CPSC 219: Introduction to Computer Science for Multidisciplinary Studies II

Project – Final Submission Guidelines

Weight: 5+15%

Final Code Base Submission (75 pts) – 15%

- Program is a JavaFX event-driven object-oriented GUI (20)
  - Program is launched from main() but main() mostly just launches GUI Scene
  - Has well developed .fxml created in SceneBuilder (5)
  - Has JavaFX Controller event-driven design pattern (5)
  - Has working options to trigger add data menu commands (5)
  - Can view all data (1)
  - Can view 4 special options (4)
- Program is object-oriented (10)
  - Helper functions can be static
  - However data must be stored in OO data structures
    - (java data structures like Arrays/ArrayList/HashSets/HashMaps can be used, but the majority of data storage should be in objects, with these just used to organize them)
  - proper use of inheritance, abstract, protected, public, private
  - proper use of final (constants)
  - proper use of static
  - proper use of interfaces, enums
  - proper usage of toString
- View data (10)
  - [2] The stored data can be viewed in its regular stored state (ex. a table of entries)
  - [8] Four new features of data can be accessed. For example, different sorted views of the data. A summation of data features. A prediction of future data. A warning that something was missed or that the rate of something is too low.
- Save/load data to file exists (5)
  - Program should have GUI menu option which will save data to a (comma separate value) .csv file
  - Program should have GUI menu option which will load data from a (comma separate value) .csv file
  - Program should be able to be run with command line argument that will start program using previous data saved to a (comma separate value) .csv file
- Proper usage of hashCode, equals, compareTo (5)
  - Need to be able to demonstrate how completing at least one of the above gives you a feature to your objects that you use
  - Like sorting an ArrayList, or using a HashMap with an object of your own

- **Have a sketch of planned GUI [this doesn't have to be your final GUI] (out of 2.5)**
    - draw.io useful website for this
- **Have a UML sketch of FINAL program structure (out of 2.5)**
    - Does not have to give internal details about internal java classes
    - Make a class box diagram for each data object and gives details for those
    - Make a connection diagram between the project class name boxes with the arrows to show association/composition/aggregation/inheritance/etc.
    - draw.io useful website for this
- **Deployment (out of 5)**
    - **Have a jar file**
        - **Package up program into executable .jar file**
    - **Have a readme.md**
        - **Readme.md explains how program (.jar file) is executed (run from cmd line)**
    - **Java a GUI 'about' option**
        - **About menu indicates contact info and other important program info in GUI**
- JUnit Testing (out of 5)
    - Unit tests for (non-input/non-output/non-GUI) functions
- Gitlab Usage (out of 5)
    - Gitlab account exists, private project exists, at least 1 commit, small commits, regular commits
- Style/Commenting (out of 5)
    - Name/Date/Tutorial, Functions commented, Javadoc, Inline commenting, doesn't use inline conditionals, limited magic numbers, don't change function names, don't change filenames, etc.
    -
- Partnership penalties
    - During the demo the TA will at times ask different members of your group (partnership) to describe how something works. Each partner should expect to have contributed to unit testing, git commits, commenting, and program functionality. In general, the penalties will be
        - -5 Partner is judged to have not contributed in one area (ex. javafx, git, commenting, code)
        - -10 Partner didn't contribute in two areas
        - -15 partner didn't contribute in three areas
        - -20 partner didn't contribute in all four areas
    - Contribution penalties are recorded separately for each student. Judgement is made by TA on basis of students being able to explain something about part of code that is being viewed in more detail than just re-iterating the readable syntax.

Your project should be packaged up into a zip file that is submitted into D2L dropbox. This submission should contain your complete source code, as well as the requested sketches, readme.md, .jar file, and link to your final project git. Your TA/instructors should both be added to your repository.

Reflection (25 pts) – 5%

- Individual report in which student demonstrates their experience during project.
  1. What design decisions did you make that you consider unique?
  2. Were these design decisions successful?
  3. What would you do differently in design next time?
  4. Other features you would consider adding?
  5. What did you learn from Git/JUnit/JavaFX and working with a partner to make code?
  - Please consider your reflections from the point of view of what you learned as you moved from procedural design into to OO-design and then, finally an event-driven GUI layer on top of OO design.
  - Limit your reflection to one pdf page submitted in D2L.