Informatics Institute of Technology

School of Computing

Software Development II Coursework Report

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

Date of submission : 3/25/2024

Student ID : <20232125> / <w2052215>

Student First Name :Dilshan

Student Surname : Manohara

Tutorial group (day, time, and tutor/s): G-16 / Tuesday / 10.30-12.30 / Mr.Torin Weerasinghe

"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 : Obada Mudalige Dilshan Manohara

Student ID : 20232125

Self-assessment form and test plan

1) Self-assessment form

Task	Self-assessment (select one)	Comments
1	☒Fully implemented ☐Partially implemented ☐Not attempted	Task 1 is done and fully implemented.
2	☒Fully implemented ☐Partially implemented ☐Not attempted	Create a string array to store all the options for the user menu. Use an enhanced for loop to go through each option in the array. Display each option as part of the user menu at the beginning of the program.
Insert here a screenshot of your welcome message and menu:		

```
Welcome to the Plane Management application
**************************************************
*                  MENU OPTIONS                  *
**************************************************
                1) Buy a seat
                2) Cancel a seat
                3) Find first available seat
                4) Show seating plan
                5) Print ticket information and total sales
                6) Search tickets
                0) Quit
**************************************************
Please select an option
```

3	☒Fully implemented ☐Partially implemented ☐Not attempted	In task 3, you'll ask the user to pick a row letter and seat number.Then, you'll check if that seat is available.If it is, you'll book the seat and tell the user it's successful.If it's not available or doesn't exist, you'll tell the user the seat is invalid and can't be booked.
4	☒Fully implemented	After booking a seat, we should be able to cancel it if needed.So,

	☐Partially implemented ☐Not attempted	in this method, we'll provide an option to cancel a previously booked seat.If the user chooses to cancel, we'll make the seat available again.This way, users can change their mind and free up seats they've booked before.
5	☒Fully implemented ☐Partially implemented ☐Not attempted	If the user picks option 3, the program finds the first available seat.This helps users see which seat they can take without checking the whole seating plan.
6	☒Fully implemented ☐Partially implemented ☐Not attempted	At the start of the program, the seating plan of the plane is shown.Users can see which seats are available (empty) and which are not (occupied).Users can also check if their chosen seat has been successfully booked or not.

Insert here a screenshot of the seating plan:

```
Please select an option
4
Seating Plan:
00000000000000
 000000000000
 000000000000
00000000000000
```

7	☒Fully implemented ☐Partially implemented ☐Not attempted	In the Person class, we use getters and setters to manage the data.When a user enters information, like their name or age, these methods store and retrieve that data for us.
8	☒Fully implemented ☐Partially implemented ☐Not attempted	This method prints out all the details of a person and their ticket by accessing the data using getters.
9	☒Fully implemented ☐Partially implemented ☐Not attempted	The buy_seat method extends and takes the person's name, surname, and email as input.The cancel_seat method extends and removes the ticket from the list of tickets when a user cancels their booking.
10	☒Fully implemented ☐Partially implemented ☐Not attempted	This method prints out information about all the tickets that have been sold during the session. It also displays the total number of tickets sold and the total amount of money earned from those tickets during the session.
11	☒Fully implemented ☐Partially implemented ☐Not attempted	In this method, users can input a row letter and seat number to check if someone has bought that seat yet. Additionally, users can view details of all the seats that have been booked.
12	☒Fully implemented ☐Partially implemented ☐Not attempted	When a user books a ticket, the program saves the information in a file. The file's name includes the row and seat number of the booked ticket. For example, if the

		user books row 'B' seat number 1, the file will be named B1.txt and will contain the ticket 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
Task 2). When run the programme		Display menu and, prompt for the user input.	Welcome to the Plane Management application ********** * MENU OPTIONS * ********** 1) Buy a seat 2) Cancel a seat 3) Find first available seat 4) Show seating plan 5) Print ticket information and total sales 6) Search tickets 0) Quit ********** Please select an option	☒Pass ☐Fail
Task 3). User enter the number 1 for his option	1	Display prompt for the row letter: seat number:	Enter the row letter (A-D): Enter the seat number (1-14):	☒Pass ☐Fail
Enter invalid row letter and seat number as input.	Q 20	Display message like "Invalid row or seat number.'	Invalid row or seat number.	☒Pass ☐Fail
Enter valid row letter and seat number	A 3	Display message like "Seat A3 has been successfully"	Seat A3 has been successfully sold.	☒Pass ☐Fail

as input.		sold.		
Task 4). The user enter the number 2 for his option Enter invalid row letter and seat number as input.	2	Prompt for the enter row letter and seat Number.Then display “Seat (ex:A4) canceled successfully. Prompt for the enter row letter and seat Number.Then display “Seat (ex:Q4) canceled successfully.	Seat A4 canceled successfully. Invalid row letter. Please try again.	☒Pass ☐Fail
Task 5). User enter the number 3 for his option	3	Display a message like this: “First available seat found: A1” (if A1 is not booked)	First available seat found: A1	☒Pass ☐Fail
Task 6). User enter the number 4 for his option in beginning	4	Display a message “Seating Plan” and display seating plan.	Seating Plan: OOOOOOOOOOOOOO OOOOOOOOOOOO OOOOOOOOOOOO OOOOOOOOOOOOOO	☒Pass ☐Fail

Part B testing

Test case / scenario	**Input**	**Expected Output**	**Output**	**Pass/Fail**
Task 7) Create a new class file called Person and add a method that prints the information from person.		Print the information about the person with ticket information when the user enters the number ‘5’ as an option.	Enter person's name: Enter person's surname: Enter person's email:	☒Pass ☐Fail
Task 8). Create a new class file called Ticket with row, seat number and Person.		Print the information about the person with ticket information when the user enters the number ‘5’ as an option.	Enter the row letter (A-D): Enter the seat number (1-14): Enter person's name: Enter person's surname: Enter person's email:	☒Pass ☐Fail
Task 9). 1). Extend the buy seat method, when buying a ticket, it asks for all information of a Person.	If you are entered the valid correct input as a input like this: Insert your first name: Dilshan Insert your surname:	valid correct input as a input like this: Insert your first name: Dilshan Insert your surname: Manohara Insert your email:	Seat A3 has been successfully sold.	☒Pass ☐Fail

Insert your first name: Insert your surname: Insert your email:	Manohara Insert your email: kkk@gmail.com	kkk@gmail.com Display a message like this: “Seat(ex: A3) has been successfully sold.!”		
Task 9) 2). Extend the cancel seat method such that when cancelling a seat a ticket removes the ticket from the array list of ticket.	in the beginning you buy Row A seat number 1, now you give these Row letter and seat number as the output in here… Row letter: A Seat number: 1	Display a message like this: “Seat A1 canceled successfully.”	Seat A1 canceled successfully.	☒Pass ☐Fail
Task 10). print_tickets_info that prints the information of all tickets that have been sold during the session	In the beginning, you buy Row A seat number 1, and gives these inputs in there Insert your first name: Dilshan Insert your surname: Manohara Insert your email: kkk@gmail.com then you enter number ‘5’ in here.. input : ‘5’	Enter the row letter (A-D): A Enter the seat number (1-14): 1 Enter person's name: Dilshan Enter person's surname: Manohara Enter person's email: kkk@gmail.com Ticket information saved to file: A1.txt Seat A1 has been	Enter the row letter (A-D): A Enter the seat number (1-14): 1 Enter person's name: Dilshan Enter person's surname: Manohara Enter person's email: kkk@gmail.com Ticket information saved to file: A1.txt Seat A1 has been	☒Pass ☐Fail

		successfully sold.	successfully sold.	
Task 11). Method search ticket asks the user to input a row letter and a seat number and searches if someone has bought that seat.	In the beginning, you buy Row A seat number 1, and give these inputs in there Row letter: A Seat number: 1	Display ticket information like this: Please select an option 6 Enter the row letter (A-D): A Enter the seat number (1-14): 1 Ticket: A1 - Price: €200.0 - Passenger: Dilshan Manohara - Email: kkk@gmail.com	Please select an option 6 Enter the row letter (A-D): A Enter the seat number (1-14): 1 Ticket: A1 - Price: €200.0 - Passenger: Dilshan Manohara - Email: kkk@gmail.com	☒Pass ☐Fail
Task 12). Add a Method save in the class ticket that saves the information of the ticket every time a ticket is sold.	In the beginning, you buy Row A seat number 1, then create a file the name of the row and the seat number.	Display a message like this ,when end of the buy seat, Ticket information saved to file: A1.txt	Enter the row letter (A-D): A Enter the seat number (1-14): 1 Enter person's name: Dilshan Enter person's surname: Manohara Enter person's email: kkk@gmail.com Ticket information saved to file: A1.txt Seat A1 has been successfully sold.	☒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 :

planemanagement.java

```
import java.util.InputMismatchException;

import java.util.Scanner;


public class w2052215_PlaneManagement {


    private Ticket ticket;

    private static int[][] seats = {

            {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

            {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

            {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0},

            {0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0}

    };

    private static Ticket[][] tickets = new Ticket[4][14]; // 2D array to store tickets


    public static void main(String[] args) {

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

        Scanner scanner = new Scanner(System.in);


        int option = 1; //1 is set to options initially to ensure that the loop runs at least onc
```

```
do {
    System.out.println("*************************************************");
    System.out.println("*                MENU OPTIONS                *");
    System.out.println("*************************************************");
    System.out.println("      1) 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 ticket information and total sales");
    System.out.println("      6) Search tickets");
    System.out.println("      0) Quit");
    System.out.println("*************************************************");
    System.out.println("Please select an option");

    try {
        option = scanner.nextInt();

        switch (option) {
            case 1:
                buySeat(scanner);
```

```
    break;
case 2:
    cancelSeat(scanner);
    break;
case 3:
    findAvailableSeat();
    break;
case 4:
    displaySeats();
    break;
case 5:
    printTicketInfo();
    break;
case 6:
    searchTickets(scanner);
    break;
case 0:
    System.out.println("Exiting program");
    break;
default:
    System.out.println("Invalid option. Please try again.");
```

```
            }
        } catch (InputMismatchException ex) {
            System.out.println("Invalid input. Please enter a valid number.");
            scanner.nextLine(); // Clear the input buffer
        }
    } while (option != 0);
}

public static void buySeat(Scanner scanner) {
    // Ask the user for input
    System.out.println("Enter the row letter (A-D): ");
    char rowLetter = scanner.next().toUpperCase().charAt(0); // Convert input to uppercase
    System.out.println("Enter the seat number (1-14): ");
    int seatNumber = scanner.nextInt();



    // Validate row and seat number
    if(rowLetter < 'A' || rowLetter > 'D'){System.out.println("Invalid row or seat number.");
        return;
    }
```

```
else if(rowLetter =='A' || rowLetter =='D'){
    if(seatNumber < 1 || seatNumber >14){
        System.out.println("Invalid row or seat number.");
        return;
    }
}else if(rowLetter == 'B' || rowLetter == 'C'){
    if(seatNumber < 1 || seatNumber > 12){
        System.out.println("Invalid row or seat number.");
        return;
    }
}


// Convert row letter to array index
int row = rowLetter - 'A';


// Check if the seat is available
if (seats[row][seatNumber - 1] == 1) {
    System.out.println("Seat is already occupied. Please choose another seat.");
} else {
    // Mark the seat as sold
```

```
seats[row][seatNumber - 1] = 1;


// Ask for person information

System.out.println("Enter person's name:");

String name = scanner.next();

System.out.println("Enter person's surname:");

String surname = scanner.next();

System.out.println("Enter person's email:");

String email = scanner.next();


// Create a Person object

person person = new person(name, surname, email);


// Define price based on seat location

double price = calculatePrice(row, seatNumber);


// Create a Ticket object

Ticket ticket = new Ticket(row, seatNumber, price, person);


// Add the ticket to the tickets array

tickets[row][seatNumber - 1] = ticket;
```

```
        ticket.save();


        System.out.println("Seat " + rowLetter + seatNumber + " has been successfully sold.");
    }
}


private static double calculatePrice(int row, int seatNumber) {
    double price = 0;
    if(seatNumber <= 5) {
        price = 200;
    }
       else if(seatNumber >= 6 && seatNumber <= 9){
            price = 150;
        }
            else{
                price = 180;
    }
    return price;
}
```

```
private static void cancelSeat(Scanner scanner) {

    // Implement cancelSeat method to remove the ticket from the array of tickets
    System.out.println("Enter row letter (A-D): ");
    char rowLetter = scanner.next().toUpperCase().charAt(0);
    int row = rowLetter - 'A';
    if (row < 0 || row >= seats.length) {
        System.out.println("Invalid row letter. Please try again.");
        return;
    }

    System.out.println("Enter seat number: ");
    int seatNumber = scanner.nextInt();

    // Validate row and seat number
    if (seatNumber < 1 || seatNumber > seats[row].length) {
        System.out.println("Invalid seat number. Please try again.");
        return;
    }
```

```
    // Check if the seat is available
    if (seats[row][seatNumber - 1] == 0) {
        System.out.println("Seat is already available. Please choose another seat to cancel.");
    } else {
        seats[row][seatNumber - 1] = 0; // Mark the seat as available
        tickets[row][seatNumber - 1] = null; // Remove the ticket from the tickets array
        System.out.println("Seat " + rowLetter + seatNumber + " canceled successfully.");
    }
}

private static void findAvailableSeat() {
    boolean seatFound = false;

    for (int i = 0; i < seats.length; i++) {
        for (int j = 0; j < seats[i].length; j++) {
            if (seats[i][j] == 0) {
                char rowLetter = (char) ('A' + i);
                System.out.println("First available seat found: " + rowLetter + (j + 1));
                seatFound = true;
                break;
            }
```

```
        }
        if (seatFound) {
            break;
        }
    }

    if (!seatFound) {
        System.out.println("No available seats found.");
    }
}

private static void displaySeats() {
    System.out.println("Seating Plan:");

    for (int i = 0; i < seats.length; i++) {
        if (i == 1 || i == 2) {
            System.out.print(" "); // Add a space at the start of row B and C
        }

        for (int j = 0; j < seats[i].length; j++) {
            if (seats[i][j] == 0) {
```

```
            System.out.print("O");
        } else {
            System.out.print("X");
        }
    }
    System.out.println(); // Move to the next row after printing seats in the current row
  }
}

private static void printTicketInfo() {
  double totalPrice = 0;

  // Iterate through the tickets array to print ticket information and calculate total price
  for (int i = 0; i < tickets.length; i++) {
    for (int j = 0; j < tickets[i].length; j++) {
      if (tickets[i][j] != null) { // Check if the seat is sold
        Ticket ticket = tickets[i][j];
        char rowLetter = (char) ('A' + ticket.getRow());
        int seatNumber = ticket.getSeat() + 1;

        // Print ticket information
```

```
            System.out.println("Ticket: " + rowLetter + seatNumber +
                    " - Price: €" + ticket.getPrice() +
                    " - Passenger: " + ticket.getPerson().getName() +
                    " " + ticket.getPerson().getSurname() +
                    " - Email: " + ticket.getPerson().getEmail());

            // Calculate total price
            totalPrice += ticket.getPrice();
        }
    }
  }

  // Print total price
  System.out.println("Total price of tickets sold during the session: €" + totalPrice);
}

private static void searchTickets(Scanner scanner) {
  // Ask the user to input a row letter and seat number
  System.out.println("Enter the row letter (A-D): ");
  char rowLetter = scanner.next().toUpperCase().charAt(0); // Convert input to uppercase
  System.out.println("Enter the seat number (1-14): ");
```

```
int seatNumber = scanner.nextInt();


// Validate row and seat number
if (rowLetter < 'A' || rowLetter > 'D' || seatNumber < 1 || seatNumber > 14) {
    System.out.println("Invalid row or seat number.");
    return;
}


// Convert row letter to array index
int row = rowLetter - 'A';


// Check if the seat is sold
if (tickets[row][seatNumber - 1] != null) {
    // Seat is sold, print ticket and person information
    Ticket ticket = tickets[row][seatNumber - 1];
    System.out.println("Ticket: " + rowLetter + seatNumber +
            " - Price: €" + ticket.getPrice() +
            " - Passenger: " + ticket.getPerson().getName() +
            " " + ticket.getPerson().getSurname() +
            " - Email: " + ticket.getPerson().getEmail());
} else {
```

```
            // Seat is available
            System.out.println("This seat is available.");
        }
    }

}
```

Ticket.java

```
  private int row;
    private int seat;
    private double price;
```

```
private person person;

// Constructor
public Ticket(int row, int seat, double price, person person) {
    this.row = row;
    this.seat = seat;
    this.price = price;
    this.person = person;
}

// Getters and Setters
public int getRow() {
    return row;
}

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

public int getSeat() {
    return seat;
```

```
}

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

public double getPrice() {
    return price;
}

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

public person getPerson() {
    return person;
}

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

```
// Constructor, getters, and setters


public void save() {
    String filename = (char) ('A' + row) + String.valueOf(seat) + ".txt";


    try {
        FileWriter writer = new FileWriter(filename);
        writer.write("Ticket Information\n");
        writer.write("Row: " + (char) ('A' + row) + "\n");
        writer.write("Seat: " + seat + "\n");
        writer.write("Price: €" + price + "\n");
        writer.write("Passenger: " + person.getName() + " " + person.getSurname() + "\n");
        writer.write("Email: " + person.getEmail() + "\n");
        writer.close();
        System.out.println("Ticket information saved to file: " + filename);
    } catch (IOException e) {
        System.out.println("An error occurred while saving the ticket information to file.");
        e.printStackTrace();
    }
}
```

```
}
```

Person.java

```
public class person {
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;
}

// Getters and Setters
public String getName() {
    return name;
}

public void setName(String name) {
    this.name = name;
}

public String getSurname() {
    return surname;
}
```

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

    public String getEmail() {
        return email;
    }

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

}
```

<<END>>