Exercise – JUnit test for ArrayStack

The four jar-files (`Stack1.jar`, `Stack2.jar`, `Stack3.jar` and `Stack4.jar`) each contain the class files for the interface `StackADT<T>` and one version of the implementation of `ArrayStack<T>`. These implementations (jar files) can be found here:

- https://ict-engineering.dk/jar/Stack1.jar
- https://ict-engineering.dk/jar/Stack2.jar
- https://ict-engineering.dk/jar/Stack3.jar
- https://ict-engineering.dk/jar/Stack4.jar

I can tell you that one of the implementations should have no errors, another implementation contains 2 errors and the remaining two each contain 1 error. Note that one error could course more of your JUnit test methods to fail. There is a prize (to be given next time we have a physical meeting) to the first person in your class (or first group) finding all the correct errors.

Step 1

Read and understand the specification for interface `StackADT<T>`

- http://ict-engineering.dk/javadoc/Collections/utility/collection/StackADT.html

and the further specifications for all four implementations of `ArrayStack<T>`, which are

- The stack may contain `null` elements
- Duplicate elements are allowed
- A stack is never full
- Default capacity is 100 (initial capacity when calling the zero-args constructor)
- After trying to add more elements when the size is equal to the capacity, a new array with twice the capacity is created
- `toString` method return a string with the elements separated with comas and encapsulated in a set of curly braces. Top element first, example: `"{C, B, A}"`

Step 2

a) Make a JUnit test and test one version of `ArrayStack` (add one of the 4 jar files as a library to the IntelliJ module). *Note: An easy way to generate the JUinit test is first to create a dummy class use ALT-ENTER … to generate the JUnit test, and just delete the dummy class again.*

b) Specify if the implementation has failed the test and if so, in which method(s) in `ArrayStack` to look for errors (what you expect to be wrong)

Remember to test using ZOMB+E and thereby also the following

- Boundaries (using Boundary value analysis)
- Equivalence partitioning
- `null` elements and duplicates
- All legal types of exceptions
- And if the internal array expands correctly

Step 3

Reuse the test for the remaining 3 jar files, one module for each jar file to test.