Project #3 – 120 points

Due Date – Monday, March 24 at 11:59 pm. (Spring Break is March 15 ~ March 23, 2025)

Submission

(1) Zip the project folder and submit the zipped file to Canvas.
(2) The zip file must include the following grading items. [**100 points**]
- All Java source files in the "**src"** folder from Project 2, **EXCLUDING** TransactionManager.java and RunProject2.java.
- Three JavaFX-related files, which replace the TransactionManager.java in Project 2
  - **Main.java**, which contains the **main()** with **launch()** method to run your project.
  - **Controller.java**, which contains the event handlers.
  - **view.fxml**, the XML file containing the scene graph.
- Test Specification for the functional testing. [**15 points**]
- Javadoc folder. [**5 points**]

(3) The submission button on Canvas will disappear after **March 24, 11:59 pm**. Do not wait until the last minute to submit the project. You are responsible for ensuring the project is well-received in Canvas by the due date and time. **You get 0 points** if you do not have a submission on Canvas. **Projects sent through emails or other methods will not be accepted.**

Project Description

Your team will revamp the interactive user interface in Project 2 and develop a GUI (graphical user interface) with JavaFX. The GUI shall replace the TransactionManager class in Project 2. Revamping the user interface does not change the functionality of the software. With the new GUI, the software shall perform the same functionality specified in Project 1 and Project 2.

Project Requirement

1. You MUST follow the Coding Standard posted on Canvas under Modules/Week #1. **You will lose points** if you violate the rules listed in the coding standard.
2. You are required to follow the Academic Integrity Policy. See the **Additional Note #14** in the syllabus posted on Canvas. If your team uses a repository hosted on a public website, you MUST set the repository to private. Setting it to public is considered a violation of the academic integrity policy. The consequences of violation of the Academic Integrity Policy are: **(i) all parties involved receive 0 (zero) on the project, (ii) the violation is reported, and (iii) a record on your file of this violation.**
3. Each source file (.java file) can only include one public Java class, or you will **lose 2 points**.
4. Include all the Java classes from Project 2 and use them in this project, EXCEPT the TransactionManager class, JUnit test classes, and RunProject2 class. You can reuse the code from TransactionManager class in the Controller class, but you CANNOT directly include TransactionManager class as part of Project 3, or you will **lose 10 points.**
5. The grader will use the test cases in Project2TestCases.txt to grade Project 3. If you lose points in Project 2, you must fix the problems, or you will lose points again for the same cause!
6. This project uses the **MVC** (Model-View-Controller) **design pattern**. You must include ONE .fxml file for the "View," ONE controller.java for the "Controller," and ALL the classes from Project 2 as the "Model." In addition, there will be ONE additional Java source file that contains the **main()** method to "launch" the GUI. **You will get 0 points** if you don't follow the MVC design pattern.
7. You shall design the GUI for this project using a combination of JavaFX UI controls. If you use only a TextField and a Button to take command lines, **you will get 0 points** for this project.

8. The GUI must include FileChooser, TextField, Button, RadioButton, TextArea, TabPane, and GridPane, or **-3 points** for each violation.
9. You MUST set a "title" for the GUI (the window/stage) or **-2 points.**
10. You CANNOT **use System.out** (write to terminal) or **System.in** (read from terminal) ANYWHERE in any of the classes, **or you will lose 3 points for each violation, with a maximum of losing 10 points**. All read (input) and write (output) must be done on the GUI. You may change the names of the **print()** methods and their return types to String.
11. You are required to generate the **Javadoc**. Your Javadoc must include the documentation for the constructors and private and public methods of all Java classes (*.java files.)
    - DO NOT need to comment **Main.java**,
    - MUST comment **Controller.java**.
    - DO NOT include the *.fxml files when you generate the Javadoc; those are NOT Java source files.
    - Generate the Javadoc in a single folder and include it in your project folder for submission.
    - You are responsible for double-checking your Javadoc after you generate them. The grader will navigate the Javadoc with the "index.html." You will **lose 5 points** for not including the Javadoc, OR the grader cannot navigate your Javadoc through the "index.html."
12. **Functional Testing**.
    - Design test cases to test the functionality through the GUI. Test the software as a black box, focusing on the I/O behaviors.
    - You MUST create a test specification and design at least 15 test cases, each focusing on a specific testing objective. The **Test Specification** is worth **15 points.** You MUST use the table template in the Coding Standard to organize the test cases, or you **will get 0 points** for this part.
    - Use the test cases in **Project2TestCases.txt** as a reference to design your test cases and run the tests.
    - The GUI shall reject any invalid data and display proper error messages. You will **lose 2 points** for each invalid condition not rejected or each error message not properly displayed.
    - You are responsible for thoroughly testing your software beyond the **15 test cases** required. Your software must always run in a sane state and **should not crash in any circumstances**. The grader will try to produce exceptions while using the GUI. You must handle all exceptions. The software shall continue to run until the grader stops the execution or closes the GUI window. **You will lose 2 points** for each exception not caught.