COSC2391 Further Programming

School of Computing Technologies RMIT University

Assignment 2 - Semester 1 2024

1. Introduction

You are required to implement a basic Java program using Java SE 17 or later. This assignment is designed to:

- Evaluate your ability to develop desktop applications using JavaFX.
- Evaluate your skills in storing persistent copy of data.
- Test your knowledge to implement advanced OO design patterns.

This is an individual assignment. Your final submission is worth 42%; an **in-lab** milestone check in Week 10 is worth a further 3%; a final post-submission interview is worth 5%; giving a total of 50% for Assignment 2.

2. Academic Integrity (more)

The submitted assignment must be **your own work**. No marks will be awarded for any work which is not created by you.

Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students (**or enables such copying)**, the internet, the output of AI systems, or other resources without proper reference. Sometimes, students study and work on assignments together and submit similar files which may be regarded as plagiarism. Please note that you should always create your own assignment even if you have very similar ideas. Plagiarism-detection tools will be used to check all submissions. Penalties may be applied in cases of plagiarism.

3. Background

This assignment builds on Assignment 1. You will build a GUI application for Burrito King Restaurant. The core business logit (e.g., adding orders, calculating preparation time, ~~printing out sales report~~) remains the same as Assignment 1. This time, suppose you are a **customer** of the restaurant, not the manager. A sample solution of Assignment 1 will be provided, which you can use as reference.

4. Task Specifications

Basic functional requirements are listed below.

- The application can have many users.

- Each user can create a profile, with a unique username, password, and first and last name.
- Once the username and password are created, the user can log in.
- Each user is shown a dashboard after login. The dashboard should display the user's first name, last name, and any active order that has not been collected by the user. For each active order displayed here, show an order summary here, including order number, ordered food items, total price, and order status (See below).
- Each user can perform the following actions:
    - Edit profile (change first name/last name/password, username is not changeable)
    - Add food items to shopping basket. The program can guide the user to place the order by selecting food items (burritos, fries, and soda) and specifying quantity. The program allows the user to update the shopping basket (e.g., removing food items, updating quantity) as long as the user has not checked out.
    - Place a new restaurant order. A user will be notified of the total price and the waiting time before placing the order. Once confirmed by the user, proceed to payment, where you collect (**fake**) credit card information (card number, expiry date, and cvv). (Do not use real credit card numbers!) You will need to perform simple validation to check (1) if the card number has 16 digits; (2) if the expiry date is a future date; (3) if cvv has 3 digits. For simplicity, you can assume the payment will be successful. Once an order is placed, a unique order number will be generated and assigned. The GUI should also ask the user to specify a "fake" time at which the order is placed.
    - View all orders. The user can view the following details of all orders: day and time that the order was placed, total price of the order, and status of the orders. The status can be "placed" (await for collection), "collected" and "cancelled". The orders should be displayed in the reverse order of time they were placed.
    - Collect an order. This will move an order status from "placed" to "collected". The GUI must allow the user to enter a "fake" time at which the order is collected. Note that collect-time must be at least order-placed time + time to prepare. An order that has been cancelled can not be collected.
    - Cancel an order. A user can cancel an order that has not been "collected". After an order has been cancelled it can no longer be "collected".
    - Export all orders. The user should be able to export the historical orders to a file. The user should be able to select which orders to export. The user should be able to select the destination of the file and specify the file name. The user should be able to select what order information (e.g., ordered items or price) to be exported. The saved file should be in csv format.
    - Log out.

There are advanced features for VIP users. When a new user is registered, the user is a non-VIP user by default. Once a non-VIP user logs in, the user has the option of upgrading to a VIP user. To upgrade, the application asks if the user agrees to receive promotions:

Would you like to receive promotion information via email?

The user will become a VIP once they agree and provide a valid email address. Then the user can access VIP functionalities:

Please log out and log in again to access VIP functionalities.

A VIP user has the following additional functionalities:

- Ordering a meal (a burrito, one serve of fries, and a soda), with a discount of $3.
- Collecting credits for all orders. The user can earn one credit for each dollar spent.
- Use credits when paying for orders. The program should ask the user if he/she wants to redeem credits when checking out. If yes, ask the user to specify the number of credits to redeem. Every 100 credits can be used as one dollar. If credits are redeemed in payment, the final order price of this order should be recorded as the actual amount paid by user. In other functionalities, the final order price will be used. For example, when "Viewing all orders", user will only see the actual price paid for that order. In "Collecting credits", the user can collect credits for the actual paid amount.

5. Assessment Details

This assignment is structured in three milestones: in-lab milestone check in week 10, full program submission in week 12, and post-submission interview in week 14. It is recommended for you to follow the stages below to incrementally build the GUI application. But you can also plan the work in your own way.

5.1 Setup

You will use GitHub to store program code in this assignment. To get started, accept the assignment invitation by clicking this link: https://classroom.github.com/a/OjLPx4GP . You will need to create a GitHub account if don't have. After accepting this invitation, create a git repository in **FurtherProgramming2410-classroom**. Make your repository **private**. Otherwise, your repository will be considered as plagiarism.

Instructions to create a git repository and upload your project to GitHub can be found in **Work with GitHub and GitHub Classroom.pdf** from Canvas -> Assignment 2. Throughout the program development, you will need to make regular commitments to GitHub to facilitate transparency in the development process.

5.2 Part A (In-Lab milestone – Week 10)

This milestone aims to assess your ability to define classes, establish relationships among them, and create a blueprint for the program's implementation. For this milestone, you are required to design the structure of your program. Specifically, you need to identify all the necessary classes required for your application. For each class, you need to identify key data attributes and methods. You also need to determine the relationships among the classes (e.g., association, aggregation, composition, and inheritance). For this milestone, you will prepare a documentation of your program design with

a clear description of classes and class relationships. While you don't need to strictly follow this design as you progress to later stages, you should propose a program design that reflects your current understanding to the best of your ability at this stage.

5.3 Part B

Following your class design in 5.2 Part A, you will work on the backend part of the application (excluding data storage, which will be covered in Part D). You will implement the key functionalities mentioned in the task specification based on Assignment 1.

5.4 Part C

You will work on the frontend part of the application in this stage. You will design the user interface of the program that serves the program functionality. You will connect the frontend with the key functionalities from Part B, e.g., rendering data in a clean, concise, user-friendly format in the graphical interface.

5.5 Part D (Full program submission – Week 12)

You will connect the front-end and back-end to build a functional program. You will use JDBC to store program data, so that the application can be restarted in the same state it was in the end of the previous execution. You will incorporate OO principles to design and optimize classes.

5.6 Part E (Post-submission interview – Week 14)

You will be interviewed after program submission. During the interview, you will demonstrate the functionality and GUI design of your program. In addition, it is crucial for you to demonstrate your programming skills by clearly explaining how you design the program and implement different functionalities.

6. General Requirements

This section summarizes the general requirements of Assignment 2 in four aspects: (1) GUI; (2) Functionality; (3) Program Design; and (4) Others. Please refer to Canvas -> Assignment 2 -> Rubric for detailed mark allocation.

Important notes: Marks for GUI and Functionality are checked during the post-submission interview. If you do not attend the final interview, you will be awarded ZERO mark for GUI and Functionality. You will also get mark deductions if you cannot EXPLAIN how you implement the functionalities.

6.1 GUI

- Your program should include appropriate JavaFX components to make the application easy-to-navigate. For example, each window should have a window title; menu options (if applicable) are selected from a menu bar.

- Your program should respond clearly to every user's action. For example, if the user types a wrong password, the login window should display a clear error message and ask the user to type again.
- Your program should provide visual consistency. Avoid using different styles and labels for similar elements on different windows of the application.
- The UI design of your program should serve the purpose of the application.

6.2 Functionality

All functions required by specification are implemented correctly covering **different** cases that might happen during the usage of the program.

6.3 Program Design

Appropriate choice of data structures that serves the purpose of the application. Class designs that adhere to the SOLID principles.

- You are required to follow MVC pattern to connect the frontend with the backend.
- You are required to use one of design patterns such as Singleton pattern, besides the MVC pattern.
- You are required to implement at least one interface or abstract class to demonstrate your understanding of OO principles.
- You are required to apply SOLID principles when appropriate in order to enhance the maintainability and extensibility of your program, decrease coupling amongst classes, and minimize code repetition across classes.
- You are required to choose appropriate data structures to enhance the running efficiency of the program.

6.4 Others

- High source code quality with adequately commented and properly indented codes and appropriate class/method/variable names.
- Regular commitments to GitHub to show the progress of program development.