CPSC 219: Introduction to Computer Science for Multidisciplinary Studies II

Project

Weight: 50%

(10%+10%+10%+5%+15%)

(3 x demo, 1 reflection report, 1 code submission)

Collaboration

This is a pair project. This means you can freely discuss your project and share your code with your project partner. You are expected to share your code with your partner as you program. It is important to remember that there will be demonstration/reflection components in which you, as an individual, will need to demonstrate knowledge of the part of the project that you coded. Therefore, the expectation is that students will discuss their plans, and code with frequent communication. However, the whole project should not be completed by only one of the partners. The rubric will have grade portions that will be shared between students, as well as parts that will be individual.

If you have concerns about your partnered group, this should be dealt with early. The project size will be scaled such that if one member leaves the group the remaining individual can still complete the project. Some of the skills we want you to demonstrate due to programming with your partner will have to be simulated via other considerations.

Outside of your partner, you should not share code, collaborate, or download code from the internet to complete your project (that isn't cleared by the instructor or TA). If you are considering using a library to add a non-core feature to your assignment, then you should clear this usage with your instructor/TA. For example, if you want to use a Java library that represents dates, this is allowed as this is a non-core feature. If you want to use a library that handles file input, or storage of your data, then this is NOT allowed, as this is the core code you are designing and programming.

In general, discussing the project requirements with others is a reasonable thing to do, and an excellent way to learn. However, the work you hand in must ultimately be **your group's work**. This is essential for you to benefit from the learning experience, and for the instructors and TAs to grade you fairly. Handing in work that is not your **group's original work** but is represented as such, is plagiarism and academic misconduct. Penalties for academic misconduct are outlined in the university calendar.

Your Gitlab repository (other Git repositories) must be private, and only shared with the TA and your group member.

Here are some tips to avoid plagiarism in your programming assignments.

1. Cite all sources of code you hand in that are not your original work. You can put the citations into comments in your program. For example, if you find and use code found on a website, include a comment that says, for example:

   ```
   # The following code is from
   https://www.quackit.com/python/tutorial/python_hello_world.cfm.
   ```

   Use the complete URL so that the marker can check the source.

2. A tool like chat-GPT can be used to improve small code blocks. For example, five lines of code. If you get help from code assistance like Chat-GPT, you should comment above the block of code you requested assistance on debugging or improving and cite the tool used to get that suggestion. Using a tool like chat-GPT to write the majority of your assignment requirements will be treated as plagiarism if found without citation, and with citation, it will be treated as 0 for the component the student did not complete. Code improvement of short length will get credit if commented/cited properly.
3. Citing sources avoids accusations of plagiarism and penalties for academic misconduct. **However, you may still get a low grade if you submit code not primarily developed by yourself. Cited material should never be used to complete core assignment specifications. Before submitting, you can and should verify any code you are concerned about with your instructor/TA.**
4. Discuss and share ideas with other programmers as much as you like, but make sure that when you write your code, it is your own. A good rule of thumb is to wait 20 minutes after talking with somebody before writing your code. If you exchange code with another student, write code while discussing it with a fellow student, or copy code from another person's screen, this code is not yours.
5. **Collaborative coding is strictly prohibited**. **Your assignment submission must be strictly your code**. Discussing anything beyond assignment requirements and ideas is a strictly forbidden form of collaboration. This includes sharing code, discussing the code itself, or modelling code after another student's algorithm. **You can not use (even with citation) another student's code.**
6. Making your code available, even passively, for others to copy or potentially copy is also plagiarism.
7. We will look for plagiarism in all code submissions, possibly using automated software designed for the task. For example, see Measures of Software Similarity (MOSS - https://theory.stanford.edu/~aiken/moss/).
8. Remember, if you are having trouble with an assignment, it is always better to go to your TA and/or instructor for help rather than plagiarizing. A common penalty is an F on a plagiarized assignment.

Late Penalty

The group project has a couple of due dates. The demonstrations will be scheduled with the TA. When a TA indicates when these are happening, you must attend a tutorial to do this demonstration. If you do not appear, you will get a grade of 0. Alternate demonstration dates will not be scheduled if you miss your demonstration. Students will demonstrate in the tutorial time, that by enrollment, they've confirmed they can attend. If your group is a partner across two tutorials with different TAs, then you must demonstrate during one of those two times. You may be required to demonstrate in one of those two tutorials based on distribution of grading load between the two TAs.

The final code submission and reflection are due on the last day of lectures. A late submission of these will be a 10% penalty within the first 24 hours, and 20% penalty if it is within the next 24 hours (within 48 hours). After that, these components will be assigned a 0.

Groups

The maximum size of a group is 2. This is a partnership project. We prefer if both your group members are registered in PeopleSoft in the same tutorial. Your enrolled TA needs to be able to meet you for your demonstration within tutorial time. There is a possibility that you and a partner can partner across two different tutorial times. However, suppose you do partner across tutorial times. In that case, you should be prepared for demonstration times being arranged that will allow us to fairly manage the workload of TA grading/demonstrations properly. If you have the same TA in each of your tutorials, then this partnering is the same as if you were in the same tutorial.

Once we are sufficiently into the course, your TA will collect from you a statement of your group members so that they can verify your partnership is correctly formed. Anyone who does not form their group before this time will then be assigned a random group.

If your group members are not participating/communicating, or dropping the course, then you are responsible for keeping the TA appraised of this type of scenario. The TA and I can manage a fair assessment of your project going forward. In extreme circumstances, if you do not participate in progressing on the project with your partner, you will be split as group members to complete the remaining parts of the project such that you can be assessed on your singular input alone.

Goal

The goal of this project is to mirror the skills of Assignments 1,2 and 3. You will not be provided starter code for the project. Your goal is to create your own 'tracking software.' This software can be anything from tracking fitness, eating, hygiene, athletic sports statistics, esports statistics, etc.

There will be three stages to this project.

1. First, make a program in **Java** with a standard CPSC 217/231 procedural structure. Use **Git** version control properly to store this project. Perform unit testing via **JUnit** to establish the correctness of portions of the assignment code created. This program will capture input from **System.in**, store the data in data structures like Arrays/ArrayList/HashMaps/etc., and then allow the user to request summaries of data or other interesting queries such as regularity of participation, etc. and print this to **System.out**. You program will be a procedural program. You will not be making any new Classes and Objects, but will be expected to use existing Java Classes like String, ArrayList, HashMap, etc.
2. Rethink this first program to use classes and objects. The goal will be to represent stored data in properly designed classes/objects. This storage should be more than just

packaging data into a class. There should be proper encapsulation and inheritance used in the design. Other OO design principles are expected to be followed. Input/Output should be separated from data storage, and other parts of the project, beyond just data storage, should have proper OO design. Add the ability to read previously stored data from a .csv input file, and backup currently stored data to .csv file. Filenames should be indicated by command line argument configuration parameters when the program is stated.

3. Add a GUI system via JavaFX that allows data to be entered, and results to be viewed, so that input and output interactions no longer require the user to use the shell/terminal. The requirement is a GUI window, areas to input new information, minimal verification that data is valid, an event trigger to store data, a trigger to view stored data, and a trigger to view summarized data. Loading/Backing up data to .csv files via event trigger should also be added. Note that most of the back-end data management is from part 2, part 3 is just a new interface to access these options.

Each one of these stages should utilize skills that the prior assignment involved (or were explored in lectures). For example, assignment one concerns Git, Junit, arrays, looping, functions, and conditional. You will add lecture material on storage features like ArrayLists and System.out/System.in I/O interaction to this. Another example, assignment two will be about classes/object design, including encapsulation and inheritance. Assignment 3 will be about JavaFX.

Technology

Java 20, Git, JUnit 5, JavaFX

Submission Instructions

The three demonstrations will be in-person in your tutorial time and require no code submission. The TA will view your code and see it running during the demonstration.

You must submit your project's final code and reflection electronically using **Gitlab** and **D2L** for the final deadline. Use the Project Final Code and Project Reflection dropbox in **D2L** for a final codebase electronic submission. You will also share a link to your **Gitlab** codebase with your TA in that D2L submission. In **D2L**, you can submit multiple times over the top of a previous submission. Do not wait until the last minute to attempt to submit. You are responsible if you attempt this, and time runs out.

Your project must be completed in **Java** (not Kotlin or others) and be executable with **Java version 20.** You must use a **private Gitlab** hosted at **csgit.ucalgary.ca** (not GitHub or another Git host). You must use **JUnit 5** and not other unit-testing libraries.

Description

Git Requirement: As we move through topics, we will introduce Git as a version control system. For this assignment, we will require you to upload your code to the university **csgit.ucalgary.ca** hosting site as a **private** repository shared with your TA (in addition to submitting it via D2L Dropbox). *If you are comfortable with Git before we cover it, you can start this process early and do all your code storage in Gitlab.* If Git is new to you, you will be taught it, and we will have more limited expectations of its usage. However, we will still expect your testing development to use regular commits. This means regular small code commits from both partners.

For the second and third demonstrations, we expect to see semi-regular usage of Git. This means small code commits from both partners.

The minimum expectation for Git usage is that when you submit your code, the TA can go and view it in Gitlab and see that you've made multiple commits as you edited it before the submission deadline to D2L. To use csgit.ucalgary.ca, you will need a functional **UCIT account** username/password.

Program Coding Requirement:

Additional Specification

- You must comment on your code with **Javadoc** comments.
- **Use in-line comments** to indicate code blocks and describe decisions or complex expressions.
- **Do not use inline conditionals**
- **Break and continue are allowed**.
- Put your **name, date, and tutorial** into the comments for the **Javadoc** of the classes for your TAs to identify your work.
- **You should not import ANY libraries to complete the core project. Using these could result in a grade of 0 for that portion of the assignment. Citing code from the internet to complete a core function will also result in 0. You can import libraries for tertiary things. Ex, like Date representation.**
- Use constants appropriately. Your TA may note one or two magic numbers as a comment, but regular usage will result in lost marks.
- Have an appropriate style, like variable naming, that is clear.
- You should have reasonable error-checking of input by demonstration two. For demonstration one, exceptions will not have been covered, so TAs will not examine error checking. For demonstration one, you can write code that expects proper user usage. For demonstration two, you should write code that expects the user to make mistakes, like typing a letter when asked for a number or giving an improper filename.

Grading

Demonstration 1 (out of 50)

Usage of GIT, usage of JUnit, proper Style, gets input via System.in, stores input in arrays/ArrayList/other, stored data is viewable via System.out, information can be calculated/retrieved about stored data via System.out

Demonstration 2 (out of 50)

Usage of objects/classes, file I/O, exceptions, command line arguments

Demonstration 3 (out of 50)

Usage of JavaFX to enable input and output viewing of stored data, proper usage of event-driven design

Final Codebase Submission (out of 75) Due end of lectures

25 points are assigned to progress from each demonstration level. The same rubric ideas will be assessed but for a final working codebase.

Reflection (out of 25) Due end of lectures

Individual report in which student demonstrates their experience during the project. What decisions did they make, if they were successful, what they might do differently in design, and what other features they would consider adding. Individuals will also report on what they learned from Git/JUnit and work with a partner to make code. A specific area of interest will be the view of the project development as the design moved from procedural, to OO, to event-driven GUI layer on top of OO design.