4COSC005W Software Development II Coursework 23/24

Module Code / Title:	4COSC005W / Software Development II
Assessment Component:	Coursework
Weighting:	50%
Qualifying mark:	30%
Academic Year:	2023 - 2024
Semester:	2
Module Leader(s):	Dr. Ester Bonmati
Handed out:	Monday 12th February 2024
Due date:	Monday 18th March 2024 at 1pm Coursework demonstrations will be during week 10 and 11
Learning outcomes:	LO1: Choose appropriate algorithms and data structures for problemsolving and implement these using a programming language. LO3: Develop solutions to common programming problems using classes to implement fundamental object-orientated concepts. LO4: Implement common data structures and data sorting algorithms. LO5: Undertake basic requirements gathering and data modelling exercises.
Expected deliverables:	You must submit the following files: - Self-evaluation form (word or pdf) - Your project exported into a Code.zip file. Zip file should be able to open back using intelliJ If you do not submit a NetBeans project, it is your responsibility to know how to submit the correct files. **You must also attend a demo. Not attending the demo will result in 0 marks.**
Method of submission:	Submission on Blackboard.
Marks	We will aim to give the marks in 15 working days, but due to the large number of students in this module this may not be always possible. All marks will remain provisional until formally agreed by an Assessment Board.
Feedback	Constructive feedback will be given with the marks.

Assessment regulations

Refer to section 4 of the "How you study" guide for undergraduate students for a clarification of how you are assessed, penalties and late submissions, what constitutes plagiarism etc.

Penalty for late submissions

If you submit your coursework late but within 24 hours or one working day of the specified deadline, 10 marks will be deducted from the final mark, as a penalty for late submission, except for work which obtains a mark in the range 40 – 49%, in which case the mark will be capped at the pass mark (40%). If you submit your coursework more than 24 hours or more than one working day after the specified deadline you will be given a mark of zero for the work in question unless a claim of Mitigating Circumstances has been submitted and accepted as valid. It is recognised that on occasion, illness or a personal crisis can mean that you fail to submit a piece of work on time. In such cases you must inform the Student Centre in writing on a mitigating circumstances form, giving the reason for your late or non-submission. You must provide relevant documentary evidence with the form. This information will be reported to the relevant Assessment Board that will dec ide whether the mark of zero shall stand. For more detailed information regarding University Assessment Regulations, please refer to the following website:http://www.westminster.ac.uk/study/current-students/resources/academic-regulations

Coursework description

Important notes

- The use of generative AI (such as ChatGPT) for code generation is strictly **prohibited**. Generative AI can be used to: explain difficult concepts and find bugs in your code.
- Only use standard arrays and **do not use dynamic arrays** (e.g., ArrayList , Map, etc).
- Implement all the tasks in a single project.
- Use descriptive names for your variables and user-defined methods and add comments to explain your code, and use a good style.
- Implement and re-use your self-defined methods within your coursework when possible.
- Reference within your code any code adapted from external or other sources, or any technology that you may have used to implement your solution.
- Use appropriate debugging methods if you need to debug your code.
- For each task, read first what is requested, think about the design (you can use pen + paper) and then implement it.
- It is your responsibility to understand how to open BB submitted files using your IDE (i.e IntelliJ).

Context

A new company managing a private plane has asked you to create a Java program to manage and track seat reservations effectively. Your assignment involves designing and implementing this program. You will be working with the company's aircraft, which is shown below:

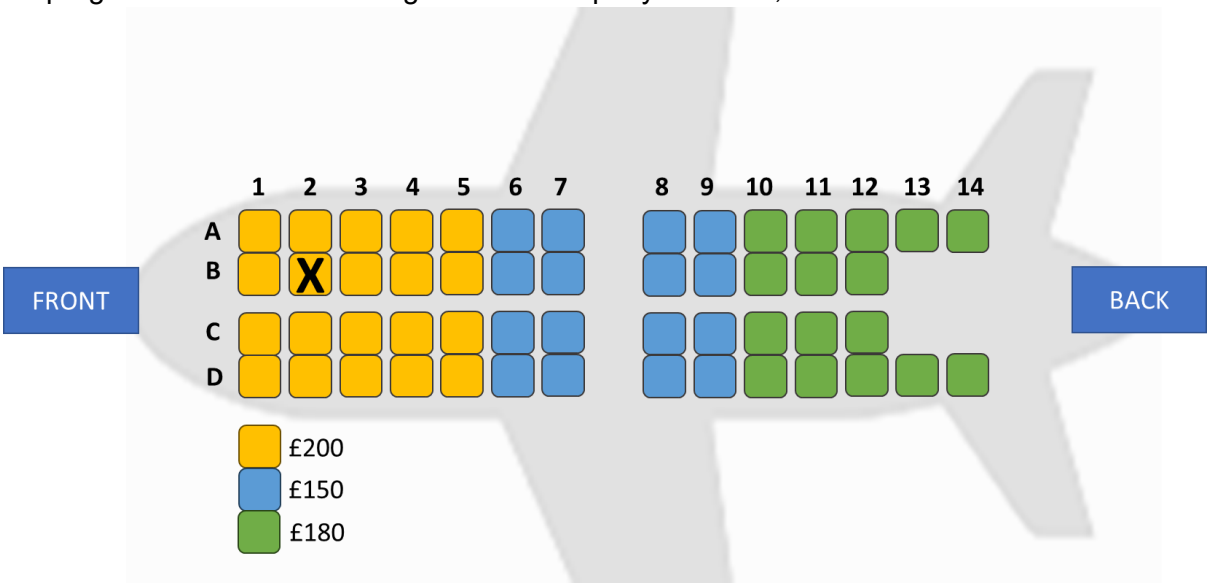

Figure 1. Plane seat plan. The plan has 4 rows with different numbers of seats (between 1 and 14 seats). In red is shown seat B2.

In the above picture, you can notice four rows of seats labelled as A, B, C and D, and it is important to note that these rows have varying seat capacities (either 12 or 14 seats). Seats are coded according to their row and seat number and have different prices according to their colour. For example, the seat marked with an 'X' is considered seat B2 and costs £200.

Considering the picture above, complete the following tasks:

Part A: Main program (40 marks)

Task 1) Create a new project titled <your_student_id_number>_PlaneManagement (e.g., w123456_PlaneManagement). Your project should have a class file named PlaneManagement (or <your_student_id_number>_PlaneManagement.java). This class should contain a **`main`** method that initiates the program with the display of the following message: 'Welcome to the Plane Management application' at the start of the program.

Implement the seat management system using standard arrays (**the use of dynamic arrays such as ArrayList or similar is prohibited, and their use will be penalised**) to keep track of the seats have been sold, and the seats that are still available. Use 0 to indicate that a seat is available, and 1 to indicate that a seat has been sold.

At the start of the program all seats should be available (0). The main method will be the method called when the program starts (entry point) for **all tasks** described in this coursework.

4 marks

Task 2) Add a user menu in your `main` method that displays the following menu and asks the user to select an option. Option '0' should terminate the program without crashing or giving an error. The rest of the options will be implemented in the next tasks.

```
**********************************************
*                MENU OPTIONS                *
**********************************************
     1) Buy a seat
     2) Cancel a seat
     3) Find first available seat
     4) Show seating plan
     5) Print tickets information and total sales
     6) Search ticket
     0) Quit
**********************************************
Please select an option:
```

Tip: Think carefully which control structure you will use to decide what to do after the user selects an option (Lecture Variables and Control Structures).

8 marks

Task 3) Create a method called **`buy_seat`** that asks the user to input a row letter and a seat number. Check that the row and seat entered are valid and that the seat is available (free). Record the seat as sold (as described in Task 1). Call this method when the user selects '1' in the main menu.

6 marks

Task 4) Create a method called **`cancel_seat`** that makes a seat available. It should ask the user to input a row number and a seat number. Check that the row and seat are valid, and that the seat is not available already. Record the seat as available (as described in Task 1). Call this method when the user selects '2' in the main menu.

6 marks

Task 5) Create a method called **`find_first_available`** that find the first seat which is still available. You should search in row A first, then B, then C and then D. Call this method when the user selects '3' in the main menu.

8 marks

Task 6) A) Create a method called **`show_seating_plan`** that shows the seats that are available and the seats that have been sold. Display available seats with the character 'O' and the sold seats with 'X'. Call this method when the user selects '4' in the main menu.

The output at the start of the program should have the following format:

```
OOOOOOOOOOOOO
OOOOOOOOOOOO
OOOOOOOOOOOO
OOOOOOOOOOOOOO
```

Please note that there is a space at the start of row B and C to match the plan seat plan.

After buying a ticket for row A, seat 4, the output should look like this:

```
OOOXOOOOOOOOO
OOOOOOOOOOOO
OOOOOOOOOOOO
OOOOOOOOOOOOOO
```

8 marks

Part B: Classes and Objects (38 marks)

Part B of coursework needs to be integrated with part A. Do not create a new project for part B. In part B, you will be adding two classes (Ticket and Person) in your project that stores extra information when buying a seat.

Task 7) Create a new class file called **`Person`** (Person.java) with the following attributes: name, surname, and email. Add a constructor that takes the 3 variables as input to create an object Person. Add all the getters and setters of the class Person. Add a method that prints the information from Person.

4 marks

Task 8) Create a new class file called **`Ticket`** (Ticket.java) with the following attributes: row, seat, price, and Person. Person is an object created using the class `Person` from Task 7.

Add all the getters and setters of the class Ticket. Add a method that prints the information of a Ticket (including the ~~information of the Person~~).

4 marks

Task 9) In the main program, add a array of Tickets to store all the tickets sold in that session. Then:

1) Extend the **`buy_seat`** method such that when buying a ticket, it asks for all the information of a Person, creates a new Ticket with the corresponding price and seat information and adds the ticket to the new array of Tickets.
2) Extend the **`cancel_seat`** method such that when cancelling a ticket, it removes the ticket from the array list of tickets.

10 marks

Task 10) Create a method called **`print_tickets_info`** that prints the information of all tickets that have been sold during the session, and calculates the total price of the tickets sold during the session Example, if ticket A1 and B7 were sold during the session, the total amount would be £350 (A1 = £200 + B2 = £150). Call this method when the user selects '5' in the main menu.

6 marks

Task 11) Create a method called **`search_ticket`** that asks the user to input a row letter and a seat number and searches if someone has bought that seat. If someone has bought the seat, it will print the Ticket and Person information, otherwise will display 'This seat is available'. Call this method when the user selects '6' in the main menu.

6 marks

Task 12) Add a method **`save`** in the class Ticket that saves the information of the Ticket (including the Person) in a file. The name of the file should be the name of the row and the seat number (e.g., A2.txt for a ticket sold in row A seat 2). Call this method every time a ticket is sold.

8 marks

Part C: Testing, style, and self-evaluation (12 marks)

Download the self-evaluation form from Blackboard and complete the following tasks:

Task 13) Download the self-evaluation form and test plan from Blackboard and complete it by describing the tests cases that you have carried out. ***10 marks.***

You will be evaluated on your coding style. Ensure that you use a good coding style in the submitted coursework to facilitate understanding: add comments to key parts of the code, use indentation, use informative names for variables and methods, create and reuse your selfdefined methods (to avoid repetition), etc. ***2 marks.***

Part D: Coursework demonstration (10 marks)

You will need to demonstrate your understanding of the submitted code. Your tutor will arrange a coursework demonstration, that will take place during weeks 10 and 11. During the coursework demonstration, your tutor will ask you to execute your program and questions on your code.

Failure to attend the demonstration will result in 0 marks for the coursework.

Marking scheme

The marking scheme of this coursework can be found in Blackboard under the coursework folder.

Submitting your coursework

1) Create a ZIP file of your project created based on the used IDE(i.e IntelliJ): Make sure the downloaded zip file is in working condition. You need to do the demonstration using downloaded zip file from Black Board during VIVA.

2) Rename your code.zip file and the self-assessment form in following format.

Code File : <IITNO>_<UoWNo>.zip
Self Assessment form : <IITNO>_<UoWNo>.pdf

3) Go to Blackboard's module page and select Assessment (coursework) > Coursework submission. **Attach below files as separate files in the same submission.**

i. Coursework report/ Self-assessment form (word or pdf)
ii. Code.zip file

4) Additionally Coursework report MUST be also submitted to Turnitin link provided separately.

5) Submit your work.

END