Informatics Institute of Technology

School of Computing

Software Development II Coursework Report

Module : 4COSC010C.2: Software Development II (2023)

Date of submission : 25.03.2024

Student ID : 20230614 / w2053166

Student First Name : Pamodi

Student Surname : Pansiluni

Tutorial group (day, time, and tutor/s): G20 – Monday – 10.30 a.m – 12.30 p.m.

Ms.Lakna Gammeda

Mr.Dilshard Ahamed

"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 : Pamodi Pansiluni

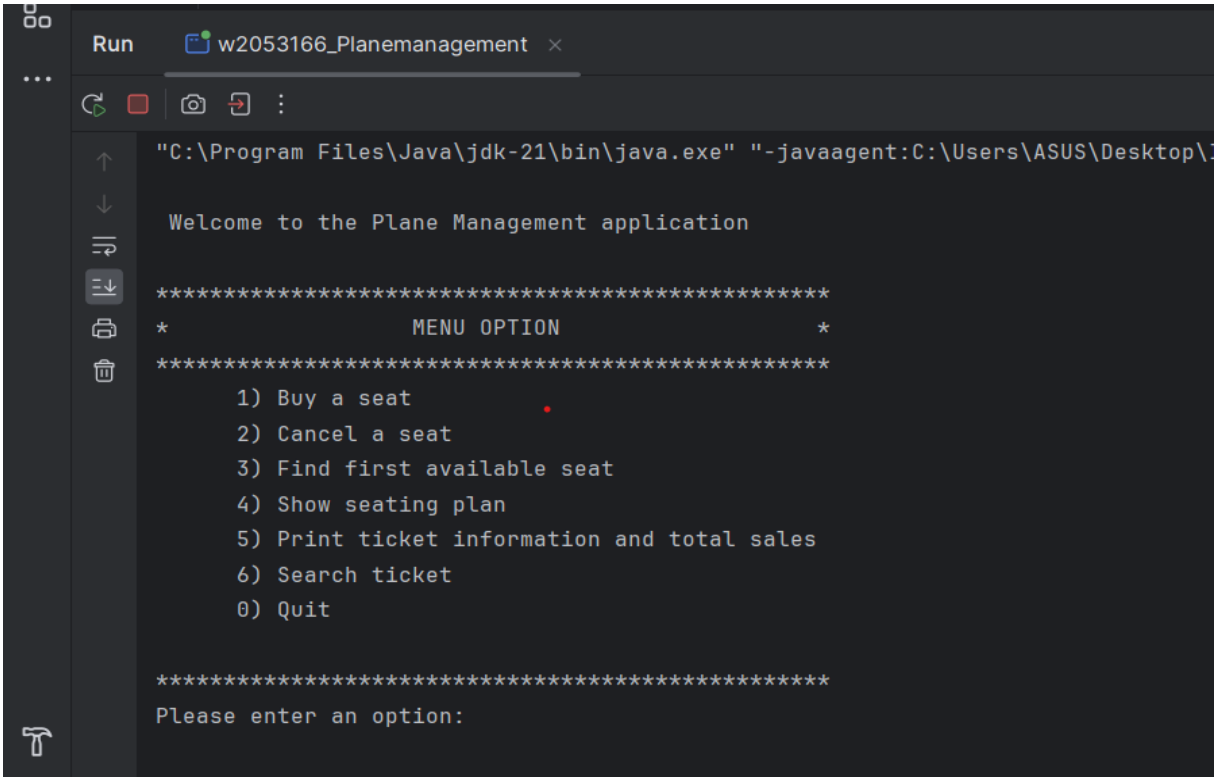
Student ID : 20230614 / w2053166

Self-assessment form and test plan

1) Self-assessment form

Task	Self-assessment (select one)	Comments
1	☒Fully implemented ☐Partially implemented ☐Not attempted	Welcome message is displayed when the main method runs
2	☒Fully implemented ☐Partially implemented ☐Not attempted	Display the user menu and ask the user to enter an option. Program terminates when the user enter 0.

Insert here a screenshot of your welcome message and menu:



3	☒Fully implemented ☐Partially implemented ☐Not attempted	When user enter '1', method buy_seat is called and booked the seat if it is available.
4	☒Fully implemented ☐Partially implemented ☐Not attempted	When the user enter '2', method cancel_seat is called and if the seat is booked cancel the booking and make the seat available.

5	☒Fully implemented ☐Partially implemented ☐Not attempted	When the user enter ‘3’, method findFirstAvailableSeat called and give the first available seat as the output.
6	☒Fully implemented ☐Partially implemented ☐Not attempted	When the user enter ‘4’, the method show_seating_plan called and display the available seats by character ‘O’ and the sold seats by character ‘X’ .

Insert here a screenshot of the seating plan:

```
**************************************************
Please enter an option: 4
Seating Plan:
O O O O O O O O O O O O O O
O O O O O O O O O O O O
O O O O O O O O O O O O
O O O O O O O O O O O O O O
```

7	☒Fully implemented ☐Partially implemented ☐Not attempted	Person class is created by the given attributes. Constructors, getters and setters also created.
8	☒Fully implemented ☐Partially implemented ☐Not attempted	Ticket class is created by the given attributes.Constructors,getter, setters also created.
9	☒Fully implemented ☐Partially implemented ☐Not attempted	Created a array of tickets. When buying and cancelling seats it add and remove the tickets from the array respectively by extending the methods.
10	☒Fully implemented ☐Partially implemented ☐Not attempted	Method print_ticket_info is called when the user enter ‘5’, and it prints ticket information of booked seats and total sales.
11	☒Fully implemented ☐Partially implemented ☐Not attempted	Method search_ticket is called when the user enter ‘6’, and it search the ticket given y the user and notify the information.
12	☒Fully implemented ☐Partially implemented ☐Not attempted	Save method save ticket information in to a text file when the user successfully booked a seat.

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**
Welcome message	Run	Welcome to the Plane Management application.	Welcome to the Plane Management application.	☒Pass ☐Fail
Invalid menu option	Option = h	Invalid option / error message	Invalid option. Please.try again.	☒Pass ☐Fail
Invalid menu option	Option = 8	Invalid option / error message	Invalid option. Option must be a number.	☒Pass ☐Fail
Program Termination	Option = 0	Exit	Exit	☒Pass ☐Fail
Buy seat	1. Option = 1 Row = M 2. Option = 1 Row = 2 3. Option = 1 Row = C Seat = 13 4. Option = 1 Row = D Seat = P 5. Option = 1 Row = A Seat = 11	1. Invalid option / error message. 2. Invalid option / error message. 3. Invalid option / error message 4. Invalid option / error message 5. Seat successfully booked.	1. Invalid row letter. Please enter again. 2. Invalid row letter. Please enter again. 3. Invalid seat number. Please enter again. 4. Invalid seat number. Please enter again. 5. Seat successfully booked.	☒Pass ☐Fail
Booking a seat that already booked	Option = 1 Row = A Seat = 11	Seat is already booked.	Sorry, this seat is already booked.	☒Pass ☐Fail

Cancel seat	1. Option = 2 Row = C Seat = 5 2. Option = 1 Row = A Seat = 11	1. Seat is not booked. 2. Seat successfully cancelled.	1. This seat is already available. Please enter another seat number. 2. Seat successfully cancelled	☒Pass ☐Fail
Find first available seat	Option = 3	Note : if A1 is booked First available seat is A2	First available seat : A2	☒Pass ☐Fail
Show seat plan	Option = 4	Print available seats with 'O' and sold seats with 'X'	Print available seats with 'O' and sold seats with 'X'	☒Pass ☐Fail

Part B testing

Test case / scenario	**Input**	**Expected Output**	**Output**	**Pass/Fail**
Buy a seat	Option = 1 Row = B Seat = 10	Display the prompt to get the user information and add into the array.	Display the prompt to get the user information and add into the array.	☒Pass ☐Fail
Cancel seat	Option = 2 Row = B Seat = 10	Seat cancelled. Information get removed from the array.	Seat cancelled. Information get removed from the array.	☒Pass ☐Fail
Print ticket info	Option = 5	Print all information of booked tickets and display total sales.	Print all information of booked tickets and display total sales.	☒Pass ☐Fail
Search ticket	Option = 6 Row = C Seat = 12	Print the information of the ticket and the status.	Print the information of the ticket and the status.	☒Pass ☐Fail

Save	Option = 1 Row = B Seat = 6 Name = Pamodi Surname = Pansiluni Email = pamodipansiluni@gmail.com	Save information of each seat in separate text file when the seat booked successfully.	Print all information of booked tickets and display total sales.	☒Pass ☐Fail
Cancel and delete seat	Option = 2 Row = B Seat = 6	Delete the text file when the seat successfully cancelled	Delete the text file when the seat successfully cancelled	☒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 :

W2053166_PlaneManagement.java

```
import java.util.InputMismatchException;

import java.util.Scanner;

public class w2053166_Planemanagement {
    public static Scanner input = new Scanner(System.in);
    //Define 2D array to represent the seating plan
    public static int [][] seat_plan = new int[4][];

    //Define an array to store ticket information with the size of 52
    private static final Ticket[] tickets = new Ticket[52];
    private static int ticketIndex = 0;

    public static void main(String[] args) {
        //Initialization of the seating arrangement for the each row of
seats
        seat_plan[0] = new int[14];
        seat_plan[1] = new int[12];
        seat_plan[2] = new int[12];
        seat_plan[3] = new int[14];

        System.out.println("\n   Welcome   to   the   Plane   Management
application");

        //Menu
        do {
            System.out.println();
            System.out.println(num(50));
            System.out.println("*                              MENU  OPTION
*");
            System.out.println(num(50));
```

```
System.out.println("""
              1) Buy a seat\s
              2) Cancel a seat
              3) Find first available seat
              4) Show seating plan
              5) Print ticket information and total sales
              6) Search ticket
              0) Quit\s
        """);
System.out.println(num(50));


//Main loop for prompt the options continuesly
try {
    System.out.print("Please enter an option: ");
    int option = input.nextInt();

    switch (option) {
        case 0:
            System.out.println("Exit");
            return;
        case 1:
            buy_seat(input);
            break;
        case 2:
            cancel_seat(input);
            break;
        case 3:
            findFirstAvailableSeat();
            break;
        case 4:
            show_seating_plan();
            break;
        case 5:
            print_tickets_info();
            break;
        case 6:
            search_ticket(input);
```

```
                        break;
                    default:
                        System.out.println("Invalid option. Please try
again.");
                }
            } catch (InputMismatchException e) {
                System.out.println("Invalid option. Option must be a
number.");
                input.nextLine();
            }
        } while (true);
    }

    private static String num(int count) {
        //Recurtion to generate "*" caharacters based on the given count
        if (count == 0) {
            return "";
        } else {
            return "*" + num(count - 1);
        }
    }

    private static void buy_seat(Scanner scanner) {
        //Method to handle the process of buying a seat by user
        char row;
        int seatNumber;
        int max_seat;
        String name, surname, email;

        //Loop to check whether the input row is valid or not
        do {
            System.out.println("Enter the row letter: ");
            row = scanner.next().toUpperCase().charAt(0);

            if (row < 'A' || row > 'D') {
                System.out.println("Invalid row letter. Please enter
again.");
            }
```

```
        } while (row < 'A' || row > 'D');

        max_seat = switch (row) {
            case 'A', 'D' -> 14;
            case 'B', 'C' -> 12;
            default -> 0;
        };

        //Loop to check whether the input seat number is valid or not
        System.out.println("Enter the seat number: ");
        while (true) {
            try {
                seatNumber = scanner.nextInt();
                if (seatNumber < 1 || seatNumber > max_seat) {
                    System.out.println("Invalid   seat   number.   Please
enter again: ");
                } else {
                    break;
                }
            } catch (InputMismatchException e) {
                System.out.println("Invalid  seat  number.  Please  enter
again: ");
                scanner.next();
            }
        }

        //Seat booking process
        if (seat_plan[row - 'A'][seatNumber - 1] == 1) {
            System.out.println("Sorry, This seat is already booked.");
        } else {
            //Inputting customer details
            scanner.nextLine();
            System.out.println("Enter your name: ");
            name = scanner.nextLine();
            System.out.println("Enter surname: ");
            surname = scanner.nextLine();
            System.out.println("Enter email: ");
            email = scanner.nextLine();
```

```
            //Creating Person and Ticket objects
            Person person = new Person(name, surname, email);
            Ticket    ticket    =    new    Ticket(row,    seatNumber,
calculatePrice(seatNumber), person);
            tickets[ticketIndex++] = ticket;

            //Adding ticket to the array
            seat_plan[row - 'A'][seatNumber - 1] = 1;
            System.out.println("Seat   "   +   row   +   seatNumber   +   "
successfully booked");
            ticket.save();
        }
    }

    private static double calculatePrice(int seatNumber) {
        //Method to calculate the price of a seat according to the seat
number
        int price;
        if (seatNumber < 6){
            price = 200;
        }
        else if (seatNumber<10){
            price = 150;
        }
        else {
            price = 180;
        }
        return price;
    }

    private static void cancel_seat(Scanner scanner) {
        char row;
        int seatNumber;
        int max_seat;

        //Loop to check whether the input row is valid or not
        do {
```

```
            System.out.println("Enter the row letter: ");
            row = scanner.next().toUpperCase().charAt(0);
            if (row < 'A' || row > 'D') {
                System.out.println("Invalid  row  letter.  Please  enter
again.");
            }
        } while (row < 'A' || row > 'D');
        max_seat = switch (row) {
            //Idenfifing the maximum seats according the row
            case 'A', 'D' -> 14;
            case 'B', 'C' -> 12;
            default -> 0;
        };

        //Loop to check whether the input seat number is valid or not
        while (true) {
            System.out.println("Enter the seat number: ");
            while (true) {
                try {
                    seatNumber = scanner.nextInt();
                    if (seatNumber < 1 || seatNumber > max_seat) {
                        System.out.println("Invalid seat number. Please
enter again: ");
                    } else {
                        break;
                    }
                } catch (InputMismatchException e) {
                    System.out.println("Invalid   seat   number.   Please
enter again: ");
                    scanner.next();
                }
            }

            //Handling the seat cancelling process
            if (seat_plan[row - 'A'][seatNumber - 1] == 0) {
                System.out.println("This  seat  is  already  available.
Please enter another seat number: ");
            } else {
```

```
                seat_plan[row - 'A'][seatNumber - 1] = 0;
                System.out.println("Seat " + row + seatNumber + " successfully canceled");

                // Removing the cancelled ticket from the array
                for (int i = 0; i < ticketIndex; i++) {
                    if (tickets[i].getRow() == row && tickets[i].getSeat() == seatNumber) {
                        tickets[i] = null;
                        break; // Exiting loop after removing the ticket
                    }
                }
                break; // Exit the loop if the ticket cancelled succesfully
            }
        }
    }

    //Method to find and display the first availabke seat in the seat plan
    public static void findFirstAvailableSeat() {
        char row = 'A'; //Start searching from row 'A'
        int seatNumber = findAvailableSeatInRow(row, 0);

        if (seatNumber == -1) {
            System.out.println("Sorry, all seats are taken.");
        } else {
            System.out.println("First available seat: " + row + seatNumber);
        }
    }


    private static int findAvailableSeatInRow(char row, int seatNumber) {
        //recurtion to find the first available seat in the specified row
        if (seatNumber >= seat_plan.length) {
            return -1;
```

```
        }
        if (seat_plan[row - 'A'][seatNumber] == 0) { // If seat is
available it will represent as 0
            return seatNumber + 1;
        }
        return findAvailableSeatInRow(row, seatNumber + 1); //Continue
the recursion
    }


    private static void show_seating_plan() {
        //Method to display the seat plan
        System.out.println("Seating Plan:");

        for (int[] ints : seat_plan) {
            for (int j = 0; j < ints.length; j++) {
                if (ints[j] == 0) {
                    System.out.print("O   "); //Available seats are
represented by 0
                } else {
                    System.out.print("X  "); // unavailable seats are
represented by X
                }
            }
            System.out.println(); //Move to next row
        }
    }

    private static void print_tickets_info() {
        //Method to display all information about sold tickets and
calculate the total
        double totalSales = 0.0;

        for (Ticket ticket : tickets) {
            if (ticket != null) { // Check if the ticket is not null
                ticket.print_Ticket_Info();  //Print ticket information
                totalSales += ticket.getPrice(); //Adding ticket price to
total sales
```

```
            }
        }
        System.out.println("Total Sales: £" + totalSales);
    }

    private static void search_ticket(Scanner scanner) {
        //Method to search for a ticket by row and seat number
        char row;
        int seatNumber;
        int max_seat;
        String name, surname, email;

        //Loop to check whether the input row is valid or not
        do {
            System.out.println("Enter the row letter: ");
            row = scanner.next().toUpperCase().charAt(0);

            if (row < 'A' || row > 'D') {
                System.out.println("Invalid  row  letter.  Please  enter
again.");
            }
        } while (row < 'A' || row > 'D');

        max_seat = switch (row) {
            case 'A', 'D' -> 14;
            case 'B', 'C' -> 12;
            default -> 0;
        };

        //Loop to check whether the input seat number is valid or not
        System.out.println("Enter the seat number: ");
        while (true) {
            try {
                seatNumber = scanner.nextInt();
                if (seatNumber < 1 || seatNumber > max_seat) {
                    System.out.println("Invalid   seat   number.   Please
enter again: ");
                } else {
```

```
                    break;
                }
            } catch (InputMismatchException e) {
                System.out.println("Invalid seat number. Please enter
again: ");
                scanner.next();
            }
        }

        boolean found = false;
        for (int i = 0; i < ticketIndex; i++) {
            Ticket ticket = tickets[i];
            if (ticket != null && ticket.getRow() == row &&
ticket.getSeat() == seatNumber) {
                ticket.print_Ticket_Info(); //if found print ticket
information
                found = true;
                break;
            }
        }

        if (!found) {
            System.out.println("This seat is available."); //if ticket is
not found, seat is available
        }
    }
}
```

Person.java

```
public class Person {
    //Attributes
    private String name;
    private String surname;
    private String email;

    //Constructor
    public Person(String name, String surname, String email) {
        this.name = name;
        this.surname = surname;
        this.email = email;
    }

    //Getter for name attribute
    public String getName() {
        return name;
    }

    //Setter for name attribute
    public void setName(String name) {
        this.name = name;
    }

    //Getter for the surname attribute
    public String getSurname() {
        return surname;
    }

    //Setter for the surname attribute
    public void setSurname(String surname) {
        this.surname = surname;
    }
```

```
    //Getter for the email attribute
    public String getEmail() {
        return email;
    }

    //Setter for the email attribute
    public void setEmail(String email) {
        this.email = email;
    }

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

Ticket.java

```
import java.io.FileWriter;
import java.io.IOException;

//Attributes
public class Ticket {
    private char row;
    private int seat;
    private double price;
    private Person person;

    //Constructor
    public Ticket(char row, int seat, double price, Person person)
{
```

```
        this.row = row;
        this.seat = seat;
        this.price = price;
        this.person = person;
    }

    //Getter for the row attribute
    public char getRow() {
        return row;
    }

    //Setter for the row attribute
    public void setRow(char row) {
        this.row = row;
    }

    //Getter for the seat attribute
    public int getSeat() {
        return seat;
    }

    //Setter for the seat attribute
    public void setSeat(int seat) {
        this.seat = seat;
    }

    //Getter for the price attribute
    public double getPrice() {
        return price;
    }

    //Setter for the price attribute
    public void setPrice(double price) {
        this.price = price;
    }
```

```
    //Getter for the person attribute
    public Person getPerson() {
        return person;
    }

    //Setter for the person attribute
    public void setPerson(Person person) {
        this.person = person;
    }

    //Method to print the ticket information
    public void print_Ticket_Info() {
        System.out.println("\nTicket Information: ");
        System.out.println("Row: " + getRow());
        System.out.println("Seat: " + getSeat());
        System.out.println("Price: £" + getPrice());
        System.out.print("\nPerson Information: \n");
        person.printInfo();
    }

    //Method to save ticket information
    public void save() {
        String fileName = getRow() + "" + getSeat() + ".txt";
        try (FileWriter writer = new FileWriter(row + "" + seat +
".txt")) {
            writer.write("Ticket Information of seats" + "\n");

            writer.write("\t Row " + getRow() + "\n");
            writer.write("\t Seat " + getSeat() + "\n");
            writer.write ("\t Price £ " + getPrice() + "\n");

            writer.write("\t Person Information: " + "\n");
```

```
            writer.write("\t First name: " + person.getName() +
"\n");
            writer.write("\t Surname: " + person.getSurname() +
"\n");
            writer.write("\t Email: " + person.getEmail() + "\n");

            writer.close(); //close the file writer
        } catch (IOException e) {
            e.printStackTrace(); //handling IO exception
        }
    }
}
```

<<END>>