PROG 24178

Project Requirements

1. Project Overview

Our PROG 24178 project is a group project in which two students collaborate equally to create an object-oriented program of medium complexity. It should be a GUI program that uses principles, best practices and technologies learned in the course. Certain elements are a **required** part of your project, for example it must implement a GUI of reasonable complexity, use a dynamic data structure like ArrayList, Has multiple classes with defined class relationships, exception handling, and use file input and/or output (further details below in "General Requirements"). The rest is up to you.

Each student group will choose their own idea for their application and submit a project proposal outlining their idea. For the GUI portion of the project, you will use the JavaFX platform. You can design your GUI graphically with SceneBuilder, or implement it dynamically in code. This choice, along with other detailed plans, must be clearly stated in the project proposal.

Above all your project should be interesting and fun, so try to choose an idea you find interesting or potentially useful. Don't make your project too big or ambitious, remember it should take about as much time as two or three assignments. If you feel your project may be too big then you could simplify certain portions or leave parts for future implementation. For example, a GUI button could display a message "Not yet implemented". After you are done, you may want to add your project to your SLATE ePortfolio (Sheridan co-curricular record) to show off to others.

2. General Requirements

1. **GUI Portion:** The project will consist of at least 2 GUI windows or dialogs (minimum one per team member). In JavaFX this means at least 2 stages, where each stage has its own separate FXML file and controller class. Error messages or simple confirmation dialogs don't count.
2. **Non-GUI Portion:** Minimum one class per team member – so minimum two back-end (data management) classes, not counting GUI classes or controller classes.
3. **File I/O:** Either file input, output, or both. For example, a program that manages information about customers of a business could store the customer information in a file and then re-load the file next time your run the program.

4. **Data structures:** ArrayList (dynamic array) or HashMap. For example a multi-player game could use an ArrayList of Player objects to store information about all the players. Often (but not always) you may use a dynamic data structure like ArrayList to store information you read from a file. Don't use regular Java arrays.
5. **Exception Handling:** For full marks your project should contain enough exception handling and error recovery so that bad user input (or bad input files etc.) don't crash the program. You should display a user-friendly error message instead.
6. **Commenting:** The code shall be commented as follows.
    a. Each Java class shall have a header comment which lists the principal author(s) of the code and briefly describes what the code in the file is for, using JavaDoc commenting style.
    b. Each method and field shall be commented using JavaDoc style comments. For methods include the purpose of method and a brief description of each parameter. For fields give the purpose of the field and units if applicable (e.g. length in meters, price in dollars).
    c. Code inside methods shall be commented as needed (e.g. tricky steps) using non-Javadoc style comments.
    d. FXML files do not need to be commented.
7. **Coding standards and quality:** All code shall follow our coding standards (as posted on SLATE). Variable names should be descriptive. Classes should be well designed and well structured, with use of object-oriented techniques such as encapsulation, inheritance, abstract classes, and interfaces where appropriate. Programs should not use repetitive code and should use field variables only where necessary. Programs should not define variables or methods that are never used; logic should be written concisely and not cluttered with unnecessary code.

3. Project Deliverables

Each student group will hand in/perform the following deliverables for their project. Only **one** team member needs to submit any given item in SLATE, I don't need two copies. Both team members must contribute equally on all deliverables, otherwise marks may be deducted from the non-contributing team member. The project mark breakdown is given below.

1. **Project proposal**, due in week 8. Worth 10% of the project grade. Should be done using Microsoft Word (hand in **one** .docx file only, no additional files. If you wish to use a different document format you must get prior approval from your professor. Diagrams can be done separately but must be inserted into your main report (no separate files or loose pages). *Your professor must approve your proposal before you continue with the other parts of your*

project. The project presented and handed in must match the approved project proposal. Please see "Project Proposal" below for additional details.

2. **Project Interim code demonstration,** in week 11. Worth 20% of the project grade. Source code of the project submitted with 50% of the overall program functionality implemented. Please see 'Project Interim code demo' below.
3. **Project presentation**, in week 13. Up to 40% deduction off other project deliverable marks for a poor/missed presentation. Please see "Project Presentation" below for details.
4. **Project report**, due in week 13. Worth 30% of the project mark. Should be done using Microsoft Word (hand in **one** .docx file only, no additional files). If you wish to use a different document format you must get prior approval from your professor. Please see "Project Report" below for additional details.
5. **Program source code**, handed in the same time as the report. Source code and program functionality is worth 35% of the project mark. I must be able to compile and run your program. For details see "Project Submission", below.
6. **Generated JavaDoc documentation**, handed in along with the source code. Worth 5% of the project mark. For details see "Project Submission", below.

4. Project Proposal

Prepare and submit a proposal document for the project with the following structure. The document should be created using Microsoft Word (hand in one .docx file using SLATE, no additional files), or on paper (hand in to your professor in class).

1. A cover page with the project title and all group member names.
2. Project Overview. A few paragraphs describing the purpose of the application, the development environment (VSCode, Java 17, etc.) and the technologies used (JavaFX, SceneBuilder, file I/O, collections, graphics etc.) Your description **must** make clear how you will be incorporating the required elements:
    a) File I/O – What information will you save to files on disk, and/or what information will you load from files? Detailed file format information is not required.
    b) Dynamic array or hash map – What information will be stored in a dynamic data structure? How will it be managed?
3. User Interface. Provide one drawing for every window (stage/activity/dialog) you envision to be part of your project. You can draw it on paper, draw using Visio, or you can design it in Scene Builder and provide a screenshot.
4. Project Class Structure. Use UML class diagram(s) to present the rough class design you envision for the application. You should include all significant classes and their relationships (arrows). Fields and methods are **not** required to be listed. Diagrams can be done on paper, or with a UML drawing tool like StarUML or Visio (UML Model Diagram template). It must be clear which classes are GUI classes and which classes are non-GUI (data management) classes. For JavaFX all controller classes should have names ending with "Controller".

Although each project will be unique, all JavaFX projects will have
- FXML files for the GUI layout (one file per window)
- Controller classes for the GUI event handlers (one per window)
- Data management classes (typically two or more)

When designing a larger application like your course project avoid putting data management code in the controller (GUI) classes.

5. Division of responsibilities. To the extent possible, list the planned division of responsibilities, e.g. who will work on which classes, who will design/implement the GUI, etc. Work should be divided between group members as evenly as possible – one group member should not do most of the work!

The proposal is not final. If it needs to be modified to be accepted, I will let you know. During development certain functionality may be changed, modified, added or removed. Your original design may change – this is a part of a normal development process. The proposal must be updated and resubmitted if there are any major changes. The professionalism of your submission, and clarity of written communication is important. If your proposal is incomplete or missing any major element by the project due date the maximum mark you can receive on the proposal is 6/10.

6. Project Interim Code Demo

This demonstration of the project will be done to the course professor only and will include at least 2 of the major functional requirements needed for the project as given in the proposal. This will also include submission of the project code in SLATE as per the Project Submission guidelines given below.

7. Project Report

The project report structure and format is similar to the proposal. It should include:

1. A cover page with the project title and group member names
2. Final project overview and description, including a description of how dynamic arrays and file I/O were used
3. Actual GUI screen snapshots for every window (stage/activity/dialog)
4. Final UML class diagram(s) including all classes you implemented and their relationships (fields and methods are **not** required to be listed)
5. A separate conclusion section **for each group member** describing important lessons you learned, difficulties you encountered, things you would do differently in the future etc.

The project report sections should be written by the same group member who did the work described. The professionalism of your submission, and clarity of written communication is extremely important.

6. Project Presentation

The project presentation must be performed jointly by both group members. In the presentation group members should:

1. Describe the purpose of the application.
2. Present a high-level overview of windows (stages/activities/dialogs) and classes used in the application.
3. Demonstrate the application functionality. Each student in the group should demo the functionality he/she worked on.
4. Briefly describe the technologies used in the application (as per the General Requirements given above).
5. Describe any difficulties you had to overcome when developing your project.
6. Briefly describe how your project is different from your proposal (Note: remember to submit an updated proposal with any significant changes).

Don't worry, the presentation is a fun chance to show off your work!

7. Project Submission

Your submission must follow all the project requirements outlined above. Each group should work independently. There must be no excess collaboration (copying) between groups or between members of different groups. Submit your project using SLATE (click on the Dropbox tab, then click on the Project link).

ZIP the entire VSCode project folder. Do not use other archive formats like RAR. Attach the ZIP file to your SLATE dropbox. Also attach your **project report Word document**.

Your project must be submitted before the due date/time. Any projects submitted after this time are considered late. See our class plan on SLATE for my rules governing late assignments/projects. Your project presentation **must** be given in class on the specific day it is scheduled.