Informatics Institute of Technology

School of Computing

Software Development II Coursework Report

Module : 4COSC010C.2: Software Development II (2024)

Date of submission : 23 / 03 /2024

Student ID : 20230171/ w2052917

Student First Name : Dineth

Student Surname : De Alwis

Tutorial group (day, time, and tutor/s): Tuesday,10.30am-12.30am, Mr. Ruwan Egodawatte

"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."

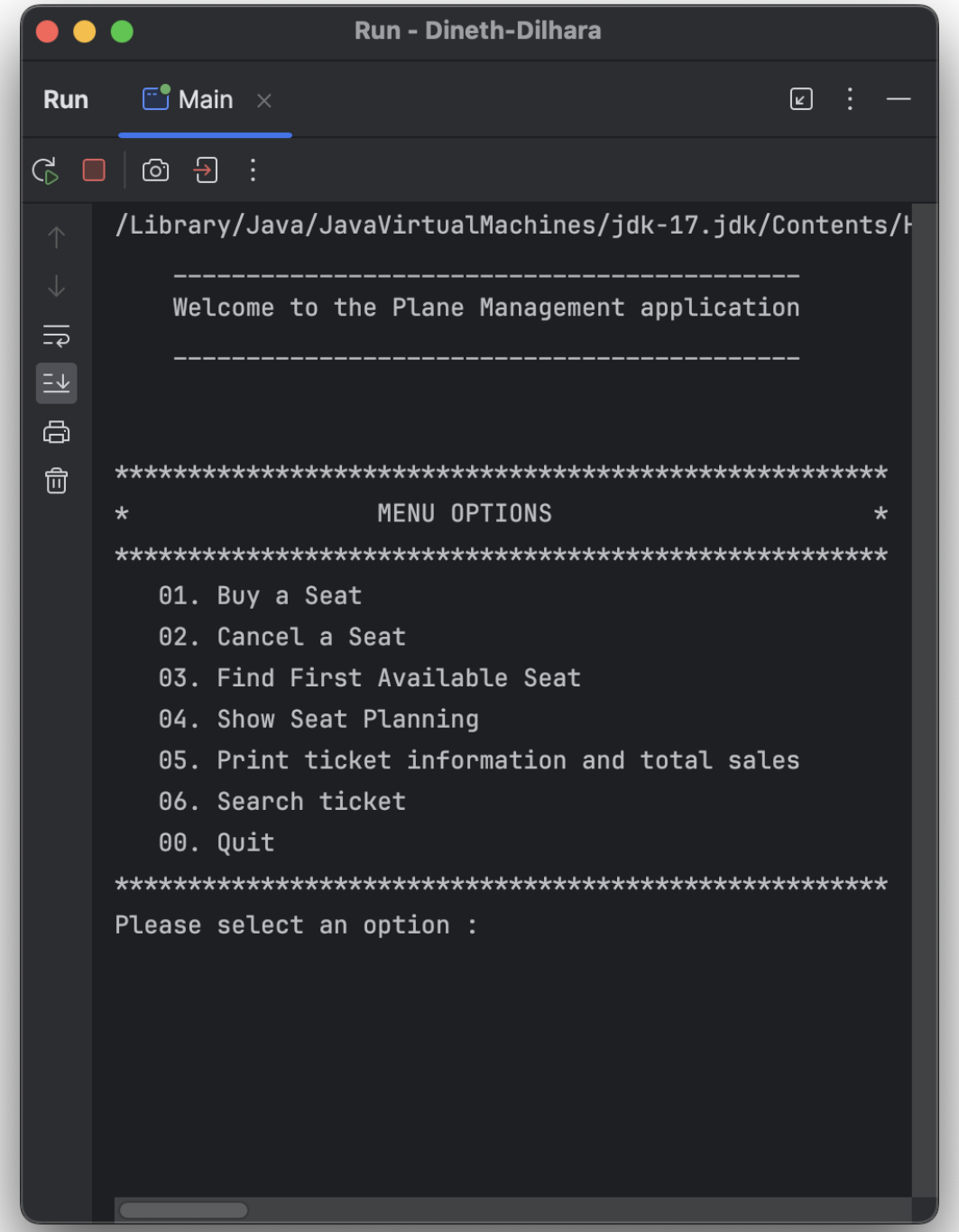
Name : K.D. Dineth Dilhara De Alwis

Student ID : 20230171 / w2052917

Self-assessment form and test plan

1) Self-assessment form

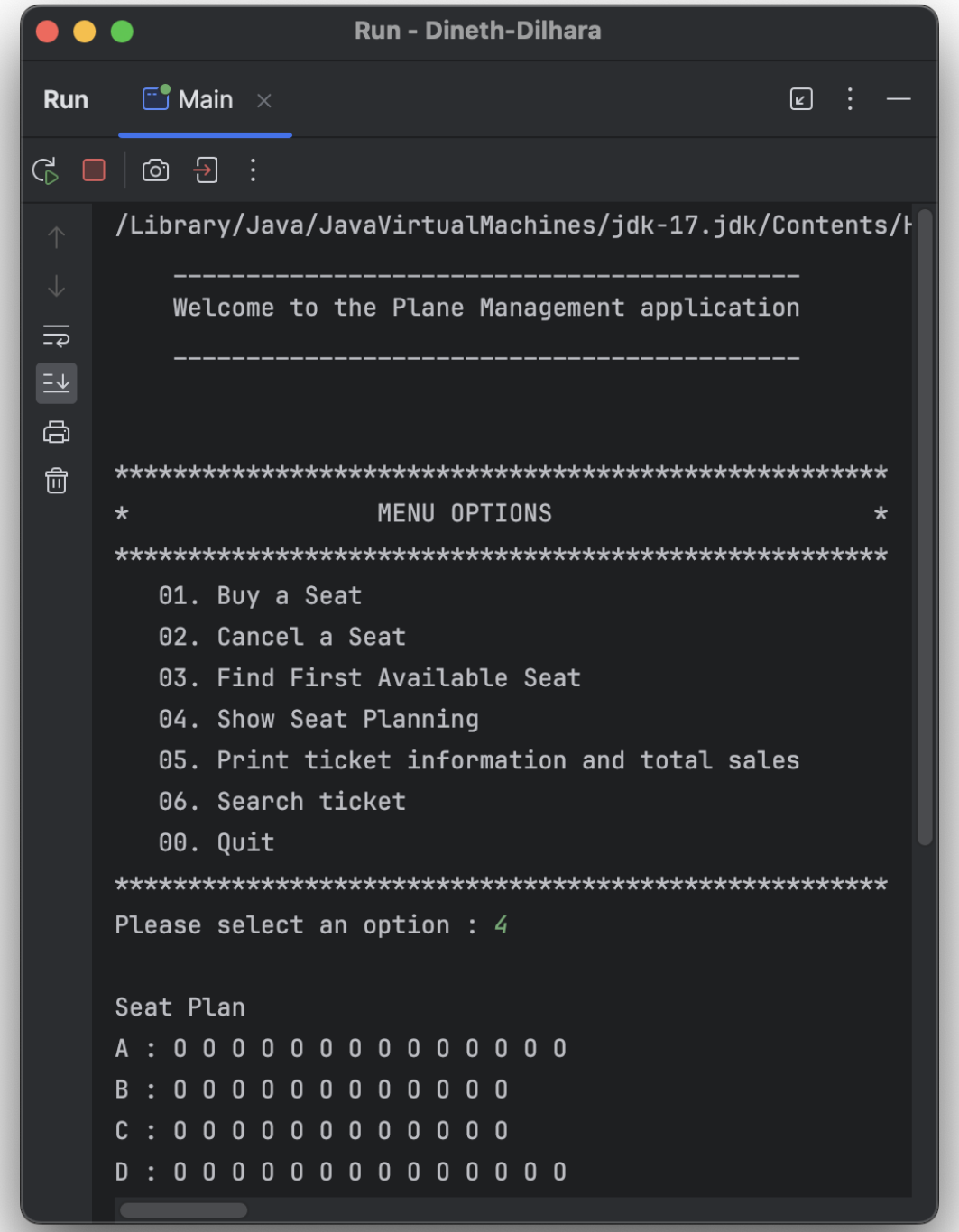
Task	Self-assessment (select one)	Comments
1 **Main, welcome message and array declaration and initialisation**	☒Fully implemented. ☐Partially implemented ☐Not attempted	declaration of arrays and main shows welcome message correctly
2 **Menu**	☒Fully implemented. ☐Partially implemented. ☐Not attempted	Menu works correctly and exit when option is 0 without errors
Insert here a screenshot of your welcome message and menu:		


Run - Dineth-Dilhara
Run
Main
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/H

Welcome to the Plane Management application

****
* MENU OPTIONS *
****
01. Buy a Seat
02. Cancel a Seat
03. Find First Available Seat
04. Show Seat Planning
05. Print ticket information and total sales
06. Search ticket
00. Quit
****
Please select an option :

3 **Method buy_seat**	☒Fully implemented. ☐Partially implemented. ☐Not attempted	Row Letter and seat validation and seat update correctly
4 **Method cancel_seat**	☒Fully implemented. ☐Partially implemented. ☐Not attempted	Row Letter and seat validation and seat update correctly
5 **Method find_first_available**	☒Fully implemented. ☐Partially implemented ☐Not attempted	Works correclty
6 **Method show_seating_plan**	☒Fully implemented. ☐Partially implemented. ☐Not attempted	Updates correctly with right format
Insert here a screenshot of the seating plan:		


Run - Dineth-Dilhara
Run
Main
/Library/Java/JavaVirtualMachines/jdk-17.jdk/Contents/H

Welcome to the Plane Management application

****
* MENU OPTIONS *
****
01. Buy a Seat
02. Cancel a Seat
03. Find First Available Seat
04. Show Seat Planning
05. Print ticket information and total sales
06. Search ticket
00. Quit
****
Please select an option : 4
Seat Plan
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 **Class Person**	☒Fully implemented ☐Partially implemented ☐Not attempted	all getters and setters and constructor and properties define correctly.
8 **Class Ticket**	☒Fully implemented ☐Partially implemented ☐Not attempted	all getters and setters and constructor and properties define correctly.
9 **Buy/Cancel seats update**	☒Fully implemented ☐Partially implemented ☐Not attempted	Works Correctly
10 **Method print_tickets_info**	☒Fully implemented ☐Partially implemented ☐Not attempted	Prints every seat ticket detail with person details that has been sold in session
11 **Method search_ticket**	☒Fully implemented ☐Partially implemented ☐Not attempted	Works Correctly
12 **Method save**	☒Fully implemented ☐Partially implemented ☐Not attempted	Correct implementation and saves the file with the correct information

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**
01.Menu (Select an Option)	Option = 1	After option 1 ask row letter for buy seat	After option 1 ask row letter for buy seat	☒Pass ☐Fail
02.Menu (Select an invalid Option)	Option = a	Ask user to enter valid Option	Ask user to enter valid Option	☒Pass ☐Fail

03.BuySeat	Option = 1 Row Letter = A Seat Number = 5	Bought seat and update seat plan	Bought seat and update seat plan	☒Pass ☐Fail
04.BuySeat (Already bought seat)	Option = 1 Row Letter = A Seat Number = 5	Display is seat is not available	Display is seat is not available	☒Pass ☐Fail
05.CancelSeat	Option = 2 Row Letter = A Seat Number = 5	Cancel seat and update seat plan	Cancel seat and update seat plan	☒Pass ☐Fail
06.CancelSeat (Already Available seat)	Option = 2 Row Letter = A Seat Number = 5	Display this seat is available at the moment	Display this seat is available at the moment	☒Pass ☐Fail
07.Method frist_find_seat	Option = 3	Display first available seat	Display first available seat	☒Pass ☐Fail
08.Method Show_Seat_ Plan	Option = 4	Display updated seat plan	Display updated seat plan	☒Pass ☐Fail
09.Input Invalid input for Row Letter	Option = 1 Row Letter = W	Ask user to input valid row letter and ask to enter it to system	Ask user to input valid row letter and ask to enter it to system	☒Pass ☐Fail
10.Input Invalid input for Seat number	Option = 1 Row Letter = A Seat Number = 20	Ask user to input valid seat number and ask to enter it to system	Ask user to input valid seat number and ask to enter it to system	☒Pass ☐Fail

Part B testing

Test case / scenario	Input	Expected Output	Output	Pass/Fail
01.Extend the buy seat method with Person class	Option = 1 Row Letter = A Seat Number = 1	When buy a seat Ask user to enter name, surname and mail and add them into ticket array	When buy a seat Ask user to enter name, surname and mail and add them into ticket array	☒Pass ☐Fail
02.Extend the Cancel seat	Option = 2 Row Letter = A Seat Number = 1	When cancelling seat makes that	When cancelling seat makes that seat	☒Pass ☐Fail

method with Person class		seat ticket detilas to null	ticket detilas to null	
03.Method Print_ Ticket_information	Option = 5	Print every bought person and ticket details And total sales	Print every bought person and ticket details And total sales	☒Pass ☐Fail
04.Method Search Ticket (Already bought seat)	Option = 6 Row Letter = A Seat Number = 1	Print bought person and ticket details	Print bought person and ticket details	☒Pass ☐Fail
05.Method Search Ticket (Already Available seat)	Option = 6 Row Letter = A Seat Number = 12	Print seat is available at the moment	Print seat is available at the moment	☒Pass ☐Fail
06.Method save	Option = 2 Row Letter = A Seat Number = 3 Inputs person details	After bought seat save a text file with person and ticket information with the name of row letter and seat number	After bought seat save a text file with person and ticket information with the name of row letter and seat number	☒Pass ☐Fail
07.Exit from the Program	Option = 0	Exit from the program without giving any error message	Exit from the program without giving any error message	☒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.

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

3) Code:

01.Main Class

```
import java.io.File;
import java.util.InputMismatchException;
import java.util.Scanner;

public class Main {
    public static void main(String[] args) {

        System.out.println("""
                                    ____________________________________________
                                    Welcome to the Plane Management application
                                    ____________________________________________
                                 """);
        initializeSeats();
        deleteAllFilesAtBegin();
        displayMenu();
    }
    static Ticket[][] tickets = new Ticket[4][];// 2D array call ticket to
store ticket information
    static int[][] seatStructure = new int[4][];// 2D array to store and
update seat availability
    private static void initializeSeats() {
        // initialize each row in jagged 2D arrays that hold ticket details
and seat availability
        for (int i = 0; i < 4; i++) {
            seatStructure[i] = new int[seatPerRow[i]];
            tickets[i]= new Ticket[seatPerRow[i]];
        }
    }
    static int[] seatPerRow = {14,12,12,14};// helps to initialize seat in
jagged array
    static char[] rows = {'A','B','C','D'};// to validate row letter
    static double totalPrice = 0;// variable to store total sales
    private static void displayMenu() {
        Scanner scanner = new Scanner(System.in);
        // display menu
        System.out.print("""
                \n*****************************************************
                *                  MENU OPTIONS                      *
                *****************************************************
```

```
                    01. Buy a Seat
                    02. Cancel a Seat
                    03. Find First Available Seat
                    04. Show Seat Planning
                    05. Print ticket information and total sales
                    06. Search ticket
                    00. Quit
                ****************************************************
                """);

        int option = 0;
        boolean isValidInput = false;

        while (!isValidInput) {// ask input from user until entered correct
option
            try {
                System.out.print("Please select an option : ");
                option = scanner.nextInt();

                if (option >= 0 && option <= 6) {
                    isValidInput = true;
                } else {
                    System.out.println("Invalid option Please Try Again !");
                }
            } catch (InputMismatchException e) {
                System.out.println("Invalid input. Please enter a valid
integer.");
                scanner.nextLine();
            }
        }
        switch (option) {// switch case to call right method
            case 1 -> buySeat();
            case 2 -> cancelSeat();
            case 3 -> find_first_available();
            case 4 -> show_seating_plan();
            case 5 -> printTicketInfo();
            case 6 -> searchTicket();
            case 0 -> { return; }
            default -> System.out.println("Invalid option. Please select
again.");
        }
        displayMenu();
    }
    public static void buySeat() {
        Ticket ticket = new Ticket();// create reference variable to access
Ticket class
        String rowLetter = ticket.getRowLetter();
        int seatNumber = ticket.getSeatNumber();

        // convert unicode to access rows in seatStructure 2D array
        char row = Character.toUpperCase(rowLetter.charAt(0));
        int rowArrayIndex = row - 'A';

        if (seatStructure[rowArrayIndex][seatNumber - 1] == 0) {// check
availability of seat
            seatStructure[rowArrayIndex][seatNumber - 1] = 1;
```

```
            Person person = new Person();// Create reference variable to
access Person class
            System.out.print("\nSeat number " + seatNumber + " in Row " +
rowLetter + " is received for ");
            person.personInfo();

            ticket.setRowLetter(rowLetter);
            ticket.setSeatNumber(seatNumber);
            ticket.setPerson(person);

            totalPrice += ticket.getPriceTag();//Update total price when
ticket has been sold

            tickets[rowArrayIndex][seatNumber - 1] = ticket;//store ticket
details in ticket Array

            ticket.save();// Save a text file when a seat is booked

        } else {
            System.out.println("Seat in a Row " + rowLetter + ", the seat
number " + seatNumber + " is already taken.");
        }
    }
    public static void cancelSeat(){
        Ticket ticket = new Ticket();// create reference variable to access
Ticket class
        String rowLetter = ticket.getRowLetter();
        int seatNumber = ticket.getSeatNumber();

        // convert unicode to access rows in seatStructure 2D array
        char row = Character.toUpperCase(rowLetter.charAt(0));
        int rowArrayIndex = row - 'A';

        if (seatStructure[rowArrayIndex][seatNumber-1]==1) {// check
availability of seat
            seatStructure[rowArrayIndex][seatNumber-1]=0;

            System.out.println("You have successfully canceled Seat number "
+ seatNumber + " in row " + rowLetter );

            tickets[rowArrayIndex][seatNumber-1]= null;//Update ticket
details in ticket Array to null

            totalPrice-= ticket.getPriceTag();//Update total price when
ticket has been canceled

            ticket.delFile();// Delete a text file when a seat is canceled
        } else {
            System.out.println("Seat in a Row " + rowLetter + ",the  seat
number " + seatNumber + " is available at the moment.");
        }
    }
    public static void show_seating_plan() {// Show seaplane according to
seatStructure
        System.out.println("\nSeat Plan");
        for (int i = 0; i < rows.length; i++) {
            System.out.print(rows[i] + " : ");
```

```
            for (int seat : seatStructure[i]) {
                System.out.print(seat == 0 ? "O " : "X ");// display O for 0
and X for 1 in seatStructure
            }
            System.out.println();
        }
    }
    public static void printTicketInfo() {// print every ticket information
        for (Ticket[] rowTickets : tickets) {
            for (Ticket ticket : rowTickets) {
                if (ticket != null) {// ignore null ticket in ticket array
                    ticket.ticketInfo();
                }
            }
        }
        System.out.println("Total Price is :" +totalPrice);// display updated
total price
    }
    public static void searchTicket() {
        String rowLetter = Ticket.validateRow();
        int seatNumber = Ticket.setValidSeatNum(rowLetter);

        // convert unicode to access rows in seatStructure 2D array
        char row = Character.toUpperCase(rowLetter.charAt(0));
        int rowArrayIndex = row - 'A';

        Ticket ticket = tickets[rowArrayIndex][seatNumber - 1];

        if (ticket != null) {
            ticket.ticketInfo();// display details of ticket
        } else {
            System.out.println("Seat " + seatNumber + " in Row " + rowLetter
+ " is available at the moment.");
        }
    }
    public static void find_first_available() {// find first available seat
using linear search
        for (int i = 0; i < rows.length; i++) {
            char row = rows[i];
            int[] seatRow = seatStructure[i];
            for (int j = 0; j < seatRow.length; j++) {
                if (seatRow[j] == 0) {
                    System.out.println("First available seat in row " + row +
" is seat number " + (j + 1));
                    return;
                }
            }
        }
        System.out.println("No available seats found.");
    }
    public static void deleteAllFilesAtBegin() {// delete all text file when
program is start
        for (char j : rows) {
            for (int k = 0; k < seatStructure[0].length; k++) {
                if (j == 'B' || j == 'C') {
                    if (k == 12 || k == 13) {
                        continue;
```

```
                    }
                }
                String filename = (j + String.valueOf(k + 1));// generate all
text files can be existed

                File file = new File(filename + ".txt");
                if (file.exists()) {// check whether file is existing
                    file.delete();// delete file
                }
            }
        }
    }

}
```

02. Person Class

```
import java.util.Scanner;
public class Person {
    private String name;
    private String surName;
    private String mail;
    // Create a Constructor
    public Person(){
        this.setName();
        this.setSurName();
        this.setMail();
    }
    // create getters and setters to access properties of person class
    private void setName() {
        this.name = capitalizeInput(askInput("First Name"));
    }
    public String getName() {
        return name;
    }
    private void setSurName() {
        this.surName = capitalizeInput(askInput("Surname"));
    }
    public String getSurName() {
        return surName;
    }
    private void setMail() {
        this.mail = validateEmail(askInput("Email Address"));
    }
    public String getMail() {
        return mail;
    }
    public void personInfo() {// to display person information
        System.out.printf("%s %s (%s)%n", getName(), getSurName(),
getMail());
    }
    private String capitalizeInput(String input){// change input name surname
first letter to Uppercase
```

```
        if (input.isEmpty()){
            return input;
        } else {
            return Character.toUpperCase(input.charAt(0)) +
input.substring(1);
        }
    }
    private String askInput(String field) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter " + field + ": ");
        return scanner.next();
    }
    private String validateEmail(String email) {// use regex expressions for
email validation
        if (email.matches("[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,}"))
{
            return email;
        } else {
            System.out.println("Invalid email address. Please enter again.");
            return validateEmail(askInput("Email Address"));
        }
    }
}
```

03. Ticket Class

```
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.util.InputMismatchException;
import java.util.Scanner;
public class Ticket {
    private String rowLetter;
    private int seatNumber;
    private double priceTag;
    private Person person;
    public Ticket() {// Create a Constructor
        this.rowLetter = validateRow();
        this.seatNumber = setValidSeatNum(rowLetter);
        this.priceTag = setPriceTag(seatNumber);
        this.person = null;
    }
    // create getters and setters to access properties of person class
    public void setPerson(Person person){
        this.person = person;
    }
    public void setRowLetter(String rowLetter) {
        this.rowLetter = rowLetter;
    }
    public void setSeatNumber(int seatNumber) {
        this.seatNumber = seatNumber;
        this.priceTag = setPriceTag(seatNumber);
```

```
    }
    private double setPriceTag(int seatNumber) {// select price according to
seat number
        if (seatNumber >= 1 && seatNumber <= 5) {
            return 200.00;
        } else if (seatNumber >= 6 && seatNumber <= 9) {
            return 150.00;
        } else {
            return 180.00;
        }
    }
    public String getRowLetter() {
        return rowLetter;
    }
    public int getSeatNumber() {
        return seatNumber;
    }
    public double getPriceTag() {
        return priceTag;
    }
    public Person getPerson() {
        return person;
    }
    public static String validateRow() {// check validation of row letter
        Scanner scanner = new Scanner(System.in);
        String rowLetter;

        while (true) {
            System.out.print("Enter row (A, B, C, D): ");
            rowLetter = scanner.next().toUpperCase();

            if (rowLetter.matches("[A-D]")) {
                return rowLetter;
            } else {
                System.out.println("Invalid Row , Please try again !");
            }
        }
    }
    public static int setValidSeatNum(String rowLetter) {// check validation
of seat number
        boolean isNum = false;
        int seatNumber = 0;

        while (!isNum) {
            Scanner scanner = new Scanner(System.in);
            System.out.print("Enter Seat Number: ");
            try {
                seatNumber = scanner.nextInt();
                if (rowLetter.equals("A") || rowLetter.equals("D")) {
                    if (seatNumber >= 1 && seatNumber <=
Main.seatStructure[0].length) {
                        isNum = true;
                    } else {
                        System.out.println("Invalid seat number, Please try
again !");
                    }
                } else {
```

```
                    if (seatNumber >= 1 && seatNumber <=
Main.seatStructure[1].length) {
                        isNum = true;
                    } else {
                        System.out.println("Invalid seat number, Please try
again !");
                    }
                }
            } catch (InputMismatchException e) {
                System.out.println("Invalid input! Please enter a valid
integer.");
                scanner.nextLine();
            }
        }
        return seatNumber;
    }
    public void ticketInfo() { // display ticket information details
        String info = """
        Ticket Information:
        -------------------
        Row Letter: %s
        Seat Number: %d
        Price Tag: $%.2f
        Person Information:
        Name: %s
        Surname: %s
        Email: %s
        -------------------
        """;

        System.out.printf(info, getRowLetter(), getSeatNumber(),
getPriceTag(),
                getPerson().getName(), getPerson().getSurName(),
getPerson().getMail());
    }
    public void save() { // save a text file with ticket details
        try{
            FileWriter file = new
FileWriter(getRowLetter()+getSeatNumber()+".txt");
            file.write("Ticket information:\n");
            file.write("Row "+getRowLetter()+"\n");
            file.write("Seat "+getSeatNumber()+"\n");
            file.write("Price £"+getPriceTag()+"\n\n");

            file.write("Person information:\n");
            file.write("Name "+getPerson().getName()+"\n");
            file.write("Surname "+getPerson().getSurName()+"\n");
            file.write("Email "+getPerson().getMail()+"\n");
            file.close();

        }
        catch (IOException e){
            System.out.println("Error while writing to a file");
        }
    }
    public void delFile(){// delete file when cancel a ticket
        File file = new File(getRowLetter()+getSeatNumber()+".txt");
```

```
        file.delete();
    }


}
```

<<END>>