**SE401: Software Quality Assurance & Testing**

**Activity 7.3: White Box Testing**

For this activity, you will be using the source code of the methods. For each method you should first derive a suite of test cases which together have full statement coverage of the source code. Then you should write additional test cases (if any are needed) which together with the ones already there have full branch coverage of the code. The test cases should be executable with JUnit. If any of the test cases for a method fails, try to find the bug and correct the source code.

Keep in mind that you may need a method, A, to set up the test of another method, B. If the method A has a bug it can be very tricky to test method B. Wherever possible you can first fix the bug in a method and then do the tests that use it in the setup (or test oracle). Also keep in mind that a bug in the implementation may imply that full statement or branch coverage is not possible. This it in itself an indication of the bug's existence. In such case, it's enough if you have full coverage after fixing the bug.

The activity is about the class Set.

> Set

Set represents sets of integers. The elements of a set are stored in an ArrayList. They are sorted and without duplicates to speed up some operations. Two methods might need an explanation:

 **public** **void** section(Set s) { ... }

removes from this set any element that is equal to an element in s.

 **public** **boolean** containsArithTriple() { ... }

returns true iff there are three elements, x, y and z, in this set such that y - x = z - y.

Note that you can use toArray to easily examine the state of a class instance.

**a)**Use statement and branch coverage to derive test cases for insert.

**b)**Use statement and branch coverage to derive test cases for member.

**c)**Use statement and branch coverage to derive test cases for section.

**d)**Use statement and branch coverage to derive test cases for containsArithTriple.

**e)**If you find any bugs, try to correct them in the implementation. Also keep in mind that it might be your specification that is incorrect. Make a new version of Set.java which passes all your tests.

|  |
| --- |
| DebuggingWhen an array is sorted, finding an element in that array becomes possible in logarithmic time by performing a [binary search](https://en.wikipedia.org/wiki/Binary_search_algorithm). The class BinarySearch implements a binary search for arrays of integers. However, there is a bug in the implementation of the method search. In the method main, replace the value 4 in the variable value by any other of the elements in array, and search will not be able to find it.Your task is to debug the previous implementation in order to find the error in method search. In your report for this activity you must include a description of how you have debugged the previous method (e.g. break points used, etc).**Reporting**Upload an archive Activity 7.1.zip (or rar, tar.gz, tar as you wish). The archive must have the following structure:Activity 7.1.zip | |-Set | |- <your\_1st\_Set\_test\_class>.java | |- ... | |- <your\_mth\_Set\_test\_class>.java | |- SetTestSuite.java | |- Set.java | \- report.txt | \-BinarySearch |- BinarySearch.java \- report.txt* Set and BinarySearch are directories
* Set/SetTestSuite.java is the test suite class. It looks like the following replacing the contents within angle brackets < > with your own class names in the same directory:

 import org.junit.runner.RunWith; import org.junit.runners.Suite; @RunWith(Suite.class) @Suite.SuiteClasses({ <your\_1st\_Set\_test\_class>.class, ... <your\_mth\_Set\_test\_class>.class }) public class SetTestSuite { // nothing goes here  } Executing with JUnit the SetTestSuite class makes JUnit run the test cases contained in the classes specified inside the @Suite.SuiteClasses annotation arguments.* Set/<your\_1st to nth\_test\_class>.java are all classes implementing test cases for exercise 1 (replacing the contents within angle brackets < > with your own class names)
* Set/<your\_1st to mth\_Set\_test\_class>.java are all classes implementing test cases for exercise 2 (replacing the contents within angle brackets < > with your own class names)
* report.txt is the report file (where you have to motivate how you engineered the test and report bugs you found / fixed).

**Remark:** No need to be more complex than this; avoid package declarations, avoid to add unnecessary files such as hidden files or folders produced by your text editor, for instance. |