SYSC 4101

Laboratory 1

A `Java` class, named `OrdSetSimple`, implements an ordered set of integers: it is a set so there is no duplicate; it is ordered and integer values are ranked in increasing order; it has an upper bound size. Among other things, it supports a basic sets operation, named `difference`, which determines the difference between two sets: the difference between sets `s1` and `s2` contains all the elements in `s1` that are not in `s2` (e.g., `{1,3,4,5}-{2,3,4}={1,5}`).
The upper bound size of the set of integer values is given as an argument of the constructor. Once created with a given upper bound size `UB`, if a client of `OrdSetSimple` tries to add more than `UB` elements, the element is not added and an error message is written on the standard output.

The material for the first lab contains two files for this case study: the `OrdSetSimple.java` file; the `OrdSetSimple_Driver.java` file, which is a sample test driver for method `difference` of the `OrdSetSimple` class. A test driver contains all the test cases we wish to execute. The sample test driver contains only one test case.

Assuming you have created a folder named `Lab1` and unzipped the contents of the lab material in `Lab1`, you should see one subfolder named `CaseStudy`. The code of the case study is structured in several folders[1] under a folder called `CaseStudy`. The zip file contains a commands.rtf file that shows the commands you will need to execute (from command line) during this lab, either for Windows-like machines or for Unix-like machines. The instructions give the commands that are required to (1) compile the system under test, (2) to compile the test driver, (3) to instrument (with jacoco[2]) the compiled system under test, (4) to execute the tests on the instrumented version of the system under test, and (5) to produce a structural coverage report that indicates what has been executed and what has not.

What to do…

You are asked to test the code of class `OrdSetSimple`. Use your own judgment to define test cases for all the public functions.
You are asked to ensure that your test suite, that is your set of test cases, achieves 100% instruction coverage as well as 100% branch coverage on the entire code of class `OrdSetSimple`.
Use the report produced by `jacoco` to guide you in the definition of test cases.

What to submit… (This is individual work.)

1. Your test suite as a single Java file that can be executed as discussed above (i.e., it is not a JUnit test file). Each test case should specifically indicate, in source code comments, what inputs and expected outputs the test case uses.
2. The report folder.

[1] The rationale for having different folders will become clearer as you continue reading. Some folders are initially empty; this is normal. That will also allow you to transfer knowledge acquired during this lab, especially the use of jacoco, to other, possibly larger projects.
[2] Jacoco jar files provided in the zip file.

3. Any comment (in plain text) you may have to reflect on your experience in this laboratory, including, though not necessarily restricted to:
   - How easy (or difficult) it was to reach the structural coverage objective, especially for private functions.
   - Given we (including me when producing the code of this exercise) cannot ensure code is free of fault, whether you have found a fault. If your tests reveal a fault, you are asked to indicate which test case reveals the fault and why. Also explain how you would fix the fault.

Zip all your files into one zip file and name your zip file with the following nomenclature: **SYSC4101-lab1-yourLastName-yourFirstName.zip**.
You are asked to show your results to the TA before submitting.