4COSC005 Software Development II
Coursework 22/23

Module Code / Title:	4COSC005 / Software Development II
Assessment Component:	Coursework
Weighting:	50%
Qualifying mark:	30%
Academic Year:	2022 - 2023
Semester:	2
Module Leader(s):	Dr. Ester Bonmati
Handed out:	Monday 6th February 2023 (week 3)
Due date:	**Monday 20th March 2023 at 1pm** (week 9) **Coursework demonstrations will be during week 10 and 11**
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) - Your NetBeans 13 exported into a Code.zip file. You can do this by selecting the project, then select File > Export project > To Zip (see detailed instructions at the end of the document).
Method of submission:	Submission 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

- Use inbuilt methods when possible.
- Use descriptive names for your variables and user-defined methods.
- Add comments to explain your code and use a good style.
- Re-use the methods you implement within your coursework when possible.
- Use Netbeans IDE 13.
- 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 your debugging rubber duck 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.

Context

A new theatre company called 'New Theatre' 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 in which we can see that the theatre is composed of 3 rows, each with a different number of seats: 12, 16 and 20 retrospectively.
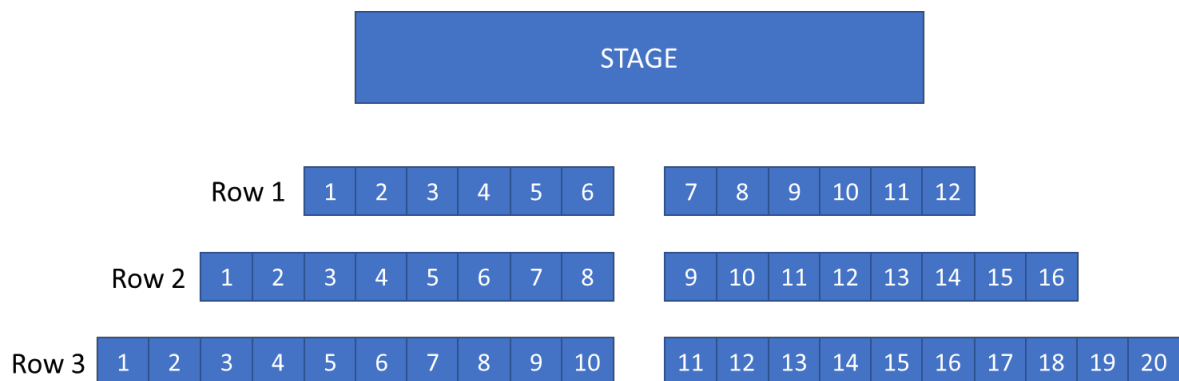

Figure 1 Theatre seat plan. The New Theatre has 3 rows, row 1 has 12 seats, row 2 has 16 seats and row 3 has 20 seats. The stage is in front of row 1.

Part A: Main program (40 marks)

Task 1)   Create a new project named Theatre with a class (file) called **Theatre** (Theatre.java) with a `main` method that displays the following message 'Welcome to the New Theatre' at the start of the program. Add 3 arrays (one for each row) in your program to keep record of the seats that have been sold and the seats that are still free. Row 1 has 12 seats, row 2 has 16 seats and row 3 has 20 seats. 0 indicates a free seat, 1 indicates 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 8 options: 1) Buy a ticket, 2) Print seating area, 3) Cancel ticket, 4) List available seats, 5) Save to file, 6) Load from file, 7) Print ticket information and total price, 8) Sort tickets by price, 0) Quit.

Then, 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. Example:

```
-------------------------------------------------

Please select an option:

     1) Buy a ticket

     2) Print seating area

     3) Cancel ticket

     4) List available seats

     5) Save to file

     6) Load from file

     7) Print ticket information and total price

     8) Sort tickets by price

     0) Quit

-------------------------------------------------

Enter 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).

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 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) 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 '2' in the main menu. The output should show the following when no tickets have been bought:

```
OOOOOOOOOOOO

OOOOOOOOOOOOOOO

OOOOOOOOOOOOOOOOOOO
```

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

OOOOOXOOOOOO

OOOOOOOOOOOOOOOO

OOOOOOOOOXOOOOOOOOOO

B) Update your code to align the display such that shows the seats in the correct location, and the stage, as shown below:

```
     * * * * * * * * * *

     *   STAGE   *

     * * * * * * * * * *

     OOOOOX  OOOOOO

   OOXXXXXX  OOOOOXXX

 XXOOOOXXXO  XXXOOOXXXX
```

Task 5) 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 '3' in the main menu.

Task 6) Create a method called **show_available** that for each of the 3 rows displays the seats that are still available. Call this method when the user selects '4' in the main menu.

Example at the start of the program:

```
Enter option: 4

Seats available in row 1: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12.

Seats available in row 2: 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16.

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.
```

Task 7) Create a method called **save** that saves the 3 arrays with the row's information in a file. Call this method when the user selects '5' in the main menu.

Task 8) Create a method called **load** that loads the file saved in Task 7 and restores the 3 arrays with the row's information. Call this method when the user selects '6' in the main menu.

Part B: Classes and Objects (40 marks)

Task 9) Create a new class file called `Person` (Person.java) with a constructor and the following attributes: name, surname, and email. Add a constructor that takes the 3 variables as input to create an object Person.

Task 10) Create a new class file called `Ticket` (Ticket.java) with a constructor and the following attributes: row, seat, price, and Person. Person will be is an object created using the class `Person` from Task 9.

Task 11) Add a method in class Ticket called `print` that prints all the information from a ticket: Person name, Person surname, Person email, row, seat, and price.

Task 12) In the main program, add an array list of tickets to save all the Tickets. Extend the `buy_ticket` method such that when buying a ticket, it asks for the information of a Person, creates a new ticket 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.

Task 13) Create a method called `show_tickets_info` that prints all the information for all tickets and calculates and shows the total price of all tickets after the ticket's info. Example, if ticket 1 costs £20 and ticket 2 costs £10, the total price will be £30. Call this method when the user selects '7' in the main menu.

Task 14) Create a method called `sort_tickets` that returns a new list of Tickets sorted by Ticket price in ascending order (cheapest first). Print the tickets information once have been sorted (including price). Call this method when the user selects '8' 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 15) In the self-evaluation form, fill the columns "Test input", "Expected output" and "Output obtained" with the different tests cases that you used to test your program. Check that the expected output and the output obtained are the same.

Example:

2	☒ Fully implemented and working ☐ Partially implemented ☐Not attempted	1: 2: Option 0 selected	1: Menu with 9 options 2: Program ends	1: Menu with 9 options 2: Program ends	Menu working correctly

See an example of a more complete form in Blackboard.

Task 16) 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), etc.

Task 17) Complete the self-evaluation form and attach it with your submission. **Tasks 15 and 17 cannot be marked if you do not submit the self-evaluation form.**

Part D: Coursework demonstration (10 marks)

Your tutor will arrange a coursework demonstration, that will take place during weeks 10 and 11. You must attend the demo. If you do not attend the demo, tasks 9 to 15 will not be marked.

During the coursework demonstration, your tutor will ask you to demonstrate your program and to show and answer a few questions on your code.

Marking scheme

This coursework will be marked based on the following criteria:

Task	Criteria	Marks
1	Main correctly implemented showing a welcome message and arrays correctly created	4
2	Menu correctly implemented with all options and working	6
3	Method 'buy_ticket' correctly records the seat as taken	9
4	Method 'print_seating_area' shows correct seats available and seats taken	5
5	Method 'cancel_ticket' correctly sets the ticket as available by updating the corresponding array	6
6	Method 'show_available' correctly implemented	2
7	Method 'save' opens a file and saves the arrays into a file correctly	4
8	Method 'load' opens a file and loads the arrays correctly	4
9	Object Person implemented with the correct attributes and constructor	6
10	Object Ticket implemented with the correct attributes and constructor	6
11	Method 'print' implemented in class Ticket	3
12	New array for Ticket objects created, and functions updated	12
13	Method 'show_tickets_info' shows the ticket info and total amount correctly calculated	8
14	Method 'sort_tickets' correctly implemented using a sorting algorithm	5
15	Tests for Part A and B	4
16	Style use of comments, variables+function names, use of constants and self-defined functions	4
17	Self-evaluation form completed and provided	2
DEMO	Performance on demonstrating the code and explaining the code. **Tasks 9-15 will not be marked if the demo is not attended.**	10
	TOTAL	**100**

Submitting your coursework

1) ZIP your NetBeans project: Select the project and then select File > Export project > To Zip. Name the .zip folder as Code.zip. Select Export. Please check where the .zip folder will be stored in your computer.
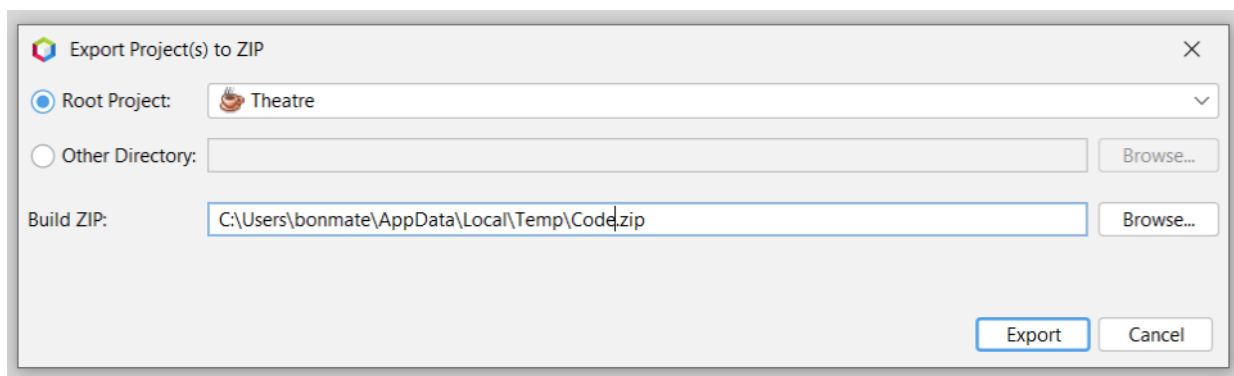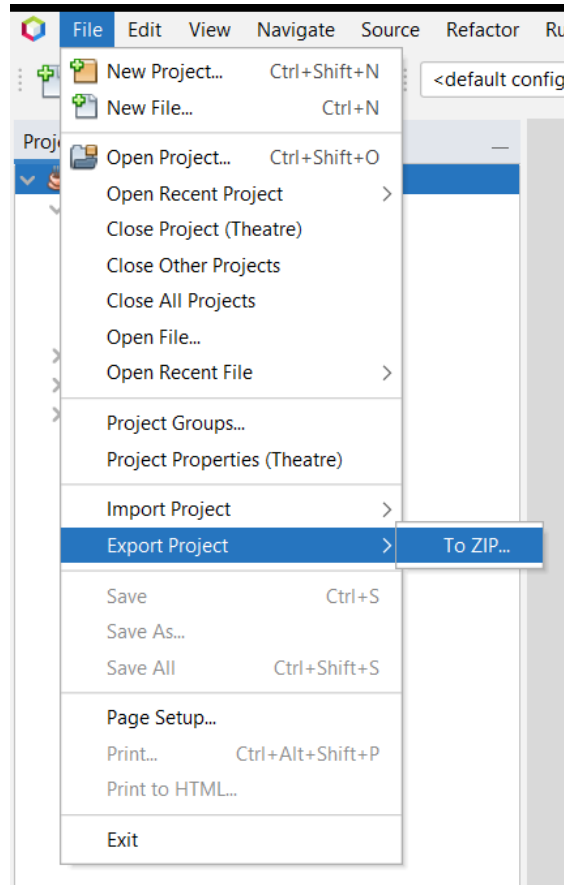
2) Go to Blackboard's module page and select Assessment (coursework) > Coursework submission. Attach your Self-assessment form (word or pdf) and your Code.zip file in into the same submission. Submit your work.

END