

Advanced Programming

# REPORT

## Description of the problem

Creating a sample booking system for MC cinema involves two classes: `Demo` with the main function and `Cinema`, which holds and manages cinema information. The aim is to conduct JUnit tests on three functions in the `Cinema` class and draw a UML diagram to illustrate the system's structure.

## Cinema Class

```
package com.assignment.mypckg;

public class Cinema {

    //member variables
    private int numOfRows;
    private int availableSeats;
    char[] seats;
    private String dateOfShow;
    private double priceStandard;
    private double pricePensioner;
    private double priceFrequent;
    private double totalRefundAmount = 0;
    private int standardSeats = 0, pensionerSeats = 0, frequentSeats = 0;
    private int refundedStandardSeats = 0, refundedPensionerSeats = 0,
    refundedFrequentSeats = 0;
    private String seatNumbers;

    //Constructor to initialize
    public Cinema(int numOfRows, int availableSeats, char[] seats, String
    dateOfShow, double priceStandard,
        double pricePensioner, double priceFrequent) {
        this.numOfRows = numOfRows;
        this.availableSeats = availableSeats;
        this.seats = seats;
        this.dateOfShow = dateOfShow;
        this.priceStandard = priceStandard;
        this.pricePensioner = pricePensioner;
        this.priceFrequent = priceFrequent;
    }

    //getter methods
    public int getAvailableSeats() {
        return availableSeats;
    }

    public char[] getSeats() {
        return seats;
    }

    public double getPriceStandard() {
        return priceStandard;
    }

    public double getPricePensioner() {
        return pricePensioner;
    }
}
```

```

    }

    public double getPriceFrequent() {
        return priceFrequent;
    }

    //setter method
    public void setAvailableSeats(int availableSeats) {
        this.availableSeats = availableSeats;
    }

    //Method to display available seats
    public void displayAvailableSeats() {
        System.out.println();

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

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

        int seatCounter = 0;
        int rowGap = 1;
        for(int i = 0; i < numOfRows * 10; i++) {
            if(seatCounter == 0){
                System.out.print(" "); //First row space
            }
            System.out.print((i + 1) + ":" + seats[i] + " ");
            seatCounter++;
            //Row gap - passage
            if (seatCounter / 5 == rowGap) {
                System.out.print(" ");
                rowGap++;
            }
            //To check if 10 seats per row
            if (seatCounter % 10 == 0) {
                System.out.println(); //Moves to the next row
            }
        }
        System.out.println();
        //Total number of seats
        System.out.println("Number of available seats: " + availableSeats);
        System.out.println();
    }

    //Method to book seats
    public void bookSeats(int seatNum, char category) {
        switch (category) {
            case 'S':
                standardSeats++;
                break;

            case 'P':
                pensionerSeats++;

```

```

        break;

    case 'F':
        frequentSeats++;
        break;

    default:
        System.out.println("Invalid seat category. Please try again");
        return;
    }

    seats[seatNum - 1] = category;
    availableSeats--; //Updating the number of available seats

}

public void printReciepts() {
    //Calculate totalCost based on booked seats
    double totalCost = (standardSeats* priceStandard) + (pensionerSeats *
pricePensioner) + (frequentSeats * priceFrequent);

    //Printing the reciept details
    System.out.println("Receipt\n*****");
    System.out.println("Date: " + dateOfShow);
    System.out.println("Number of seats booked: " + (standardSeats +
pensionerSeats + frequentSeats));
    System.out.println(standardSeats + " * Standard @ $" +
priceStandard + " = $" + (standardSeats * priceStandard) + " seat(s): " +
getBookedSeats('S'));
    System.out.println(pensionerSeats + " * Pensioner @ $" +
pricePensioner + " = $" + (pensionerSeats * pricePensioner) + " seat(s): " +
getBookedSeats('P'));
    System.out.println(frequentSeats + " * Frequent @ $" +
priceFrequent + " = $" + (frequentSeats * priceFrequent) + " seat(s): " +
getBookedSeats('F'));

    System.out.println("                        Total : $" + totalCost);

    System.out.println();

    //Resetting the seat count
    standardSeats = 0;
    pensionerSeats = 0;
    frequentSeats = 0;
}

//Method to print the number of seats for a specific category
public String getBookedSeats(char category) {
    String seatNumbers = "";
    boolean firstSeat = true;

    for (int i = 0; i < seats.length; i++) {
        if (seats[i] == category) {
            if (!firstSeat) {
                seatNumbers += ", ";
            } else {

```

```

        firstSeat = false;
    }
    seatNumbers += (i + 1);
}

return seatNumbers;
}

//Method to refund seats
public void refundSeats(char category) {

    switch (category) {
        case 'S':
            standardSeats--;
            refundedStandardSeats++;
            break;

        case 'P':
            pensionerSeats--;
            refundedPensionerSeats++;

            break;

        case 'F':
            frequentSeats--;
            refundedFrequentSeats++;
            break;
    }
    availableSeats++;
}

//Method to print the refunded receipt
public void printRefundReceipt(int refundedSeats) {

    //Calculate total refund amount
    double totalRefundAmount = (refundedStandardSeats* priceStandard) +
(refundedPensionerSeats * pricePensioner) + (refundedFrequentSeats *
priceFrequent);

    //Printing the refund details
    System.out.println("Refund Receipt\n*****");
    System.out.println("Date: " + dateOfShow);
    System.out.println("Number of seats refunded: " + refundedSeats);
    System.out.println(refundedStandardSeats + " * Standard @ $" +
priceStandard + " = $" + (refundedStandardSeats * priceStandard) + "
seat(s): " + getRefundedSeats('S'));
    System.out.println(refundedPensionerSeats + " * Pensioner @ $" +
pricePensioner + " = $" + (refundedPensionerSeats * pricePensioner) + "
seat(s): " + getRefundedSeats('P'));
    System.out.println(refundedFrequentSeats + " * Frequent @ $" +
priceFrequent + " = $" + (refundedFrequentSeats * priceFrequent) + "
seat(s): " + getRefundedSeats('F'));

    System.out.println("Total Refund: $" + totalRefundAmount);

    //Resetting the refund seat count

```

```

        refundedStandardSeats = 0;
        refundedPensionerSeats = 0;
        refundedFrequentSeats = 0;
    }

    //Method to get the number of seats for a specific category
    public String getRefundedSeats(char category) {

        String seatNumbers = "";
        boolean firstSeat = true;

        for (int i = 0; i < seats.length; i++) {
            if (seats[i] == category) {
                if (!firstSeat) {
                    seatNumbers += ", ";
                } else {
                    firstSeat = false;
                }
                seatNumbers += (i + 1);
            }
        }
        return seatNumbers;
    }

    //Method to print the statistic report
    public void displayReport() {

        double averageSeatPrice = 0.0;

        System.out.println(); //Extra line for formatting

        //Calculating the total number of seats booked
        int totalSeatsBooked = numOfRows * 10 - availableSeats;

        //Calculating the percentage of seats sold
        double percentageSold = ((double) totalSeatsBooked / (numOfRows *
10)) * 100;
        String percentageSeatsSold = String.format("%.2f%", percentageSold);

        if (totalSeatsBooked > 0) {
            averageSeatPrice = ((totalSeatsBooked * priceStandard) +
(totalSeatsBooked * pricePensioner)
            + (totalSeatsBooked * priceFrequent)) / totalSeatsBooked;
        }

        //Printing the report details
        System.out.println("Statistic Report\n*****");
        System.out.println("Number of seats sold      : " +
totalSeatsBooked);
        System.out.println("Percentage of seats sold : " +
percentageSeatsSold + "%");
        System.out.println("Average seat price      : $" +
averageSeatPrice);

        System.out.println();
    }
}

```

## Demo class

```
package com.assignment.mypckg;

import java.util.Scanner;

public class Demo {

    //main method
    public static void main(String[] args) {

        Scanner input = new Scanner(System.in);

        int seatNumber = 0;
        char category;

        System.out.println("Welcome to the MC Ticket Booking System");

        //Entering the number of rows
        System.out.println("Enter the Number of Rows");
        int numOfRows = input.nextInt();

        int availableSeats = numOfRows * 10;
        char[] seats = new char[availableSeats]; //Initialize the seat
Array

        for (int i = 0; i < numOfRows * 10; i++) {
            seats[i] = '-'; //Initializing all the seats as not booked
        }

        input.nextLine();

        System.out.println("Enter the Date of Show");
        String dateOfShow = input.nextLine();

        //Entering the price for different seat categories
        System.out.println("Enter the price for a Standard Seat (S): ");
        double priceStandard = input.nextDouble();

        System.out.println("Enter the price for a Pensioner Seat (P): ");
        double pricePensioner = input.nextDouble();
    }
}
```

```
System.out.println("Enter the price for a Frequent patron Seat  
(F): ");  
  
double priceFrequent = input.nextDouble();  
  
System.out.println();  
  
//creating a cinema object  
Cinema c = new Cinema(numOfRows, availableSeats, seats,  
dateOfShow, priceStandard,  
pricePensioner, priceFrequent);  
  
boolean program = true;  
  
while(program) {  
  
    System.out.println("1. Display available seats");  
System.out.println("2. Book a seat");  
System.out.println("3. Refund a seat");  
System.out.println("4. Display a report");  
System.out.println("5. Exit");  
System.out.println("Enter your choice:");  
  
int choice = input.nextInt();  
  
//Handles user choices  
switch(choice) {  
case 1:  
    c.displayAvailableSeats();  
break;  
  
case 2:  
System.out.println();  
System.out.println("Enter the number of seats to book");  
int seatsToBook = input.nextInt();  
  
//Checking seat availability  
if(seatsToBook <= availableSeats) {  
    System.out.println("Enter the required seat number");  
  
    //Seat Number  
for(int i = 0; i < seatsToBook; i++) {  
        System.out.println("Seat " + (i + 1) + ":");  
seatNumber = input.nextInt();  
  
        if(seatNumber > 0 && seatNumber <= numOfRows *  
10) {  
            if(seats[seatNumber - 1] == '-') {  
                availableSeats--; //Updating the  
available seats  
  
                //Seat category  
System.out.println("Enter the seat  
category");  
  
                while (true) {  
                    category =
```



```

        if (category == 'S' || category == 'P'
|| category == 'F' ) {
            c.bookSeats(seatNumber, category);
//Calling the book seats method
            break;
        } else {
            System.out.println("Invalid seat
category. Please try again");
        }
    }
    }else {
        System.out.println("Seat " + seatNumber +
" is already booked.");
        i--; //Re-enter the seat number
    }
    }else {
        System.out.println("Invalid seat number. Seat "
+ seatNumber + " does not exist.");
        i--; //Re-enter the seat number
    }
    }
    System.out.println();
    c.printReciepts(); //Printing the receipt for the currently
booked seats
    }else {
        System.out.println("Seats Unavailable. There are only
" + availableSeats + " seat(s) available");
        System.out.println();
        break;
    }
    break;

    case 3:
        System.out.println();

        int refundedSeats = 0;

        System.out.println("Enter the number of seats to refund:");
        int seatsToRefund = input.nextInt();

        //Counting the booked seats
        int bookedSeats = 0;
        for (char seat : seats) {
            if (seat != '-') {
                bookedSeats++;
            }
        }
        if(seatsToRefund <= bookedSeats) {
            System.out.println("Enter the booked seat number to
refund:");

            for(int i = 0; i < seatsToRefund; i++) {
                System.out.println("Seat " + (i + 1) + ":" );
                seatNumber = input.nextInt();

                if (seatNumber > 0 && seatNumber <= numOfRows *
10) {

```

```

        if(seats[seatNumber - 1] != '-') {
            category = seats[seatNumber - 1];
            seats[seatNumber - 1] = '-';
            refundedSeats++; //Updating the
refunded seat count
                                c.refundSeats(category); //Calling
the refund seats methods
        }else {
            System.out.println("Seat " +
seatNumber + " was not booked. Enter the correct seat number.");
            i--; //Re-enter the seat number
        }
    }else {
        System.out.println("Invalid seat number.
Please enter a valid seat number.");
        i--; //Re-enter the seat number
    }
    }
    availableSeats += refundedSeats; //Updating the
available seats
    System.out.println();
    c.printRefundReceipt(refundedSeats); //Printing the
refund receipt
    }else {
        System.out.println("Refund quantity cannot exceed the
number of seats booked");
    }
    System.out.println();
    break;

    case 4:
        c.displayReport();
        break;

    case 5:
        program = false;
        System.out.println("Leaving the system. Appreciate your
interaction!");
        break;

    default:
        System.out.println("Invalid selection. Please choose a valid
option.");
    }
}
}
}
}

```

## Output

```
<terminated> Demo (2) [Java Application] C:\Users\DELL\Desktop\Softwares\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831
Welcome to the MC Ticket Booking System
Enter the Number of Rows
10
Enter the Date of Show
2023-12-14
Enter the price for a Standard Seat (S):
1500.00
Enter the price for a Pensioner Seat (P):
1200.00
Enter the price for a Frequent patron Seat (F):
1000.00

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
1

*****
*                               *
*               SCREEN          *
*                               *
*****

    1:- 2:- 3:- 4:- 5:-          6:- 7:- 8:- 9:- 10:-
11:- 12:- 13:- 14:- 15:-        16:- 17:- 18:- 19:- 20:-
21:- 22:- 23:- 24:- 25:-        26:- 27:- 28:- 29:- 30:-
31:- 32:- 33:- 34:- 35:-        36:- 37:- 38:- 39:- 40:-
41:- 42:- 43:- 44:- 45:-        46:- 47:- 48:- 49:- 50:-
51:- 52:- 53:- 54:- 55:-        56:- 57:- 58:- 59:- 60:-
61:- 62:- 63:- 64:- 65:-        66:- 67:- 68:- 69:- 70:-
71:- 72:- 73:- 74:- 75:-        76:- 77:- 78:- 79:- 80:-
81:- 82:- 83:- 84:- 85:-        86:- 87:- 88:- 89:- 90:-
91:- 92:- 93:- 94:- 95:-        96:- 97:- 98:- 99:- 100:-

Number of available seats: 100
```

```
<terminated> Demo (2) [Java Application] C:\Users\DELL\Desktop\Softwares\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre
1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
2

Enter the number of seats to book
5
Enter the required seat number
Seat 1:
40
Enter the seat category
S
Seat 2:
41
Enter the seat category
P
Seat 3:
43
Enter the seat category
P
Seat 4:
49
Enter the seat category
F
Seat 5:
46
Enter the seat category
P

Receipt
*****
Date: 2023-12-14
Number of seats booked: 5
1 * Standard @ $1500.0 = $1500.0 seat(s): 40
3 * Pensioner @ $1200.0 = $3600.0 seat(s): 41, 43, 46
1 * Frequent @ $1000.0 = $1000.0 seat(s): 49
Total : $6100.0
```

```

<terminated> Demo (2) [Java Application] C:\Users\DELL\Desktop\Softwares\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre
1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
1

*****
*                               *
*                               *
*****

1:- 2:- 3:- 4:- 5:-          6:- 7:- 8:- 9:- 10:-
11:- 12:- 13:- 14:- 15:-      16:- 17:- 18:- 19:- 20:-
21:- 22:- 23:- 24:- 25:-      26:- 27:- 28:- 29:- 30:-
31:- 32:- 33:- 34:- 35:-      36:- 37:- 38:- 39:- 40:S
41:P 42:- 43:P 44:- 45:-      46:P 47:- 48:- 49:F 50:-
51:- 52:- 53:- 54:- 55:-      56:- 57:- 58:- 59:- 60:-
61:- 62:- 63:- 64:- 65:-      66:- 67:- 68:- 69:- 70:-
71:- 72:- 73:- 74:- 75:-      76:- 77:- 78:- 79:- 80:-
81:- 82:- 83:- 84:- 85:-      86:- 87:- 88:- 89:- 90:-
91:- 92:- 93:- 94:- 95:-      96:- 97:- 98:- 99:- 100:-

Number of available seats: 95

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
2

Enter the number of seats to book
96
Seats Unavailable. There are only 95 seat(s) available

```

```

<terminated> Demo (2) [Java Application] C:\Users\DELL\Desktop\Softwares\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre
1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
1

*****
*                               *
*                               *
*****

1:- 2:- 3:- 4:- 5:-          6:- 7:- 8:- 9:- 10:-
11:- 12:- 13:- 14:- 15:-      16:- 17:- 18:- 19:- 20:-
21:- 22:- 23:- 24:- 25:-      26:- 27:- 28:- 29:- 30:-
31:- 32:- 33:- 34:- 35:-      36:- 37:- 38:- 39:- 40:S
41:P 42:- 43:P 44:- 45:-      46:P 47:- 48:- 49:F 50:-
51:- 52:- 53:- 54:- 55:-      56:- 57:- 58:- 59:- 60:-
61:- 62:- 63:- 64:- 65:-      66:- 67:- 68:- 69:- 70:-
71:- 72:- 73:- 74:- 75:-      76:- 77:- 78:- 79:- 80:-
81:- 82:- 83:- 84:- 85:-      86:- 87:- 88:- 89:- 90:-
91:- 92:- 93:- 94:- 95:-      96:- 97:- 98:- 99:- 100:-

Number of available seats: 95

```

```
<terminated> Demo (2) [Java Application] C:\Users\DELL\Desktop\Softwares\eclipse\plugins\org.eclipse.justi.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre
1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
2

Enter the number of seats to book
2
Enter the required seat number
Seat 1:
31
Enter the seat category
P
Seat 2:
41
Seat 41 is already booked.
Seat 2:
32
Enter the seat category
P

Receipt
*****
Date: 2023-12-14
Number of seats booked: 2
0 * Standard @ $1500.0 = $0.0 seat(s): 40
2 * Pensioner @ $1200.0 = $2400.0 seat(s): 31, 32, 41, 43, 46
0 * Frequent @ $1000.0 = $0.0 seat(s): 49
Total : $2400.0

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
1
```

```
*****
*                                     *
*                               SCREEN                               *
*****
1:- 2:- 3:- 4:- 5:-                6:- 7:- 8:- 9:- 10:-
11:- 12:- 13:- 14:- 15:-           16:- 17:- 18:- 19:- 20:-
21:- 22:- 23:- 24:- 25:-           26:- 27:- 28:- 29:- 30:-
31:P 32:P 33:- 34:- 35:-           36:- 37:- 38:- 39:- 40:S
41:P 42:- 43:P 44:- 45:-           46:P 47:- 48:- 49:F 50:-
51:- 52:- 53:- 54:- 55:-           56:- 57:- 58:- 59:- 60:-
61:- 62:- 63:- 64:- 65:-           66:- 67:- 68:- 69:- 70:-
71:- 72:- 73:- 74:- 75:-           76:- 77:- 78:- 79:- 80:-
81:- 82:- 83:- 84:- 85:-           86:- 87:- 88:- 89:- 90:-
91:- 92:- 93:- 94:- 95:-           96:- 97:- 98:- 99:- 100:-

Number of available seats: 93

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
2

Enter the number of seats to book
101
Seats Unavailable. There are only 93 seat(s) available

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
2
```

<terminated> Demo (2) [Java Application] C:\Users\DELL\Desktop\Softwares\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.8.v20230831-1047\jre

Enter the number of seats to book

101

Seats Unavailable. There are only 93 seat(s) available

1. Display available seats

2. Book a seat

3. Refund a seat

4. Display a report

5. Exit

Enter your choice:

2

Enter the number of seats to book

1

Enter the required seat number

Seat 1:

101

Invalid seat number. Seat 101 does not exist.

Seat 1:

5

Enter the seat category

S

Receipt

\*\*\*\*\*

Date: 2023-12-14

Number of seats booked: 1

1 \* Standard @ \$1500.0 = \$1500.0 seat(s): 5, 40

0 \* Pensioner @ \$1200.0 = \$0.0 seat(s): 31, 32, 41, 43, 46

0 \* Frequent @ \$1000.0 = \$0.0 seat(s): 49

Total : \$1500.0

1. Display available seats

2. Book a seat

3. Refund a seat

4. Display a report

5. Exit

Enter your choice:

1



```
*****
*                                     *
*                               SCREEN                               *
*                                     *
*****

1:- 2:- 3:- 4:- 5:S          6:- 7:- 8:- 9:- 10:-
11:- 12:- 13:- 14:- 15:-    16:- 17:- 18:- 19:- 20:-
21:- 22:- 23:- 24:- 25:-    26:- 27:- 28:- 29:- 30:-
31:P 32:P 33:- 34:- 35:-    36:- 37:- 38:- 39:- 40:S
41:P 42:- 43:P 44:- 45:-    46:P 47:- 48:- 49:F 50:-
51:- 52:- 53:- 54:- 55:-    56:- 57:- 58:- 59:- 60:-
61:- 62:- 63:- 64:- 65:-    66:- 67:- 68:- 69:- 70:-
71:- 72:- 73:- 74:- 75:-    76:- 77:- 78:- 79:- 80:-
81:- 82:- 83:- 84:- 85:-    86:- 87:- 88:- 89:- 90:-
91:- 92:- 93:- 94:- 95:-    96:- 97:- 98:- 99:- 100:-

Number of available seats: 92

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
2

Enter the number of seats to book
2
Enter the required seat number
Seat 1:
9
Enter the seat category
F
Seat 2:
10
Enter the seat category
F
```

```

Receipt
*****
Date: 2023-12-14
Number of seats booked: 2
0 * Standard @ $1500.0 = $0.0 seat(s): 5, 40
0 * Pensioner @ $1200.0 = $0.0 seat(s): 31, 32, 41, 43, 46
2 * Frequent @ $1000.0 = $2000.0 seat(s): 9, 10, 49
      Total : $2000.0

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
1

*****
*                               *
*                               *
*****

1:- 2:- 3:- 4:- 5:S          6:- 7:- 8:- 9:F 10:F
11:- 12:- 13:- 14:- 15:-    16:- 17:- 18:- 19:- 20:-
21:- 22:- 23:- 24:- 25:-    26:- 27:- 28:- 29:- 30:-
31:P 32:P 33:- 34:- 35:-    36:- 37:- 38:- 39:- 40:S
41:P 42:- 43:P 44:- 45:-    46:P 47:- 48:- 49:F 50:-
51:- 52:- 53:- 54:- 55:-    56:- 57:- 58:- 59:- 60:-
61:- 62:- 63:- 64:- 65:-    66:- 67:- 68:- 69:- 70:-
71:- 72:- 73:- 74:- 75:-    76:- 77:- 78:- 79:- 80:-
81:- 82:- 83:- 84:- 85:-    86:- 87:- 88:- 89:- 90:-
91:- 92:- 93:- 94:- 95:-    96:- 97:- 98:- 99:- 100:-

Number of available seats: 90

```

```

<terminated> Demo (2) [Java Application] C:\Users\DELL\Desktop\Softwares\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre
1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
3

Enter the number of seats to refund:
2
Enter the booked seat number to refund:
Seat 1:
9
Seat 2:
10

Refund Receipt
*****
Date: 2023-12-14
Number of seats refunded: 2
0 * Standard @ $1500.0 = $0.0 seat(s): 5, 40
0 * Pensioner @ $1200.0 = $0.0 seat(s): 31, 32, 41, 43, 46
2 * Frequent @ $1000.0 = $2000.0 seat(s): 49
Total Refund: $2000.0

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
1

```

```

<terminated> Demo (2) [Java Application] C:\Users\DELL\Desktop\Softwares\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.8.v20230831-1047\jre
*****
*                               SCREEN                               *
*****

1:- 2:- 3:- 4:- 5:S          6:- 7:- 8:- 9:- 10:-
11:- 12:- 13:- 14:- 15:-    16:- 17:- 18:- 19:- 20:-
21:- 22:- 23:- 24:- 25:-    26:- 27:- 28:- 29:- 30:-
31:P 32:P 33:- 34:- 35:-    36:- 37:- 38:- 39:- 40:S
41:P 42:- 43:P 44:- 45:-    46:P 47:- 48:- 49:F 50:-
51:- 52:- 53:- 54:- 55:-    56:- 57:- 58:- 59:- 60:-
61:- 62:- 63:- 64:- 65:-    66:- 67:- 68:- 69:- 70:-
71:- 72:- 73:- 74:- 75:-    76:- 77:- 78:- 79:- 80:-
81:- 82:- 83:- 84:- 85:-    86:- 87:- 88:- 89:- 90:-
91:- 92:- 93:- 94:- 95:-    96:- 97:- 98:- 99:- 100:-

Number of available seats: 92

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
3

Enter the number of seats to refund:
9
Refund quantity cannot exceed the number of seats booked

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit
Enter your choice:
3

```

<terminated> Demo (2) [Java Application] C:\Users\DELL\Desktop\Softwares\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.8.v20230831-1047\j

Enter the number of seats to refund:

9

Refund quantity cannot exceed the number of seats booked

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit

Enter your choice:

3

Enter the number of seats to refund:

1

Enter the booked seat number to refund:

Seat 1:

41

Refund Receipt

\*\*\*\*\*

Date: 2023-12-14

Number of seats refunded: 1

0 \* Standard @ \$1500.0 = \$0.0 seat(s): 5, 40

1 \* Pensioner @ \$1200.0 = \$1200.0 seat(s): 31, 32, 43, 46

0 \* Frequent @ \$1000.0 = \$0.0 seat(s): 49

Total Refund: \$1200.0

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit

Enter your choice:

1

<terminated> Demo (2) [Java Application] C:\Users\DELL\Desktop\Softwares\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86\_64\_17.0.8.v20230831-1047\j

\*\*\*\*\*  
\* SCREEN \*  
\*\*\*\*\*

1:- 2:- 3:- 4:- 5:S	6:- 7:- 8:- 9:- 10:-
11:- 12:- 13:- 14:- 15:-	16:- 17:- 18:- 19:- 20:-
21:- 22:- 23:- 24:- 25:-	26:- 27:- 28:- 29:- 30:-
31:P 32:P 33:- 34:- 35:-	36:- 37:- 38:- 39:- 40:S
41:- 42:- 43:P 44:- 45:-	46:P 47:- 48:- 49:F 50:-
51:- 52:- 53:- 54:- 55:-	56:- 57:- 58:- 59:- 60:-
61:- 62:- 63:- 64:- 65:-	66:- 67:- 68:- 69:- 70:-
71:- 72:- 73:- 74:- 75:-	76:- 77:- 78:- 79:- 80:-
81:- 82:- 83:- 84:- 85:-	86:- 87:- 88:- 89:- 90:-
91:- 92:- 93:- 94:- 95:-	96:- 97:- 98:- 99:- 100:-

Number of available seats: 93

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit

Enter your choice:

4

Statistic Report

\*\*\*\*\*

Number of seats sold : 7

Percentage of seats sold : 7.00%%

Average seat price : \$3700.0

1. Display available seats
2. Book a seat
3. Refund a seat
4. Display a report
5. Exit

Enter your choice:

5

Leaving the system. Appreciate your interaction!

## **TestCinema class**

```
package com.assignment.mypckg;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Assertions;
import org.junit.jupiter.api.Test;

class TestCinema {

    //assertEquals
    @Test
    public void testBookSeats() {

        Cinema c = new Cinema(10, 100, new char[100], "2023/07/15", 15.00, 12.00,
10.00);

        int availableSeats = c.getAvailableSeats();
        int seatsToBook = availableSeats - 95;
        for(int i = 1; i <= seatsToBook; i++) {
            c.bookSeats(i, 'S');
        }

        char[] seats = c.getSeats();
        for(int i = 0; i < seatsToBook; i++) {
            Assertions.assertEquals('S', seats[i]);
        }

    }
}
```

```
//Fail Test case
```

```
@Test
```

```
public void testBookSeatsFail() {
```

```
    Cinema c = new Cinema(5, 50, new char[50], "2023/07/20", 15.00, 12.00, 10.00);
```

```
    int availableSeats = c.getAvailableSeats();
```

```
    int seatsToBook = availableSeats - 45;
```

```
    for(int i = 1; i <= seatsToBook; i++) {
```

```
        c.bookSeats(i, 'S');
```

```
    }
```

```
    char[] seats = c.getSeats();
```

```
    for(int i = 0; i < seatsToBook; i++) {
```

```
        Assertions.assertEquals('F', seats[i], "seat category");
```

```
    }
```

```
}
```

```
//assertTrue
```

```
@Test
```

```
public void testAvailableSeatsIncreaseAfterRefund() {
```

```
    Cinema c = new Cinema(5, 50, new char[50], "2023/07/20", 15.00, 12.00, 10.00);
```

```
    //Assuming there are initially 5 available seats
```

```
    int initialAvailableSeats = 5;
```

```
    c.setAvailableSeats(initialAvailableSeats);
```

```

// Refund 2 standard seats
c.refundSeats('S');
c.refundSeats('P');

//Checking if the available seats have increased by 2
assertTrue(c.getAvailableSeats() == initialAvailableSeats + 2);
}

//assertFalse
@Test
public void testDisplayReport() {
    Cinema c = new Cinema(5, 50, new char[50], "2023/12/31", 20.0, 15.0, 10.0);

    //Assuming
    int availableSeats = c.getAvailableSeats();
    int totalSeatsBooked = availableSeats - 40;
    double percentageSold = (totalSeatsBooked / availableSeats) * 100;
    double averageSeatPrice = (c.getPriceStandard() + c.getPricePensioner() +
c.getPriceFrequent() / 3);

    Assertions.assertTrue(totalSeatsBooked >= 0, "Total seats sold should be greater than or
equal to zero");

    Assertions.assertFalse(percentageSold >= 10 && percentageSold <= 100, "Percentage
sold should be between 0 and 100");

    Assertions.assertFalse(averageSeatPrice <= 0, "Average seat price should be greater than
or equal to zero");
}

//assertFalse - fail
@Test


```


```
public void testDisplayReportFail() {  
    Cinema c = new Cinema(10, 100, new char[100], "2023/12/31", 20.0, 15.0, 10.0);  
  
    //Assuming  
    int availableSeats = c.getAvailableSeats();  
    int totalSeatsBooked = availableSeats - 40;  
    double percentageSold = (totalSeatsBooked / availableSeats) * 100;  
    double averageSeatPrice = (c.getPriceStandard() + c.getPricePensioner() +  
c.getPriceFrequent() / 3);  
  
    Assertions.assertTrue(totalSeatsBooked >= 0, "Total seats sold should be greater than or  
equal to zero");  
    Assertions.assertFalse(percentageSold <= 0 || percentageSold >= 100, "Percentage sold  
should be between 0 and 100");  
    Assertions.assertFalse(averageSeatPrice <= 0, "Average seat price should be greater than  
or equal to zero");  
    }  
}
```

## Test Cases Output

Finished after 0.148 seconds


Runs: 1/1      ✖ Errors: 0      ✖ Failures: 0





>  TestCinema [Runner: JUnit 5] (0.025 s)

Finished after 0.142 seconds

Runs: 1/1      ✖ Errors: 0      ✖ Failures: 1



▼  TestCinema [Runner: JUnit 5] (0.029 s)

-  testBookSeatsFail() (0.029 s)




Finished after 0.129 seconds

Runs: 1/1

✖ Errors: 0

✖ Failures: 0


>  TestCinema [Runner: JUnit 5] (0.019 s)

Finished after 0.15 seconds

Runs: 1/1

✖ Errors: 0

✖ Failures: 0


>  TestCinema [Runner: JUnit 5] (0.025 s)


Finished after 0.182 seconds

Runs: 1/1

✖ Errors: 0

✖ Failures: 1

✓  TestCinema [Runner: JUnit 5] (0.039 s)

✖  testDisplayReportFail() (0.039 s)

## UML DIAGRAM

### Cinema Class

Cinema
<ul style="list-style-type: none"><li>- numofRows: int</li><li>- availableSeats: int</li><li>- seats: char[]</li><li>- dateOfShow: String</li><li>- priceStandard: double</li><li>- pricePensioner: double</li><li>- priceFrequent: double</li><li>- totalRefundAmount: double</li><li>- standardSeats: int</li><li>- pensionerSeats: int</li><li>- frequentSeats: int</li><li>- refundedStandardSeats: int</li><li>- refundedPensionerSeats: int</li><li>- refundedFrequentSeats: int</li></ul>
<ul style="list-style-type: none"><li>+ Cinema(numofRows:int, availableSeats:int, char[] seats, dateOfShow:String, priceStandard:double, pricePensioner:double, priceFrequent:double)</li><li>+ getAvailableSeats(): int</li><li>+ getSeats(): char[]</li><li>+ setAvailableSeats(availableSeats:int): void</li><li>+ displayAvailableSeats(): void</li><li>+ bookSeats(seatNum:int, category:char): void</li><li>+ printReceipts(): void</li><li>+ getBookedSeats(category:char): String</li><li>+ refundSeats(category:char): void</li><li>+ printRefundReceipt(refundedSeats:int): void</li><li>+ getRefundedSeats(category:char): String</li><li>+ displayReport(): void</li></ul>

## Demo Class

Demo
- main(String[]): void