The Problem Set 1 for Exercise

1. Introduction

The problems in this set require the students to implement test cases under the Junit 5 framework for testing the Java class that is a revision of the Triangle Problem.

2. Specification of the input and the output

The Java class to be tested is still named Triangle (compiled from source file Triangle.java, which is a minor revision of the version in Programming Assignment 1). The Triangle class implements a public method, *triangle(int a, int b, int c)*. The following table specifies the expected action by this method. The actions "**report-out-of-range**" and "**report invalid**" have different meanings in different parts of this exercise (see Section 3).

<table>
<tr><td colspan="4">Conditions of Input</td><td>Expected Action</td></tr>
<tr><td colspan="4">! (1≤ a ≤200) or !(1≤ b ≤200) or !(1≤ c ≤200)</td><td>report our-of-range</td></tr>
<tr><td rowspan="4">(1≤ a ≤200) and
(1≤ b ≤200) and
(1≤ c ≤200)</td><td colspan="3">a=b=c</td><td>return EQUILATERAL</td></tr>
<tr><td>!(a=b=c)</td><td colspan="2">!(a+b>c and a+c>b and b+c>a)</td><td>report invalid</td></tr>
<tr><td rowspan="2">a+b>c and
a+c>b and
b+c>a</td><td>a=b or b=c or a=c</td><td>return ISOSELES</td></tr>
<tr><td>a≠b and b≠c and a≠c</td><td>return SCALENE</td></tr>
</table>

NOTE: In the above table, the column labeled "Expected Action" lists symbolic constants whose integer values are defined in the Triangle class as the following:

```
public static final int SCALENE = 0;
public static final int ISOSELES = 1;
public static final int EQUILATERAL = 2;
```

3. Tasks for This Exercise

This exercise has **two parts**, each constituting a set of test cases that can be run separately from the other parts. In your current directory "Practice1", where this problem set specification is found, you can find two subdirectories, named "Part1"and "Part2", corresponding to these two parts.

For each part, find in the corresponding subdirectory two Java source files. The file Triangle.java implement the Triangle solution that is to be tested. *It may or may not contain implementation defects.* Hence if your testing reports a failure, it is perfectly normal, as long as your test methods are composed correctly.

The second Java source file, TriangleTest.java, is incomplete. Your task is to complete this source file, using JUnit 5 annotations, such that the test cases are implemented as specified below for each part. To test your work, run command "make" to invoke the Makefile provided to

you in that subdirectory. (If you have copied the problem set to your own computing device, instead of working on the Data server, you need to change the path for the Junit 5 command line interface JAR file.)

For the CSV files you are required to compose, there is no requirement on the order between the data lines for the test cases.

Header lines at the top of the CSV file are permitted, so long as your test method properly handles them.

NOTE: The file "output.txt", generated by the test run, may contain special characters (which are ANSI color codes). If you want to properly view the content of the file, simply use the "cat" command.

The following subsections specify the task for each part of this assignment.

Part 1

For this part, the action "report out-of-range" is expected to throw an `IllegalArgumentException`. The action "report invalid" is expected to return the integer value -1.

The task is to write TriangleTest.java to test the cases (a, b, c) = (100, 100, 1), (100, 100, 100), (150, 200, 130), (100, 100, 201), (100, 100, 200), (120, 120, 100), (2, 50, 2), (201, 201, 201).

If a test case is expected to cause a "report out-of-range" action, then your test class must verify that the correct exception is thrown by the triangle() method *without checking any error messages.* For all other test cases, compose a CSV file to include all these cases, and the TriangleTest class should perform all these tests.

Part 2

This part differs from Part 1 with only a single change explained below. Otherwise, the task is the same as that in Part 1.

For this part, the action "report out-of-range" is expected to throw an `IllegalArgumentException`, with the error message "`Out of Range`" generated (NOTE: a single space between the words).

If a test case is expected to cause a "report out-of-range" action, then your test class must verify that the correct exception is thrown by the triangle() method with correct error message generated. For all other test cases, compose a CSV file to include all these cases, and the TriangleTest class should perform all these tests.