Informatics Institute of Technology

School of Computing

Software Development II Coursework Report

Module	: 4COSC010C.2: Software Development II (2023)
Date of submission	: 24/03/2024
Student ID	: 20221948 / w2053013
Student First Name	: Sachintha
Student Surname	: Piyathunga

Tutorial group (day, time, and tutor/s): Group - G32, Monday, 1.30 p.m. to 3.30 p.m., Mr. Nazhim Kalam

"I confirm that I understand what plagiarism / collusion / contract cheating is and have read and understood the section on Assessment Offences in the Essential Information for Students. The work that I have submitted is entirely my own. Any work from other authors is duly referenced and acknowledged."

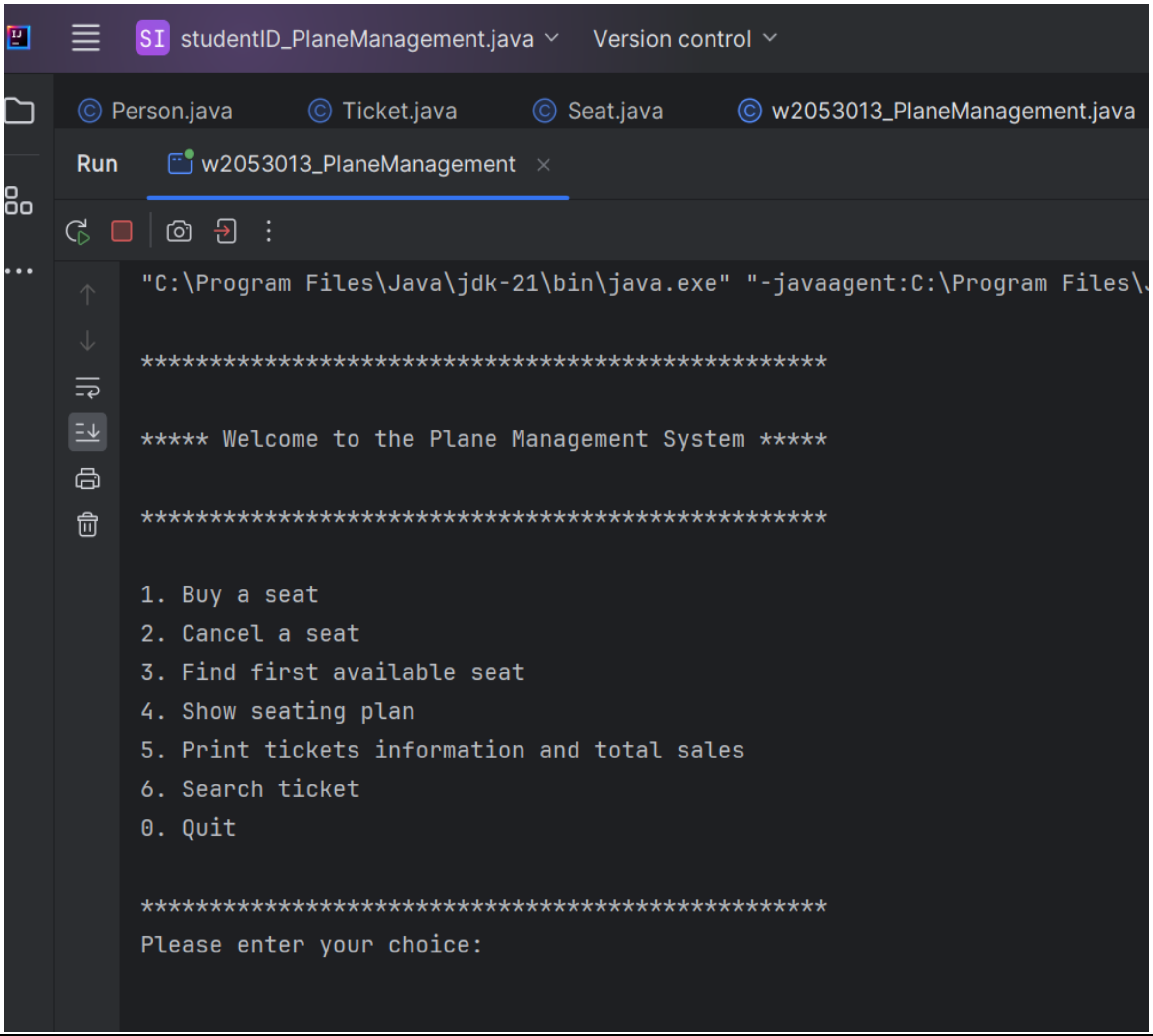
Name	: S S U Sachintha Chamod Piyathunga
Student ID	: 20221948 / w2053013

Self-assessment form and test plan

1) Self-assessment form

Task	Self-assessment (select one)	Comments
1	☒Fully implemented ☐Partially implemented ☐Not attempted	Project is created with; correct project title, correct project classes and add '0' and '1' to available seats.
2	☒Fully implemented ☐Partially implemented ☐Not attempted	Menu options are perfectly added to get the user's choice.

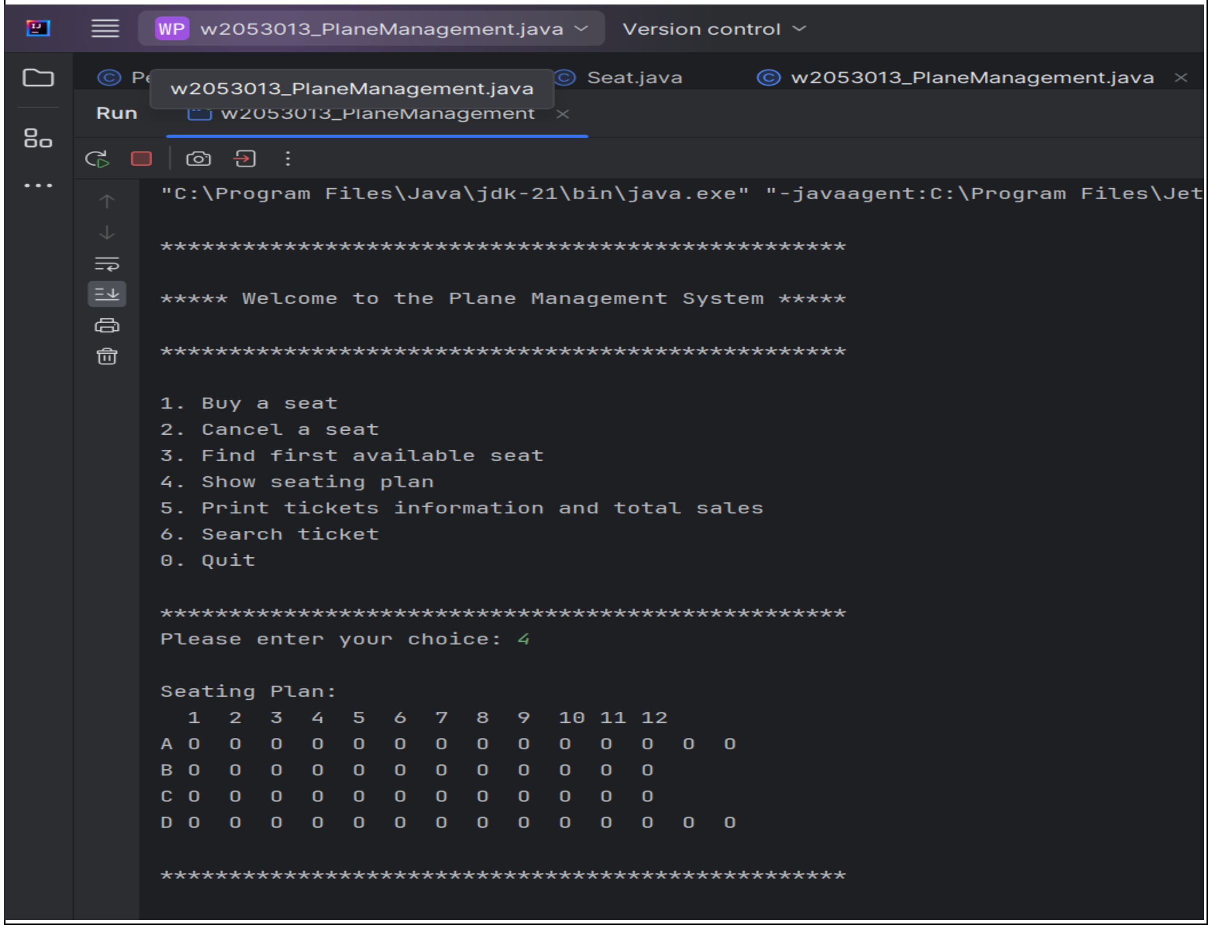
Insert here a screenshot of your welcome message and menu:



```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\

**************************************************

***** Welcome to the Plane Management System *****

**************************************************

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 enter your choice:
```

3	☒Fully implemented ☐Partially implemented ☐Not attempted	In buy a seat method, ask from user seat row letter, seat number ant user personal information(name and email).
4	☒Fully implemented ☐Partially implemented ☐Not attempted	In cancel a seat method, ask from user row letter and seat number and cancel the seat, and also delete the user information from that seat.
5	☒Fully implemented ☐Partially implemented ☐Not attempted	In this method Find first available seat for the user.
6	☒Fully implemented ☐Partially implemented ☐Not attempted	In showing seating plan method; show all free seats and booked seats.

Insert here a screenshot of the seating plan:


WP w2053013_PlaneManagement.java
Version control
w2053013_PlaneManagement.java
Seat.java
w2053013_PlaneManagement.java
Run
w2053013_PlaneManagement

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C:\Program Files\Jet

*****************************************************

***** Welcome to the Plane Management System *****

*****************************************************

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 enter your choice: 4

Seating Plan:
  1  2  3  4  5  6  7  8  9  10 11 12
A O  O  O  O  O  O  O  O  O  O  O  O  O  O
B O  O  O  O  O  O  O  O  O  O  O  O
C O  O  O  O  O  O  O  O  O  O  O  O
D O  O  O  O  O  O  O  O  O  O  O  O  O  O

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

7	☒Fully implemented ☐Partially implemented ☐Not attempted	Perfectly created the class for get user name, surname and email to get a seat.
8	☒Fully implemented ☐Partially implemented ☐Not attempted	Perfectly created the Ticket class for store the ticket information.
9	☒Fully implemented ☐Partially implemented ☐Not attempted	Added a another array to store the sold ticket information and when seat is cancelled ticket is deleted.
10	☒Fully implemented ☐Partially implemented ☐Not attempted	Printing all the ticket information and show available seats.
11	☒Fully implemented ☐Partially implemented ☐Not attempted	Add method to check and search the ticket information, to check seat is available or not.
12	☒Fully implemented ☐Partially implemented ☐Not attempted	Add a save method for save ticket information and passenger information in text file.

2) Test Plan

Complete the test plan describing which testing you have performed on your program.
Add as many rows as you need.

Part A Testing

Test case / scenario	**Input**	**Expected Output**	**Output**	**Pass/Fail**
1. Buy a seat	Option – 1 Row letter – A Seat number - 1	Booking row A seat 1 (A1)	Expected outcome	☒Pass ☐Fail
2. Cancel a seat	Option – 2 Row letter – A Seat number - 1	Cancel seat name call A1	Expected outcome	☒Pass ☐Fail
3. Find the first available seat	Option - 3	Displaying the first available seat.	Expected outcome	☒Pass ☐Fail
4. Show seating plan	Option - 4	Displaying all available seats and booking seats.	Expected outcome	☒Pass ☐Fail

5. Buying a seat using method 1	Option – 1 Row letter – A Seat number - 20	Display the seat can not be identified.	Expected outcome	☒Pass ☐Fail
6. Cancelling a seat using method 2	Option – 2 Row letter – A Seat number - 3	Displaying seat is already available.	Expected outcome	☒Pass ☐Fail
7. Quite method	Option - 0	Quitting the program.	Expected outcome	☒Pass ☐Fail

Part B testing

Test case / scenario	Input	Expected Output	Output	Pass/Fail
1. Print ticket information method	Option - 5	Display full information of the ticket and total sales.	Expected outcome	☒Pass ☐Fail
2. Search ticket method	Option – 6 Row letter – A Seat number - 4	Display that seat is available.	Expected outcome	☒Pass ☐Fail
3. Search ticket method	Option – 6 Roe letter – A Seat number - 4	Display that seat id booked.	Expected outcome	☒Pass ☐Fail
4. Buy seat method	Option – 1 Row letter - B Seat number – 8 Name – sachi Surname – piyathunga Email – sachipiya@gmail.comk	Display ticket bought successfully and save informations.	Expected outcome	☒Pass ☐Fail
5. Search ticket method	Option – 6 Row letter - G	Display row can not be identified.	Expected outcome	☒Pass ☐Fail
6. Search ticket method	Option – 6 Row letter - A Seat number - 40	Display seat can not be identified.	Expected outcome	☒Pass ☐Fail
7. Saving information in text file	Option - 0	Save all information in text file.	Expected outcome	☒Pass ☐Fail

Are there any specific parts of the coursework which you would like to get feedback?

You will need to demonstrate your understanding of the submitted code. Your tutor will arrange a coursework demonstration. During the coursework demonstration, your tutor will ask you to execute your program and questions on your code.

3) Code :

Person.java

```java
//w2053013 - Sachintha  chamod
//define a class name as Person to represent person info in ticket
class Person
{
    //declare a variable to store person information
    private String name; //name of the person
    private String surname; //surname of the person
    private String email; //email of the person

    //add method for initializing person information
    Person(String name, String surname, String email)
    {
        this.name = name;
        this.surname = surname;
        this.email = email;
    }

    //method for access private member variable
    public String getEmail() {
        return email;
    }

    public String getName() {
        return name;
    }

    public String getSurname() {
        return surname;
    }

    //set method to modify private member variable
    public void setName(String name) {
        this.name = name;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    //method to print person information
    void printPersonInfo()
    {
        System.out.println("Name: " + name);
        System.out.println("Surname: " + surname);
        System.out.println("Email: " + email);
    }
}
```

Ticket.java

```
//w2053013 - Sachintha chamod
//import necessary classes for file writing and handling exceptions.
import java.io.FileWriter;
import java.io.IOException;

//define a class as Ticket to represent a Ticket for a seat
class Ticket
{
    //declare variables to store the ticket information
    private char row; //row of the seat
    private int seat; //get seat number
    private int price; //price of the ticket
    private Person person; //person that get the ticket

    //method for initializing ticket information
    Ticket(char row, int seat, Person person)
    {
        this.row = row;
        this.seat = seat;
        this.person = person;
        //decide the price of the ticket based on the seat number
        if (seat == 1 || seat == 2 || seat == 3 || seat == 4 || seat == 5)
        {
            this.price = 200;
        } else if (seat == 6 || seat == 7 || seat == 8 || seat == 9)
        {
            this.price = 150;
        } else
        {
            this.price = 180;
        }
    }

    //add method to access private member variable
    public char getRow() {
        return row;
    }

    public int getPrice() {
        return price;
    }

    public int getSeat() {
        return seat;
    }

    public Person getPerson() {
        return person;
    }
    //set method to modify private member variable

    public void setPerson(Person person) {
        this.person = person;
    }

    public void setPrice(int price) {
        this.price = price;
    }

    public void setRow(char row) {
        this.row = row;
    }

    public void setSeat(int seat) {
        this.seat = seat;
    }

    //method for print ticket information
    void printTicketInfo()
    {
        System.out.println("Row: " + row);
        System.out.println("Seat: " + seat);
        System.out.println("Price: " + price);
        person.printPersonInfo();
    }

    //method to save ticket information in the file
    void saveTicketInfoToFile() throws IOException
    {
        //create fileWriter object to write a file
        FileWriter fileWriter = new FileWriter(row + "" + seat + ".txt");

        //write the ticket information in the file
        fileWriter.write("Row: " + row + "\n");
        fileWriter.write("Seat: " + seat + "\n");
        fileWriter.write("Price: " + price + "\n");
        fileWriter.write("Name: " + person.getName() + "\n");
        fileWriter.write("Surname: " + person.getSurname() + "\n");
        fileWriter.write("Email: " + person.getEmail() + "\n");
        //close the file writer object
        fileWriter.close();
    }
}
```

Seat.java

```
//w2053013 - Sachintha chamod
//Define a class name as seat to represent a seat in plane
class Seat
{
    int value; //get an integer variable to represent the availability of the
seat.

    Seat() {
        this.value = 0;
    }  //method for initializing the value of the seat to 0.

    void sellSeat() {
        this.value = 1;
    } //add method to mark the seat as sold, setting value to 1.

    void freeSeat() {
        this.value = 0;
    } //add method to mark the sea as free, setting value to 0.

    boolean isAvailable() {
        return this.value == 0;
    } //method to check seat availability, returning ture if value is 0.
}
```

W2053013_PlaneManagement.java

```
//w2053013 - Sachintha chamod

import java.io.IOException;//add this class for handling input-output
exceptions
import java.util.InputMismatchException;//add this class to check input
provided by the user does not match the expected type
import java.util.Scanner;//add this class to be used for obtaining user input
from the keyboard


//main class for manage for seat reservation on a plane
public class w2053013_PlaneManagement
{

    //define a 2D array to represent the seating plan in the plane
    private static final Seat[][] seatingPlan = {
            {new Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new
Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new
Seat(), new Seat(), new Seat()},
            {new Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new
Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new
Seat()},
            {new Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new
Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new
Seat()},
            {new Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new
Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new Seat(), new
Seat(), new Seat(), new Seat()}
    };
    private static final Ticket[] tickets = new Ticket[52]; //an array for
store all the sold tickets during the process
    private static final Scanner scanner = new Scanner(System.in); //scanner
object to get user inputs

    public static void main(String[] args)  //main method to start the
program
    {
        //display the welcome message
        System.out.println("\n" + "*".repeat(50) + "".repeat(16));
```

```java
        System.out.println("\n" + "*".repeat(5) + " Welcome to the Plane 
Management System " + "*".repeat(5));

        int choice; //variable to store user choice
        do //loop until the user choose quit
        {
            System.out.println("\n" + "*".repeat(50) + " ".repeat(16));
            System.out.println("\n1. Buy a seat");

            System.out.println("2. Cancel a seat");
            System.out.println("3. Find first available seat");
            System.out.println("4. Show seating plan");
            System.out.println("5. Print tickets information and total 
sales");
            System.out.println("6. Search ticket");
            System.out.println("0. Quit");
            System.out.println("\n" + "*".repeat(50));
            System.out.print("Please enter your choice: ");
            //get user choice
            choice = acquireChoice();
        } while (choice > 0); //continue the loop  until user choose to quit

        scanner.close(); //close the scanner object
    }

    private static int acquireChoice() //method to get user choice
    {
        int choice = 0;
        try
        {
            //try to get user input
            choice = scanner.nextInt(); //Consume newline character
            scanner.nextLine();
            //process for user choice
            switch (choice)
            {
                case 1:
                    buySeat();
                    break;
                case 2:
                    cancelSeat();
                    break;
                case 3:
                    findFirstAvailable();
                    break;
                case 4:
                    showSeatingPlan();
                    break;
                case 5:
                    printTicketsInfo();
                    break;
                case 6:
                    searchTicket();
                    break;
                case 0:
                    System.out.println("Quiting...");
                    break;
                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        } catch (InputMismatchException e)
        {
            //handling invalid inputs exception
            System.out.println("Invalid choice. Please try again.");
            scanner.nextLine();
            choice = 1; //set default choice to buy a seat
        }
        return choice;
    }

    //method to add a ticket to the  array of tickets
    private static void addTicket(Ticket ticket)
    {
        for (int i = 0; i < tickets.length; i++)
        {
            if (tickets[i] == null)
            {
                tickets[i] = ticket;
                break;
            }
        }
    }

    //method tho remove a ticket from the array of tickets
    private static void removeTicket(char row, int seatNumber)
    {
        for (int i = 0; i < tickets.length; i++)
        {
            if (tickets[i].getRow() == row && tickets[i].getSeat() ==
seatNumber)
            {
                tickets[i] = null;
                break;
            }
        }
    }

    //method to check if a string contains only non-integer characters
    private static boolean nonIntegerStringCorrect(String str)
    {
        if (str.isEmpty())
        {
            return false;
        } else return str.matches("[a-zA-Z]+");
    }

    //method to check if roe character is correct
    private static boolean rowCorrect(char row)
    {
        return switch (row)
        {
            case 'A', 'C', 'B', 'D' -> true;
            default -> false;
        };
    }

    //method to get row index based on the roe character
    private static int getRowIndex(char row)
    {
        return switch (row)
        {
            case 'A' -> 0;
            case 'B' -> 1;
            case 'C' -> 2;
            case 'D' -> 3;
            default -> -1;
        };
    }

    //method to check if a seat number is correct for a given roe
    private static boolean seatNumberCorrect(char row, int seatNumber)
    {
        if (row == 'A' || row == 'B')
        {
            return seatNumber >= 1 && seatNumber <= 14;
        } else
        {
            return seatNumber >= 1 && seatNumber <= 12;
        }
    }

    //method to get the row from the user
    private static char acquireRow()
    {
        System.out.print("Enter row (A-D): ");
        String rowStr = scanner.nextLine().toUpperCase();
        if (!nonIntegerStringCorrect(rowStr))
        {
            System.out.println("row cannot be empty or have numbers");
            return 0;
        }
        char row = rowStr.charAt(0);
        if (!rowCorrect(row))
        {
            System.out.println("row cannot be identified");
            return 0;
        }
        return row;
    }

    //method to get the seat number from user
    private static int acquireSeat(char row)
    {
        try
        {
            System.out.print("Enter seat number: ");
            int seat = scanner.nextInt();
            if (!seatNumberCorrect(row, seat))
            {
                System.out.println("seat cannot be identified");
                return -1;
```

```java
            }
            scanner.nextLine();
            return seat;
        } catch (Exception e)
        {
            scanner.nextLine();
            return -1;
        }
    }

    //method to get a string input from the user for various purposes
    private static String acquireStr(String enterWhat)
    {
        System.out.print("Enter your " + enterWhat + " : ");
        String str = scanner.nextLine();
        if (!nonIntegerStringCorrect(str))
        {
            System.out.println(enterWhat + " cannot be empty or have
numbers");
            return null;
        }
        return str;
    }

    //method to get an email from user (email format)
    private static String acquireEmail()
    {
        System.out.print("Enter your email : ");
        String str = scanner.nextLine();
        if (!str.matches("^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-
zAZ]{2,6}$"))
        {
            System.out.println("email cannot be identified");
            return null;
        }
        return str;
    }

    //method to handle the process of buying seat
    private static void buySeat()
    {
        char row = acquireRow();
        if (row == 0)
        {
            return;
        }
        int rowIndex = getRowIndex(row);
        int seat = acquireSeat(row);
        if (seat == -1)
        {
            return;
        }

        Seat chosenSeat = seatingPlan[rowIndex][seat - 1];
        if (chosenSeat.isAvailable())
        {
            String name = acquireStr("name");
            if (name == null)
            {
                return;
            }
            String surname = acquireStr("surname");
            if (surname == null)
            {
                return;
            }
            String email = acquireEmail();
            if (email == null)
            {
                return;
            }

            Person person = new Person(name, surname, email);
            chosenSeat.sellSeat();

            Ticket ticket = new Ticket(row, seat, person);
            try
            {
                addTicket(ticket);
                ticket.saveTicketInfoToFile();
            } catch (IOException e)
            {
                throw new RuntimeException(e);
            }
            System.out.println("Ticket bought successfully!");
        } else {
            System.out.println("Seat already sold or invalid seat.");
        }
    }

    //method to handle the process of  canceling a seat
    private static void cancelSeat()
    {
        char row = acquireRow();
        if (row == 0)
        {
            return;
        }
        int rowIndex = getRowIndex(row);
        int seat = acquireSeat(row);
        if (seat == -1)
        {
            return;
        }

        if (!seatingPlan[rowIndex][seat - 1].isAvailable())
        {
            removeTicket(row, seat);
            seatingPlan[rowIndex][seat - 1].freeSeat();
            System.out.println("Seat cancelled successfully!");
        } else
        {
            System.out.println("Seat already available or invalid seat.");
        }
```

```java
    }

    //method to find first available seat
    private static void findFirstAvailable()
    {
        for (int row = 0; row < seatingPlan.length; row++)
        {
            for (int seat = 0; seat < seatingPlan[row].length; seat++)
            {
                if (seatingPlan[row][seat].isAvailable())
                {
                    System.out.println("First available seat: Row " + (char)
('A' + row) + ", Seat " + (seat + 1));
                    return;
                }
            }
        }
        System.out.println("No available seats.");
    }

    //method to display the seating plan
    private static void showSeatingPlan()
    {
        System.out.println("\nSeating Plan:");
        System.out.print("  ");
        for (int i = 1; i <= seatingPlan[1].length; i++)
        {
            System.out.printf("%-3d", i);
        }
        System.out.println();
        for (int row = 0; row < seatingPlan.length; row++)
        {
            System.out.print((char) ('A' + row) + " ");
            for (Seat seat : seatingPlan[row])
            {
                System.out.printf("%-3s", seat.value == 0 ? "O" : "X");
            }
            System.out.println();
        }
    }

    //method to print ticket information
    private static void printTicketsInfo()
    {
        // This method is already implemented in Part B
        int totalAcquiredTicketPrices = 0;
        for (Ticket ticket: tickets){
            if (ticket == null)
            {
                continue;
            }
            ticket.printTicketInfo();
            System.out.println();
            totalAcquiredTicketPrices += ticket.getPrice();
        }
        System.out.println("Total acquired ticket prices : " +
totalAcquiredTicketPrices);
```

```java
    }

    //method to search for a ticket
    private static void searchTicket()
    {
        // This method is already implemented in Part B
        char row = acquireRow();
        if (row == 0)
        {
            return;
        }
        int rowIndex = getRowIndex(row);
        int seat = acquireSeat(row);
        if (seat == -1)
        {
            return;
        }

        if (seatingPlan[rowIndex][seat - 1].isAvailable())
        {
            System.out.println("Seat available");
            return;
        }

        for (Ticket ticket: tickets)
        {
            if (ticket == null)
            {
                continue;
            }
            if (ticket.getRow() == row && ticket.getSeat() == seat)
            {
                ticket.printTicketInfo();
                break;
            }
        }
    }

}
```