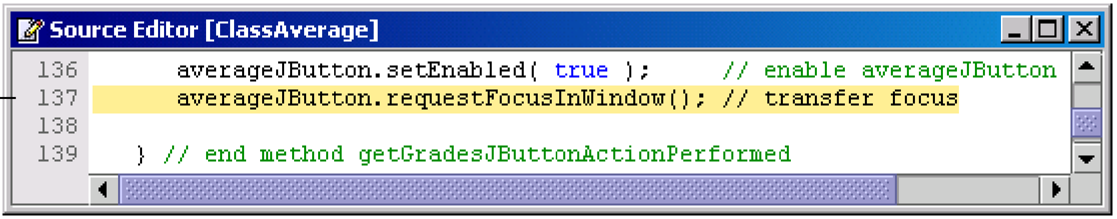
To enable a `JButton` or other GUI component, pass `true` to method `setEnabled`



```
      while ( counter <= 10 );                    // end do...while

      averageJButton.setEnabled( true );          // enable averageJButton

   } // end method getGradesJButtonActionPerformed
```

Figure 9.18 Transferring focus to a `JButton`.

Transfer focus to a GUI component by using method `requestFocusInWindow`

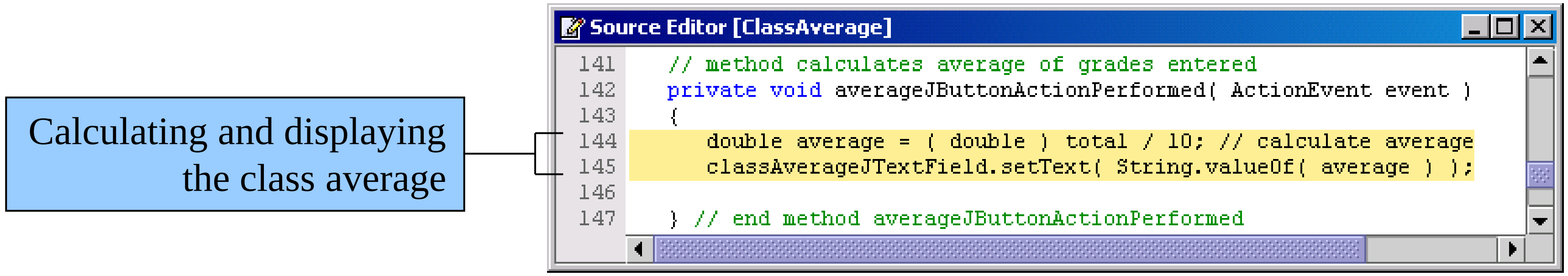


```
      averageJButton.setEnabled( true );          // enable averageJButton
      averageJButton.requestFocusInWindow(); // transfer focus

   } // end method getGradesJButtonActionPerformed
```

- Transferring the focus
  - Call the `JButton`'s `requestFocusInWindow` method

9.3 Creating the **Class Average** Application (Cont.)

Figure 9.19 Calculating and displaying the class average.



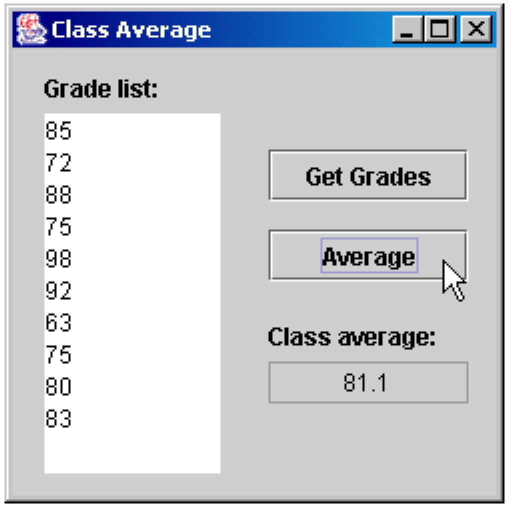
Calculating and displaying the class average
Source Editor [ClassAverage]

```
// method calculates average of grades entered
private void averageJButtonActionPerformed( ActionEvent event )
{
   double average = ( double ) total / 10; // calculate average
   classAverageJTextField.setText( String.valueOf( average ) );

} // end method averageJButtonActionPerformed
```

- Cast operator
  - Convert the operand (in this case `total`) to the type placed within the parentheses of the cast.

9.3 Creating the **Class Average** Application (Cont.)

Figure 9.20 Completed **Class Average** application.



```
85
72
88
75
98
92
63
75
80
83
```

81.1