COSC2288 Further Programming

SP1, 2024

Assessment Task (AT) 1:

Assignment 1 Specification V1.2 (updated 18/3/24)

Live Music Venue Matcher

✺	Assessment type: **individual assignment; no group work**. Submit online via Canvas → Assignments → Assignment 1. Marks are awarded for meeting requirements as closely as possible according to assignment specifications and the supplied rubric. Clarifications/updates may be made via announcements/relevant discussion forums.
📅	Due date: **Week 6, Tuesday, 2 April at 23:59**. Late submission penalty will be applied unless special consideration has been granted. In a separate assignment submission page you will need to submit a **recorded demo presentation**. You will describe and explain the key concepts adopted in your code, demonstrate code execution, and articulate lessons learnt in the milestones. Instructions for milestone checking will be provided separately in the milestone submissions.
✪	Weighting: 15 points for Assignment 1. 3 points for AT1 recorded demo presentation.

1. Overview

NOTE: Carefully read this document. In addition, regularly follow the Canvas assignment discussion board for assignment related clarifications and discussion.

In this assignment you are required to work individually to implement a basic object-oriented Java program that will run on terminal screen, an Event Venue Management System. You will practice your knowledge of Java basics, object-oriented (OO) concepts, Java Collections Framework, exceptions, file input/output, and unit testing, whilst being able to turn a software design and specification into a working software application. You will use Java SE 17 or later.

This specification document may be revised later to help clarify requirements, remove ambiguity, improve format, etc.

Background Story

You have been hired by a Melbourne-based venue broker company as an analyst programmer to build a system to help find suitable venues to host live music events for a variety number of audience per show.

The system's only user at this stage will be the manager, who works with various clients based on job orders.

Software Requirements

The system will be used by the manager to perform the following tasks:

- List current events requiring a venue based on job requests data. At first, all venues are available for booking.

- Select and show event details including the main artist and supporting acts.

- List venues based on category and name search.

- Select and show a venue details including availability, capacity and hire price.

- Import and extract venue data from csv files.

- Import and extract job request data from csv files.

- Once imported each venue record is given an auto-generated unique id number.

- Once imported each request record is given an auto-generated unique id number.

- Auto-matching feature: Get recommendation for potential venues for all the active job requests. This should be done by comparing attributes between request data and venue data, including availability, capacity, event type (large live concert or gig), and category (indoor or outdoor), For example, if all 4 of 4 criteria are matching, the suitability of the venue for that particular event is 100%.

- Hire the venues on behalf of the client.

- Calculate commissions based on brokering and other fees.

- All functions must be thoroughly tested.

Out of scope:

- The system is not required to have persistent data or ability to record orders in an external file or database at this stage.

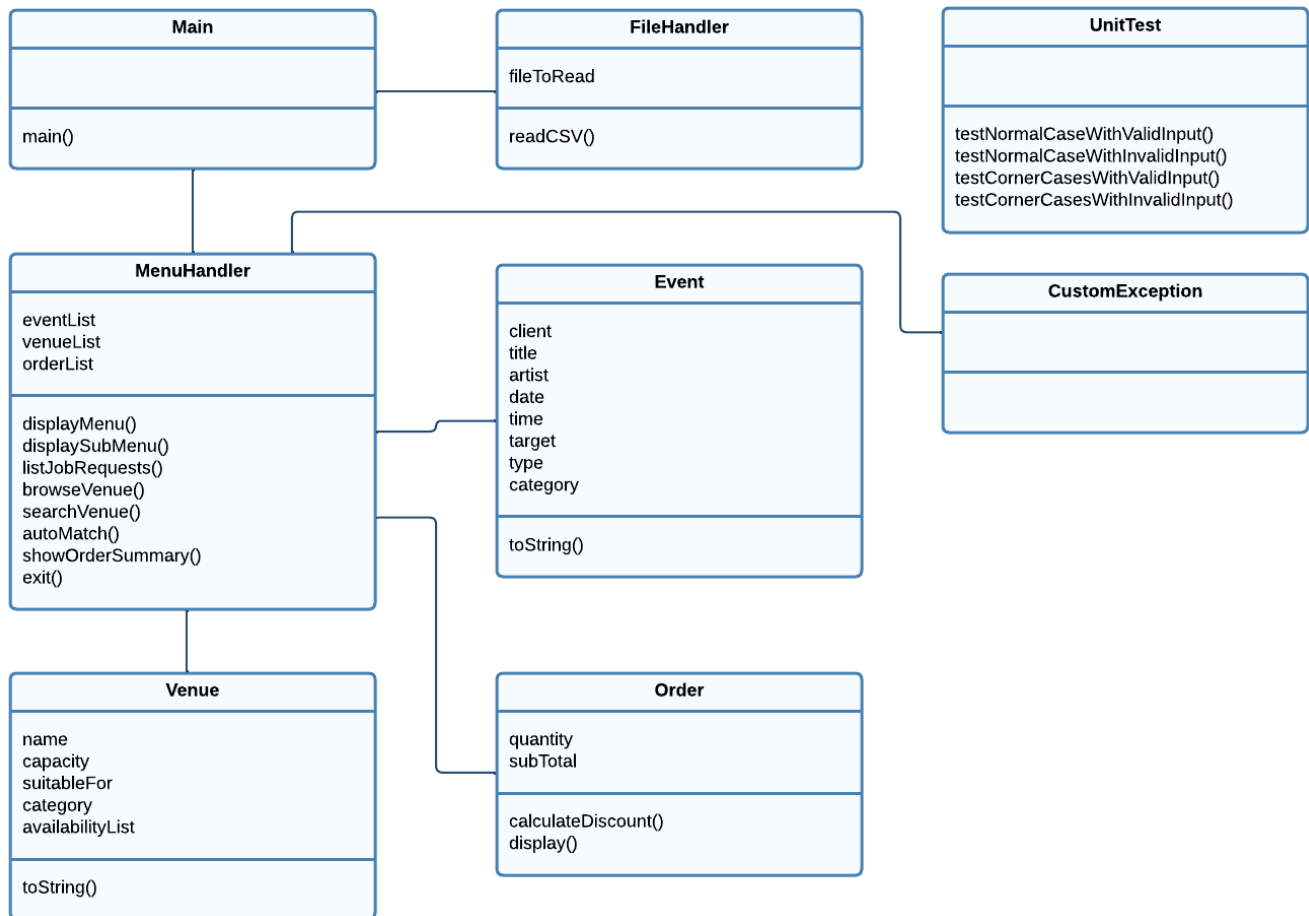- No requirement for ability to delete or update order records at this stage.

Software Design

Based on this information, your co-worker have come up with initial analysis of the requirement and designed a class diagram in Unified Modelling Language (UML) format.

You were to implement a software program that will run on terminal screens that rely heavily

on keyboard inputs.

The following class diagram is not complete. It only represents basic attributes, operations of each class and its relationship with other classes required in your program. As the only Java programmer in this company, you are to decide on the correct data type for each attribute, method return type, and parameter arguments, and if you need to introduce inheritance, abstract class or interface in your improved design, so that your program will work efficiently, easy to extend, adhering to the standard rule in designing modular software systems: 'low coupling, high cohesion'.

Main

main()

FileHandler

fileToRead

readCSV()

UnitTest

testNormalCaseWithValidInput()
testNormalCaseWithInvalidInput()
testCornerCasesWithValidInput()
testCornerCasesWithInvalidInput()

MenuHandler

eventList
venueList
orderList

displayMenu()
displaySubMenu()
listJobRequests()
browseVenue()
searchVenue()
autoMatch()
showOrderSummary()
exit()

Event

client
title
artist
date
time
target
type
category

toString()

CustomException

Venue

name
capacity
suitableFor
category
availabilityList

toString()

Order

quantity
subTotal

calculateDiscount()
display()

Software Specification

Business rules to incorporate in your program are as follows:

- Brokerage commission is 10% from total venue order.

- When a client made multiple job requests, they are eligible for 1% discount, hence the manager will only receive 9% commission payable by the client.

- When the manager run the auto-match function, the system will scan through a list of venue candidates for each event listed in the job requests based on: capacity, category and suitability keywords and make recommendation for the most suitable one.

- The menu item selections for categories and venues should be coming from a dynamically populated list based on the provided data file. At the end of this assignment, none of these items should be hard-coded, because data files can grow / shrink and frequently getting updated.

- At first, all venues are available for booking.

- Once a venue is booked for a particular date and time, this venue is no longer available for that date and time.

The following is a sample interaction with the Venue Matcher. You do not have to follow this precisely, but it illustrates the required functionality. Text in **bold and green** would be input from the user. For convenience, the system should prompt to offer if the user wants to make a booking after displaying lists in function menu items 1,2, 3 and 4 as such "Do you want to make a booking (y/n)?", then proceed to booking prompts linking order record with a suitable venue record. Once the booking is confirmed, it will be recorded in a list of orders and will remain in the memory as long as the program is running.

```
Welcome to Venue Matcher
-------------------------------------------------------
> Select from main menu
-------------------------------------------------------
1) List current job requests
2) Browse venue by category
3) Search venue by name
4) Auto-match events with suitable venues
5) Show order summary
6) Exit
Please select: 7
Please select a valid menu option.
-------------------------------------------------------
> Select from main menu
-------------------------------------------------------
```

```
1) List current job requests
2) Browse venue by category
3) Search venue by name
4) Auto-match events with suitable venues
5) Show order summary
6) Exit
Please select: 2
--------------------------------------------------------
> Select by category
--------------------------------------------------------
1) Outdoor
2) Indoor
3) Convertible
4) Go to main menu
Please select: 2
--------------------------------------------------------
> Select from venue list
--------------------------------------------------------
1) The Espy
2) The Forum
3) Corner Hotel
4) The Tote

Please select: 1
--------------------------------------------------------
> The Espy
Capacity: 2000
Category: indoor
Suitable for: gigs, live music, bands
--------------------------------------------------------
   1) Hire for $550.00/hour
   2) Back to venue list
Please select: 1

Hiring Details
Please enter number of hours: 3
Please enter date: 20/12/2024
Please enter time: 8:00pm
Enter a request number for this
order: 1
Event name: Thirsty Merc In the Summertime
Artist name: Thirsty Merc
Requester name: Mousetrap Heart Events


Confirm order (y/n): y
Order confirmed.

--------------------------------------------------------
> Select from main menu
--------------------------------------------------------
1) List current job requests
2) Browse venue by category
```

```
3) Search venue by name
4) Auto-match events with suitable venues
5) Show order summary
6) Exit
Please select: 3
Please enter a venue name: forum
--------------------------------------------------------
> Select from matching list
--------------------------------------------------------
   1) The forum
   2) Go to main menu
Please select: 2


--------------------------------------------------------
> Select from main menu
--------------------------------------------------------
1) List current job requests
2) Browse venue by category
3) Search venue by name
4) Auto-match events with suitable venues
5) Show order summary
6) Exit
Please select: 5


--------------------------------------------------------
Order Summary
--------------------------------------------------------
Job#1
Client:      Mousetrap Heart Events
Venue:       The Espy
Event name: Thirsty Merc In the Summertime
Artist:      Thirsty Merc
Event date: 20/12/2024
Event time: 8:00pm

3 hours venue hire @ $550     $1650.00
Brokering commission 10%:      $165.00


Job#2
Client:      Party People United
Venue:       The Forum
Event name: New Years Eve Rock and Dance Gigs
Artist:      Various local musicians
Event date: 31/12/2024
Event time: 8:00pm

5 hours venue hire @ $280     $1400.00
Brokering commission 10%:      $140.00


Total earnings to date: $305.
--------------------------------------------------------
```

2. Assessment Criteria

This assessment will determine your ability to implement Object-Oriented Java code according to the specifications shown below. In addition to functional correctness (i.e., getting your code to work) you will also be assessed on code quality. Specifically:

- You are required to demonstrate your understanding of basic object-oriented programming principles, including encapsulation and abstraction.
- You are required to use well-tested Java collections.
- You are required to validate all user inputs and catch and handle all exceptions.
- You are required to demonstrate that you have done adequate unit testing using the JUnit framework.
- You are required to read all system data from files and parse the information correctly.
- You are required to write properly documented code.

3. Learning Outcomes

This assessment is relevant to the following Learning Outcomes:

CLO1: Explain the purpose of OO design and apply the following OO concepts in Java code: inheritance, polymorphism, abstract classes, interfaces, and generics.

CLO2: Describe the Java Collections Framework (JCF) and apply this framework in Java code.

CLO4: Demonstrate proficiency using an integrated development environment such as Eclipse for project management, coding and debugging.

4. Assessment details

You are required to implement the functions mentioned in the task specification in Section 1 - Overview.

- You will incorporate basic object-oriented concepts (e.g., abstraction and encapsulation). In particular, you will define several classes to support the OO implementation. You will also need to choose appropriate data structures to store relevant information. You MUST use Java Collections.
- Your program should perform basic input validation (e.g., out-of-bound option) and provide clear console messages for invalid inputs and further actions.
- The order summary should be well-formatted (HINT: you can use System.out.printf method for formatting).
- You can hard code the venue and job request information at the beginning (Week 2- Week 3 just to get things going). You MUST use the information provided in the files. But after you know how to import data into a collection, please replace the hard coded data with dynamically populated collection based on the relevant CSV columns. – More marks will be given if you can work this out and implement it successfully.
- Main Menu Item #4 the auto-matching function requires an algorithm to compare the values of

relevant attributes of 2 types of objects from relevant collections. Once you worked out the search function for Main Menu Item #3, you can take this on as an extra challenge which will earn you more marks. HINT: You may write your own search and compare algorithm, or use existing Java List API methods such as removeAll, retainAll and workout the difference between 2 lists. Please do not use 3rd party library as this would be considered out-of-scope.

- You will need to document the code properly by following Code Conventions for the Java Programming Language by Oracle:
https://www.oracle.com/java/technologies/javase/codeconventions-introduction.html
- You are required to write unit tests for all classes that include functions using JUnit framework. Your unit tests should have a good coverage of the typical use cases of your classes (e.g., calculating the order price for multiple food items ordered from multiple restaurants) and test all reasonable corner cases (e.g., handling invalid method arguments). You do not need to write unit tests for getters and setters.
- You are required to read all information from the given files for venues and demands (HINT: you can use Scanner class and its relevant methods to read from text file and parse the information).
- You are required to catch and handle all exceptions in your program. For example, if you use Scanner class to read the text file, you will need to catch and handle the *FileNotFoundException* specified by Scanner class. Avoid throwing an unspecific Exception; throw a specific exception (e.g., throw a *NumberFormatException* instead of an *IllegalArgumentException*).

5. Submission format

- You must submit a zip file of your project via Canvas.
- Please include a README with instructions on how to run and test your program.
- You can submit your assignment as many times as you would like prior to the due date. The latest submission will be graded.
- You MUST NOT submit compiled files (*.class files). Please check your submission carefully, make sure all java files have been included.

6. Referencing guidelines

You are free to refer to textbooks or notes and discuss the design issues (and associated general solutions) with your fellow students or on Canvas; however, the assignment should be your OWN WORK.

The submitted assignment must be your own work. For more information, please visit the academic integrity website: https://www.rmit.edu.au/students/student-essentials/assessment-and-results/academic-integrity.

Plagiarism is treated very seriously at RMIT. Plagiarism includes copying code directly from other students, internet 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 work even if you

have very similar ideas.

Plagiarism-detection tools will be used for all submissions. Penalties may be applied in cases of plagiarism.

7. Academic integrity and plagiarism (standard warning)

Academic integrity is about honest presentation of your academic 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), summarized, 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.

If you do not acknowledge the sources of your 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 behaviors, including:

- Failure to properly document a source

- Copyright material from the internet or databases

- Collusion between students