Verification & Validation Word Counter

Teacher: Sira Vegas

Student: Ricardo Carvalho

A program in Java to count words and their frequency of appearance in a .txt file

Source Code

```java
package org.example;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;
import java.util.Scanner;

6 usages  ricardoomrques *
public class Counter {
    ricardoomrques *
    public static void main(String[] args) throws IOException {
        count(args[0]);
    }
```

```java
public static String count(String filename) throws IOException {
    Map<String,Integer> counterWords = new HashMap<>();
    Integer countWord = 0;
    try {
        File myObj = new File(filename);
        FileReader fr=new FileReader(myObj);
        BufferedReader br=new BufferedReader(fr);  //creates a buffering character input stream
        StringBuffer sb=new StringBuffer();    //constructs a string buffer with no characters
        String line;
        while ((line = br.readLine()) != null) { // count words
            line = line.replaceAll( regex: "\\s",  replacement: " ");
            if (line.isEmpty()) continue;
            else countWord += line.split( regex: " ").length;
            for (int i = 0; i < line.split( regex: " ").length; i++) {
                String temp = line.split( regex: " ")[i].replaceAll( regex: "\\W",  replacement: "");
                temp = temp.toLowerCase();
                if (counterWords.containsKey(temp)) counterWords.put(temp,counterWords.get(temp) + 1);
                else counterWords.put(temp,1);
            }
        }
        fr.close();
        for (Map.Entry<String,Integer> item : counterWords.entrySet())
            System.out.println("Word = " + item.getKey() +
                    " Appears = " + item.getValue() + "\n");
    } catch (FileNotFoundException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
    System.out.println("Words in file: " + countWord + "\n\n");
    return "Words in file: " + countWord + "\n\n";
}
```

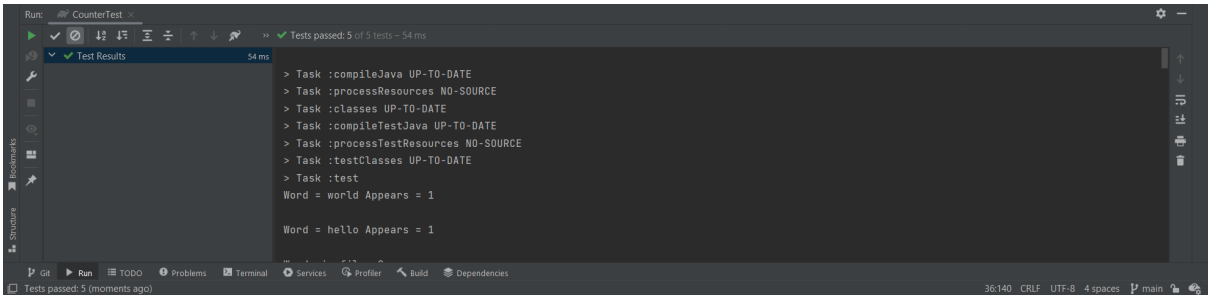
```java
public class CounterTest {
    @Test
    void testFileWithNoWords() throws IOException {
        var counter = new Counter();
        assertEquals(counter.count( filename: "C:\\Users\\ricar\\OneDrive\\Documentos\\Word-Counter\\src\\main\\Files\\file1.txt"), actual: "Words in file: 0\n\n");
    }

    @Test
    void testFileHelloWorld() throws IOException {
        var counter = new Counter();
        assertEquals(counter.count( filename: "C:\\Users\\ricar\\OneDrive\\Documentos\\Word-Counter\\src\\main\\Files\\file2.txt"), actual: "Words in file: 2\n\n");
    }

    @Test
    void testFile3() throws IOException {
        var counter = new Counter();
        assertEquals(counter.count( filename: "C:\\Users\\ricar\\OneDrive\\Documentos\\Word-Counter\\src\\main\\Files\\file3.txt"), actual: "Words in file: 53\n\n");
    }

    @Test
    void testFile4() throws IOException {
        var counter = new Counter();
        assertEquals(counter.count( filename: "C:\\Users\\ricar\\OneDrive\\Documentos\\Word-Counter\\src\\main\\Files\\file4.txt"), actual: "Words in file: 104\n\n");
    }

    @Test
    void testFile5() throws IOException {
        var counter = new Counter();
        assertEquals(counter.count( filename: "C:\\Users\\ricar\\OneDrive\\Documentos\\Word-Counter\\src\\main\\Files\\file5.txt"), actual: "Words in file: 273\n\n");
    }
```



Basically, what our program does is reading the absolute path of the file, and then reads the file line by line. If the line is empty, the program moves forward. If it doesn’t, we remove every special character from the line, and then we split it by whitespaces. This gives us an array of words. So, the number of words in that line is the length of the array, and we increment it as we read the lines. Then, we add the words to a map, so we can keep track of how many times that word appeared. The tests were made with JUnit, and we are just comparing the number of words that the program returns with the real number of words that the file has.