COMP20250 15% JUnit Testing CA

Last Updated: November 2021

1 Key Submission Details

Deadline: Dec 3rd 2021, 5pm (uploaded to Brightspace).

Late Submissions Standard UCD policy on late submissions applies; see `https://www.ucd.ie/t4cms/latesub_po.pdf`. Your submission is deemed late if at least one deliverable is submitted late.

Plagiarism: The submission must be yours and yours alone. If you are unsure what is or is not plagiarism, the following is a none exhaustive list of example activities you cannot do:

- Copy the completed files of another student and submit them as your own
- Share copies, images or print outs of your code with another student (by e-mail, FB messenger, WhatsApp etc.)
- A group of students working on a single solution and then all submitting the same work or subset of the same work (regardless of whether variable, method, class names or ordering have been changed)
- Students collaborating at too detailed a level. For example, consulting each other after each line / block / segment of code and/or sharing the results.

For more details see:

- `https://csintranet.ucd.ie/sites/default/files/cs-plagiarism-policy_sept2020.pdf`, and
- `https://www.ucd.ie/governance/resources/policypage-plagiarismpolicy/`
- `https://www.ucd.ie/secca/studentconduct/`

Any submission suspected of plagiarism will be submitted to the School of Computer Science Plagiarism subcommittee for further investigation.

2 Task

For this assignment, you should answer one of the past 70% assessment questions from the 2020 paper on Brightspace. The question is determined by your student number. If your student number:

- ends with a 0 or a 1: do part A question 1,
- ends with a 2 or a 3: do part A question 2,
- ends with a 4 or a 5: do part B question 3,
- ends with a 6 or a 7: do part B question 4,
- ends with a 8 or a 9: do part B question 5.

You should answer the question two times such that you would expect:

1. a first solution to receive a B grade or as high as you can,
2. a second solution receive a D grade.

Note: these grades are a guide for you (i.e., not a "must") to help you have two solutions of differentiable quality. For simplicity, there is no distinction between a B, a B+ or a B-, i.e. a B or better grade should fall somewhere in the 70-100% range, and a D somewhere in the 40-55% range. All parts of both solutions **should**:

- compile, and
- be complete, i.e. an answer must be provided to parts that expect a coded solution (however, it can be incorrect as long as it compiles). You cannot differentiate the quality of the solution based on the last part of each question, as this is reserved for me to assess the quality of your code in a real assessment scenario.

To substantiate your claim on the quality of the solutions, prepare and execute ONE set of JUnit tests that evidences the rank order of your solutions. Both solutions should be tested with the same set of unit tests.

You can have two JUnit test .java files, if needed (these can be hard-coded to call the various solutions). You can also give each of your solutions unique names for this assignment (i.e. you are not obliged to following the naming instructions in the exam), just make sure this is made clear in your **video** to avoid any confusion during the grading process. However, all methods with a `@Test` annotation must be the same if multiple instances of the JUnit tests are present.

If you are unsure how to start, recreate the examples as unit tests. Other things you could consider: look at any requirement in the question text, and try to derive a test for that. Explore the possibility of incorrect input and whether the solution can handle that.

3 Deliverables

3.1 Video Demonstration

Prepare a short video (max 5 min – see OBS instructions on Brightspace) that discusses the testing strategy of your JUnit tests specifically with respect to identifying better vs. worse solutions. You should discuss:

- the coverage of your unit tests
- how you have assigned marks to the unit tests to derive a score for the solution (map this to the UCD CS scale: `https://csintranet.ucd.ie/CSGrading/`)
- how (briefly) each test provides an insight into the solution quality

Finally, provide a summary of the findings / results and discuss why one solution is better than another providing a rank order that refers to the test results. The discussion on solution quality **should only** be on the basis of JUnit test results not comments, code formatting or any other non-functional properties of the code.

3.2 Code

Provide a .java file of your JUnit tests. Each test should be appropriately commented to note:

1. which question part they are evaluating, and
2. what they are seeking to test (this can be a repeated version of detail in the video)

Provide a .java file for each of your solutions. Each solution should be commented to identify which part(s) of the code correspond to the part(s) of question. You should also include a comment at the start of the .java file to note whether the solution is to be considered the B (or better), or D grade solution.

4 Grading Criteria

See next page.

Criteria	A+, A, A-	B+, B, B-	C+, C, C-	D+, D, D-	≤ FM+
Testing Strategy (10/15)	A very thorough set of unit tests is implemented. They capture all aspects of all parts of the question. A large number of fringe cases, erroneous input, and valid input scenarios are captured as well as expected outputs and behaviours. In general, the presented unit tests enable a very established basis for comparing solutions.	A thorough set of unit tests is implemented. They capture most aspects of most parts of the question. A reasonable number of fringe cases, erroneous input, and valid input scenarios are captured as well as expected outputs and behaviours. In general, the presented unit tests enable a solid basis for comparing solutions. More tests would have enabled a more informed discussion.	A good attempt at unit testing most parts of the question is present. Some fringe cases, and erroneous input scenarios are captured as well as valid input scenarios expected outputs and behaviours. In general, the presented unit tests enable a reasonable basis for comparing solutions. A more detailed testing strategy would have enabled a deeper, more informed discussion.	Evidence of unit testing for major parts of the question (e.g. only parts b and c). Limited testing of fringe cases, erroneous input, and valid input scenarios are captured as well as expected outputs and behaviours. The presented tests were able to provide some detail to compare solutions, but somewhat superficially.	Testing is superficial or (mostly) not present. At least two question parts (i.e. a, b, c, or d) have not been tested. One or more files don't compile or cannot be run due to errors or similar.
Video Presentation (5/15)	A very well executed video presentation that shows the running of the tests, discusses at depth the testing strategy and accurately explains all relevant choices and/or decisions concerning the design of the JUnit tests to differentiate the quality of each solution.	A well executed video presentation that shows the running of the tests, accurately explains most key relevant choices and/or decisions concerning the design of the JUnit tests to differentiate the quality of each solution.	A video presentation that shows the running of the tests, accurately explains a selection of relevant choices and/or decisions concerning the design of the JUnit tests to differentiate the quality of each solution. There are some JUnit test cases, however, that are either not addressed in the discussion or not adequately addressed to understand their inclusion and/or role.	A video presentation that shows the running of the tests, with a discussion that is accurate, but lacks significant depth and/or detail.	A video presentation that doesn't really show the running of the tests, and has a discussion that seems arbitrary, largely inaccurate and/or is not well informed by the performed JUnit tests.