ENSF 380 WS 2025 Individual Project Instructions

Deliverables (What files to submit and in what format)

Overview

This document describes Submission 1 (Week 9) and Summative (Week 13).

Work independently on these sub0issions. You are expected to adhere to the course policy for individual work. The course outline provides information on appropriate and inappropriate forms of collaboration and generative AI use.

You may make use of the model solutions which were provided for the group assignments and/or material developed by your group as the base for your individual submissions. This includes code (submitted as GP #3 or provided as a solution to GP #3), tests (provided as a solution to GP #2), and UML (submitted as GP #1 or provided as a solution to GP #1). Alternately, you may choose to begin a completely new implementation of the project.

Keep in mind that when you submitted GP #1 and received a solution to it, we had not introduced several important OOP concepts such as inheritance and interfaces. The solution you received for each part of the group project was restricted to topics which had been covered. ***A well-designed solution to the individual assignment will require not only extending the code but also rethinking the design and refactoring the code.***

Your individual project submissions should include all functionality from the original (group) project, except where a feature request has replaced/updated the original functionality. The 'Features' document describes the new requirements for the individual project.

Submission 1 is intended to give you feedback on your design and overall understanding of the project requirements. Your UML diagram will be used to understand your OOP design, while the tests will be used to understand how you expect the program to function. Our focus will be on the changes you have made; you will not be marked on parts of the project which are carried over from earlier versions. Because the individual project is less structured than the group project, no model solution will be provided. You can make use of the feedback you receive to improve your work between Submission 1 and the Summative submission.

Make sure that your file is named <UCID>.zip - for example, if your UCID is 12345678, your submission will be 12345678.zip.

Tip: We recommend that you use a private repository hosted on a public platform like GitHub or GitLab to develop your code. ***Do not use a public repository, as this could allow someone to copy your work!*** If your work is copied, this could result in academic misconduct sanctions even if you did not intentionally share the code.

Submission 1

Submit a zip file. Your zip file should contain:

- A UML class diagram (PDF)
- Unit tests (Java)

UML diagram

Submit a UML diagram based on the revised project requirements. The UML diagram should show class names, visibility, attributes, data types, default values, cardinality and class relationships. You should show all exceptions being thrown at the method (not class) level. You can depict the database as a rectangle with a "use" relationship drawn from any class which directly interacts with it. You do need to include any class or classes which only implement `main()`. The diagram should not be hand-drawn and must be legible.

Tip: Remember that you cannot guarantee that a class will only be used as depicted in your design. For this reason, each class needs to enforce any constraints related to data validity, and cannot expect that it will always receive correct data.

You may find Lessons 16, 18, 19, and 20 particularly helpful in completing this submission.

Your diagram will be treated as an artifact which demonstrates your OOP design. This means that it will be evaluated based on the extent to which it adheres to OOP principles. Diagram correctness is essential for this evaluation. If the diagram is incomplete (e.g., does not include visibility information), inconsistent (e.g., arrow directions do not align with data members), or illegible (e.g., spanning multiple pages with overlapping arrows), we will not be able to assess your comprehension of OOP design principles and you will not receive points for any design aspect we were unable to evaluate.

Unit tests

Submit Java unit tests for the classes and methods depicted in your UML diagram. The tests should adhere to best practices for testing and be valid Java code. You may make use of the two external testing libraries introduced in class (hamcrest and junit-4.XX). Your code should all be part of the `edu.ucalgary.oop` package. There should be one test file associated with each class.

You may find Lessons 5, 14 and 23 particularly helpful in completing this submission.

Your tests will be evaluated both for your adherence to testing best practices at the individual test level and the extent to which they demonstrate an understanding of the edge cases and boundary conditions related to features.

Summative Submission

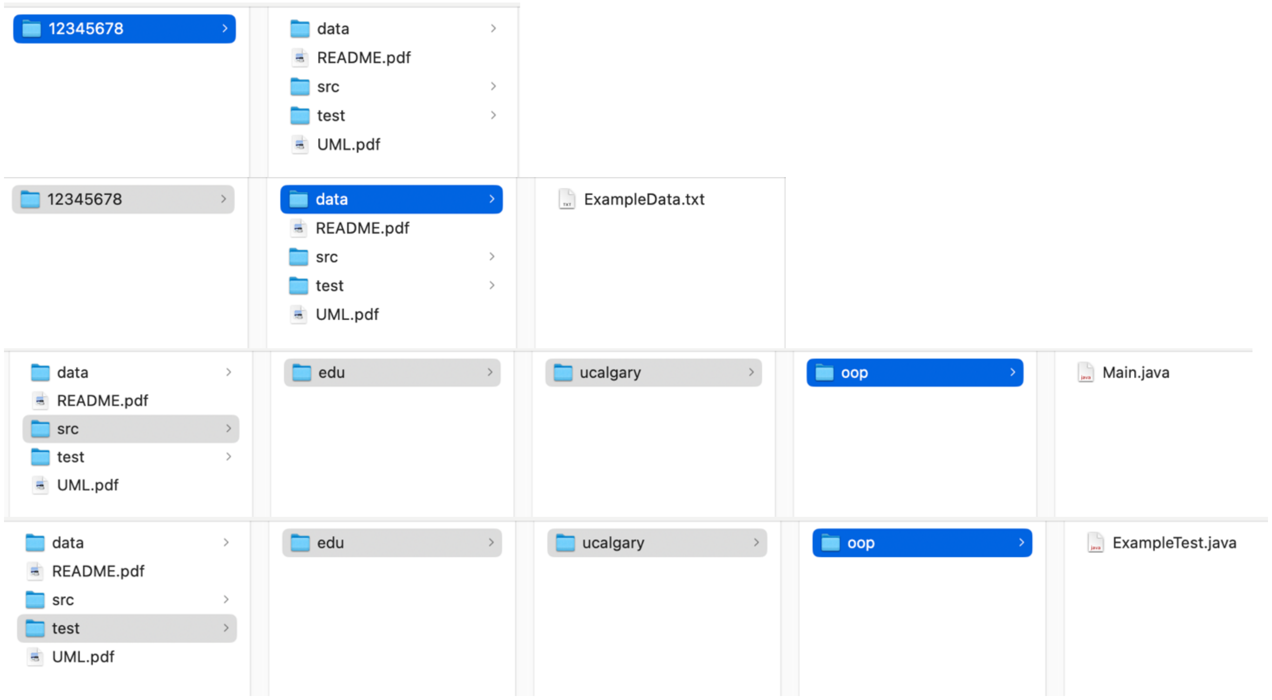
Submit your complete project as a zip file. Your zip file should unzip to contain a directory with your UCID, which should contain:

- The complete program, in `src/edu/ucalgary/oop/` with the executable `main()` in `Main.java`
- Unit tests, in `test/edu/ucalgary/oop/`
- UML diagram, in the root directory and named `UML.pdf`
- A statement about implementation, in the root directory and named `README.pdf` (optional)
- Any files which are needed to run the code, in `data/`

Use the standard directory structure and names we have provided to ensure that the TA focuses on your program's content, rather than on finding components. Below are a series of screenshots showing the directory structure, for a student with a UCID of 12345678.

Do not include any extra/unnecessary files such as files output by your application, .class, .sql or .jar files. We will use our own .sql file with test data to test your application.

Refer to the rubric for information about how your final submission will be assessed.



UML diagram

The UML diagram should adhere to all the requirements described for Submission 1. You may opt to automatically generate the final version of the UML diagram from your code using a tool such as an IDE extension. However, please review any generated diagrams for completeness and correctness; some tools may not include certain aspects (e.g., cardinality, default values or visibility).

Your UML diagram will be used to assess your understanding of OOP design. It will be evaluated against *all* features, whether you implement them or not. If you do not implement all features, your design should nonetheless show how you intended to implement them.

Code and tests

Your code should all be part of the `edu.ucalgary.oop` package. It should be possible to generate developer documentation by running `javadoc`. It must be possible to compile your code and to compile your code against your tests on the command-line using Java 21. You may not create dependencies on any libraries outside of the java.* namespace other than the ones introduced in this class (PostgreSQL connector, junit-4.X, hamcrest and javax.* libraries included in the standard (SE) Java distribution). The code should include only relevant includes (e.g., non-test files should not have a dependency on test libraries). Each public class must be placed in its own file.

You should not add, remove, or rename database tables or fields. We will use our own database data to test your program. Database access should be mediated through the 'oop' account with 'ucalgary' as the password.

File access should be based on a system-independent relative path from the working directory (i.e., `../data` is the file directory from the working directory of `src`).

Your tests will be used to assess your understanding of test coverage, code coverage, and test design.

Your code will be used to assess your ability to write Java in order to implement a program. **If your code cannot be compiled and run (with a main), you will receive zero for all code-related assessment criteria.** No partial credit will be given based on a manual evaluation of the code.

Your developer documentation assessment will be based on javadocs generated from the source code (excluding tests).

Statement

If you wish to provide any information to the TA to assist with understanding your program, do so in a statement (README.pdf). In particular, if your UML diagram differs from your implementation (for instance if you did not implement all features), it should be stated here.

Files

Include any files which your program expects as input. When a feature defines a standard file format, we may replace your data with data of our own. Do not include any output files your program creates.