4COSC010C Software Development II
Deferral/Referral Coursework 22/23

Module Code / Title:	4COSC010C / Software Development II
Assessment Component:	Deferral/Referral coursework
Weighting:	50%
Qualifying mark:	30%
Academic Year:	2022 - 2023
Semester:	2
Module Leader(s):	Dr. Ester Bonmati
Handed out:	Monday 28th June 2023
Due date:	**Monday 24th July 2023 at 1pm**
Learning outcomes:	LO1: Choose appropriate algorithms and data structures for problem solving 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) - The following 3 Java files: ○ Cinema.java (Part A) ○ Person.java (Part B) ○ Ticket.java (Part B)
Method of submission:	Submission in on Blackboard.
Marks	Marks will be given 15 working days (3 weeks) after the submission deadline. All marks will remain provisional until formally agreed by an Assessment Board.
Feedback	Constructive feedback will be given 15 working days (3 weeks) after the submission deadline.

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

General notes

- **Do not use dynamic arrays (e.g., ArrayList) in this coursework, only standard arrays are allowed (e.g., `int[] array_name`).**
- Use descriptive names for your variables and user-defined methods.
- Add comments to explain your code and use a good coding style.
- Re-use the methods you implement 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 (e.g., ChatGPT).
- For each task, read first what is requested, think about the design (you can use pen + paper) and then implement it.

Context

A new cinema company called 'The New London Cinema' has asked you to design and implement a new Java program to manage and control the seats that have been sold and the seats that are still available for one of their theatre sessions. They have provided you with their floorplan, which can be seen in the picture below. The picture shows the 3 rows with the number of seats for each row.

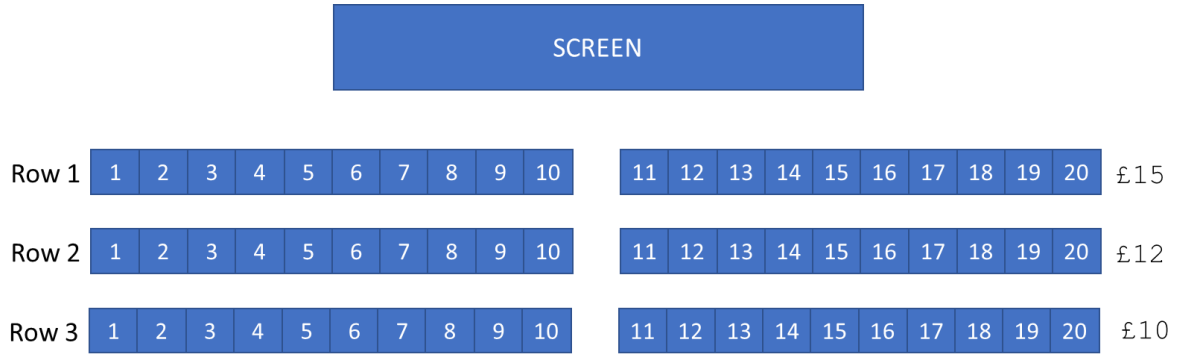

Figure 1 The New London Cinema seat plan.

Part A: Main program (40 marks)

Task 1) Create a new project named Cinema with a class (file) called **Cinema** (Cinema.java) with a **`main`** method that displays the following message 'Welcome to the New London Cinema' at the start of the program.

Use standard arrays (not dynamic arrays such as ArrayList) in your program to keep record of the seats that have been sold and the seats that are still free. Use value 0 in the array to indicate an available seat. Use value 1 to indicate an occupied (sold) seat. At the start of the program all seats should be 0 (free). This main method will be the method called when the program starts (entry point) for all Tasks described in this coursework.

Task 2) Add a menu in your `main` method. The menu should print the following options:

```
------------------------------------------------
Please select an option:
     1) Buy a ticket
     2) Cancel ticket
     3) Print seating area
     4) List available seats
     5) Print person information
     6) Print ticket information and total price
     7) Sort tickets by price
     8) Quit
------------------------------------------------
Enter option:
```

Ask the user to select one of the options. Option '0' should terminate the program without crashing or giving an error. The rest of the options will be implemented in the next tasks. Include a screenshot of your program's output in the self-assessment form.

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).

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

Task 4) Create a method called **`cancel_ticket`** that makes a seat available again. It should ask the user to input a row number and a seat number. Check that the row and seat are correct, and that the seat is not available. Record the seat as occupied (as described in Task 1). Call this method when the user selects '2' in the main menu. Include a screenshot of your program's output in the self-assessment form.

Task 5) A) Create a method called **`print_seating_area`** that shows the seats that have been sold, and the seats that are still available. Display available seats with the character 'O' and the sold seats with 'X'. Call this method when the user selects '3' in the main menu.

The output should show the following when no tickets have been bought (start of the program):

```
************
*  SCREEN  *
************
OOOOOOOOOO OOOOOOOOOO
OOOOOOOOOO OOOOOOOOOO
OOOOOOOOOO OOOOOOOOOO
```

After purchasing a ticket for row 1, seat 6, and a ticket for row 3, seat 10, using option '1' in the menu, the method should display the following:

```
************
*  SCREEN  *
************
OOOOOXOOOO OOOOOOOOOO
OOOOOOOOOO OOOOOOOOOO
OOOOOOOOOX OOOOOOOOOO
```

Include a screenshot of your program's output in the self-assessment form.

Task 6) Create a method called **`show_available`** that lists the seats available for each row. Call this method when the user selects '4' in the main menu.

Output example at the start of the program (all seats available):

```
Enter option: 4
Seats available in row 1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20.
Seats available in row 2: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20.
Seats available in row 3: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20.
```

Part B: Classes and Objects (50 marks)

Task 7) Create a class called **`Person`** (Person.java) with the following attributes: name, surname, and email. Add a constructor that takes the 3 variables as input to creates an object Person. Add the getters and the setters for each attribute in the class.

Add a method in class Person called **`print`** that prints all the information from a person.

Task 8) Create a new class file called **`Ticket`** (Ticket.java) with a constructor and the following attributes: row, seat, price, and Person. Person will be an object created using the class `Person`. Add the getters and the setters for each attribute in the class. Ticket prices are shown in Figure 1.

Add a method in class Ticket called **`print`** that prints all the information from a ticket: row, seat, price, and person details.

Task 9) In the main program, add a standard array list of tickets (not dynamic arrays such as ArrayList) to save the sold tickets information. Extend the **`buy_ticket`** method such that when buying a ticket, it asks for the information of a person, creates a new person, creates a new ticket (with the person) and adds the ticket in the new array list of tickets. Extend the **`cancel_ticket`** method such that when cancelling a ticket, it removes the ticket from the array list of tickets. Ticket prices for each row are shown in Figure 1.

Task 10) Create a method called **`show_person_info`** that asks for a seat number and a row number, and if the seat has been sold, prints the person information. Call this method when the user selects '5' in the main menu. Validate the seat and row number.

Task 11) Create a method called **`calculate_price`** that calculates and prints the total price of the tickets sold. Call this method when the user selects '6' in the main menu.

Task 12) Create a method called **`sort_tickets`** that sorts the list of sold tickets by price (lowest price first) using one of the algorithms shown during the lecture (lecture week 8). Print the tickets information once have been sorted (including price). Call this method when the user selects '7' in the main menu.

Part C: Testing, style, and self-evaluation (10 marks)

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

Task 13) In the self-evaluation form, fill the tables in TESTING (Part A and B) with the different tests cases that you used to test your program. Check that the expected output and the output obtained are the same and report if your implementation passes or not the test.

Examples:

Input	Expected output	Output obtained	Pass/Fail
Select option 1 in the menu and buy a ticket for row 3, seat 10 and print the seating area selecting option 3.	*The output should show that row 3, seat 10 is taken and the rest are available.*	*The output shows that row 3, seat 10 is taken and the rest are still free.*	*Pass.*
Select option 1 and enter row 'a'	*Message saying that the value entered is invalid.*	*The program crashes.*	*Fail*

Task 14) You will be evaluated on your coding style. Ensure that you are using 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 own methods (to avoid repetition), use of constants and global variables, etc.

Task 15) Complete the self-evaluation form with the screenshots requested and attach it with your submission. **Tasks 13 and 15 cannot be marked if you do not submit the self-evaluation form.**

Marking scheme

This coursework will be marked based on the following criteria:

Task	Criteria	Marks
1	Main correctly implemented showing a welcome message and array/s correctly created	5
2	Menu correctly implemented with all options and working correctly	7
3	Method 'buy_ticket' correctly records the seat as occupied	9
4	Method 'cancel_ticket' correctly records the seat as available	9
5	Method 'print_seating_area' shows correct available and occupied seats	6
6	Method 'show_available' correctly lists the available seats	4
7	Class Person implemented with the correct attributes, constructor, and methods	10
8	Class Ticket implemented with the correct attributes, constructor, and methods	10
9	New array for Ticket objects created, and methods correctly updated	12
10	Method 'show_person_info' shows the person details	4
11	Method 'calculate_price' calculates the price correctly	4
12	Method ‘sort_tickets’ sorts the tickets correctly	10
13	Tests for Part A and B	4
14	Style: use of comments, variables and methods names, use of constants and self-defined functions	4
15	Self-evaluation form completed	2
	TOTAL	**100**

Submitting your coursework

Go to the Blackboard's module page and select Deferral/Referral Coursework) > Coursework submission. Attach your **Self-assessment form (word or pdf)** and your 3 java files: **Cinema.java, Person.java and Ticket.java** into the same submission and submit your work.

END