# 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.

Graphical user interface, application

Description automatically generated

*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 OOOOOOOOOOOOOOOO OOOOOOOOOOOOOOOOOOOO

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. |
| in | row | 2: | 1, | 2, | 3, | 4, | 5, | 6, | 7, | 8, 9, 10, 11, 12, 13, |
| in | row | 3: | 1, | 2, | 3, | 4, | 5, | 6, | 7, | 8, 9, 10, 11, 12, 13, |

Seats available 14, 15, 16.

Seats available

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.