Exercises, SDJ1

Exercise: javadoc

a) Find a solution to one of your exercises, e.g. the GradeList exercise - and copy this into a new module.

b) Write Javadoc comments in the class(es).

- An overall comment in top of the file
  - General description for the class
  - @author with the author of the code
  - @version with version number and the date
- Comments before each method
  - Description for the method
  - @param for each of the parameters (if any)
  - @return (if any)
  - @throws for each of the type of exceptions and when (if the method throws exceptions – which is not relevant for the SDJ1 exercises you have made until now)

c) Generate Javadoc documents. Open and inspect the result.

Exercise: JUnit – Blackbox

a) Find a solution to one of your exercises, e.g. the GradeList exercise - and copy this into a new module (Alternative, if it is the same solution as the Javadoc exercise then reuse the module).

b) Create a JUnit test suite in IntelliJ

- Create test methods for each constructor and methods - using the ZOMB+E approach
  - Zero
  - One
  - Many (using equivalence partitioning not to test for all values)
  - Boundary (both sides of the boundaries)
  - Exceptions (could be relevant for a few methods, e.g. NullPointerException or IndexOutOfBoundsException)

c) Run the test.

Exercise: JUnit – Whitebox

a) Find a solution to one of your exercises, e.g. the GradeList exercise - and copy this into a new module.

b) Select one or two methods to whitebox test (a method that at least contains a loop and maybe also some if statements). On paper, write down all possible paths as different test cases

- One test case where you enter an if statement (where the condition is true)
- Another test case where you skip an if statement (where the condition is false)
- Another test case where you skip a loop (where the condition is false)
- Another test case where you have exactly one loop cycle

- A few test cases where you have “many” loop cycles (e.g. 3, 4, 8)
- Another test case where you have the maximum expected loop cycles (boundary)
- Another test case where you have the maximum + 1 loop cycles, if possible (boundary)
- If the method takes a String or another object, then test cases where these parameters are null
- If the method could throw exceptions then test cases forcing the exception to be thrown
    - E.g. `getGrade(index)` called with an index out of bounds

c) Create a JUnit test suite in IntelliJ, type in test method for each of the test cases, run and inspect.