UOW-logo

Informatics Institute of Technology

School of Computing

Software Development II Coursework Report

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

Date of submission :

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

Student First Name :

Student Surname :

Tutorial group (day, time, and tutor/s):

"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 :

Student ID :

## Self-assessment form and test plan

# Self-assessment form

|  |  |  |
| --- | --- | --- |
| Task | Self-assessment (select one) | Comments |
| 1 | Fully implemented  Partially implemented  Not attempted |  |
| 2 | Fully implemented  Partially implemented  Not attempted |  |
| Insert here a screenshot of your welcome message and menu: | | |
| 3 | Fully implemented  Partially implemented  Not attempted |  |
| 4 | Fully implemented  Partially implemented  Not attempted |  |
| 5 | Fully implemented  Partially implemented  Not attempted |  |
| 6 | Fully implemented  Partially implemented  Not attempted |  |
| Insert here a screenshot of the seating plan: | | |
| 7 | Fully implemented  Partially implemented  Not attempted |  |
| 8 | Fully implemented  Partially implemented  Not attempted |  |
| 9 | Fully implemented  Partially implemented  Not attempted |  |
| 10 | Fully implemented  Partially implemented  Not attempted |  |
| 11 | Fully implemented  Partially implemented  Not attempted |  |
| 12 | Fully implemented  Partially implemented  Not attempted |  |

# 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 |
| Buy a ticket | Row 2 Seat 5  Raj Kumar  [raj@gmail.com](mailto:raj@gmail.com) | The seat has been booked | The seat has been booked | Pass  Fail |
| Cancel a ticket | Row 1 Seat 5 | This seat is already available or invalid seat number | This seat is already available or invalid seat number | Pass  Fail |
| Cancel a ticket | Row 2 Seat 5 | The seat has been cancelled | The seat has been cancelled | Pass  Fail |
| Buy a ticket | Row 2 Seat 5  Raj Kumar  [raj@gmail.com](mailto:raj@gmail.com) | The seat has been booked | The seat has been booked | Pass  Fail |
| Print seating area |  | Only 2nd Row Seat 5 being marked X | Only 2nd Row Seat 5 being marked X | Pass  Fail |
| Find first available seat |  | Row 1, Seat 1 | Row 1, Seat 1 | Pass  Fail |
| Buy a ticket | Row 1 Seat 1  Kishore Rajiv  [kishore@gmail.com](mailto:kishore@gmail.com) | The seat has been booked | The seat has been booked | Pass  Fail |
| Find first available seat |  | Row 1, Seat 2 | Row 1, Seat 2 | Pass  Fail |

## Part B testing

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test case / scenario | Input | Expected Output | Output | Pass/Fail |
| Print tickets information |  | Total price of tickets sold: $0 | Total price of tickets sold: $0 | Pass  Fail |
| Buy a ticket | Row 1 Seat 1  Kishore Rajiv  [kishore@gmail.com](mailto:kishore@gmail.com) | The seat has been booked | The seat has been booked | Pass  Fail |
| Print tickets information |  | Row: 1, Seat: 1, Price: $10  Name: Kishore, Surname: Rajiv, Email: [kishore@gmail.com](mailto:kishore@gmail.com)  Total price of tickets sold: $10 | Row: 1, Seat: 1, Price: $10  Name: Kishore, Surname: Rajiv, Email: [kishore@gmail.com](mailto:kishore@gmail.com)  Total price of tickets sold: $10 | Pass  Fail |
| Cancel a ticket | Row 1 Seat 1 | The seat has been cancelled | The seat has been cancelled | Pass  Fail |
| Print tickets information |  | Total price of tickets sold: $0 | Total price of tickets sold: $0 | Pass  Fail |
| Buy a ticket | Row 1 Seat 1  Kishore Rajiv  [kishore@gmail.com](mailto:kishore@gmail.com) | The seat has been booked | The seat has been booked | Pass  Fail |
| Buy a ticket | Row 3 Seat 5  Raj Kumar  [raj@gmail.com](mailto:raj@gmail.com) | The seat has been booked | The seat has been booked | Pass  Fail |
| Search for a ticket | Row 1 Seat 1 | Row: 1, Seat: 1, Price: $10  Name: Kishore, Surname: Rajiv, Email: [kishore@gmail.com](mailto:kishore@gmail.com) | Row: 1, Seat: 1, Price: $10  Name: Kishore, Surname: Rajiv, Email: [kishore@gmail.com](mailto:kishore@gmail.com) | Pass  Fail |
| Search for a ticket | Row 1 Seat 2 | This seat is available | This seat is available | Pass  Fail |
| Sort tickets by price |  | Row: 1, Seat: 1, Price: $10  Name: Kishore, Surname: Rajiv, Email: hgdss  Row: 3, Seat: 5, Price: $12  Name: Raj, Surname: Kumar, Email: raj@gmail.com | Row: 1, Seat: 1, Price: $10  Name: Kishore, Surname: Rajiv, Email: hgdss  Row: 3, Seat: 5, Price: $12  Name: Raj, Surname: Kumar, Email: raj@gmail.com | 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.**

1. **Code :**

**Main Class**

import java.util.Scanner;

public class CinemaManagement {

private static final int ROWS = 5;

private static final int SEATS = 16;

private static final int[] PRICES = {10, 10, 12, 14, 15}; // Prices for each row

private static int[][] seats = new int[ROWS][SEATS];

private static Ticket[] tickets = new Ticket[ROWS \* SEATS];

private static int ticketCount = 0;

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

System.out.println("Welcome to The London Lumiere");

while (true) {

displayMenu();

int choice = scanner.nextInt();

switch (choice) {

case 1:

buy\_ticket(scanner);

break;

case 2:

cancel\_ticket(scanner);

break;

case 3:

print\_seating\_area();

break;

case 4:

find\_first\_available();

break;

case 5:

print\_tickets\_info();

break;

case 6:

search\_ticket(scanner);

break;

case 7:

sort\_tickets();

break;

case 8:

System.out.println("Exiting program.");

return;

default:

System.out.println("Invalid option. Please try again.");

}

}

}

private static void displayMenu() {

System.out.println("\n1. Buy a ticket");

System.out.println("2. Cancel a ticket");

System.out.println("3. Print seating area");

System.out.println("4. Find first available seat");

System.out.println("5. Print tickets information");

System.out.println("6. Search for a ticket");

System.out.println("7. Sort tickets by price");

System.out.println("8. Exit");

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

}

private static void buy\_ticket(Scanner scanner) {

System.out.println("Enter row number (1-5): ");

int row = scanner.nextInt() - 1;

System.out.println("Enter seat number (1-16): ");

int seat = scanner.nextInt() - 1;

if (isValidSeat(row, seat) && seats[row][seat] == 0) {

System.out.println("Enter your name: ");

String name = scanner.next();

System.out.println("Enter your surname: ");

String surname = scanner.next();

System.out.println("Enter your email: ");

String email = scanner.next();

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

int price = PRICES[row];

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

seats[row][seat] = 1;

tickets[ticketCount++] = ticket;

System.out.println("The seat has been booked");

} else {

System.out.println("This seat is not available or invalid seat number");

}

}

private static void cancel\_ticket(Scanner scanner) {

System.out.println("Enter row number (1-5): ");

int row = scanner.nextInt() - 1;

System.out.println("Enter seat number (1-16): ");

int seat = scanner.nextInt() - 1;

if (isValidSeat(row, seat) && seats[row][seat] == 1) {

seats[row][seat] = 0;

for (int i = 0; i < ticketCount; i++) {

if (tickets[i].getRow() == row && tickets[i].getSeat() == seat) {

tickets[i] = tickets[--ticketCount]; // Remove the ticket

tickets[ticketCount] = null; // Nullify the last element

System.out.println("The seat has been cancelled");

return;

}

}

} else {

System.out.println("This seat is already available or invalid seat number");

}

}

private static void print\_seating\_area() {

System.out.println("Seating area (O = available, X = sold):");

System.out.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n\*\t\tSCREEN \t \t\*\n\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

for (int row = 0; row < ROWS; row++) {

for (int seat = 0; seat < SEATS; seat++) {

if (seat == 8) System.out.print(" "); // Gap between seats 8 and 9

System.out.print(seats[row][seat] == 0 ? 'O' : 'X');

}

System.out.println("\t" + "($" + PRICES[row] + ")");

}

}

private static void find\_first\_available() {

for (int row = 0; row < ROWS; row++) {

for (int seat = 0; seat < SEATS; seat++) {

if (seats[row][seat] == 0) {

System.out.println("First available seat: Row " + (row + 1) + ", Seat " + (seat + 1));

return;

}

}

}

System.out.println("No available seats");

}

private static void print\_tickets\_info() {

int total = 0;

for (int i = 0; i < ticketCount; i++) {

tickets[i].printTicketInfo();

total += tickets[i].getPrice();

}

System.out.println("Total price of tickets sold: $" + total);

}

private static void search\_ticket(Scanner scanner) {

System.out.println("Enter row number (1-5): ");

int row = scanner.nextInt() - 1;

System.out.println("Enter seat number (1-16): ");

int seat = scanner.nextInt() - 1;

if (isValidSeat(row, seat) && seats[row][seat] == 1) {

for (int i = 0; i < ticketCount; i++) {

if (tickets[i].getRow() == row && tickets[i].getSeat() == seat) {

tickets[i].printTicketInfo();

return;

}

}

}

else if(isValidSeat(row, seat) && seats[row][seat] == 0){

System.out.println("This seat is available");

}

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

}

private static void sort\_tickets() {

for (int i = 0; i < ticketCount - 1; i++) {

for (int j = 0; j < ticketCount - i - 1; j++) {

if (tickets[j].getPrice() > tickets[j + 1].getPrice()) {

Ticket temp = tickets[j];

tickets[j] = tickets[j + 1];

tickets[j + 1] = temp;

}

}

}

for (int i = 0; i < ticketCount; i++) {

tickets[i].printTicketInfo();

}

}

private static boolean isValidSeat(int row, int seat) {

return row >= 0 && row < ROWS && seat >= 0 && seat < SEATS;

}

}

**Ticket Class**

public class Ticket {

private int row;

private int seat;

private int price;

private Person person;

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

this.row = row;

this.seat = seat;

this.price = price;

this.person = person;

}

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 int getPrice() {

return price;

}

public void setPrice(int price) {

this.price = price;

}

public Person getPerson() {

return person;

}

public void setPerson(Person person) {

this.person = person;

}

public void printTicketInfo() {

System.out.println("Row: " + (row + 1) + ", Seat: " + (seat + 1) + ", Price: $" + price);

person.printPersonInfo();

}

}

**Person Class**

public class Person {

private String name;

private String surname;

private String email;

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

this.name = name;

this.surname = surname;

this.email = email;

}

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 printPersonInfo() {

System.out.println("Name: " + name + ", Surname: " + surname + ", Email: " + email);

}

}

<<END>>