School of Computing Technologies

COSC1295 Advanced Programming 2025 S1

Assessment 2

	Assessment Type: **Individual assignment; no group work.** Submit online via Canvas → Assignments → Assignment 2. Marks awarded for meeting requirements as closely as possible. Clarifications/updates may be made via announcements/relevant discussion forums.
	Due date: **Wednesday 4 June**; Deadlines will not be advanced but they may be extended. Please check Canvas → Assignments → Assignment 2 for the most up to date information. As this is a major assignment, a university standard **late penalty** of 10% per each working day applies for up to 5 working days late, unless special consideration has been granted.
	Weighting: **50 marks** out of 100

Background Information

For this assessment you need to write a Java application with Graphic User Interface (GUI) which adheres to the object-oriented programming principles as specified in the specification of Assignment 1. In addition, you are required to practice on:

- developing GUI applications using JavaFX.
- implementing event driven programming.
- storing persistent copy of data.
- utilising advanced OO design patterns.

You are being assessed on your ability to write a program in an object-oriented manner.

The following guidelines apply throughout this assessment:

a) Any collections of objects in your program should be stored properly.
b) You must **not** use 3rd party libraries unspecified in this specification.

Learning Outcomes

1. Demonstrate knowledge of basic OO concepts and structure design in Java programming.
2. Devise solutions to computing problems under specific requirements.
3. Encode the devised solutions into executable programs and test the programs.
4. Demonstrate understanding of coding conventions and ethical considerations in programming.
5. Design graphical user interface to enable intuitive user interactions.
6. Develop test cases to validate the design and the execution of the program.
7. Practice on basic design patterns.
8. Manage time properly.

1. Overview

In this assignment, you will build a GUI application for The Super Event booking system. The app sells event tickets as shown below. The app will be able to:

- List all events.
- Book an event with number of seats.
- Checkout.
- Display orders.
- Export orders.
- Advanced features (see details in Section 4).

2. Getting Started

Your program should be able to read data from a text file "**events.dat**" (provided). The content of the file is shown as below. Each row is a line in the file. Columns are separated by semi-colons "**;**". Each event has a title, a location and a day information showing when and where the event will be held. The price for a same performance may differ depends on the day and the location. The total tickets column is the maximum capacity of the venue. #Total minus #Sold is obviously the number of tickets remaining, e.g., the event in the first row of the following table has 68 (99 - 22) tickets remaining.

EVENT	VENUE	DAY	$PRICE (AUD)	#SOLD TICKETS	#TOTAL TICKETS
Jazz Night with Joe	Theatre Nova	Mon	45	22	90
Jazz Night with Joe	Theatre Nova	Tue	50	30	90
Jazz Night with Joe	The Media Club	Fri	80	70	100
Jazz Night with Joe	The Media Club	Sun	85	60	100
Slow Rock Concert	Kaleide Theatre	Thu	100	100	200
Slow Rock Concert	Kaleide Theatre	Sat	150	150	200
Mozart Chamber Music	Studio 1	Mon	35	40	50
Mozart Chamber Music	Concert Hall	Sat	50	200	300
Mozart Chamber Music	Theatre Riverside	Thu	40	90	90

3. Task Specification

Basic functional requirements are listed below.

- The application requires login with username and password. A wrong login (e.g. incorrect combination of username and password) should not be able to go further. Note, all users share one system. Hence when one user booked some seats for an event (after checkout), other users can see the reduction of available seats as well.

- A new user can sign up. When signing up, the user should provide details including username, password, and a preferred name.

- Each user is shown a dashboard after login. The dashboard should display a personalized welcoming message that addresses the user by preferred name. The dashboard also displays the events that are stored in the data. Note, you can assume there will never be no more than 15 events, e.g. you don't need a paginator to show events on multiple pages.

- A user can book events by adding them into the cart. Your program allows the user to place an order by selecting an event and specifying the quantity. The program should allow the user to update the shopping cart (e.g., removing bookings, updating quantity) before checking out. When there is a modification to the cart, your program should verify the availability of tickets and notify the user if any selected event is fully booked.

    - Example 1. Assume that there are 10 seats of "Mozart Chamber Music" available on Monday (40 sold for Studio 1 which has 50 seats). A user will get a warning message if 11 seats were added into the cart.

    - Example 2. Assume that there are 10 seats of "Mozart Chamber Music" available on Monday. The user added 10 seats in the cart. Then the user further booked 2 more seats to the cart. A warning will be shown before checkout as there are no enough seats. Checkout will fail.

    - Example 3. Assume that there are 10 seats of "Mozart Chamber Music" available on Monday. The user booked 10 seats and checked out successfully. Then the user wanted to add 2 more seats for the same event. A warning will be shown and checkout cannot proceed.

- A user can check out. If there is no problem, the user will be notified of the total price. Once confirmed by the user, proceed to payment, where a simple validation is performed to check:

    - a 6-digit confirmation code, which supposedly is sent to the user's phone or email (sending the confirmation code is not required in this assignment). The validation is considered successful, as long as a valid six-digital is entered, e.g. 230134, or 792466, but not 120.23 or ab1023 or 94784 or 1124985.

    - the booked event is at least today or later. Our simple system only works for one-week, starting from Monday. Hence if today is Monday, when we log into the app, we can book all events scheduled from Monday (Mon) to Sunday (Sun). If today is Wednesday, we can NOT book any events scheduled for Monday (Mon) and Tuesday (Tue). If today is Saturday, the only shows that we can book are Saturday (Sat) and Sunday (Sun) events.

- A user can view all the orders. The user can view the following details of all historical orders: (1) a four-digit order numbers, starting from 0001, 0002, and so on; (2) date and time the order was placed; (3) the event booked along with the number of seats; (4) the total price of each order. The orders should be displayed in reverse chronological order, with the most recent order appearing first.

- A user can export all the orders to a text file, same as the above displayed content, including (1) the four-digit order number; (2) the date & time of the order; (3) the booked events with their quantities in that order, and (4) the total price of the order. A user can specify a preferred file name, and preferred location for saving the file to export.

- A user can log out and exit the program gracefully. When login back again, the user (either the same or different) can see the ticket status just before the logout, meaning the system is persistent.

Notes on requirements.

For GUI, we do not impose a particular design or a template. You can arrange the components according to your own design. However the design cannot be too simple, e.g. using the GUI as a text based terminal inside of a JavaFX window.

For backend data storage, you are required to implement JDBC for data management, including order data, event data and user management. For marking purpose, **please use SQLite ONLY.**

For OO design patterns, you are required to implement the MVC design pattern. In addition, you will implement **at least one more design pattern** (in addition to MVC).

4. Advanced Tasks

4.1 Admin features: The extended version of your program supports a special user, the admin. The account is initialized by login with the special username and password: `admin` and `Admin321`:

The admin user can perform the following actions:

- Once log in, the admin user should be welcomed with an admin dashboard.
- The admin can view all events, same as what a normal user can view. However, the admin should not have a shopping cart. In addition, the same performance will be grouped together. That is, there is no repetition in titles. For example, the show "Jazz Night with Joe" at the Theatre Nova will only appear once but with two options: "Mon" and "Tue". The show "Mozart Chamber Music" will also appear once, but with three options: "Studio 1" – "Mon", "Concert Hall" – "Sat" and "Theatre Riverside" – "Thu".
- The admin can disable one or multiple events. Once an event is disabled, normal users would not be able to see that event. Note the event is just disabled, NOT deleted. The admin can still enable the event later for normal users. Note, there is no need to check whether there are existing bookings for the to-be-disabled event, assuming refund is properly handled. Restoring a disabled event simply restores all the previous bookings.
- The admin can update the list of events through GUI
    - add an event, providing the name, venue, day, price and venue capacity (ticket sell should be zero). Check for duplicates.
    - delete an event. Give warning but OK to proceed if there are existing bookings for the event.
    - modify an existing event, e.g. adjust the capacity of a venue, change the venue, the day and the price. Check for possible duplicates.
- The admin can view the existing orders of all users.

4.2 User features:

The advanced version allows a normal user to change the password. Once password changed and application restarted, you application should accept the new password. Note, the new password should never be stored in its original text. Some kind of encryption should apply, e.g. simply shifting the alphabetic position, a -> d, b -> e, 1 -> 4 etc.

4.3 JUnit

You are required to write at least 5 test cases for the validation mentioned above, e.g. password validation, availability check, or existence check.

Assessment Criteria

This assessment will determine your ability to:

1. Follow coding, convention and behavioral requirements in this specification and in the class.
2. Independently solve a problem by using concepts taught over the first several weeks.
3. Write and debug Java code independently.
4. Document code.
5. Provide references where due.
6. Manage the code development process and meet deadlines.
7. Seek clarification from your "client"/"supervisor" (us) when needed via discussion forums.
8. Create a program by recalling concepts taught in class, understanding and applying concepts relevant to solution, analysing components of the problem, evaluating different approaches.

Marking Guide

This assessment is divided up into several components:

• Basics GUI Design (not too cumbersome or too ugly)	(7 marks)
• Object oriented design and patterns	(7 marks)
• Booking functionalities (signup, login, and dashboard)	(3 marks)
• JDBC for data storage and management	(2 marks)
• Event booking (availability check)	(3 marks)
• Checkout (confirmation code and day validation)	(3 marks)
• Viewing all orders	(2 marks)
• Export orders	(2 marks)
• Code quality (code readability, comment)	(2 marks)
• In-class milestone check	(5 marks)
• Advanced: admin GUI and functionalities	(10 marks)
• Advanced: user functionalities (password updates)	(2 marks)
• Advanced: JUnit test cases	(2 marks)

Submission Instructions

Submit via zip file

Export the project as an archive; the file name should be the same as the project name, for example **s1234567-APAssignment2.zip** and submit the zip file to Canvas under Assignments.

To export a project as a zip file in Eclipse/IntelliJ, click **File** -> **Export** -> **Project to zip file**.

You are required to submit regularly from Week 10, at least **one submission per week** unless you would like to complete early. The first weekly submission is due by Week 10. Canvas will keep all submissions. **No extension request will be granted** if there is no sufficient amount of submissions.

Marking is purely based on the last version. Early submissions are only for progress management.

You are required to include a README either as comments at the top of your main Java file (with `public static void main(String args[])`, or a separate `README.TXT` file in the same folder as the main Java file to specify what you have implemented in Assignment 2 by Y(es), N(o) or a number, as below:

```
// <Your Student ID>, <Your Name>
// Y Signup
// Y Login
// Y Read the events from database and display
// Y Events can be booked and bookings can be modified
// Y Bookings are validated
// Y Checkout is validated with confirmation code and day information
// Y Remaining seats of events is updated once an order is made
// Y User can view all orders
// Y User can export orders to file
// N Admin GUI & admin login implemented
// N Admin display implemented (no duplicate event titles)
// N Event disable function implemented
// N Event adding & deletion functions implemented
// N Event modification function implemented
// N Viewing orders of all users implemented
// N User password update and encryption implemented
// N Junit test cases included
// N Design pattern (in addition to MVC)
```

Academic Integrity and Plagiarism (Standard RMIT Warning)

Your code will be automatically checked for similarity against other submissions so please make sure your submitted project is entirely your own work.

Academic integrity is about honest presentation of your work. It means acknowledging the work of others while developing your own insights, knowledge, and ideas. You should take extreme care that you have:

- Acknowledged words, data, diagrams, models, frameworks and / or ideas of others you have quoted (i.e., directly copied), summarised, paraphrased, discussed, or mentioned in your assessment through the appropriate referencing methods.
- Provided a reference list of the publication details so your reader can locate the source if necessary. This includes material taken from internet sites.

Without acknowledging the sources of the material, you may be accused of plagiarism because you have passed off the work and ideas of another person without appropriate referencing, as if they were your own.

RMIT University treats plagiarism as a very serious offence constituting misconduct. Plagiarism covers a variety of inappropriate behaviours, including:

- Failure to properly document a source.
- Copyright material from the internet or databases.
- Collusion between students.

For further information on our policies and procedures, please refer to the University website.

Assessment Declaration

When you submit your project electronically, you agree to the RMIT Assessment Declaration.