CSCI 2134 Final Exam

3:30PM-6:30PM Atlantic Daylight Time, April 19, 2023

Submitted via Git .

Before the Exam:

Clone the Final Exam repository

https://git.cs.dal.ca/courses/2023-winter/csci-2134/final/????.git

where ???? is your CSID.

Be sure that that stub code compiles and that the stub Junit tests run in IntelliJ.

At the Start of the Exam

Perform a **pull** on the repository to update it. You will find that your code base has been updated and that there is much more code in them. There should be several files in the **`src`** directory and one file in the **`test`** directory (**`Tests.java`**). This is the code base for your exam.

All code modifications must take place in the java files. All written answers are to be placed in the comment at the top of the `Tests.java` file.

Exam Rules

- The exam should be written using whatever development tools that you prefer.
- All exams will be reviewed by the course instructor
- The exam is open book. You are permitted access to
  - Your course notes, including any provided links from Brightspace
  - All code that you have written previously
  - Any books that you have at the time of writing

  You are NOT permitted to
  - Perform web searches or get help from the Internet
  - Receive assistance from any other person either locally or remotely
  - To take any other unfair advantage of the situation
- The exam is 3 hours (180 minutes) in length.
  - When you first begin the exam you **must** pull the new code, add your name and Banner ID to Tests.java, git add, commit, and push.
  - If you have accommodations for writing exams, these will be reflected in your exam settings.
  - If you encounter any issues, please let the course instructor know immediately either via Teams or email.
- To ask questions in the exam, email the instructor or use the private chat feature in Teams. Please do NOT post your questions in the Teams public discussion forum. The course instructors will reply to you directly.
- The exam is out of 100 marks.

Good Luck! May the Source be with You!

Questions

1. **[10 marks]** Using Git, clone and pull the repository for your exam. Add your name and Banner ID to Tests.java, git add, commit and push. You have used Git.
   NOTE: To get full marks you **MUST** perform a commit **AND** push after each question. This way, if something goes wrong, I can check each part separately and give you the marks you deserve.
   ADDITIONAL NOTE: Sometimes the Git server is slow when many students access it at the same time, particularly at the beginning and end of the exam period. If you are unable to push then please inform the course instructor and continue on to the next question. You still must perform a commit after every question.
   Note: Your **TARGET** method will be specified at the top of the **`Tests.java`** file.

2. **[15 marks]** Testing
   a. Give three (3) written English test cases for the TARGET method (see above).
      Each written test case description should be one line long in the area for written answers.
   b. Implement the unit tests in the test file.
3. **[15 marks]** Debugging: Debug the issues causing your tests to fail. There are at least a couple bugs in the code. Add more tests if necessary. List bugs you found and fixed. Give
   - a brief description of each bug
   - the method where the bug occurs
   - how you fixed the problem
4. **[15 marks]** Defensive programming with assertions:
   a. Identify three (3) locations in the code where **different** assertions are appropriate. Give
      i. the method where the assertion should be used
      ii. what the assertions should assert
   b. Write the assertions in the code.
5. **[15 marks]** Defensive programming with exceptions
   a. Suggest one (1) exception that is appropriate for the TARGET method and one (1) additional **different** exception somewhere else in the code. State the exception name and condition that throws the exception.
   b. Implement the exceptions in the code.
   c. Add two additional unit tests to test that the exceptions are thrown when appropriate.
6. **[15 marks]** Procedural refactoring
   a. Identify three (3) **different** procedural refactoring opportunities to be done in the code. Give
      i. a brief description of each issue
      ii. the method where the issue is
      iii. how to fix the issue
   b. Perform the refactorings on the code. Be sure to do regression testing.
7. **[15 marks]** Identify three (3) **different** class-level refactorings that can be done in the code. Give
   i. a brief description of each issue
   ii. where the issue is
   iii. what SOLID principle (if any) is violated
   iv. whether a class implementation or class interface refactoring is needed
   v. how to fix the issue (**note: do not perform the refactoring**)