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	: 20222153/ w2052268
Student First Name	: Dodanduwa
Student Surname	: Wadisingha

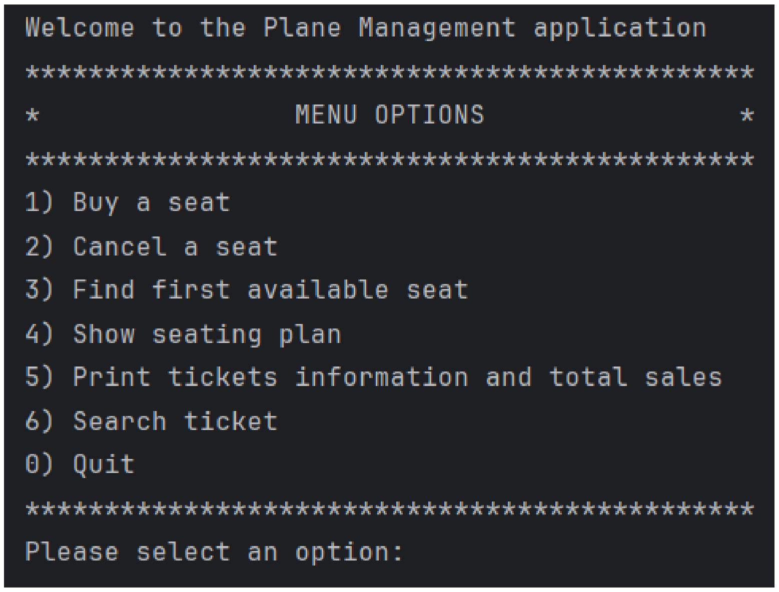
Tutorial group (day, time, and tutor/s): 22(Tuesday, 8.30-10.30, Mr. Ammar Raneez and Ms. Rashmi Perera)

"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."

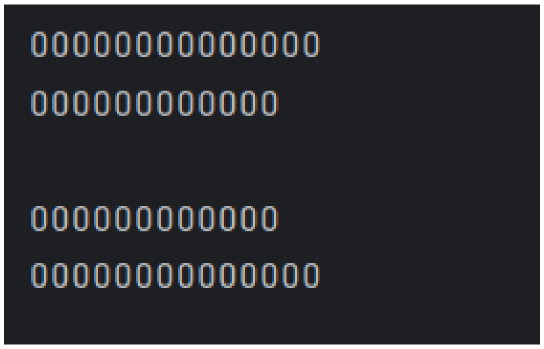
Name	: D.W.A. Nawoda Nethmini Wadisingha
Student ID	: 20222153/w2052268

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 completed and Created project 'w2052268_PlaneManagement' with a 'PlaneManagement' class, displaying a 'Welcome' message and implemented seat management using standard arrays. All seats were initially available (0).
2	☒Fully implemented ☐Partially implemented ☐Not attempted	Task 2 was completed by adding a user menu to the main method of the 'PlaneManagement' class. The menu displays options for the user to select, with option '0' allowing the user to quit the program. I used a while loop to continuously display the menu and handle user input.
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 tickets information and total sales
6) Search ticket
0) Quit
**********************************************
Please select an option:
```

3	☒Fully implemented ☐Partially implemented ☐Not attempted	Task 3 completed and implemented the buy_seat method to handle seat booking, validation of input, and check of seat availability when booking a seat. The method easily combines with option '1' on the main menu.
4	☒Fully implemented ☐Partially implemented ☐Not attempted	Task 4 completed and implemented the cancel_seat method to handle seat cancellation, including input validation, seat availability check, file deletion, and ticket removal. The method integrates smoothly with the main menu option ‘2’.
5	☒Fully implemented ☐Partially implemented ☐Not attempted	Task 5 completed and implemented find_first_available method to locate the first available seat, searching rows A through D sequentially. The method integrates smoothly with the main menu option ‘3’.
6	☒Fully implemented ☐Partially implemented ☐Not attempted	Task 6 completed and implemented the show_seating_plan method to display the plane's seating plan, marking available seats with ‘O’, and sold seats with ‘X’. The method integrates smoothly with the main menu option ‘4’.
Insert here a screenshot of the seating plan: 		

```
OOOOOOOOOOOOOO
OOOOOOOOOOOO

OOOOOOOOOOOO
OOOOOOOOOOOOOO
```

7	☒Fully implemented ☐Partially implemented ☐Not attempted	Task 7 completed and implemented the Person class file with attributes, constructor, getters, setters, and ‘printInfo’ method for storing and displaying the person information.
8	☒Fully implemented ☐Partially implemented ☐Not attempted	Completed Task 8 by creating the Ticket class file, which defines attributes for row, seat, price, and a Person object to store ticket holder information.
9	☒Fully implemented ☐Partially implemented ☐Not attempted	Completed Task 9 by adding an array of Tickets to store all tickets sold in a session. Modified buy_seat to prompt for Person information, create a new Ticket, and add it to the Tickets array. Also modified cancel_seat to remove a ticket from the array.
10	☒Fully implemented ☐Partially implemented ☐Not attempted	Completed Task 10 by creating the print_tickets_info method, which prints ticket information for all tickets sold and calculates the total price of tickets sold during the session.
11	☒Fully implemented ☐Partially implemented ☐Not attempted	Completed Task 11 by creating the search_ticket method, which allows users to input a row letter and seat number to search for ticket information. The method prints Ticket and Person information if the seat is sold or displays 'This seat is available' if the seat is not sold.
12	☒Fully implemented ☐Partially implemented ☐Not attempted	Completed Task 12 by adding a save method to the Ticket class, which saves ticket information (including Person details) to a file. The file name is based on the row and seat number of the ticket.

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**
Test display the welcome message and menu option at the start of the program	Run the program.	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 tickets information and total sales 6) Search ticket 0) Quit ** Please select an option:	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 tickets information and total sales 6) Search ticket 0) Quit ** Please select an option:	☒Pass ☐Fail
Test buy_seat method	Please select an option: 1 Enter a Row Letter(A/B/C/D): A Enter a seat number: 1 Person name: Nawoda Person surname: wadisingha Person email: nawoda@gmail.com	Displaying the following messages and saving the text file. You bought the ticket. Ticket information is saved to A1.txt.	Displayed the following messages and saved the text file. You bought the ticket. Ticket information is saved to A1.txt.	☒Pass ☐Fail
Test cancel_s	Please select an option: 2	Displaying the following message.	Displayed the following message.	☒Pass ☐Fail

eat method	Enter a Row Letter(A/B/C/D): B Enter a seat number: 10	Seat Already Available.	Seat Already Available	
Test cancel_seat method	Please select an option: 2 Enter a Row Letter(A/B/C/D): A Enter a seat number: 1	Displaying the following messages and deleting the created text file. File deleted successfully. Cancellation Done.	Displayed the following messages and deleted the created text file. File deleted successfully. Cancellation Done.	☒Pass ☐Fail
Test find_first_available method	Please select an option: 3	Displaying the following message. First available seat: A1	Displayed the following message. First available seat: A1	☒Pass ☐Fail
Test show_seating_plan method	Please select an option: 4	Displaying the following format. OOOOOOOOOOOOOO OOOOOOOOOOOO OOOOOOOOOOOO OOOOOOOOOOOOOO	Displayed the following format. OOOOOOOOOOOOOO OOOOOOOOOOOO OOOOOOOOOOOO OOOOOOOOOOOOOO	☒Pass ☐Fail
After booking the 'D12' seat, test show_seating_plan method	Please select an option: 4	Displaying the following format. OOOOOOOOOOOOOO OOOOOOOOOOOO OOOOOOOOOOOO OOOOOOOOOOOXOO	Displayed the following format. OOOOOOOOOOOOOO OOOOOOOOOOOO OOOOOOOOOOOO OOOOOOOOOOOXOO	☒Pass ☐Fail
Test invalid integer input for the menu option.	Please select an option: 10	Displaying the following message and again displaying the welcome message and menu option. Wrong Choice.	Displayed the following message and again displayed the welcome message and menu option. Wrong Choice.	☒Pass ☐Fail
Test invalid character input for the menu option.	Please select an option: de	Displaying the following message and again displaying the welcome message and menu option. Invalid input. Please enter a number.	Displayed the following message and again displayed the welcome message and menu option. Invalid input. Please enter a number.	☒Pass ☐Fail

Test invalid input for the Row letter.	Enter a Row Letter(A/B/C/D): h	Displaying the following message and again displaying the enter a-row letter option. invalid input Enter a Row Letter(A/B/C/D):	Displayed the following message and again displayed the enter a-row letter option. invalid input Enter a Row Letter(A/B/C/D):	☒Pass ☐Fail
Test invalid input for the Row letter.	Enter a Row Letter(A/B/C/D): dab	Displaying the following message and again displaying the enter a-row letter option. invalid input Enter a Row Letter(A/B/C/D):	Displayed the following message and again displayed the enter a-row letter option. invalid input Enter a Row Letter(A/B/C/D):	☒Pass ☐Fail
Test invalid input for the seat number for row A.	Enter a Row Letter(A/B/C/D): a Enter a seat number: 15	Displaying the following message and again displaying the enter a-seat number option. Invalid seat number Enter a seat number:	Displayed the following message and again displayed the enter a-seat number option. Invalid seat number Enter a seat number:	☒Pass ☐Fail
Test invalid input for the seat number for row B.	Enter a Row Letter(A/B/C/D): b Enter a seat number: 13	Displaying the following message and again displaying the enter a-seat number option. Invalid seat number Enter a seat number:	Displayed the following message and again displayed the enter a-seat number option. Invalid seat number Enter a seat number:	☒Pass ☐Fail
Test invalid input for the seat number for row c.	Enter a Row Letter(A/B/C/D): c Enter a seat number: 13	Displaying the following message and again displaying the enter a-seat number option. Invalid seat number Enter a seat number:	Displayed the following message and again displayed the enter a-seat number option. Invalid seat number Enter a seat number:	☒Pass ☐Fail
Test invalid input for the seat number for row d.	Enter a Row Letter(A/B/C/D): b Enter a seat number: 15	Displaying the following message and again displaying the enter a-seat number option. Invalid seat number Enter a seat number:	Displayed the following message and again displayed the enter a-seat number option. Invalid seat number Enter a seat number:	☒Pass ☐Fail

Part B testing

Test case / scenario	**Input**	**Expected Output**	**Output**	**Pass/Fail**
After booking the ‘D12’and ‘B5’ seat, test print_tickets_info method	Please select an option: 5	Displaying the following information. Ticket Information: Row: B Seat: 5 Price: €200 ---------------------------------- Ticket Information: Row: D Seat: 12 Price: €180 ---------------------------------- *** Total Sales: €380	Displayed the following information. Ticket Information: Row: B Seat: 5 Price: €200 ---------------------------------- Ticket Information: Row: D Seat: 12 Price: €180 ---------------------------------- *** Total Sales: €380	☒Pass ☐Fail
Test search_ticket method to ‘B5’ seat.	Please select an option: 6 Enter a Row Letter(A/B/C/D): b Enter a seat number: 5	Displaying the following information. Ticket Information:- Row: B Seat: 5 Price: €200 Person Information:- Name: nawoda Surname: nethmini Email: nawoda@gmail.com	Displayed the following information. Ticket Information:- Row: B Seat: 5 Price: €200 Person Information:- Name: nawoda Surname: nethmini Email: nawoda@gmail.com	☒Pass ☐Fail
Test search_ticket method to available seat.	Please select an option: 6 Enter a Row Letter(A/B/C/D): c Enter a seat number: 9	Displaying the following message. Seat Available.	Displayed the following message. Seat Available.	☒Pass ☐Fail

Test saving ticket information	Buy a ‘B5’ ticket and check if the ticket information is saved in a file with the correct name	Saving the information of the ticket including the person in a file named ‘B5.txt’.	Saved the information of the ticket including the person in a file named ‘B5.txt’.	☒Pass ☐Fail
Without booking any seat test the print_tickets_info method	Please select an option: 5	Displaying the following message. ****************************** ****************** Total Sales: €0	Displayed the following message. ********************************* *************** Total Sales: €0	☒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 <u>0 for the coursework.</u>

3) Code :

Code for class File named PlaneManagement.

```
//Import necessary libraries

import java.util.*;           //For various utility classes like Scanner,Arrays

import java.io.IOException;    //For handling input/output exceptions

import java.nio.file.Files;    //For working with files

import java.nio.file.Path;     //For representing file paths

import java.nio.file.Paths;    //For creating file paths



// Define a class named PlaneManagement

public class PlaneManagement{

   //Declare variables for row and seat number

   char row;

   int seat_number;

   char euro='€';



   //Arrays to track seat availability for each row

   int[] A = new int[14];

   int[] B = new int[12];

   int[] D = new int[14];
```

```
int[] C = new int[12];


//Array to store Ticket objects for each seat

Ticket[] tickets_array = new Ticket[52];


//Method to initialize all seat arrays to zero

public void assign_zero (){

   Arrays.fill(A, 0);

   Arrays.fill(B, 0);

   Arrays.fill(C, 0);

   Arrays.fill(D, 0);

}


public static void main(String[] args){

   try {

       //Create an instance of PlaneManagement

       PlaneManagement obj1 = new PlaneManagement();

       //Initialize seat arrays to zero

       obj1.assign_zero();

       //Main menu loop
while (true) {

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

   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");

   System.out.println("***********************************************");

   System.out.println("Please select an option: ");



   int option;

   Scanner scanner = new Scanner(System.in);

   //Get user input for menu option

   if (scanner.hasNextInt()) {

      option = scanner.nextInt();
```

```
}

else {

   System.out.println("Invalid input. Please enter a number.\n");

   scanner.nextLine();

   continue;

}

//Show output based on user input

if (option == 1) {

   obj1.buy_seat();

}

else if (option == 2) {

   obj1.cancel_seat();

}

else if (option == 3) {

   obj1.find_first_available();

}

else if (option == 4) {

   obj1.show_seating_plan();

}

else if (option == 5) {
```

```
            obj1.print_tickets_info();

          }

          else if (option == 6) {

            obj1.search_ticket();

          }

          else if (option == 0) {

            break;

          }

          else {

            System.out.println("Wrong Choice.\n");

          }

        }

      }

      //Catch any unexpected exceptions

      catch (Exception e){

            System.out.println("An unexpected error occurred.");

            e.printStackTrace();

      }

    }
```

```
//Method to get user input for row and seat number

public void get_inputs(){

    boolean loop = true;

    //Loop to ensure valid row input

    while(true){

        Scanner scanner = new Scanner(System.in);

        System.out.println("Enter a Row Letter(A/B/C/D): ");

        String temp_in=scanner.nextLine();

        temp_in=temp_in.toUpperCase();



        //Check if input is valid row letter

        if(temp_in.equals("A") || temp_in.equals("B") || temp_in.equals("C") || temp_in.equals("D")){

            row=temp_in.charAt(0);

            break;

        }

        System.out.println("invalid input");

    }



    //Loop to ensure valid seat number input

    while (loop) {
```

```
loop=false;

System.out.println("Enter a seat number: ");

try {

    Scanner scanner = new Scanner(System.in);

    seat_number = scanner.nextInt();



    //Check if seat number is valid for the selected row

    if (row == 'A' || row == 'D') {

        if (!(0 < seat_number && seat_number <= 14)) {

            System.out.println("Invalid seat number");

            loop=true;

        }

    }

    else if (row == 'B' || row == 'C') {

        if (!(0 < seat_number && seat_number <= 12)) {

            System.out.println("Invalid seat number");

            loop = true;

        }

    }
```

```
        }

        catch(Exception e){

            System.out.println("Invalid seat number");

            loop = true;

        }

    }

}



// Method to calculate the price of a seat

public int calculate_price(int seat_number){

    seat_number++;  // Increment the seat number by 1 to match the seat number

    int price_tag=0;    // Initialize the price tag variable



    // Calculate the price tag

    if(seat_number<=5 && seat_number>=1){

        price_tag=200;

    }

    else if(seat_number>=6 && seat_number<=9){

        price_tag=150;

    }
    else if(seat_number>=10 && seat_number<=14){

        price_tag=180;

    }

    return price_tag;

}



// Calculate the seat index based on the row and seat number

public int set_seat_index(int seat_number,char row){

    int seat_index=seat_number;   // Initialize seat_index with the seat number

    // Increment the seat_index

    if(row=='B'){

        seat_index=seat_index+14;

    }

    else if (row=='C'){

        seat_index=seat_number+26;

    }

    else if(row=='D'){

        seat_index=seat_number+38;

    }

    return seat_index;
```

```
}


//Method to buy a seat
public void buy_seat(){
    Scanner scanner = new Scanner(System.in);
    while(true) {
        //Get row and seat number inputs
        get_inputs();
        boolean valid = true;
        seat_number=seat_number-1;
        if(valid){
            //Check if seat is available and mark it as sold
            if (row == 'A' && A[seat_number] == 0) {
                A[seat_number] = 1;
                break;
            }
            else if (row == 'B' && B[seat_number] == 0) {
                B[seat_number] = 1;
                break;
            }
            else if (row == 'C' && C[seat_number] == 0) {

                C[seat_number] = 1;

                break;

            }

            else if (row == 'D' && D[seat_number] == 0) {

                D[seat_number] = 1;

                break;

            }

            else {

                System.out.println("Seat not Available");

            }

        }

    }


    //Get person's information

    System.out.println("Person name: ");

    String name = scanner.next();



    System.out.println("Person surname: ");

    String surname = scanner.next();
```

```
    System.out.println("Person email: ");

    String email = scanner.next();



    //Calculate ticket price

    int price_tag=calculate_price(seat_number);

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



    //Calculate seat index

    int seat_index=set_seat_index(seat_number,row);



    //Create and store ticket object

    Ticket obj_ticket = new Ticket(row,(seat_number+1),price_tag,obj_person);

    tickets_array[seat_index]= obj_ticket;

}



//Method to cancel a seat booking

public void cancel_seat(){

    //Get row and seat number inputs

    get_inputs();
```

```
//Generate file name based on row and seat number

String fileName=String.valueOf(row)+String.valueOf(seat_number)+".txt";

Path pathToFile = Paths.get(fileName);



try {

   //Delete file associated with the booking

   Files.delete(pathToFile);

   System.out.println("File deleted successfully.");

}

catch (IOException e) {

 //System.out.println(fileName);

}



//Decrement seat number to match array indexing

seat_number=seat_number-1;

//Calculate seat index

int seat_index=set_seat_index(seat_number,row);



//Set the seat as available and remove associated ticket
```

```
tickets_array[seat_index]=null;

if (row == 'A' && A[seat_number] == 1) {

   A[seat_number] = 0;

   System.out.println("Cancellation Done.\n");

}

else if (row == 'B' && B[seat_number] == 1) {

   B[seat_number] = 0;

   System.out.println("Cancellation Done.\n");

}

else if (row == 'C' && C[seat_number] == 1) {

   C[seat_number] = 0;

   System.out.println("Cancellation Done.\n");

}

else if (row == 'D' && D[seat_number] == 1) {

   D[seat_number] = 0;

   System.out.println("Cancellation Done.\n");

}

else {

   System.out.println("Seat Already Available.\n");

}
```

```
}


//Method to find the first available seat

public void find_first_available(){

    //Array of rows

    char[] rows = {'A', 'B', 'C', 'D'};



    //Iterate over each row

    for (char r : rows) {

        //Iterate over each seat in the row

        for (int i = 0; i < A.length; i++) {

            // Check if the seat is available in the current row

            if (r == 'A' && A[i] == 0) {

                System.out.println("First available seat: " + r + (i + 1)+"\n");

                return;

            }

            else if (r == 'B' && B[i] == 0) {

                System.out.println("First available seat: " + r + (i + 1)+"\n");

                return;

            }
            else if (r == 'C' && C[i] == 0) {
                System.out.println("First available seat: " + r + (i + 1)+"\n");
                return;
            }
            else if (r == 'D' && D[i] == 0) {
                System.out.println("First available seat: " + r + (i + 1)+"\n");
                return;
            }
        }
    }
    //If no available seats found
    System.out.println("No available seats.\n");
}


//Method to show the seating plan
public void show_seating_plan(){
    // Display seats for row A
    for(int i=0;i< A.length;i++){
        if(A[i]==0){
            System.out.print("O");
        }
        else {
            System.out.print("X");
        }
    }
    System.out.println("");


    // Display seats for row B
    for(int i=0;i< B.length;i++){
        if(B[i]==0){
            System.out.print("O");
        }
        else {
            System.out.print("X");
        }
    }
    System.out.println("\n");


    // Display seats for row C
    for(int i=0;i< C.length;i++){
        if(C[i]==0){

            System.out.print("O");

        }

        else {

            System.out.print("X");

        }

    }

    System.out.println("");



    // Display seats for row D

    for(int i=0;i< D.length;i++){

        if(D[i]==0){

            System.out.print("O");

        }

        else {

            System.out.print("X");

        }

    }

    System.out.println("");

    System.out.println("");
```

```
}


// Method to print ticket information and total sales
public void print_tickets_info(){
    int total=0;    // Initialize a variable to store the total sales
    // Iterate over the tickets_array
    for(int i=0;i< tickets_array.length;i++){
        // Check if the current ticket is not null
        if(!(tickets_array[i]==null)){
            // Add the price of the current ticket to the total sales
            total=total+tickets_array[i].getPrice();
            // Print the information of the current ticket
            System.out.println("Ticket Information:");
            System.out.println("Row: " + tickets_array[i].getRow());
            System.out.println("Seat: " + tickets_array[i].getSeat());
            System.out.println("Price: "+euro + tickets_array[i].getPrice());
            System.out.println("------------------------------------");
        }
    }
    System.out.println("***********************************************");
```

```
        System.out.println("Total Sales: "+ euro +total+"\n");
    }


    // Method to search for a ticket
    public void search_ticket(){
        get_inputs();    // Get the row and seat number from the user input


        int seat_number1=seat_number-1;   // Adjust the seat number for array indexing
        int seat_index=set_seat_index(seat_number1,row);     // Calculate the index of the seat in the
tickets_array
        // Check if the seat is available
        if(tickets_array[seat_index]==null){
            System.out.println("Seat Available.\n");
        }
        else{
            // If the seat is not available, print the ticket information
            tickets_array[seat_index].printInfo();
            System.out.println("");
        }
    }
}
```

```
}
```

Code for class File named Person.

```
// Define a class called Person

public class Person {

   // Declare a private String variable

   private String name;

   private String surname;

   private String email;



   // Constructor that takes name, surname, and email as parameters

   public Person(String name, String surname, String email) {

      // Assign the parameter name to the instance variable

      this.name = name;

      this.surname = surname;

      this.email = email;

   }



   // Getter method for retrieving the name

   public String getName() {

      return name;

   }
}


// Setter method for setting the name

public void setName(String name) {

   this.name = name;

}


// Getter method for retrieving the surname

public String getSurname() {

   return surname;

}


// Setter method for setting the surname

public void setSurname(String surname) {

   this.surname = surname;

}


// Getter method for retrieving the email

public String getEmail() {

   return email;

}
```

```
    // Setter method for setting the email

    public void setEmail(String email) {

        this.email = email;

    }



    // Method to print the information of the person

    public void printInfo() {

        System.out.println("Name: " + name);

        System.out.println("Surname: " + surname);

        System.out.println("Email: " + email);

    }

}
```

Code for class File named Ticket.

```
import java.io.FileWriter;     // For writing to a file

import java.io.IOException;



// Define a class called Ticket

public class Ticket {

   char euro='€';    // Define a char variable for the euro symbol

   // Declare a private variables

   private char row;

   private int seat;

   private int price;

   private Person person;     // Declare a private Person object called person to store the person's
information



   // Constructor that takes row, seat, price, and person as parameters

   public Ticket(char row, int seat, int price, Person person) {

      // Assign the parameters

      this.row = row;

      this.seat = seat;

      this.price = price;

      this.person = person;
```

```
        save();   // Call the save method to save ticket information to a file
    }


    // Method to save ticket information to a file
    private void save(){
        int seatnumber=seat;    // Assign the seat number
        String fileName = row + "" + seatnumber + ".txt";   // Generate the file name based on row and seat
number
        try (FileWriter writer = new FileWriter(fileName)) {   // Try to create a FileWriter object
            // Write ticket information to the file
            writer.write("Ticket info:- \n");
            writer.write("Row: " + row + "\n");
            writer.write("Seat: " + seatnumber + "\n");
            writer.write("Price: "+euro + price + "\n");
            writer.write("Person Information:-\n");
            writer.write("Name: " + person.getName() + "\n");
            writer.write("Surname: " + person.getSurname() + "\n");
            writer.write("Email: " + person.getEmail() + "\n");
            System.out.println("You bought the ticket.");
            System.out.println("Ticket information is saved to " + fileName+".\n"); // Print confirmation
message
```

```
    }
    catch (IOException e) {    // Catch IOException if an error occurs
        System.out.println("An error occurred while saving the ticket information.");
        e.printStackTrace();  // Print the stack trace for debugging
    }
}


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


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


// Getter method for retrieving the seat
public int getSeat() {
    return seat;
```

```
}

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

// Getter method for retrieving the price
public int getPrice() {
    return price;
}

// Setter method for setting the price
public void setPrice(int price) {
    this.price = price;
}

// Getter method for retrieving the person object
public Person getPerson() {
    return person;
    }


    // Setter method for setting the person object

    public void setPerson(Person person) {

        this.person = person;

    }


    // Method to print ticket information

    public void printInfo() {

        System.out.println("");

        System.out.println("Ticket Information:-");

        System.out.println("Row: " + row);

        System.out.println("Seat: " + seat);

        System.out.println("Price: "+euro + price);

        System.out.println("Person Information:-");

        person.printInfo();   // Call the printInfo method of the person object to print person information

    }

}
```

<<END>>