cse15l-lab-reports

lab report 5

part 1

The student post:
I am testing my ListExamples.java merge method on ListExamplesTests.java. However, one failure message appeared on the terminal regarding a Timeout Exception. I guess this might be caused by some of the local variable haven't be updated. Here's a screenshot of the output and corresponding code.

test.sh M ListExamples.java M

ListExamples.java > ListExamples > merge(List<String>, List<String>)

```java
class ListExamples {

  // Takes two sorted list of strings (so "a" appears before "b" and so on),
  // and return a new list that has all the strings in both list in sorted order.
  static List<String> merge(List<String> list1, List<String> list2) {
    List<String> result = new ArrayList<>();
    int index1 = 0, index2 = 0;
    while(index1 < list1.size() && index2 < list2.size()) {
      if(list1.get(index1).compareTo(list2.get(index2)) < 0) {
        result.add(list1.get(index1));
        index1 += 1;
      }
      else {
        result.add(list2.get(index2));
        index2 += 1;
      }
    }
    while(index1 < list1.size()) {
      result.add(list1.get(index1));
      index1 += 1;
    }
    while(index2 < list2.size()) {
      result.add(list2.get(index2));
      // change index1 below to index2 to fix test
      index1 += 1;
    }
    return result;
  }


}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash

```
Jason Zhang@LAPTOP-SMNVKH98 MINGW64 ~/Documents/GitHub/lab7 (main)
$ bash test.sh
JUnit version 4.13.2
..E
Time: 0.532
There was 1 failure:
1) testMerge2(ListExamplesTests)
org.junit.runners.model.TestTimedOutException: test timed out after 500 milliseconds
        at app//ListExamples.merge(ListExamples.java:44)
        at app//ListExamplesTests.testMerge2(ListExamplesTests.java:19)

FAILURES!!!
Tests run: 2,  Failures: 1
```

The TA response:

Jason Zhang: try to replace the `index1` into `index2` in the last `while - loop` .

After changing the code and the description of the bug:
The third while loop in the original code tries to add the remaining elements from `list2` to the result arraylist. However, this while loop checks to see if `index2` is smaller than the arraylist's size. The original code includes a line `index+=1` ;, which increments `index1` each time the condition is met. As a result, the while loop runs indefinitely because `index2` never changes. Instead of incrementing `index1` , `index2` should be incremented by `index2+=1` ; this ensures that the code behaves correctly.

ListExamples.java > ListExamples > merge(List<String>, List<String>)

```java
class ListExamples {

  // Takes two sorted list of strings (so "a" appears before "b" and so on),
  // and return a new list that has all the strings in both list in sorted order.
  static List<String> merge(List<String> list1, List<String> list2) {
    List<String> result = new ArrayList<>();
    int index1 = 0, index2 = 0;
    while(index1 < list1.size() && index2 < list2.size()) {
      if(list1.get(index1).compareTo(list2.get(index2)) < 0) {
        result.add(list1.get(index1));
        index1 += 1;
      }
      else {
        result.add(list2.get(index2));
        index2 += 1;
      }
    }
    while(index1 < list1.size()) {
      result.add(list1.get(index1));
      index1 += 1;
    }
    while(index2 < list2.size()) {
      result.add(list2.get(index2));
      // change index1 below to index2 to fix test
      index2 += 1;
    }
    return result;
  }


}
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS bash

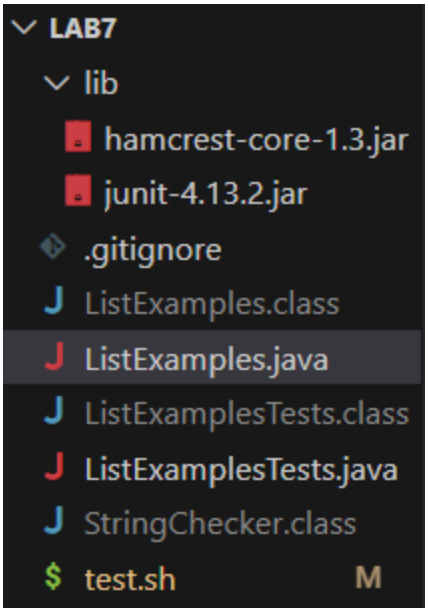
```
FAILURES!!!
Tests run: 2,  Failures: 1


Jason Zhang@LAPTOP-SMNVKH98 MINGW64 ~/Documents/GitHub/lab7 (main)
$ bash test.sh
JUnit version 4.13.2
..
Time: 0.014

OK (2 tests)
```

Setup:

The file & directory structure needed are in lab7:



The content of each file before fixing the bug:

`ListExamplesTests.java`

```
ListExamplesTests.java > ...
import static org.junit.Assert.*;
import org.junit.*;
import java.util.*;
import java.util.ArrayList;


public class ListExamplesTests {
    @Test(timeout = 500)
    public void testMerge1() {
            List<String> l1 = new ArrayList<String>(Arrays.asList(...a:"x", "y"));
        List<String> l2 = new ArrayList<String>(Arrays.asList(...a:"a", "b"));
        assertArrayEquals(new String[]{ "a", "b", "x", "y"}, ListExamples.merge(l1, l2).toArray());
    }

    @Test(timeout = 500)
        public void testMerge2() {
        List<String> l1 = new ArrayList<String>(Arrays.asList(...a:"a", "b", "c"));
        List<String> l2 = new ArrayList<String>(Arrays.asList(...a:"c", "d", "e"));
        assertArrayEquals(new String[]{ "a", "b", "c", "c", "d", "e" }, ListExamples.merge(l1, l2).toArray());
        }

}

```

`ListExamples.java`

test.sh M | ListExamples.java M | ListExamplesTests.java

ListExamples.java > ListExamples > merge(List<String>, List<String>)

```java
import java.util.ArrayList;
import java.util.List;

interface StringChecker { boolean checkString(String s); }

class ListExamples {

  // Returns a new list that has all the elements of the input list for which
  // the StringChecker returns true, and not the elements that return false, in
  // the same order they appeared in the input list;
  static List<String> filter(List<String> list, StringChecker sc) {
    List<String> result = new ArrayList<>();
    for(String s: list) {
      if(sc.checkString(s)) {
        result.add(index:0, s);
      }
    }
    return result;
  }


  // Takes two sorted list of strings (so "a" appears before "b" and so on),
  // and return a new list that has all the strings in both list in sorted order.
  static List<String> merge(List<String> list1, List<String> list2) {
    List<String> result = new ArrayList<>();
    int index1 = 0, index2 = 0;
    while(index1 < list1.size() && index2 < list2.size()) {
      if(list1.get(index1).compareTo(list2.get(index2)) < 0) {
        result.add(list1.get(index1));
        index1 += 1;
      }
      else {
        result.add(list2.get(index2));
        index2 += 1;
      }
    }
    while(index1 < list1.size()) {
      result.add(list1.get(index1));
      index1 += 1;
    }
    while(index2 < list2.size()) {
      result.add(list2.get(index2));
      // change index1 below to index2 to fix test
      index1 += 1;
    }
    return result;
  }


}
```

test.sh

```
$ test.sh M ×   ListExamples.java   ListExamplesTests.java
$ test.sh
javac -cp ".;lib/hamcrest-core-1.3.jar;lib/junit-4.13.2.jar" *.java
java -cp ".;lib/junit-4.13.2.jar;lib/hamcrest-core-1.3.jar" org.junit.runner.JUnitCore ListExamplesTests
```

The description of fixing the bug:
Change `index+=1` in the final while loop to `index+=2` to properly add elements from `list2` to the result arraylist and keep the loop from running indefinitely.

part 2

The most useful thing I learned was how the gradescope operates internally. GradeScope uses scripts/shell files with meaningful command lines to evaluate student code and assign scores based on its behavior. I'm not sure how Gradscope tests our code and assigns us grades before we attend lectures and labs (I initially assumed instructors were using magical commands or were behind the screen manually running our codes). Knowing how it works gave me a better understanding of how my programming assignments are evaluated, as well as inspiration for creating meaningful and effective test cases on my own. It's great to understand the entire process behind the scenes.