UNIVERSITY OF WESTMINSTER

INFORMATICS INSTITUTE OF TECHNOLOGY

Informatics Institute of Technology

School of Computing

Software Development II Coursework Report

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

Date of submission : 24.03.2023

Student ID : <IIT No> / <UOW No>

Student First Name : Thishakya

Student Surname : Wickramasinghearachchi

Tutorial group (day, time, and tutor/s): G 6 Tuesday: 8.30am – 10.30am

Torin Weerasinghe

Shiyam Baasith

"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 : Thishakya Dewwandi Wickramsinghearachchi

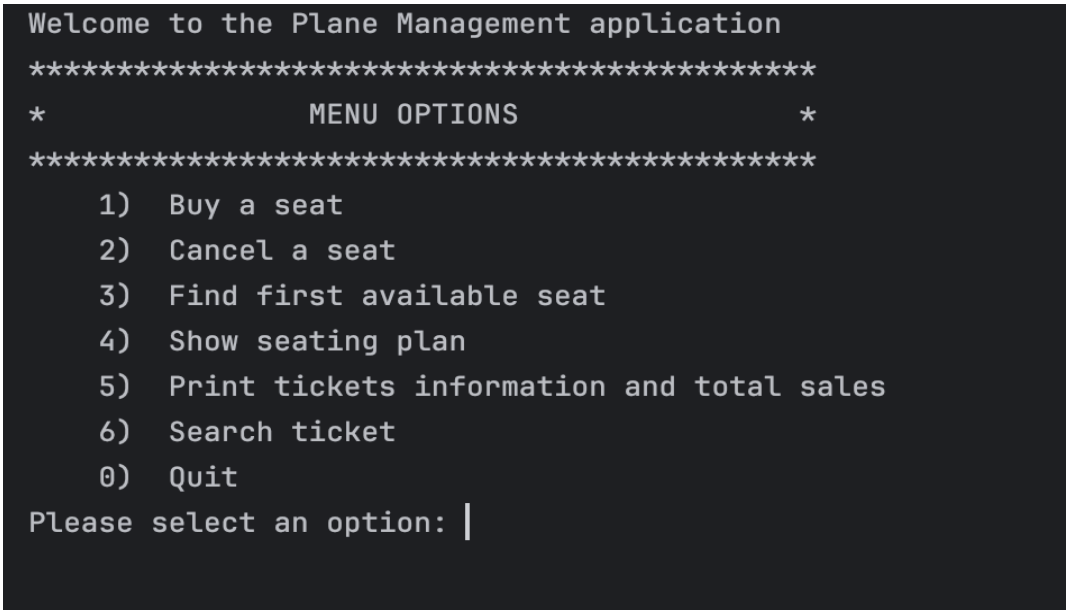
Student ID : w2051952

Self-assessment form and test plan

1) Self-assessment form

Task	Self-assessment (select one)	Comments
1	☒Fully implemented ☐Partially implemented ☐Not attempted	→ A new class called "w2051952_PlaneManagement" is created. → Prints out the welcome message, when the main method is called. → Creates a two-dimensional array for seats.
2	☒Fully implemented ☐Partially implemented ☐Not attempted	→ Displays a menu for the user. → Allows the user to select an option and the respective method is called based on the option selected, with the use of a switch case. → Terminates the do-while loop and switch case when the user enters, '0'.

Insert here a screenshot of your welcome message and menu:



3	☒Fully implemented ☐Partially implemented ☐Not attempted	→ This method is called when the user enters option '1'. → Allows the user to enter the row letter and seat no to be bought, checks if it is valid and available. If so, the seat can be booked successfully.
4	☒Fully implemented ☐Partially implemented ☐Not attempted	→ This method is called when the user enters option '2'. → Allows the user to enter the row and seat no of the seat to be cancelled and checks if it is valid and already booked. If so, the seat can be successfully cancelled.
5	☒Fully implemented ☐Partially implemented ☐Not attempted	→ This method is called when the user enters option '3'. → Allows the user to find the first seat available by traversing through the seats array.
6	☒Fully implemented ☐Partially implemented ☐Not attempted	This method is called when the user enters option '4'. This demonstrates the seats available with character 'O' and the seats sold with character 'X'.

Insert here a screenshot of the seating plan:

This picture shows when seat A5 and C8 is purchased.

```
Please select an option: 4
Seating Plan:
 O  O  O  O  X  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  X  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	→ A class called “Person” is created and the respective attributes, constructors, getters and setters are added. → The personal_infromation() method prints out the personal information of the passenger.
8	☒Fully implemented ☐Partially implemented ☐Not attempted	→ A class called “Ticket” is created and the respective attributes, constructors, getters and setters are added. → The ticket_information() method prints out the personal (calling personal_information() from Person class)and ticket information of the passenger.
9	☒Fully implemented ☐Partially implemented ☐Not attempted	→ Gets the personal information of the passenger. → Calculates the price based on the seat booked. → The relevant information is added to the tickets array. → If seat is cancelled the ticket and personal information are removed.
10	☒Fully implemented ☐Partially implemented ☐Not attempted	→ This method is called, printing the ticket information and total sales, when option ’5’ is selected. → Makes use of two "For Loops" to traverse through the two-dimensional array for tickets.
11	☒Fully implemented ☐Partially implemented ☐Not attempted	→ This method is called, allowing the user to check if a particular seat is booked or not, when option ‘6’ is selected. → This is allowed if the user enters a valid row letter and seat no only.

12	☒Fully implemented ☐Partially implemented ☐Not attempted	→ Saves the ticket information to a separate 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**
Run the program	Click run	Welcome to the Plane Management application (Prints menu)	Welcome to the Plane Management application (Prints menu)	☒Pass ☐Fail
Invalid option	option = 10	Invalid option. Please enter a number within 0-6.	Invalid option. Please enter a number within 0-6.	☒Pass ☐Fail
Terminating the program	option = 0	The program is terminating.	The program is terminating.	☒Pass ☐Fail
Buying a seat	i) option = 1 rowNo = E ii)option = 1 rowNo = A seatNo = 20 iii) option = 1 rowNo = B seatNo = 3 iv) option = 1 rowNo = B seatNo = 3	i) Enter a valid row letter. ii) Invalid seat number iii) The seat is successfully booked. iv) This seat is already booked.	i) Enter a valid row letter. ii) Invalid seat number iii) The seat is successfully booked. iv) This seat is already booked.	☒Pass ☐Fail

Cancelling a seat	i) option = 1 rowNo = A seatNo = 1 iv) option = 1 rowNo = B seatNo = 3	i) This seat was not booked previously. ii) The seat is successfully cancelled.	i) This seat was not booked previously. ii) The seat is successfully cancelled.	☒Pass ☐Fail
Finding the first available seat	option = 3	First available seat found at row A, seat 1	First available seat found at row A, seat 1	☒Pass ☐Fail
Show seating plan	option = 4	Prints the seats that are available and booked with a 'O' and 'X' respectively.	Prints the seats that are available and booked with a 'O' and 'X' respectively.	☒Pass ☐Fail

Part B Testing

Test case / scenario	**Input**	**Expected Output**	**Output**	**Pass/Fail**
Buying the seat	option = 1 rowNo = A seatNo = 12	The user is asked to enter the details and it is added to the tickets array.	The user is asked to enter the details and it is added to the tickets array successfully.	☒Pass ☐Fail
Cancelling the seat	option = 1 rowNo = A seatNo = 12	The user is asked to enter the seat no and row letter of the seat to be cancelled and it is removed from the tickets array.	The user is asked to enter the seat no and row letter of the seat to be cancelled and it is removed from the tickets array sucessfully.	☒Pass ☐Fail
Print ticket information	option = 5	Prints out the ticket	Prints out the ticket	☒Pass ☐Fail

		informametion and value of the total sales.	information and value of the total sales.	
Search ticket	option = 6	Searches the ticket and prints ticket information if booked or that the seat is available if not booked.	Searches the ticket and prints ticket information if booked or that the seat is available if not booked.	☒Pass ☐Fail
Save	Option= 1 rowNo= C seatNo= 3 name = Anne surname= Philip email =anne@gmail.com	Saves the information of the ticket in a separate text file for every seat booked.	Saves the information of the ticket in a separate text file for every seat booked.	☐Pass ☐Fail

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

I would like to gain feedback on how to further improve this code.

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 :

W2051952_PlaneManagement.java:

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

public class w2051952_PlaneManagement {
    //Task 1: Creating the seats management array:
    public static int[][] seats = new int[4][];

    //Task 9: Creating the tickets array:
    public static Ticket[][] tickets = new Ticket[4][];

    public static void main(String[] args) {
        System.out.println("Welcome to the Plane Management application");

        //Task 1: Creating the seats management array:

        seats[0] = new int[14];
        seats[1] = new int[12];
        seats[2] = new int[12];
        seats[3] = new int[14];

        //Task 9: Creating the tickets array

        tickets[0] = new Ticket[14];
        tickets[1] = new Ticket[12];
        tickets[2] = new Ticket[12];
        tickets[3] = new Ticket[14];
```

```
// Task 2: The menu is looped until the option is not 0.
int option = -1;
do {
   //Task 2: Prints out the user menu
   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 tickets information and total sales");
   System.out.println("    6)  Search ticket");
   System.out.println("    0)  Quit");


   Scanner input = new Scanner(System.in);

   try {
      //Task 2: Allows the user to select an option from the menu
      System.out.print("Please select an option: ");
      option = input.nextInt();

      //Task 2: Executes the respective method based on the option chosen
      switch (option) {
         case 1:
            buy_seat();
            break;
         case 2:
            cancel_seat();
            break;
```

```
            case 3:
               find_first_available();
               break;
            case 4:
               show_seating_plan();
               break;
            case 5:
               print_tickets_info();
               break;
            case 6:
               search_ticket();
               break;
            case 0:
               System.out.println("The program is terminating.");
               break;
            default:
               //If option is not within 0-6 an error message is displayed.
               System.out.println("Invalid option. Please enter a number within 0-6.");
         }
      } catch (InputMismatchException e) {
         //If option is not within 0-6 an error message is displayed.
         System.out.println("Invalid option. Please enter a number within 0-6.");
         input.nextLine();
      }

   }
   while (option != 0);

}
```

```
    public static void buy_seat() {
        //Task 3: This method allows the customer to buy the seat if the row letter and seat no are
correct
        Scanner input = new Scanner(System.in);

        System.out.println("Row A and D has 14 seats each.");
        System.out.println("Row B and C has 12 seats each.");

        //Task 3: The row letter of the seat is obtained from the user.
        System.out.println("Enter the row letter to be purchased (A-D): ");
        char rowNo = input.next().toUpperCase().charAt(0);

        switch (rowNo) {
            //Task 3: Checks if the row letter is within A-D:
            case 'A', 'B', 'C', 'D':
                //Task 3: Allows the user to enter the Seat No if the Row letter is correct.
                System.out.println("Enter the seat number to be purchased : ");
                int seatNo = input.nextInt();
                input.nextLine();

                //Task 3: Converts the row letter to a numerical index.
                int rowIndex = rowNo - 'A';

                //Task 3: Checks if the Seat No is within the range based on the row.
                int condition;
                if (rowIndex == 0 || rowIndex == 3) {
                    condition = 14;
                } else {
                    condition = 12;
                }
```

```
if (seatNo > 0 && seatNo <= condition) {
    if (seats[rowIndex][seatNo - 1] == 0) {
        seats[rowIndex][seatNo - 1] = 1;

        //Task 9: Gets the user input for Personal information.
        System.out.println("Enter your name: ");
        String name = input.nextLine();

        System.out.println("Enter your surname: ");
        String surname = input.nextLine();

        System.out.println("Enter your email: ");
        String email = input.nextLine();

        //Task 3: The seat is successfully booked if both seat No & row letter are correct.
        System.out.println("The seat is successfully booked.");

        Person person1 = new Person(name, surname, email);

        //Task 9: Finding the prices based on the seat numbers
        int price;
        if (seatNo <= 5) {
            price = 200;
        } else if (seatNo <= 9) {
            price = 150;
        } else {
            price = 180;
        }

        //Task 9: Adding ticket information to the tickets array
        Ticket new_ticket = new Ticket(rowNo, price, seatNo, person1);
```

```
                tickets[rowIndex][seatNo - 1] = new_ticket;

                //Task 12: Prints the ticket information after the seat is booked.
                new_ticket.save();

            } else {
                //Task 3: Prevents the user from booking a seat, that has already been booked previously.
                System.out.println("This seat is already booked.");
            }
        } else {
            //Task 3: Prevents the user from entering an invalid Seat No.
            System.out.println("Invalid seat number");
        }
        break;

      default:
        //Task 3: Prevents the user from entering an invalid Row Letter.
        System.out.println("Enter a valid row letter.");
    }
  }

  public static void cancel_seat() {
    //Task 4: This method allows the customer to cancel the seat if the row letter and seat no are correct.
    Scanner input = new Scanner(System.in);

    //Task 4: The row letter of the seat is obtained from the user.
    System.out.println("Enter the row number to be cancelled: ");
    char rowNo = input.next().toUpperCase().charAt(0);
```

```
        switch (rowNo) {
            case 'A', 'B', 'C', 'D':
                //Task 4: Allows the user to enter the Seat No if the Row letter is correct.
                System.out.println("Enter the seat number to be cancelled: ");
                int seatNo = input.nextInt();

                //Task 4: Converts the row letter to a numerical index.
                int rowIndex = rowNo - 'A';

                //Task 4: Checks if the Seat No is within the range based on the row.
                int condition;
                if (rowIndex == 0 || rowIndex == 3) {
                    condition = 14;
                } else {
                    condition = 12;
                }

                if (seatNo > 0 && seatNo <= condition) {
                    if (seats[rowIndex][seatNo - 1] == 1) {
                        seats[rowIndex][seatNo - 1] = 0;

                        //Task 9: Removes the ticket from the tickets array.
                        tickets[rowIndex][seatNo - 1] = null;

                        //Task 4: The seat is successfully cancelled if both seat No & row letter are
correct.
                        System.out.println("The seat is successfully cancelled.");
                    } else {
                        //Task 4: Checks if the seat is already booked in order to be cancelled.
                        System.out.println("This seat was not booked previously.");
```

```
                }
            } else {
                //Task 4: Prevents the user from entering an invalid Seat No.
                System.out.println("Invalid seat number.");
            }
            break;
        default:
            //Task 4: Prevents the user from entering an invalid Row Letter.
            System.out.println("Enter a valid row letter.");
    }
}

public static void show_seating_plan() {
    //Task 6: This method shows the seats that are still available and sold.
    System.out.println("Seating Plan:");

    //Task 6: Makes use of two "For Loops" to traverse through the two dimensional array for seats.
    for (int[] seat : seats) {
        for (int i : seat) {
            if (i == 0) {
                System.out.print(" O ");
            } else {
                System.out.print(" X ");
            }
        }
        System.out.println();
    }
}

public static void find_first_available() {
```

```
        //Task 5: This method allows the user to find the first seat which is still available.
        char row = 'A';

        //Task 5: Makes use of two "For Loops" to traverse through the two-dimensional array for seats.
        for (int i = 0; i < seats.length; i++) {
            for (int j = 0; j < seats[i].length; j++) {
                if (seats[i][j] == 0) {
                    System.out.println("First available seat found at row " + (char) (row + i) + ", seat " + (j + 1));
                    return;
                }
            }
        }
    }

    public static void print_tickets_info() {
        //Task 10: This method prints the ticket information & total sales.
        double total = 0.0;

        //Task 10: Makes use of two "For Loops" to traverse through the two-dimensional array for tickets.
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < tickets[i].length; j++) {
                if (tickets[i][j] != null) {
                    tickets[i][j].ticket_information();
                    total = total + tickets[i][j].getPrice();
                    System.out.println("Total: " + total);
                }
            }
        }
```

```
}

public static void search_ticket() {
   //Task 11: This method allows the user to check if the seat is already booked or not.
   Scanner input = new Scanner(System.in);

   //Task 11: The row letter of the seat is obtained from the user.
   System.out.println("Enter the row letter (A-D): ");
   char rowNo = input.next().toUpperCase().charAt(0);

   switch (rowNo) {
      //Task 11: Checks if the row letter is within A-D:
      case 'A', 'B', 'C', 'D':
         //Task 11: Allows the user to enter the Seat No if the Row letter is correct.
         System.out.println("Enter the seat number :  ");
         int seatNo = input.nextInt();
         input.nextLine();

         //Task 11: Converts the row letter to a numerical index.
         int rowIndex = rowNo - 'A';



         //Task 11: Checks if the Seat No is within the range based on the row.
         int condition;

         if (rowIndex == 0 || rowIndex == 3) {
         condition = 14;
         } else {
            condition = 12;
         }
```

```
            if (seatNo > 0 && seatNo <= condition) {
                //Task 11: Checks if the seat is available or not
                if (tickets[rowIndex][seatNo - 1] != null) {
                    tickets[rowIndex][seatNo - 1].ticket_information();
                    System.out.println("The ticket is available");
                } else {
                    System.out.println("This is seat is available to be booked.");
                }
            } else {
                //Task 11: Prevents the user from entering an invalid Seat No.
                System.out.println("Invalid seat number");
            }
            break;

            default:
                //Task 11: Prevents the user from entering an invalid Row Letter.
                System.out.println("Enter a valid row letter.");
        }

    }
}
```

Ticket.java:

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

// Task 8: Creating a new class file called "Ticket".
public class Ticket {
  // Task 8: Adding the attributes.
   private char row;
   private int price, seat;
   private Person person;

   // Task 8: Adding the constructors.

   public Ticket(char row, int price, int seat, Person person) {
      this.row = row;
      this.price = price;
      this.seat = seat;
      this.person = person;
   }

   //Task 8: Adding the getters & setters.
   public char getRow() {
      return row;
   }

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

   public int getPrice() {
      return price;
   }
```

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

    public int getSeat() {
        return seat;
    }

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

    public Person getPerson() {
        return person;
    }

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

    public void ticket_information(){
        //Task 8: This method prints out the information of the Ticket including the information of the
Person.
        System.out.println("\nTicket Information: ");
        System.out.println("\tRow: " + getRow());
        System.out.println("\tSeat: " + getSeat());
        System.out.println("\tPrice: " + getPrice());

        //Task 8: Calling personal_information() method from "Person" to print out the information of the
Person.
        person.personal_information();
    }

    public void save(){
```

```
        //Task 12: Saves the ticket information to a text file.
        try{
            FileWriter file = new FileWriter(getRow()+""+getSeat()+".txt");
            file.write("Ticket information for the seat: \n");
            file.write("\tRow: " + getRow()+"\n");
            file.write("\tSeat: " + getSeat()+"\n");
            file.write("\tPrice: " + getPrice()+"\n");
            file.write("Passenger information: "+"\n");
            file.write("\tName: " + person.getName()+"\n");
            file.write("\tSurname: " + person.getSurname()+"\n");
            file.write("\tEmail: " + person.getEmail()+"\n");
            file.close();

        }
        catch (IOException e){
            System.out.println("The text file is not created.");
        }

    }
}
```

Person.java:

```
// Task 7: Creating a new class file called "Person".
public class Person {
    // Task 7: Adding the attributes.
    private String name, surname, email;

    // Task 7: Adding the constructors.
    public Person(String name, String surname, String email) {
        this.name = name;
        this.surname = surname;
        this.email = email;
    }

    //Task 7: Adding getters & 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;
    }


    public void personal_information(){
        // Task 7: This method prints out the information from Person.
        System.out.println("\nPassenger Information:");
        System.out.println("\tName: " + getName());
        System.out.println("\tSurname: " + getSurname());
        System.out.println("\tEmail: " + getEmail());
    }
}
```

<<END>>