Further Programming
COSC2391
Assignment 1: Online Booking System for Course Management

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 or updates may be made via announcements.
Due Date	**Week 6, Sunday 9 April 11:59pm.** Late submission penalty will be applied unless special consideration has been granted. **There will be one milestone, video demo check, in week 4, Friday 24 March 11:59pm.** You will describe and explain the key concepts adopted in your code and demonstrate code execution in the milestone.
Marks	20 marks out of 100 for assignment 1. 5 marks (in addition to the 20 marks) out of 100 for milestone check.

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

1. Overview

In this assignment, you are required to work individually to implement a basic Java console program, an online course management system for students, named MyTimetable. You will practice your knowledge of Java basics, object-oriented (OO) concepts, Java Collections Framework, exceptions, and unit testing. You will use Java SE 8 or later.

2. Task specification

The system keeps a list of courses that students can choose to enroll. The list of courses is provided in courses.csv file (a csv file is a comma-separated values file). The first row in courses.csv file contains headers, and the rest of the rows contain the course information. The header row and an example record are shown below.

Course name	Capacity	Year	Delivery mode	Day of lecture	Time of lecture	Duration of lecture (hour)
Java programming	120	Year 1	Face-to-face	Wednesday	11:30	2

If a course is delivered in face-to-face mode, the capacity of the course (i.e., maximum number of students) will be specified in the csv file. If a course is delivered online, there will be no capacity constraint, and hence, the “Capacity” column will be “N/A”.

The console-based program provides the following options:

- To enroll into a course, the user can search for a course by keyword (e.g., programming). The program should list all courses with the matching input. Note that the matching should be case-insensitive. The user can then choose one from the list.
- Once a course is selected, the system will add this course to the student’s list of enrolled courses.
- The user can choose to display all the courses that the user has enrolled.
- The user can choose to withdraw from a course.
- The user can choose to exit the program. The program does not need to keep the record after termination.

3. Assessment details

In this assignment, you will incrementally build the basic Java program. The assignment is structured into 1 milestone with video demo in week 4 and final submission in week 6.

Part A

You are required to implement the functions mentioned in the task specification in Section 1 - Overview. You can start this part from week 1 and work on it until the due date.

- 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 (e.g., a list of courses). You MUST use Java Collections because the list is growable.
- The summary should be well-formatted (HINT: you can use System.out.printf method for formatting).
- You can hard code the course information. You MUST use the information provided in the file, for testing purposes; please do not create new course information.
- 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

Following is a sample interaction with the online booking system MyTimetable. 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:

```
Welcome to MyTimetable!
--------------------------------------------------------------------------------
> Select from main menu
--------------------------------------------------------------------------------
   1) Search by keyword to enroll
   2) Show my enrolled courses
   3) Withdraw from a course
   4) Exit
Please select: 2
You don't have any courses enrolled.
--------------------------------------------------------------------------------
> Select from main menu
--------------------------------------------------------------------------------
   1) Search by keyword to enroll
   2) Show my enrolled courses
   3) Withdraw from a course
   4) Exit
Please select: 1
Please provide a brand: Programming
--------------------------------------------------------------------------------
> Select from matching list
--------------------------------------------------------------------------------
   1) Java programming
   2) Programming skills
   3) Advanced Python programming
   4) Go to main menu
Please select: 1
You have enrolled in the course Java programming!
--------------------------------------------------------------------------------
> Select from main menu
--------------------------------------------------------------------------------
   1) Search by keyword to enroll
   2) Show my enrolled courses
   3) Withdraw from a course
   4) Exit
Please select: 2
--------------------------------------------------------------------------------
You have enrolled into the following course(s):
--------------------------------------------------------------------------------
   1) Java programming     Face-to-face     Wed 11:30-13:30
--------------------------------------------------------------------------------
> Select from main menu
--------------------------------------------------------------------------------
   1) Search by keyword to enroll
   2) Show my enrolled courses
```

```
   3) Withdraw from a course
   4) Exit
Please select: 3
--------------------------------------------------------------------------------
Please choose a course to withdraw:
--------------------------------------------------------------------------------
   1) Java programming     Face-to-face     Wed 11:30-13:30
Please select: 1
You have withdrawn from Java programming!
--------------------------------------------------------------------------------
> Select from main menu
--------------------------------------------------------------------------------
   1) Search by keyword to enroll
   2) Show my enrolled courses
   3) Withdraw from a course
   4) Exit
Please select: 4
```

Part B

You can start this part from week 4 and work on it until the due date.

- 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 total payment) and test all reasonable corner cases (e.g., handling invalid method arguments). You do not need to write unit tests for getters and setters.

Part C

You can start this part from week 4 and work on it until the due date.

- You are required to catch and handle all exceptions in your program. You will create customized exceptions that can be used in the program. For example, you can create an *InvalidMenuOption* exception. Then you can catch and handle the exception when a user enters an invalid menu option. Avoid throwing an unspecific Exception; throw a specific exception (e.g., throw a *NumberFormatException* instead of an *IllegalArgumentException*).

4. Submission

You will submit all the relevant material as listed below via Canvas.

- You must have Main.java file which includes the main method. This file serves as the entry point of your program.
- You must include a README with instructions on how to compile and run your program from the command line. You will be given a sample console program and a document showing instructions on how to compile and run the sample program from the command line on Canvas. You must give similar instructions (as shown in step 2 in the given document) in your README.
- You must submit a zip file of your project. Make sure you zip the top-level project folder.
- You must not submit compiled files (*.class files) and any IDE related files (e.g., .settings folder generated by Eclipse). Please check your submission carefully, make sure all java files have been included.
- In the final submission, you must submit a video demonstrating the functionality of your code and unit tests. Note that this video demo is **different** from milestone check in week 4. You will receive **zero** mark if you do not include a video in the submission.
- You can submit your assignment as many times as you would like prior to the due date. The latest submission will be graded.

After the due date, you will have 5 business days to submit your assignment as a late submission. Late submissions will incur a penalty of 10% of the full assignment 1 marks per day. After these five days, Canvas will be closed, and you will lose ALL marks.

5. Marking Guidelines

- Functionality (6/20)
- Unit testing (5/20)
- Exception handling (5/20)
- Object oriented design and choice of data structures (2/20)
- Source code quality (2/20)
- Milestone check in week 4 (5/5)

Functionality: Menu selection and user input operations correctly implemented: (1) filter courses correctly by given keyword; (2) enroll into a course; (3) withdraw from a course (4) all information is presented correctly with good formats.

Unit testing: Unit tests with a good coverage of the typical use cases of classes and testing all reasonable corner cases.

Exception handling: Aim to catch and handle all possible exception; using specific exception classes. The captured exceptions should be explained in the README file.

Object oriented design and choice of data structures: Appropriate choice of data structures and class designs.

Source code quality: Adequately documented and properly indented code; appropriate class/method/variable names.

Video demo in the final submission: Clearly perform the demo of all functions implemented and unit tests implemented. You must show your face while you demo. This is for the purpose of academic integrity. The demo must be within 5 minutes. You must use Microsoft Streams to create and share the demo. Instructions will be provided in a separate document.

Milestone check in week 4: Milestone check in week 4; clearly describe class design (i.e., classes and significant methods); clearly explain choice of data structures in JCF; demonstrating a working program that can allow a user to interact with the menu. The demo must be within 3 minutes.