```
        }
        else if (actionCommand.equals("Clear"))
            story.setText("");
        else
            story.setText("Unexpected error.");
    }

    private int getLineCount()
    {
        String storyString = story.getText();
        int count = 0;

        for (int i = 0; i < storyString.length(); i++)
            if (storyString.charAt(i) == '\n')
                count++;

        return count + 1;//The last line has no '\n'.
    }
}
```

39. We made the text field an instance variable because we needed to refer to it in the definition of the method `actionPerformed`. On the other hand, the only direct reference we had to the buttons was in the constructor. So, we need names for the buttons only in the constructor definition.

40. The GUI would try to add the string `"Enter numbers here."` as if it were a string for a number. This will cause a `NumberFormatException` to be thrown and the string `"Error: Reenter Number."` would be displayed in the text field.

41. Every time the user clicks the addition button, the following assignment is executed:

```
result = result + stringToDouble(ioField.getText());
```

So, the number in the text field is added to the total as many times as the user clicks the addition button. But, the value in the text field is the running total, so the running total is added to itself. Thus, the running total is added to the total as many times as the user clicks the addition button.

Let's say that the user starts the GUI, types in `10`, and clicks the addition button. That adds `10` to `result`, so the value of `result` is then `0` plus `10`, which is `10`, and `10` is displayed. Now the user clicks the addition button a second time. That adds `10` to `result` again, so the value of `result` is `10` plus `10`, which is `20`, and `20` is displayed. Next the user clicks the addition button a third time. This time, `20` is in the text field, and so it is added to `result`, which is also `20`. Thus, the value of `result` is now `40`, and `40` is displayed. Note that it is always the number in the text field that is added in.

42. The two calculator windows are completely independent. Each has its own instance variable `result`, which has no effect on the other's instance variable `result`.

43. If you click the close-window button in either calculator window, the entire program

`System.exit` in Display 17.19, but the following ensures that a `System.exit` that is in some library class will be invoked:

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Programming Projects

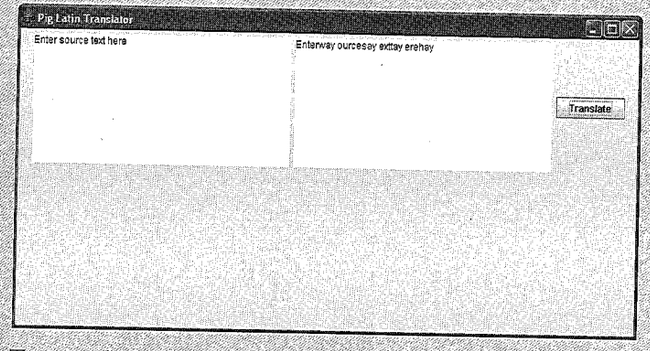
Many of these Programming Projects can be solved using AW's MyCodeMate. To access these please go to: www.mycodemate.com.

1. Design and code a Swing GUI to translate text that is input in English into Pig Latin. You can assume that the sentence contains no punctuation. The rules for Pig Latin are as follows:

   a. For words that begin with consonants, move the leading consonant to the end of the word and add "ay." Thus, "ball" becomes "allbay"; "button" becomes "uttonbay"; and so forth.

   b. For words that begin with vowels, add "way" to the end of the word. Thus, "all" becomes "allway"; "one" becomes "oneway"; and so forth.

   Use a `FlowLayout` with a `JTextArea` for the source text and a separate `JTextArea` for the translated text. Add a `JButton` with an event to perform the translation. A sample application is shown below with the text translated to Pig Latin.



   To parse the source text, note that you can use the `Scanner` class on a `String`. For example the following code:

```
Scanner scan = new Scanner("foo bar zot");
while (scan.hasNext())
{
    System.out.println(scan.next());
}
```

will output

foo

myCodeMate 2. Design and code a Swing GUI for a two-player tic-tac-toe (noughts and crosses) game on a 3 by 3 game board. The `JFrame` should use a `BorderLayout` with a `JLabel` in the NORTH region to display messages (e.g., who won the game), and a `JPanel` in the CENTER region to display the game board. For the game board in the `JPanel`, use a `GridLayout` manager with a 3 by 3 layout of `JButton`'s in each cell to display the game board. The button labels should initially be blank. When a player clicks on an empty button an appropriate "X" or "O" should be placed in the label field of the button. If there is a winner (three in a row) then the program should display the winner in the `JLabel` located at the top of the window. If all nine cells have been filled without a winner the program should indicate that there was a tie.

myCodeMate 3. Design and code a Swing GUI calculator. You can use Display 17.19 as a starting point, but your calculator will be more sophisticated. Your calculator will have two text fields that the user cannot change: One labeled `"Result"` will contain the result of performing the operation, and the other labeled `"Operand"` will be for the user to enter a number to be added, subtracted, and so forth from the result. The user enters the number for the `"Operand"` text field by clicking buttons labeled with the digits `0` through `9` and a decimal point, just as in a real calculator. Allow the operations of addition, subtraction, multiplication, and division. Use a `GridLayout` manager to produce a button pad that looks similar to the keyboard on a real calculator.

   When the user clicks a button for an operation: the operation is performed, the `"Result"` text field is updated, and the `"Operand"` text field is cleared. Include a button labeled `"Reset"` that resets the `"Result"` to `0.0`. Also include a button labeled `"Clear"` that resets the `"Operand"` text field so it is blank.

   Hint: Define an exception class named `DivisonByZeroException`. Have your code throw and catch a `DivisonByZeroException` if the user attempts to "divide by zero." Your code will catch the `DivisonByZeroException` and output a suitable message to the `"Operand"` text field. The user may then enter a new substitute number in the `"Operand"` text field. Since values of type `double` are, in effect, approximate values, it makes no sense to test for equality with `0.0`. Consider an operand to be "equal to zero" if it is in the range `-1.0e-10` to `+1.0e-10`.

4. (The Swing part of this project is pretty easy, but to do this programming project you need to know how to convert numbers from one base to another.) Write a program that converts integers from base ten (ordinary decimal) notation to base two notation. Use Swing to perform input and output via a window interface. The user enters a base ten numeral in one text field and clicks a button with `"Convert"` written on it; the equivalent base two numeral then appears in another text field. Be sure to label the two text fields. Include a `"Clear"` button that clears both text fields when clicked. (*Hint:* Include a private method that converts the string for a base ten numeral to the string for the equivalent base two numeral.)

18 Swing II