ENSF 380 WS 2024

Individual Assignments

Submissions 1 (Week 9), 2 (Week 10), and 3 (Week 13)

Instructions

Work independently on these submissions. You may incorporate the code which your group developed for GP #3, use the model solution provided by us, or start from scratch. You may use the tests provided by us (for GP #3) as a starting point. You will need to modify the code/tests to accommodate the revised requirements.

Make sure that your file is named <UCID>.<extension> - for example, if your UCID is 12345678, your submission will be 12345678.pdf or 12345678.zip, depending on the submission.

Each deliverable will be graded using a rubric. You must satisfy all of the criteria in a particular category to earn that category's minimum grade. The grading criteria can be found on the dropbox rubric.

Submission 1

Submit a UML diagram based on the revised project requirements.

You may find Lessons 10 and 12 particularly helpful in completing this submission.

You do not need to represent the database in your diagram, but you should include any classes or methods which connect to the database. You should show all exceptions being thrown at a method (not class) level.

You do need to include any class or classes which only implement `main()`. **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 are expected to make use of object-oriented principles in your design. Refer to the rubric for more information. To receive full marks, your design will have to include an interface and inheritance.

Tip: Make sure that your methods are performing a limited number of tasks so that you are able to provide good test coverage in Submission 2. Using helper methods can make your code easier to test.

Note: Submission 1 and Submission 2 will be marked at the same time.

Submission 2

Submit a zip file containing unit tests for the classes and methods depicted in your UML diagram.

You may find Lessons 18A, 18B, and 18C particularly helpful in completing this submission.

Your code should all be part of the `edu.ucalgary.oop` package. There should be one test file associated with each class.

You do not need to provide tests for functionality directly associated with database access as mock databases are beyond the scope of this course. **Tip:** Make sure that most of your design can be tested without a database by separating database access from other logic.

Tip: We recommend that you use a private repository hosted on a public platform like GitHub or GitLab to develop your code. In the event of any suspected copying, this will allow you to demonstrate when you developed the code. Do not use a public repository!

Note: Submission 2 will be compared with the UML diagram you included in Submission 1. If you defer Submission 1, you may opt to provide a UML diagram with Submission 2, to aid us in understanding your tests.

Submission 3

Submit your complete project as a zip file. Your zip file should contain:

- The complete program, in the package folder structure
- Unit tests, in the package folder structure
- UML diagram (uml.pdf)
- User documentation (README.pdf)
- Grading statement (statement.pdf - use template provided)
- Any files which are needed to run the code

Where we have provided specific names for files, please adhere to these names. Using uniform names reduces the time the TA needs to identify deliverables. Please ensure your code is in the package folder structure so that your code does not need to be manually prepared for compilation.

You may revise your unit tests and UML diagram based on feedback you received for submissions 1 and 2. You may opt to automatically generate the final version of the UML diagram from your code using a tool such as an IDE extension.

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. You may not create dependencies on any libraries outside of the java.* namespace other than the ones introduced in this class. 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.

If your code cannot be compiled and run (with a main), you will receive zero for all code-related assessment criteria in the rubric.

If your program includes database access, you should not add, remove, or rename database tables but use the ones provided by us. Database access should be mediated through the 'oop' account with 'ucalgary' as the password.

The user documentation must explain how to run the program. It should allow someone with no knowledge of your project to use the program.

Your grading statement should be brief. In it, you should state what band you aimed to achieve for your program, and a few bullet points on why you feel you achieved that mark. This should be done in reference to which optional requirements were implemented.