

```
public static int compute(String word) {  
    char[] letters = {'a', 'e', 'i', 'o', 'u'};  
    int result = 0;  
  
    for (int i = 0; i < word.length(); i++) {  
        for (int j = 0; j < letters.length; j++) {  
            if (word.charAt(i) == letters[j]) {  
                result++;  
            }  
        }  
    }  
  
    return result;  
}
```

The diagram illustrates a control flow graph overlaid on the Java code. Nodes, represented by colored circles, are placed at various points in the code: at the start of the method, before the 'letters' array declaration, before the 'result' declaration, before the first 'for' loop, at the start of the inner 'for' loop, before the 'if' statement, after the 'result++' statement, after the inner loop, after the outer loop, before the 'return' statement, and at the end of the method. Edges, shown as colored lines, represent the flow of execution: a red line connects the start node to the 'letters' declaration; a purple line connects the 'result' declaration to the first 'for' loop; yellow lines connect the first 'for' loop to the inner 'for' loop and to the 'return' statement; orange lines connect the inner 'for' loop to the 'if' statement and to the 'return' statement; and a red line connects the 'if' statement to the 'result++' statement.