Northeastern University
CS5500 – Managing Software Development
Fall 2018, Professors Jump, Katz, and Weintraub

Homework 4: Testing

Functional Testing

For each of methods specified in the `TestingHomework` interface, create a sufficient number of test cases using `JUnit5`. Use `maven` to create a project with your test cases in the appropriate `src/test` directory.

For grading, we will run — via `maven test` — your test cases against code that implements this interface. You must assume the implementation uses a class called `TestingHomeworkImpl`. The constructor has no parameters. The class has no variables. None of the methods use checked exceptions.

Finally, write in a text file whether the implementation should be accepted or not based on your test suite. Justify your answer. "Passed all the tests" is not a sufficient answer. Explain why there should be confidence in your answer.

Your submission should be a project covering only this problem and your justified accept/reject position. (10 points each - 60 points in total)

```
1  package homework.testing;
2
3  /**
4   * This interface specifies a number of methods for which you need to create
5   * a sufficient test suite.
6   */
7
8  public interface TestingHomework {
9
10   /**
11    * Calculates the square root of the input
12    *
13    * @param n the radicand
14    * @return the square root of the radicant
15    */
16   double sqrt (int n);
17
18   /**
19    * Calculates the square of the input
20    *
21    * @param n the factor
22    * @return the product of the factor times itself
23    */
```

```
24    int sqr (int n);
25
26    /**
27     * Calculates n!
28     *
29     * @param n the largest factor to consider
30     * @return n!
31     */
32    int factorial (int n);
33
34     /**
35     * Calculates the sum from 0 to n.
36     *
37     * @param n the largest addend
38     * @return the sum
39     */
40    int sumUp(int n);
41
42    /**
43      * Simple function that adds two numbers together.
44      *
45      * @param x the first addend
46      * @param y the second addend
47      * @return the sum of x and y
48      */
49     int simpleFunctionXplusY(int x, int y);
50
51     /**
52     * Replace multiple contiguous spaces in a text string with a single space
53     * That is, ␣␣ becomes ␣, ␣␣␣ becomes ␣ and so forth.
54     * @param inputText the input text
55     * @return the string with only single spaces in it
56     */
57    String despacer(String inputText);
58 }
```

Structural Testing

For the following method, `example(String)`, create the CFG (control flow graph). Turn in a document (scan of hand-drawn CFG is acceptable) with your CFG. A Java file with this code is attached.
Then, create a `maven` project copying the code into the project and then creating with the ExampleImpl featuring a sufficient number of test cases using `junit5` to achieve 100% branch coverage. Use maven to create a project with our source in the appropriate `src/main/java` directory and your test cases in the appropriate `src/test/java` directory.

For grading, we will run your test cases against this code via `maven test` and observe the branch coverage generated.

Finally, write in a text file whether the implementation should be accepted or not based on your test suite. Justify your answer.

Your submission should be a project covering only this problem, your CFG, and your justified accept/reject position. The last two may be in the same document. (40 points)

```
1  package homework.testing;
2
3  public class ExampleImpl {
4
5     public int example(String inputText) {
6        /**
7         * counts the number of multiple contiguous space substrings
8         * in a given string.
9           *
10        * That is, ␣␣ counts→ 1, ␣␣␣ counts→ 1
11          * for example, "␣␣foo␣␣␣foo" returns 2
12
13        * @param String - the string to process
14        * @return how many
15        */
16
17       boolean foundSpace = false;
18       boolean multiple = false;
19       int result = 0;
20
21       for (int i = 0; i < inputText.length(); i++) {
22
23          if (inputText.charAt(i) == ' ') {
24             if (foundSpace) {
25                multiple = true;
26             }
27             foundSpace = true;
28          } else {
29             if (foundSpace && multiple) {
30                result++;
```

```
31              }
32              foundSpace = false;
33              multiple = false;
34           }
35        }
36
37        return result;
38
39     }
40
41  }
```