The University of Saskatchewan

Saskatoon, Canada

Department of Computer Science

CMPT 270– Developing Object-Oriented Systems

Assignment 3

Date Due: May 26, 2024, 6:00pm

Total Marks: 60

General Instructions

- Assignments must be submitted using Canvas.
- Programs must be written in Java.
- VERY IMPORTANT: Canvas is very fragile when it comes to submitting multiple files. We insist that you package all of the files for all questions for the entire assignment into a ZIP archive file. This can be done with a feature built into the Windows explorer (Windows), or with the zip terminal command (LINUX and Mac). We cannot accept any other archive formats. This means no tar, no gzip, no 7zip, no RAR. Non-zip archives will not be graded. We will not grade assignments if these submission instructions are not followed.

Question 1 (10 points):

The objective of this question is to demonstrate correct use of git and adherance to an iterative, text-driven design process.

Your Tasks

You should have already cloned your git repository in Assignment 2. you will be using the same IntelliJ project in the same repository, all you need to do is create a module called Assignment3 and do your work for this assignment there.

You should start by creating a tag to indicate when you started the assignment. The following command will do this:

```
git tag A3
```

This will make it easy for instructors and markers to navigate your repository for purposes of marking and providing assistance, as well as simplify generating your gitlog at the end.

Generating GitLog

When generating your gitlog, you should only include the commits that are relevant to this particular assignment. There are a number of ways you could do this (e.g., create a separate branch for each assignment and generate a git log for the appropriate branch). If you tagged the repository before starting the assignment (see above section), you can print all commits since making the tag with the following command:

```
git log A3..HEAD > gitlog.txt
```

This command will generate a git log containing all commits after the A3 tag, and the output is redirected to a text file named `gitlog.txt`.

Expectations for Iterative Development

Your git log should demonstrate evidence of iterative development (i.e., document your work step-by-step). To get full marks here, your git log should demonstrate the following process (see Lecture 07 slides for more information):

1. Start by reading through the assignment and understand everything you are being asked to do. Ask questions and clarification early to ensure you receive an adequate response (nothing to commit here, but this is important)
2. Design your system (normally this would be creating a UML class diagram, but it is provided for you in this assignment, so you can skip this step this time)
3. Create an empty java class for each class in the UML diagram
4. Declare the attributes defined in the UML diagram in the appropriate java classes
5. Create a stub method for each method signature in the UML diagram (including appropriate javadoc comments)
6. Write regression tests for each class, as required
7. Iteratively implement one method at a time, running the regression tests until all tests pass.
8. Perform system-level testing to ensure everything is working correctly
9. Reflect on your process.

Evaluation

10 marks : git log submitted illustrates student followed a reasonable iterative process when completing the assignment.

Files to Submit

A text file called `gitlog.txt`

Question 2 (50 points):

The objective of this question is to practice implementing an object-oriented system and aligning your system with provided UML specifications. Also, practice with creating Java documentation and regression testing of classes.

Your Tasks

In the previous assignment, you implemented a program to calculate your CMPT 270 grade, using a procedural programming design. Also in the previous assignment, you redesigned this as an object-oriented system. For this question, you will implement this system, **using the provided UML class diagram**. The requirements for the system remain the same as was expressed in Assignment 2, see the assignment description or solution for Assignment 2 for a refresher on the requirement details, and implementation hints.

Hints/Tips:

1. You should start by creating a new IntelliJ module for Assignment 3
2. Download the provided GradeCalculator.java file and add it to your Assignment3 module's src/ directory
3. In the module's src/ directory, create a class for each class in the provided UML diagram (except GradeCalculator)
4. Implement all class variable/attributes
5. Create a stub method for each method indicated
6. Set up regression tests for each class that requires it.
    - Note that you only need to write tests for methods that are **implemented** in a class.
    - If a method is inheritted, assume it was already tested in the superclass.
    - Testing toString() is not required. If you want to test it, printing out the string for visual review is sufficient.
    - If a class only has a constructor and toString() method, there is no need to write a regression test for this assignment
    - Given the above information, **only 3** classes require regression tests
7. Implement one method at a time, running the regression tests as you go to ensure your implementation is correct
8. Perform System testing by running the provided GradeCalculator class
9. If everything works, submit your files, as per the submission process detailed below

Note: Your solution in the previous assignment is most likely different than the UML diagram provided for this assignment. Your solution **must** adhere to the provided diagram, **not** your solution from Assignment 2. Any deviation from the provided UML diagram will result in a deduction of marks.

Evaluation

10 marks : Files all compile and system runs without errors
10 marks : Adequate regression testing
8 marks : One mark per class for correct alignment with UML specifications (attributes and methods)
8 marks : One mark per class for adequate javadoc documentation
4 marks : Correct use of arrays in Course class
4 marks : Adequate use of inline documentation
4 marks : Correct use of inheritance, as indicated by the provided UML class diagram
2 marks : Identifying information at the top of each file

Provided Files

gradeCalculator.pdf : containing a UML class diagram of the system you are to build
GradeCalculator.java : a java class containing the main driver program for the system

Files to Submit

Your submission should include a .java file for each of the classes that are in the UML diagram (except for GradeCalculator.java, as this file was provided). There is no external documentation or UML diagrams required for this assignment.

Please ensure you have included identifying information at the top of every .java file in the form of a comment (Name, nsid, student number, etc).

Ensure the files you submit are .java files from the src/ directory in IntelliJ. Any submitted .class files cannot be opened by the marker and therefore will result in grades of 0 where relevant in the rubric.

Assignment Summary

Files Provided

gradeCalculator.pdf
GradeCalculator.java

What to Hand In

You must submit the following files, compressed in a .zip file named A3-abc123 (where abc123 is your NSID):

gitlog.txt
Course.java
Exam.java
Midterm.java
LabExam.java
FinalExam.java
Assessment.java
Quiz.java
Assignment.java

Grading Rubric

Evaluation information can be found for each question above. See Canvas for the complete grading rubric.