Assignment 1: Testing and Bug Fixing

The primary task that a software engineer is responsible for is testing and bug fixing code, especially at the beginning of their tenure at a company. It is a reliable way to get up to speed prior to implementing any new software features yourself.

You are given a little less than 2 weeks for this assignment but you only really need 1, the extra slack is for students joining the course later, students who join the course in the few days leading up to the deadline or otherwise have a valid documented excuse may choose to complete an alternative assignment in place of this. For this assignment you are required to make a **MINIMUM of 3 substantial commits** within your repository, please see assignment 0 to review the definition of a 'substantial commit'.

It is important to recall that you will be doing work like this all semester, in the event you find yourself immensely struggling with this assignment then you are not prepared for this course and are advised to drop, some struggle is okay and expected but you shouldn't be totally lost. A failing grade on the individual project will result in a lower letter grade due to the weight this project carries so please make wise decisions regarding your enrollment in this course.

Solutions will be provided to allow students who do not feel comfortable with their solution the chance to earn full marks on the other individual assignments. Also ***you are not allowed*** to consult with other students on this assignment, the time to work with other students will come later in the semester.

0. Forking the Repository and Running on Your Machine

To begin you are going to need to fork the repository provided on courseworks **Prior to moving forward please run the IndividualProjectApplication class with a single command line argument, leave it running until you see "Start Up" in the terminal then terminate the program. This initializes the database and if you run into any trouble with the database this is the process you'll need to perform. Moving forward, run the program without any command line arguments.**

1. Academic Honesty Pledge

As stated in Assignment 0 and on courseworks, Academic Honesty is an integral part of any course as such you will add to your repository a file named “honesty.txt” failure to include this file will result in an immediate 0 on the assignment. In that honesty.txt file you will include the following pledge:

I, <insert name (uni)>, have read and understood the following:

1. CS department's Policies and Procedures on Academic Honesty
2. The Course Specific Academic Honesty Policies
3. The assignment specs outlining the consequences of not submitting this pledge and other aspects of the policy

I affirm that I will abide by all the policies stated in the relevant materials from above. I understand that the relevant policies apply to: individual assignments, group projects, and individual examinations.

I also affirm that I understand that all course materials, with the exception of the individual/group project, are subject to the appropriate copyrights and thus will not post them on any public forum or publicly hosted repository, this includes but is not limited to: GitHub, stackoverflow, chegg etc.

I also affirm that I will be 100% honest when evaluating the performance of myself and my teammates when prompted by an assignment or member of the teaching staff.

Finally I affirm that I will not attempt to find any loopholes in these policies for the benefit of myself or others enrolled in the course presently or possibly in the future.

Signed: <insert name (uni)> <date>

2. Code Clean Up

Reading code is a great way to understand what it does and is an important skill to have, the previous developers of this codebase left it a terrible mess, in more ways than one. Your first job is to make the codebase clean again. Here is what we mean when we say the codebase must be clean:

1. Running the "mvn checkstyle:check" command in the terminal yields no warnings or violations, the report created by "mvn checkstyle:checkstyle" is empty, and obviously the code still runs

The particular style guide you will be checked against is the Java Style Guide by Google and it is here for your reference: Google Java Style Guide

You will need to learn how to write javadoc if you haven't learned how to do so already. Here is a source to get you started: How to Write Doc Comments for the Javadoc Tool (oracle.com)

It is crucial to note that checkstyle is NOT a bug finder, this was discussed in lecture but it is important to note it here as well.

3. Test Suite Creation

You should notice in the tests folder that there is a partially completed test suite provided for you as a base, you will need to expand this test suite to cover all the methods that exist within the codebase. It is common for software engineers to only test the happy path and we encourage you to go beyond the happy path and test edge cases which may break the code. You are required to have at minimum 55% branch coverage with the tests you write. You can view the code coverage report in the target/site/jacoco directory. Code Coverage analysis is done by JaCoCo and for your group projects you'll have a more strict 85% branch coverage requirement so it is good to get some exposure now. You will also need to make a separate test class file for each respective class present in the codebase, do not throw all your tests into a single file. **YOU MUST USE THE DATA PROVIDED TO ESTABLISH THE EXPECTED VALUES FOR YOUR TESTS.** The tests you write can be unit tests or any other test type that you find fitting as long as the requirements are met.

4. Bug Fixing

At this point all you have left to do is squash some bugs, there are different kinds of bugs that can occur either statically or not. You

should use your test suite that you have created as well as a static bug finder to fix any bugs present in the codebase. You may choose any static bug finder you wish, but you need to make sure it is free to use and you must indicate in the README.md what commands are needed to run the bug finder and analyze its results. Here is a recommended option for a static bug finder that you can use, it works on linux, macOS, and windows: PMD If there is a piece of code where deciding whether or not a bug is present is difficult, then please rely on javadoc comments to make this determination.

5. Submission Details:

On Courseworks under "Individual Assignment #1" you will submit a link to your repository, remember that if you made your repository private then you must make all members of the Teaching Staff contributors on the repo so we may grade you. At the root level of the repository you must have the honesty.txt file described in section 1 as well as your citations.txt file which contains the links and descriptions of all the resources you used during this assignment as a reminder ***you are not allowed*** to work with any other student on these individual assignments.

This assignment is due at **11:59 pm on September 13, 2024** if for some reason you still on the waitlist and not able to view the coursework submission please email your submission in the following format:

> Subject: COMS 4156: Individual Assignment #1 <uni>
> Body: <Link to Your Repository>
> To: gcn2106+4156@columbia.edu

If you joined before the end of change period and feel as if you do not have time to finish this assignment prior to the stated deadline a write an email with the following information, the alternative assignment is tentatively due **11:59 pm on September 16, 2024** the deadline to make such a request is: **11:59 pm on September 13, 2024**

> Subject: COMS 4156: Request For Alternative Assignment <uni>
> Body: <Details of Reason/Documentation>
> To: gcn2106+4156@columbia.edu
> CC: kaiser+4156@columbia.edu