The University of Saskatchewan

Saskatoon, Canada

Department of Computer Science

CMPT 270– Developing Object-Oriented Systems

Assignment 2

Date Due: September 27, 2024, 6:00pm

Total Marks: 42

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.

Git Requirements

- It is the expectation that each assignment is stored on the department git server (www.git.cs.usask.ca)
- Additionally, it is expected that students follow an iterative development approach and demonstrate this through consistent use of git (assignments will specify the main steps, each of these steps must include a commit indicating completion of the step)
- In most assignments, you will submit a .txt file containing your git log for the duration of the relevant assignment
- Instructors, TAs, and markers may clone your repository to verify your git log is correct and investigate whether the content in the git log matches what was actually done in the assignment
- This may be done as a random spot check, or in response to suspicious git logs (i.e., all commits were made within a 5 minute window)
- If you email your instructor with questions regarding a bug or strange behaviour, your instructor will likely clone your repository to investigate the issue - it is important to keep your repository up to date!

Question 1 (5 points):

The objective of this question is to demonstrate you have set up your git repository and are able to make commits and generate a git log.

Your Tasks

A git repository **should** have been created for you on www.git.cs.usask.ca (if it hasn't been, you will need to go there and log in as soon as possible so we can run a script to create the repository for you).

Some general steps to set up your repository for the term:

- Clone the repository to your development machine (either a machine in Spinks or your home computer with git installed and configured)
- Inside the repository, create a new IntelliJ project (see Tutorial 1 for detailed instructions)
- For **each** assignment, create a new module inside that IntelliJ project with the naming convention `AssignmentX`, where 'X' is the assignment number.
- All assignment work should happen inside the corresponding module (you can just add pdfs for UML diagrams into the appropriate module folder in your project)

The process of setting up git has been covered in previous courses, but if you encounter any issues setting this up, please attend a TA help desk as soon as possible to get things sorted out. The requirements for Assignment 2 are fairly light but git will be a much more significant component in future assignments.

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), but the simplest way is probably to limit the commits to those made after a particular date (likely the date you submitted the previous assignment). This can be done with the `-after` option, and specify the date after which commits should be included. On command line, the command should look something like this:

```
git log --after=''2024-09-01'' > gitlog.txt'
```

This command will generate a git log containing all commits after September 1, 2024, and the output is redirected to a text file named `gitlog.txt`. You will want to customize the date for a particular assignment, and you may want to adjust the name and path of the text file you are redirecting the output to.

Evaluation

5 marks : git log submitted as evidence that git repository was correctly cloned, and at least some commits were made indicating progress on assignment

Files to Submit

A text file called `gitlog.txt`

Question 2 (12 points):

The objective of this question is to practice creating UML class diagrams using the draw.io application.

Your Tasks

You will be creating a complete UML class diagram for a simplified version of the video game MarioKart (see system description below). Your diagram should include all of the main classes mentioned in the system description, as well as appropriate UML notation indicating how the classes are related in the system. You should also include any obvious attributes and methods for classes, complete with visibility level and data types.

System Description

You will be modelling a simplified version of the popular video game series, MarioKart. This version of the game will have the following features:

- There is a Player/Kart that drives around a track, racing against other Player/Karts (for the purposes of this question, you can use Player and Kart interchangeably as the same entity
- Each Kart contains a unique set of tires, as customized by the player. These tires can have unique effects on the Kart (e.g., higher traction, faster speed, etc). It is up to you which stats the tires affect, but you should include at least 2 unique modifiers.
- Each Kart contains a unique glider to be deployed when the Kart hits a ramp. These gliders can have unique effects on the Kart (e.g., faster glide speed, longer glide time, etc). It is up to you which stats the glider affects, but you should include at least 1 unique modifier.
- Karts can collide with item boxes on the track to get items and powerups.
- Colliding with a Hazard will make a Kart stop and briefly spin out. There are two types of Hazards: Bananas which will sit on the track unmoving, and Shells which will slide on the track, bouncing off walls until they hit a Kart.

Evaluation

3 marks : Choice of classes
3 marks : Choice of class attributes
3 marks : Choice of class methods
3 marks : Relationships between classes

Provided Files

There are no provided files for this question.

Files to Submit

You should submit your completed class diagram in a file called marioKartClass.pdf

Question 3 (20 points):

The objective of this question is to practice programming in the Java programming language in a non object-oriented context (procedural). Specifically, this question will give you some practice using conditionals, loops, console I/O, and arithmetic expressions in the Java programming language.

Your Tasks

For this question, you are tasked with creating a Java program to calculate your final grade in CMPT 270. The general design of how you structure your code is up to you, but the program must meet the following criteria:

- All code should be written in the `main` method in a class called `GradeCalculator.java`
- All input/output should happen through the console, using the Scanner class for input (as demonstrated in class) and System.out for output
- When the program starts, it should ask the user to enter the name of the course and the name should be saved as a variable
- The program should include a looping menu, as discussed in lecture. This menu should have options for the user to add marks for an assignment, add marks for a quiz, add the midterm mark, add the lab exam mark, add the final exam mark, and quit.
- When the user selects quit, the program should print out the name of the course and the calculated final mark
- If a grade was not entered (e.g., the final exam) the program should assume the mark for that assessment is 0
- The weighting of assessments should align with the weighting from the syllabus, but the weighting can be hardcoded into the program. (Note that assignments and quizzes are not weighted equally, so you will need to account for that in your calculation)
- The user will enter assignments and quizzes individually, but the program does not need to keep track of each individual assignment grade. You can if you wish, but we haven't covered how to do that in Java yet. It is possible to keep track of a running total of assignment marks and totals

Evaluation

1 marks : Correct use of Scanner class for input
1 marks : Correct use of System.out for output
1 marks : Program stores the course name, read in from user
2 marks : Program correctly reads and stores assignment grades
2 marks : Program correctly reads and stores quizz grades
3 marks : 1 mark each for reading and storing Midterm, Lab Exam, Final Exam
3 marks : Program correctly calculates grade using syllabus grading scheme
3 marks : Robustness - any missing data should be treated as 0 (including assignments and quizzes)
2 marks : Program utilizes a looping console menu correctly (including the ability to quit)
1 mark : Program correctly prints out course name and grade when quit option selected
1 mark : Documentation: program contains a reasonable amount of inline comments, as needed

Provided Files

There are no files provided by instructor for this question.

Files to Submit

Your submission should include a file called `GradeCalculator.java` with your program written in the `main` method.

Question 4 (5 points):

The objective of this question is to practice object oriented design and creating UML class diagrams in a problem you are familiar with.

Your Tasks

In the previous question, you wrote a procedural program to calculate your final grade for CMPT270. For this question, you will take that procedural program and restructure its design into an object-oriented system. For this assignment you will be creating a UML class diagram showing your design. In the next assignment, you will be actually building this system.

- Start with your solution to the previous question. Currently, the program is written in a procedural way; all of the variables and functionality is written in a single method in a single class. Go through your code and identify a reasonable way of grouping your data into classes.
- Next, identify the methods for each class that are needed to get/set the data.
- Finally, determine the interactions between the classes in your system. Which objects are contained by other classes? Which classes need to talk to another class? Is there an inheritance structure (it is not required, but you may use inheritance if it makes sense).
- Create a UML class diagram showing the outcome of the previous three steps. You only need to submit the diagram, the previous steps are used to help structure the process of creating the class diagram.

Evaluation

1 mark : for classes
1 mark : for attributes
1 mark : for methods
2 marks : structure (correct arrows between classes, correct formatting of classes)

Provided Files

There are no files provided by instructor for this question, but you will be using your solution to Question 3 to inform your process.

Files to Submit

Your submission should include a file called `gradeCalculator.pdf` with your finished class diagram.

Files Provided

You are provided a file called `financialUseCase.drawio` to use as a starting point for Question 1.

What to Hand In

You must submit the following files, compressed in a .zip file named A2-abc123 (where abc123 is your NSID):

gitlog.txt
marioKartClass.pdf
GradeCalculator.java
gradeCalculator.pdf

Grading Rubric

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