WORKED EXAMPLE 6.3   **A Sample Debugging Session**

This Worked Example presents a realistic example for running a debugger by examining a `Word` class whose primary purpose is to count the number of syllables in a word.

Problem Statement   The `Word` class uses this rule for counting syllables:

> Each group of adjacent vowels (a, e, i, o, u, y) counts as one syllable (for example, the "ea" in "peach" contributes one syllable, but the "e . . . o" in "yellow" counts as two syllables). However, an "e" at the end of a word doesn't count as a syllable. Each word has at least one syllable, even if the previous rules give a count of 0.

Also, when you construct a word from a string, any characters at the beginning or end of the string that aren't letters are stripped off. That is useful when you read the input using the `next` method of the `Scanner` class. Input strings can still contain quotation marks and punctuation marks, and we don't want them as part of the word.

Your task is to find and correct the errors in this program.

Here is the source code. There are a couple of bugs in this class.

worked_example_3/Word.java

```java
1  /**
2      This class describes words in a document. There are a couple
3      of bugs in this class.
4  */
5  public class Word
6  {
7     private String text;
8
9     /**
10        Constructs a word by removing leading and trailing non-
11        letter characters, such as punctuation marks.
12        @param s the input string
13     */
14     public Word(String s)
15     {
16        int i = 0;
17        while (i < s.length() && !Character.isLetter(s.charAt(i)))
18        {
19           i++;
20        }
21        int j = s.length() - 1;
22        while (j > i && !Character.isLetter(s.charAt(j)))
23        {
24           j--;
25        }
26        text = s.substring(i, j);
27     }
28
29     /**
30        Returns the text of the word, after removal of the
31        leading and trailing non-letter characters.
32        @return the text of the word
33     */
34     public String getText()
35     {
36        return text;
37     }
38
```

```
39    /**
40        Counts the syllables in the word.
41        @return the syllable count
42    */
43    public int countSyllables()
44    {
45        int count = 0;
46        int end = text.length() - 1;
47        if (end < 0) { return 0; } // The empty string has no syllables
48
49        // An e at the end of the word doesn't count as a vowel
50        char ch = text.charAt(end);
51        if (ch == 'e' || ch == 'E') { end--; }
52
53        boolean insideVowelGroup = false;
54        for (int i = 0; i <= end; i++)
55        {
56            ch = text.charAt(i);
57            String vowels = "aeiouyAEIOUY";
58            if (vowels.indexOf(ch) >= 0)
59            {
60                // ch is a vowel
61                if (!insideVowelGroup)
62                {
63                    // Start of new vowel group
64                    count++;
65                    insideVowelGroup = true;
66                }
67            }
68        }
69
70        // Every word has at least one syllable
71        if (count == 0) { count = 1; }
72
73        return count;
74    }
75 }
```

Here is a simple test class. Type in a sentence, and the syllable counts of all words are displayed.

worked_example_3/SyllableCounter.java

```
1   import java.util.Scanner;
2
3   /**
4       This program counts the syllables of all words in a sentence.
5   */
6   public class SyllableCounter
7   {
8       public static void main(String[] args)
9       {
10          Scanner in = new Scanner(System.in);
11
12          System.out.println("Enter a sentence ending in a period.");
13
14          String input;
15          do
16          {
17              input = in.next();
18              Word w = new Word(input);
```

```
19              int syllables = w.countSyllables();
20              System.out.println("Syllables in " + input + ": " + syllables);
21          }
22          while (!input.endsWith("."));
23      }
24  }
```

Supply this input:

```
Hello yellow peach.
```

Then the output is

```
Syllables in Hello: 1
Syllables in yellow: 1
Syllables in peach.: 1
```

That is not very promising.

First, set a breakpoint in the first line of the countSyllables method of the Word class, in line 43 of Word.java. Then start the program. The program will prompt you for the input. The program will stop at the breakpoint you just set.

First, the countSyllables method checks the last character of the word to see if it is a letter 'e'. Let's just verify that this works correctly. Run the program to line 51 (see Figure 12).
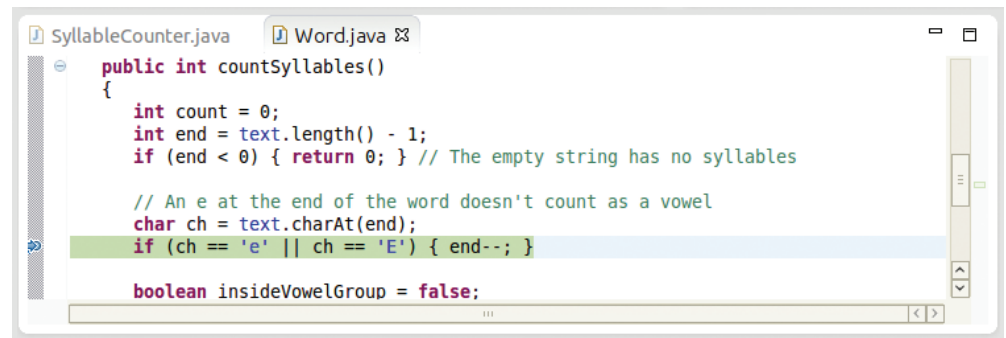

Figure 12   Debugging the countSyllables Method

Now inspect the variable ch. This particular debugger has a handy display of all current local and instance variables—see Figure 13. If yours doesn't, you may need to inspect ch
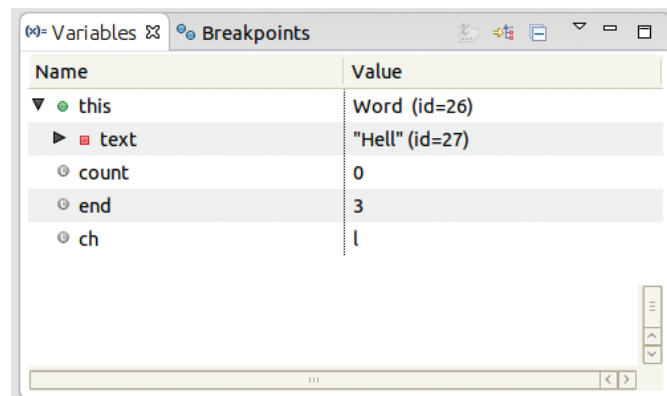

Figure 13
The Current Values of the
Local and Instance Variables

Big Java, 6e, Cay Horstmann, Copyright © 2015 John Wiley and Sons, Inc. All rights reserved.

manually. You can see that ch contains the value 'l'. That is strange. Look at the source code. The end variable was set to text.length() - 1, the last position in the text string, and ch is the character at that position.

Looking further, you will find that end is set to 3, not 4, as you would expect. And text contains the string "Hell", not "Hello". Thus, it is no wonder that countSyllables returns the answer 1. We'll need to look elsewhere for the culprit. Apparently, the Word constructor contains an error.

Unfortunately, a debugger cannot go back in time. Thus, you must stop the debugger, set a breakpoint in the Word constructor, and restart the debugger. Supply the input once again. The debugger will stop at the beginning of the Word constructor. The constructor sets two variables i and j, skipping past any nonletters at the beginning and the end of the input string. Set a breakpoint past the end of the second loop (see Figure 14) so that you can inspect the values of i and j.
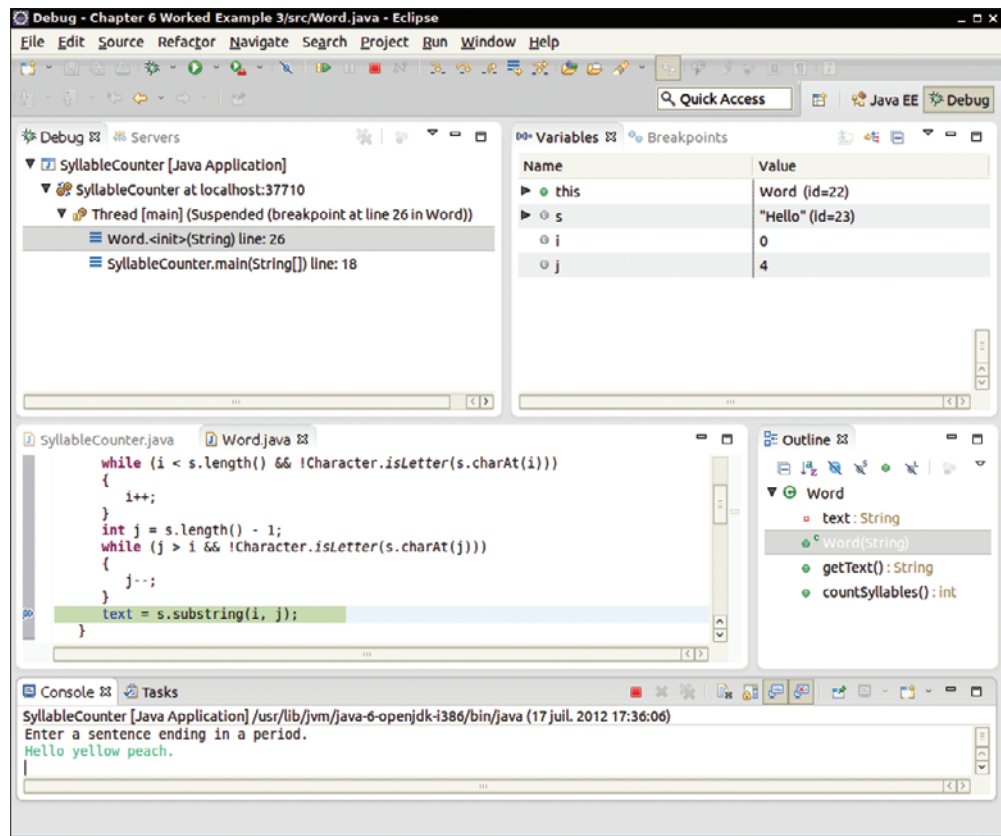

Figure 14 Debugging the Word Constructor

At this point, inspecting i and j shows that i is 0 and j is 4. That makes sense—there were no punctuation marks to skip. So why is text being set to "Hell"? Recall that the substring method counts positions up to, but not including, the second parameter. Thus, the correct call should be

```
text = s.substring(i, j + 1);
```

This is a very typical off-by-one error.

Fix this error, recompile the program, and try the three test cases again. You will now get the output

```
Syllables in Hello: 1
Syllables in yellow: 1
Syllables in peach.: 1
```

As you can see, there still is a problem. Erase all breakpoints and set a breakpoint in the count-Syllables method. Start the debugger and supply the input "Hello.". When the debugger stops at the breakpoint, start single stepping through the lines of the method. Here is the code of the loop that counts the syllables:

```
boolean insideVowelGroup = false;
for (int i = 0; i <= end; i++)
{
    ch = text.charAt(i);
    String vowels = "aeiouyAEIOUY";
    if (vowels.indexOf(ch) >= 0)
    {
        // ch is a vowel
        if (!insideVowelGroup)
        {
            // Start of new vowel group
            count++;
            insideVowelGroup = true;
        }
    }
}
```

In the first iteration through the loop, the debugger skips the if statement. That makes sense, because the first letter, 'H', isn't a vowel. In the second iteration, the debugger enters the if statement, as it should, because the second letter, 'e', is a vowel. The insideVowelGroup variable is set to true, and the vowel counter is incremented. In the third iteration, the if statement is again skipped, because the letter 'l' is not a vowel. But in the fifth iteration, something weird happens. The letter 'o' is a vowel, and the if statement is entered. But the second if statement is skipped, and count is not incremented again.

Why? The insideVowelGroup variable is still true, even though the first vowel group was finished when the consonant 'l' was encountered. Reading a consonant should set inside-VowelGroup back to false. This is a more subtle logic error, but not an uncommon one when designing a loop that keeps track of the processing state. To fix it, stop the debugger and add the following clause:

```
if (vowels.indexOf(ch) >= 0)
{
    . . .
}
else insideVowelGroup = false;
```

Now recompile and run the test once again. The output is:

```
Syllables in Hello: 2
Syllables in yellow: 2
Syllables in peach.: 1
```

Is the program now free from bugs? That is not a question the debugger can answer. Remember: Testing can show only the presence of bugs, not their absence.