



**NEW HORIZON  
COLLEGE OF ENGINEERING**

Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC  
Accredited by NAAC with 'A' Grade & Accredited by NBA



## **A MINI PROJECT**

## **REPORT**

*for*

*Mini Project in JAVA (19CSE48)*

## **MOVIE BOOKING**

*Submitted by*

**S REVANTH KUMAR**

**USN: 1NH19CS149,  
Semester-Section: 4-C**

*In partial fulfillment for the award of  
the degree of*

**Bachelor of Engineering**

*in*

**COMPUTER SCIENCE AND ENGINEERING**



**NEW HORIZON  
COLLEGE OF ENGINEERING**

Autonomous College, Affiliated to VTU | Approved by AICTE New Delhi & UGC  
Accredited by NAAC with 'A' Grade & Accredited by NBA



## ***Certificate***

*This is to certify that the mini project work titled*

### **MOVIE BOOKING**

*Submitted in partial fulfillment of the degree of  
Bachelor of Engineering in  
Computer Science and Engineering by*

**S REVANTH KUMAR**  
USN: 1NH19CS149

*DURING*

*EVEN SEMESTER 2020-2021*

*for*

*COURSE CODE: 19CSE48*

  
Signature of Reviewer

  
Signature of HOD

### **SEMESTER END EXAMINATION**

*Name of the Examiner*

*Signature with date*

1. \_\_\_\_\_

\_\_\_\_\_

2. \_\_\_\_\_

\_\_\_\_\_

## ORIGINALITY REPORT

---

12%

SIMILARITY INDEX

%

INTERNET SOURCES

12%

PUBLICATIONS

%

STUDENT PAPERS

---

## PRIMARY SOURCES

---

1

Kishori Sharan. "Beginning Java 9 Fundamentals", Springer Science and Business Media LLC, 2017

Publication

2%

---

2

Tan Kiat Shi, Willi-Hans Steeb, Yorick Hardy. "Chapter 4 Object-Oriented Programming", Springer Science and Business Media LLC, 2000

Publication

2%

---

3

Ali S. Janfada. "Elementary Synchronous Programming", Walter de Gruyter GmbH, 2019

Publication

1%

---

4

Akshay Bharadwaj K H, Deepak, V Ghanavanth, Harish Bharadwaj R, R Uma, Gowranga Krishnamurthy. "Smart CCTV Surveillance System for Intrusion Detection With Live Streaming", 2018 3rd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT), 2018

Publication

1%

---

5	Wang, C M, G R Liu, and K K Ang. "MORE READABLE, MANAGEABLE AND EXTENSIBLE CODES FOR FINITE ELEMENT ANALYSIS USING JAVA", Structural Stability and Dynamics, 2002. Publication	1 %
6	Daniel M. Solis. "Illustrated C# 2012", Springer Nature, 2012 Publication	1 %
7	Madhukar B.N., Sanjay Jain. "A Duality Theorem for the Discrete Sine Transform - IV (DST - IV)", 2016 3rd International Conference on Advanced Computing and Communication Systems (ICACCS), 2016 Publication	1 %
8	Luan P. Lima, Lincoln S. Rocha, Carla I. M. Bezerra, Matheus Paixao. "Assessing exception handling testing practices in open-source libraries", Empirical Software Engineering, 2021 Publication	1 %
9	Carsten Thomsen. "Database Programming with C#", Springer Nature, 2002 Publication	1 %
10	"Programming in C++", Walter de Gruyter GmbH, 2019 Publication	1 %

11

Kris Macleod Bell, Lars Ivar Igesund, Sean Kelly, Michael Parker. "Learn to Tango with D", Springer Science and Business Media LLC, 2007

Publication

1 %

12

Trey Nash. "Accelerated C# 2008", Springer Science and Business Media LLC, 2007

Publication

<1 %

13

Vaskaran Sarcar. "Interactive Object Oriented Programming in Java", Springer Science and Business Media LLC, 2016

Publication

<1 %

14

"Multimedia Introduction to Programming Using Java", Springer Science and Business Media LLC, 2005

Publication

<1 %

15

P. Devendran, S. Prasath, Smeya Mohanan, G. Praveen. "Articulated robotic system-wireless, manual and shadow mode", Materials Today: Proceedings, 2021

Publication

<1 %

16

M. Rabemanantsoa. "Knowledge-based system for assembly process-planning", Proceedings 1993 Software Engineering Standards Symposium, 1993

Publication

<1 %

# **ABSTRACT**

Movies have become a source for entertainment and art. People of all ages and group are been attracted to movies across whole world and coming to theatre to enjoy it. So, for people to enjoy a movie, booking tickets for a cinema they wish and to have seats reserved at time with various options for them is required. A movie booking platform facilitates selection, reservation and purchase of tickets for the movies. Waiting in queue for hours in long queues to get tickets booked is definitely a headache. This project is one of the best opportunities for those who cannot afford enough time to get their tickets reserved standing in long queues. The project provides complete information regarding currently running movies on all the screens with details of show timings and available seats. This project makes and entire process hassle free and less time consuming than the traditional methods of movie booking. The manual booking is slow and inconsistent. It is sometimes dependent on employees as human errors can occur which will ultimately cause Issues to customers. We cannot even guarantee security when we choose the manual or traditional way of movie booking. This project makes booking fast, consistent and time saving. Customers can even see what snacks and brands are available before heading to the canteen to buy them. This option lets the customer know what's available and what's not available in the canteen making the process even simpler and easier to choose.

# ACKNOWLEDGEMENT

The satisfaction and euphoria that accompany the successful completion of any task would be impossible without the mention of the people who made it possible, whose constant guidance and encouragement crowned our efforts with success.

I have great pleasure in expressing gratitude to **Dr. Mohan Manghnani**, Chairman of New Horizon Educational Institutions for providing necessary infrastructure and creating good environment.

I take this opportunity to express my profound gratitude to **Dr. Manjunatha**, Principal, New Horizon College of Engineering, for his constant support and encouragement.

I am grateful to **Dr. Amarjeet Singh**, Dean - Academics, for his unfailing encouragement and suggestions, given to me in the course of my project work.

I would also like to thank **Dr. B. Rajalakshmi**, Professor and Head, Department of Computer Science and Engineering, for her constant support.

I also express my gratitude to **Dr. Anidha Arulanandham, Professor**, Department of Computer Science and Engineering, my project guide, for constantly monitoring the development of the project and setting up precise deadlines. Her valuable suggestions were the motivating factors in completing the work.

**S REVANTH KUMAR**

**USN: 1NH19CS149**

# **CONTENTS**

<b>ABSTRACT</b>	<b>I</b>
<b>ACKNOWLEDGMENT</b>	<b>II</b>
<b>LIST OF FIGURES</b>	<b>V</b>

## **1. INTRODUCTION**

<b>1.1. PROBLEM DEFINITION</b>	<b>1</b>
<b>1.2. OBJECTIVES</b>	<b>1</b>
<b>1.3. METHODOLOGY TO BE FOLLOWED</b>	<b>1</b>
<b>1.4. EXPECTED OUTCOMES</b>	<b>1</b>

## **2. FUNDAMENTALS OF JAVA PROGRAMMING**

<b>2.1. INTRODUCTION</b>	<b>2</b>
<b>2.2. CLASS</b>	<b>2</b>
<b>2.3. OBJECT</b>	<b>3</b>
<b>2.4. INHERITANCE</b>	<b>4</b>
<b>2.5. POLYMORPHISM</b>	<b>7</b>
<b>2.6. ABSTRACT CLASS</b>	<b>8</b>
<b>2.7. JAVA PACKAGES</b>	<b>9</b>
<b>2.8. EXCEPTION HANDLING</b>	<b>10</b>
<b>2.9. THREADS</b>	<b>12</b>
<b>2.10. I/O BASICS</b>	<b>12</b>



<b>3. REQUIREMENTS AND SPECIFICATION</b>	
3.1. HARDWARE REQUIREMENTS	13
3.2. SOFTWARE REQUIREMENTS	13
<b>4. DESIGN</b>	
4.1. DESIGN GOALS	14
4.2. FLOWCHART/AGLORITHM	14
<b>5. IMPLEMENTATION</b>	
5.1. MODULE 1 FUNCTIONALITY	17
5.2. MODULE 1 FUNCTIONALITY	19
5.3. MODULE 1 FUNCTIONALITY	25
5.4. MODULE 4 FUNCTIONALITY	27
<b>6. RESULTS</b>	
6.1. DISPLAYS THE MENU	30
6.2. MOVIE DETAILS, TIMINGS AND SEATS	31
6.3. SEAT TYPES AND SEAT NUMBERS	32
6.4. TICKET DETAILS	33
6.5. CURRENTLY AVAILABLE MOVIES	34
6.6. EATERIES AND SNACKS	35
6.7. EXITS THE PROGRAM	36
<b>7. CONCLUSION</b>	37
<b>REFERENCES</b>	38

# LIST OF FIGURES

<b>Figure No</b>	<b>Figure Description</b>	<b>Page No</b>
2.4.1	Single Inheritance	5
2.4.2	Multi-level Inheritance	6
2.4.3	Hierarchical Inheritance	6
2.4.4	Multiple Inheritance	7
2.6	Abstraction	9
2.7	Packages	10
4.2	Flowchart	16
6.1	Displays Menu	30
6.2	Movie selection	31
6.3	Seat type and seat number	32
6.4	Ticket details	33
6.5	Movie running currently	34
6.6	Eateries and snacks	35
6.7	Exit program	36

# CHAPTER 1

## INTRODUCTION

### 1.1 PROBLEM DEFINITION

Booking tickets for movies is lot of time consuming, stand in queue wait of hours to get their tickets and to get the seats they wish for makes difficult for the customers to book movie tickets. It is also a difficult job for staffs to organize and keep an account and record for tickets they have issued. As manually it will take a lot of time to gather data and organize seats resulting in rough customer satisfaction.

### 1.2 OBJECTIVES

Main objective of this project is to provide easy and convenient way of booking tickets for customers. To minimize the work load on staffs, to allot and organize.

### 1.3 METHODOLOGY TO BE FOLLOWED

The main methodology used here are implementation of object-oriented programming in Java. The main aim here is to make the process of movie booking faster and hassle free, providing customers more movies and a variety of time slots which they can choose from. It provides simple solution for booking tickets in a faster manner and to choose the choose the seat we wish for.

### 1.4 EXPECTED OUTCOMES

The outcome expected in this project is to let the user view all the movies currently running in the theatre and be able to choose a movie. The customer can further choose a timing and type of seats they need. They will get an individual ticket for each seat they book with all the details including movie/show name, time, seat number and cost of the ticket.

- Book movie tickets
- Shows available seats
- Show transactions
- View all available movies/shows
- Snacks

## CHAPTER 2

# FUNDAMENTALS OF JAVA PROGRAMMING

### 2.1 INTRODUCTION

Object-oriented programming (OOP) is at the core of Java. In fact, all Java programs are to at least some extent object-oriented. Object-Oriented Programming is a methodology or paradigm to design a program using classes and objects. It simplifies software development and maintenance.

### 2.2 CLASS

A Class is a user defined “prototype” or “blueprint” for creating objects. Class declarations can include class keyword, modifiers, class name and body. While defining a class, we declare its exact form and nature, by specifying the data that it contains and the code that operates on that data. A class is declared by use of the class keyword. The general form of a class definition is shown below:

```
Class classname {  
  
    type instance-variable1;  
  
    type instance-variable2;  
  
    //...  
    type instance-variableN;  
    type methodname1 (parameter-list) {  
        // body of method  
    }  
  
    type methodname2(parameter-list) {  
  
        // body of method  
    }  
}
```

```

type methodNameN(parameter-list) {
// body of method
}
}

```

The data or variables, defined within a class are called instance variables. The code is contained within methods. Collectively, the methods and variables defined within a class are called members of the class. In most classes, the instance variables are acted upon and accessed by the methods defined for that class. Thus, the methods determine how a class data can be used.

## 2.3 OBJECT

Object is an instance of class. Objects have both state and behavior. If a dog is an object, then it's states can be idle, running etc. it's behaviors will be barking, sniffing etc.

Obtaining objects of a class is a two-step process: First, must declare a variable of the class type. Second, must acquire an actual, physical copy of object and assign it to that variable, using the new operator. The new operator dynamically (at run time) allocates memory for an object and returns a reference is the address in memory of the object allocated by new.

Syntax: `Box mybox= new Box();`

`Class-var =new classname();`

Here, class-var is a variable of the class type being created. The class name is the name of the class that is being instantiated. The class name followed by parentheses specifies the constructor for the class. It can be rewritten to show each step more clearly which includes, reference to object and allocates a Box object

`Box mybox;`

`mybox = new box();`

## 2.4 INHERITANCE

Inheritance is the process by which one object acquires the properties of another object. This is important because it supports the concept of hierarchical classification. Most knowledge is made manageable by hierarchical (that is, top-down) classifications. The class that inherits from another class is called subclass or child class. The class being inherited from is called super class. To inherit from a particular class, we have to use a keyword known as 'extends'.

Syntax: class subclass-name extends superclass-name

```
{
// methods and fields
}
```

Example program:

Class Animal

Void eat()

```
{
System.out.println("eating...");
}
```

Class Dog extends Animal {

Void bark() {

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

Class Inheritance

public class void main (String args[])

```
{
Dog d = new Dog();
d.bark();
d.eat();
}}
```

Output: barking...  
eating...

Types of Inheritance:

1. Single inheritance
2. Multi-level inheritance
3. Hierarchical inheritance
4. Multiple inheritance

1. Single inheritance: It enables a derived class to inherit properties and behavior from a single parent class. It allows a derived class to inherit the properties and behavior of a base class, thus enabling code reusability as well as adding new features to the existing code.

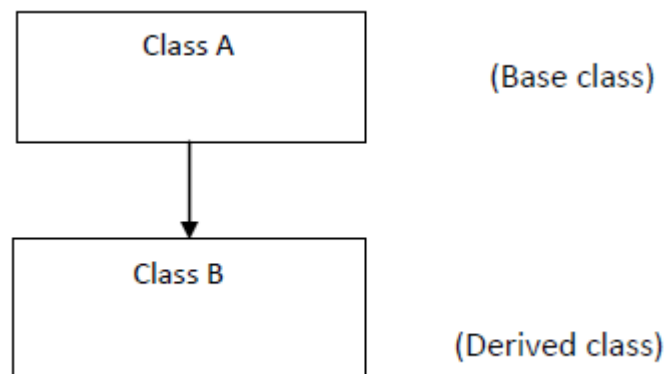


Fig. 2.4.1

2. Multi-level inheritance: It refers to a mechanism in object-oriented technology where one can inherit from derived class, thereby making this derived class, the base class for the new class.

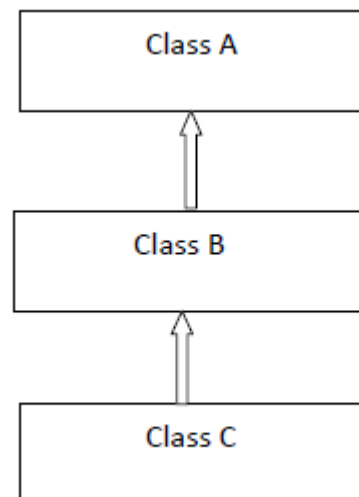


Fig. 2.4.2

3. Hierarchical inheritance: The multiple child classes inherit the single class or the single class inherited by multiple child class i.e., one parent class will be inherited by many sub classes.

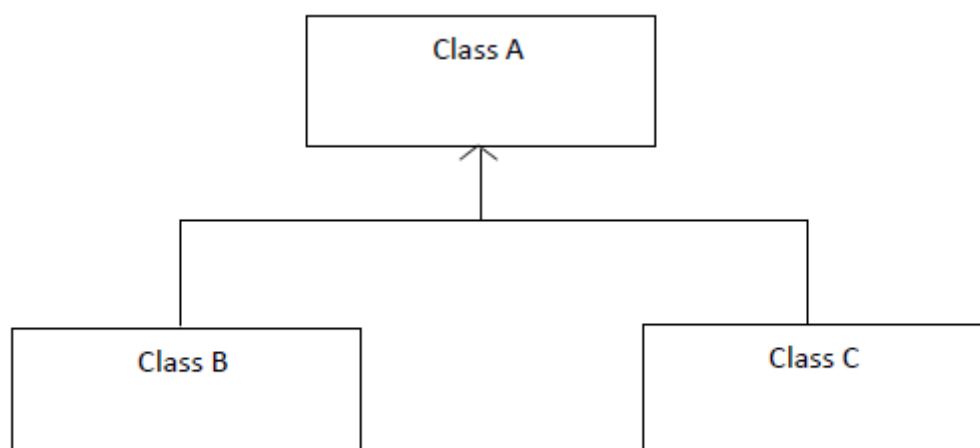


Fig. 2.4.3



4. Multiple inheritance: It is a feature of object-oriented concept, where a class can inherit properties of more than one parent class.

A class can implement any number of interfaces but can extend only one class.

Multiple inheritance is not supported because it leads to deadly diamond problem. It can be solved but it leads to complex system so multiple inheritance is been dropped.

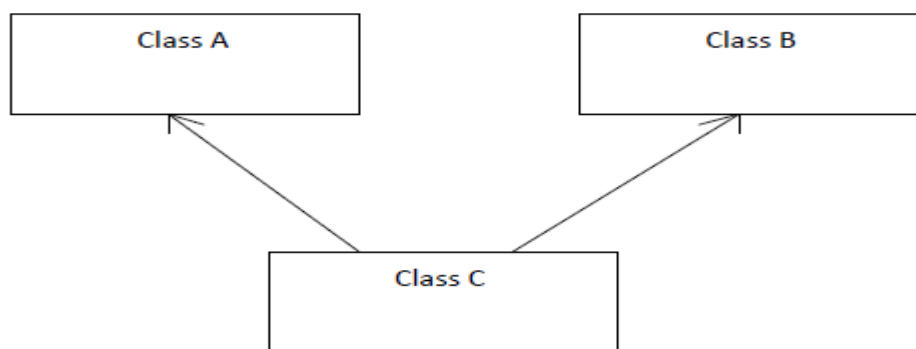


Fig. 2.4.4

## 2.5 POLYMORPHISM

Polymorphism (from Greek, meaning “many forms”) is a feature that allows one interface to be used for a general class of actions. The specific action is determined by the exact nature of the situation. There are two types of polymorphism in java: compile-time polymorphism and runtime polymorphism. We can perform in java by method overloading and method overriding.

1) Compile time polymorphism: It is a method dispatch is a process in which a call to an overloading method is resolved at compile time rather than at runtime. In this process, we done overloading of methods are called through the reference variables of a class here no need to have superclass.

**Method Overloading:** If a class have multiple methods by same name but different parameters, it is known as Method Overloading.

**Operator overloading:** Java also provides option to overload operators. For example, we can make the operator ('+') for string class to concatenate two strings.

**2) Run time polymorphism:** It is also known as dynamic method dispatch. It is a process in which a function call to the overridden method is resolved at runtime rather than compile-time. In this process, overridden method is called through the reference variable of a superclass.

**Method overriding:** if the subclass has the same method as declared in the parent class, it is known as method overriding in java. In other words, if a subclass provides the specific implementation of the method that has been declared by one of its parent class known as Method overriding.

## 2.6 ABSTRACTION

Abstraction in Java is a process in which we hide certain details or information and show only essential information to the user. An essential element of object-oriented programming is abstraction. Humans manage complexity through abstraction. Abstract class is a restricted class that cannot be used to create objects and to access it, it must be inherited from another class.

Ways to achieve abstraction:

- Abstract class
- Interface

Abstract class syntax:

```
abstract class classname{
```

Abstract method syntax:

```
abstract typename (parameter-list);
```

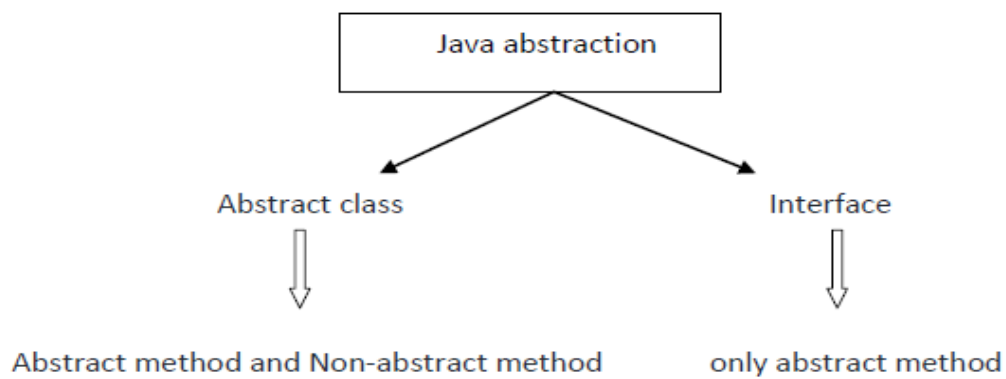


Fig. 2.6

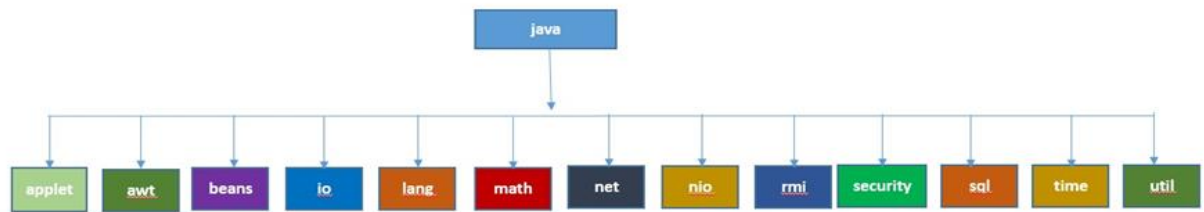
## 2.7 JAVA PACKAGES

A java package is a group of similar types of classes, interfaces and sub-packages. A package in java can be created by a user. We can create packages based on related classes. It acts a folder or directory. There are built-in packages and user defined packages.

The main advantages of this Java Package is used to categorize the classes as they can be maintained easily. They provide better protection and removes problems caused due to naming.

Packages in java can be categorized into two form:

1. Built-in packages
2. User-defined packages



Java Most Common Use Built-in Packages

Fig. 2.7

## 2.8 EXCEPTION HANDLING

An exception is an abnormal or unexpected event, which occurs during the execution of a program i.e. at run time, that disrupts the normal flow of the program's instructions. Exception indicates conditions that a reasonable application might try to catch.

Java exception is an object that describes an exceptional/abnormal condition that has occurred in a piece of code. When an exceptional condition arises, an object representing that exception is created and thrown in the method that caused the same. That method may choose to handle the exception itself, or pass it on. Either way, at some point, the exception is caught and processed. Exceptions can be generated by the Java run-time system, related to fundamental errors that violate the rules of the Java language or the constraints of the Java execution environment or they can be manually generated by your code. Java exception handling is managed via five keywords: try, catch, throw, throws and finally. In java language, an exception-handling block general form is:

```

try {

// block of code to monitor for errors

} catch {ExceptionType exob} {

// exception handler for ExceptionType 2

```

```
}  
  
//....  
  
finally {  
  
    //block of code to be executed after try catch block ends  
  
}
```

Using try and catch:

To handle a run-time error, simply enclose the code to monitor inside a try block. Immediately following the try block, includes a catch clause that specifies the exception type catch. To illustrate how easily this can be done, the following program includes a try block and a catch clause that processes the `ArithmeticException` generated by the division-by-zero error.

Throw:

It is possible for a program to throw an exception explicitly, using the throw statement. The general form of throw is shown here:

```
throw ThrowableInstance;
```

Here, `ThrowableInstance` must be an object of type `Throwable` or a subclass of `Throwable`. There are two ways to obtain a `Throwable` object:

1. Using a parameter in a catch clause
2. Creating one with the new operator.

Throws:

If a method is capable of causing an exception that it does not handle, it must specify this behavior so that callers of the method can guard themselves against that exception. To

do this by including a throws clause in the method's declaration. A throws clause lists the types of exceptions that a method might throw.

Finally:

Finally creates a block of code that will be executed after a try /catch block has completed and before the code following the try/catch block. The finally block will execute whether or not an exception is thrown. If an exception is thrown, the finally block will execute even if no catch statement matches the exception.

## 2.9 THREADS

A thread, in the context of Java, is the path followed when executing a program. It is a sequence of nested executed statements or method calls that allow multiple activities within a single process. All Java programs have at least one thread, known as the main thread, which is created by the Java Virtual Machine (JVM) at the program's start, when the main method is invoked with the main thread. Thread class provide constructors and methods to create and perform operations on a thread. Thread class extends Object class and implements interface Runnable.

Multithreading is a process of executing two or three more threads simultaneously to maximum utilization of CPU. Multithreaded applications execute two or more threads run concurrently.

## 2.10 I/O BASICS

Java I/O (Input and Output) is used to process the input and produce the output. The java.io package contains all the classes required for input and output operations. File handling in Java can be performed by using Java I/O API. I/O functions in java are used to process the input and give output. It uses the concept of streams to make I/O operations fast. We can also perform file handling in java using these streams. Examples are System.out, System.in, System.err, OutputStream ,InputStream.

## CHAPTER 3

### REQUIREMENT SPECIFICATION

#### 3.1 HARDWARE AND SOFTWARE REQUIREMENTS

Processor	: Intel i3 or above / AMD Ryzen 3 or above
RAM	: 4GB
Hard Disk	: 10 GB
Input device	: Keyboard/Mouse
Output device	: Monitor
Operating system	: Windows
Compiler	: IntelliJ or any java compiler

## CHAPTER 4

### DESIGN

#### 4.1 DESIGN GOALS

The project has been designed in an easy user accessible. It will show the movies available and show seats available and makes job for consumers to reserve there tickets easily. It also gives us a detailed ticket which gives us all the necessary information regarding the movie, time, seat number and cost of the ticket. Now customers can even see what snacks and brands are available before heading to the canteen to buy them.

#### 4.2 ALGORITHM / PSEUDOCODE

STEP 1: START

STEP 2: Displays options

- 1.Book Movie Tickets
- 2.See Movies/Show Timings
- 3.Snacks
- 4.Exit

STEP 3: Option 1: Displays all the list of movies for booking.

STEP 4: Asks for the movie number.

STEP 5: Takes the time of the movie.



STEP 6: Takes the number of seats.

STEP 7: Asks for the preferred row number, seat type and seat number.

STEP 8: Displays ticket details including the movie name, timings, seat number and cost.

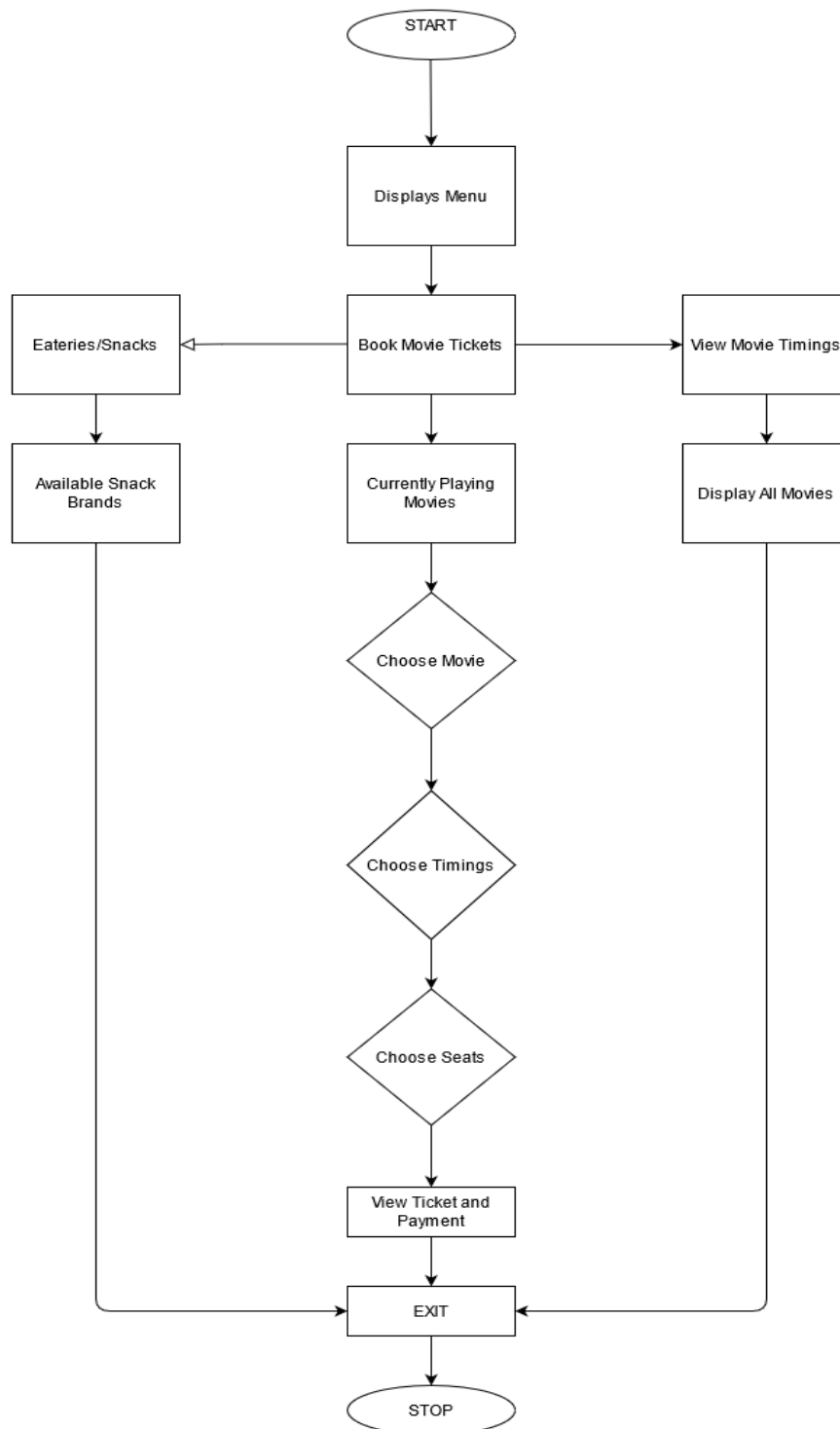
STEP 9: Option 2: See Movies/Show Timings – It will list all the movies and schedules.

STEP 10: Option 3: Displays all the snacks types and brands available to the consumer.

STEP 11: Option 4: Exit – Exits the program.

**FLOWCHART**

Fig.4.2



## CHAPTER 5

### IMPLEMENTATION

#### 5.1 MODULE 1 FUNCTIONALITY

In this module, we create an abstract class for snacks with few eateries as options. We create another class snacks\_movies and inherit the properties of snacks and create methods with brand names available in canteen.

```
package TICKET;

import java.util.*;

abstract class snacks
{
    String chips;

    String drinks;

    String popcorn;

    public void break_time()
    {
        System.out.println("Don't forget to get some snacks for the movies at our canteen!");
    }
}

class snacks_movies extends snacks
{
    public void chips_brands()
    {
```

```
        System.out.println("AVAILABLE CHIPS BRANDS:  Lays, Bingo, Yumitos, Kurkure");
    }

    public void drinks_brands()
    {
        System.out.println("AVAILABLE DRINKS BRANDS:  Coke, Pepsi, 7UP, Maaza");
    }

    public void popcorn_brands()
    {
        System.out.println("AVAILABLE POPCORN BRANDS:  Act II, Black Jewell, Jolly Time,
Pop Secret");
    }

    public static void main(String args[])
    {
        snacks_movies a=new snacks_movies();

        a.chips_brands();

        a.popcorn_brands();

        a.drinks_brands();

        a.break_time(); } }
```

## 5.2 MODULE 2 FUNCTIONALITY

In this module we create a class TI, we create method print\_ticket for showing all the details like movie name, showtimes, seat numbers, date and cost. Method display\_show\_timings has all the show/movies available to watch and their respective timings. Method check\_time uses a switch case to check the timings entered by user matches the time slot available for that particular movie. Methods show\_empty\_seats and check\_seats shows the number of seats available and number of seats available after booking respectively.

```
class TI {

    public static class ticket {

        screen s = new screen();

        void print_ticket(String movie,String showtime,int screen,String cat, String rowch, int
        colch)

        {

            Date date = new Date();

            int cost = s.getcost(cat);

            System.out.println("#####REV Cinemas#####");

            System.out.println("\t\t    MOVIE BOOKING");

            System.out.println("\tMovie: "+movie+"\t");

            System.out.print("\tTime: "+showtime);

            System.out.println("\t\t\tSeat: "+rowch+colch);

            System.out.print("\tCategory: "+cat);

            System.out.println("\t\t\tScreen: "+screen);

            System.out.println("\tDate: "+date);
```

```

        System.out.println("\tPrice: ₹"+cost);

        System.out.println("\tThank You! Hope to see you again soon!\n\n");
    }
}

public static class show {

    static String seats[][][] = new String[5][6][12];

    void display_show_timings()
    {
        System.out.println("\tMovies Now Playing          \t\t\t\t\t Timings");

        System.out.print("1.   Zack Snyder`s Justice League – \t\t\t\t 10:20   11:30   17:30
");

        System.out.println();

        System.out.print("2.   Godzilla vs. Kong –          \t\t\t\t 10:45   12:30   16:30 ");

        System.out.println();

        System.out.print("3.   Parasite –              \t\t\t\t 10:00   11:45   15:30 ");

        System.out.println();

        System.out.print("4.   Joker –                \t\t\t\t 12:00   13:30   17:30 ");

        System.out.println();

        System.out.print("5.   Knives Out –           \t\t\t\t 09:20   11:30   17:30 ");

        System.out.println();
    }
}

```

```
String check_time(int x,String t)
{
    switch(x)
    {
        case 1: if(t.equals("10:20") || t.equals("11:30") || t.equals("17:30")){
            return "true";
        }
        else
        { System.out.println("Time not available");
            return "false";
        }
        case 2: if(t.equals("10:45") || t.equals("12:30") || t.equals("16:30")){return
"true";}
        else{
            System.out.println("Time not available");
            return "false";
        }

        case 3: if(t.equals("10:00") || t.equals("11:45") || t.equals("15:30")){return
"true";}
        else
        {    System.out.println("Time not available");
            return "false";
        }
    }
}
```

```

        case 4: if(t.equals("12:00") || t.equals("13:30") || t.equals("17:30")){return
"true";}

        else

        {

            System.out.println("Time not available");

            return "false";

        }

        case 5: if(t.equals("9:20") || t.equals("11:30") || t.equals("17:30")){return
"true";}

        else

        {      System.out.println("Time not available");

            return "false";

        }

        default: return "false";

    }

}

void show_empty_seats(String x,int i)

{

    for(int j=0;j<12;j++)

        if(seats[i-1][x.charAt(0)-'A'][j]!="Full")

        {

```



```
        System.out.print(" "+(j+1));

    }

}

boolean check_seats(String a,int x,int i)

{

    if(seats[i-1][a.charAt(0)-'A'][x-1]!="Full")

        return false;

    else

        return true;

}

void fill_seat(String a,int b,int i)

{

    seats[i-1][a.charAt(0)-'A'][b-1]="Full";

}

String getmovie(int x)

{

    switch(x)

    {

        case 1: return "Zack Snyder`s Justice League";

        case 2: return "Godzilla vs. Kong";

        case 3: return "Parasite";
```

```
        case 4: return "Joker";

        case 5: return "Knives Out";

        default: return "";

    }

}

public static void main()

{

    for(int k =0; k<5;k++)

        for(int i=0;i<6;i++)

            for(int j=0;j<12;j++)

                seats[k][i][j]="Empty";

    }

}

}
```

## 5.3 MODULE 3 FUNCTIONALITY

In this module, we display the seats available for customers and let them choose between bronze, silver and platinum seats. We use method `getcat` to get the row number user wants to choose to be seated and we use `getcost` for the prices for bronze, silver, platinum being Rs.150, Rs.225, Rs.300 respectively.

```
class screen {  
  
    void display_seats()  
  
    {  
  
        System.out.println("                Screen this side");  
  
        System.out.println("\t\t ┌──────────────────────────┐");  
        System.out.println("\t\t └──────────────────────────┘");  
  
        System.out.println("\t ───────────────────────────────────┐");  
  
        System.out.println("\t A □□□□□□□□□□ Bronze – ₹ 150");  
        System.out.println("\t B □□□□□□□□□□ Bronze – ₹ 150");  
  
        System.out.println("\t ───────────────────────────────────┐");  
  
        System.out.println("\t C □□□□□□□□□□ Silver – ₹ 225");  
        System.out.println("\t D □□□□□□□□□□ Silver – ₹ 225");  
  
        System.out.println("\t ───────────────────────────────────┐");  
  
        System.out.println("\t E □□□□□□□□□□ Platinum – ₹ 300");  
        System.out.println("\t F □□□□□□□□□□ Platinum – ₹ 300");  
  
        System.out.println("\n\n Please Enter Preferred Row. Seat will be booked as per  
availability.\n");  
  
    }String getcat(String a)  
  
    {
```

```

String cat;

if(a.compareToIgnoreCase("A")==0 | a.compareToIgnoreCase("B")==0)

    cat="Bronze";

else if(a.compareToIgnoreCase("C")==0 | a.compareToIgnoreCase("D")==0)

    cat="Silver";

else if(a.compareToIgnoreCase("E")==0 | a.compareToIgnoreCase("F")==0)

    cat="Platinum";

else

    cat="N/A";

return cat;
}

int getcost(String cat)

{ int cost = 0;

    if(cat.compareToIgnoreCase("Bronze")==0)

        cost=150;

    else if(cat.compareToIgnoreCase("Silver")==0)

        cost=225;

    else if(cat.compareToIgnoreCase("Platinum")==0)

        cost=300;

    else

        return 0; return cost;

}
}

```

## 5.4 MODULE 4 FUNCTIONALITY

In this module, we have our main function. Displays all the menu options to the user to choose from like Book movie tickets, See available movies, Snacks and Exit option.

```
class run {

    public static void main(String args[]){

        Scanner in = new Scanner(System.in);

        int ch,i,n,movch; String rowch[]=new String[12];

        int colch[]=new int[12];

        String time = new String();

        String timecheck = new String();

        Tl.show objshow = new Tl.show();

        screen objscreen = new screen();

        Tl.ticket objticket = new Tl.ticket();


        snacks_movies d=new snacks_movies();

        while(true){

            System.out.println("\n\t\t Movie Ticket Booking System\n");


            System.out.println("1.Book Movie Tickets");


            System.out.println("2.See Movies/Show Timings");


            System.out.println("3.Snacks");
```

```

System.out.println("4.Exit");

System.out.println("\nAdvanced Bookings are closed as of now.");

System.out.println("\nEnter Choice");

ch = in.nextInt();

switch(ch)

{

    case 1:      objshow.display_show_timings();

                System.out.println("Choose the Movie");

                movch=in.nextInt();

                do{

                    System.out.println("Enter Show timings (hh:mm)");

                    time=in.next();

                    timecheck=objshow.check_time(movch,time);

                }while(timecheck.equals("false"));

                System.out.println("Enter Number of seats");

                n=in.nextInt();

                for(i=0;i<n;i++){

                    objscreen.display_seats();

                    rowch[i]=in.next().toUpperCase();

                    objshow.show_empty_seats(rowch[i],movch);

                    colch[i]=in.nextInt();

                    if(objshow.check_seats(rowch[i],colch[i],movch))

                    {

```

```

        System.out.println("Seat is not available");
    }

    objshow.fill_seat(rowch[i],colch[i],movch);
}

for(i=0;i<n;i++)
{
    objticket.print_ticket(objshow.getmovie(movch), time, 3,
objscreen.getcat(rowch[i]), rowch[i], colch[i]);
}

break;

case 2: objshow.display_show_timings();

break;

case 3:n=1;

for(i=0;i<n;i++) {

    d.break_time();

    d.popcorn_brands();

    d.drinks_brands();

    d.chips_brands();

}

case 4: System.exit(0);

}

}

}

}

```

## CHAPTER 6

### RESULTS

#### 6.1 DISPLAYS THE MENU

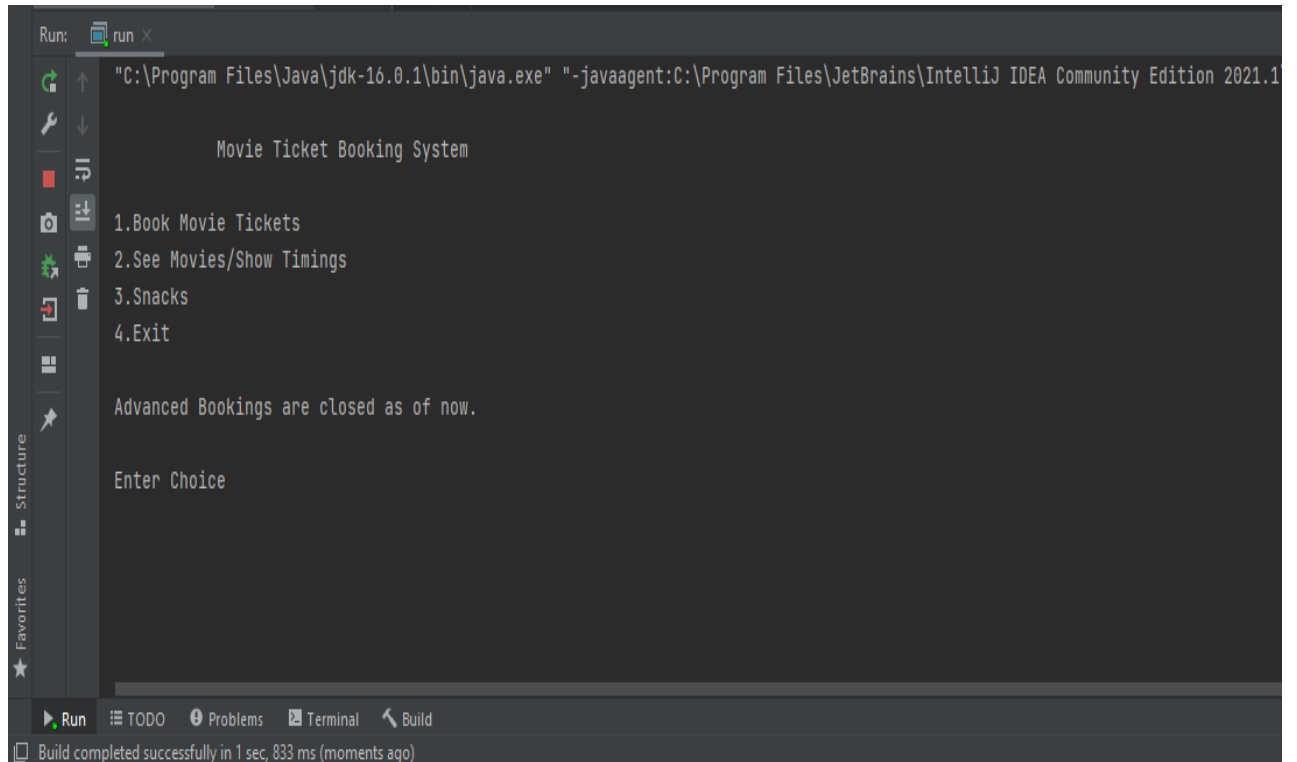
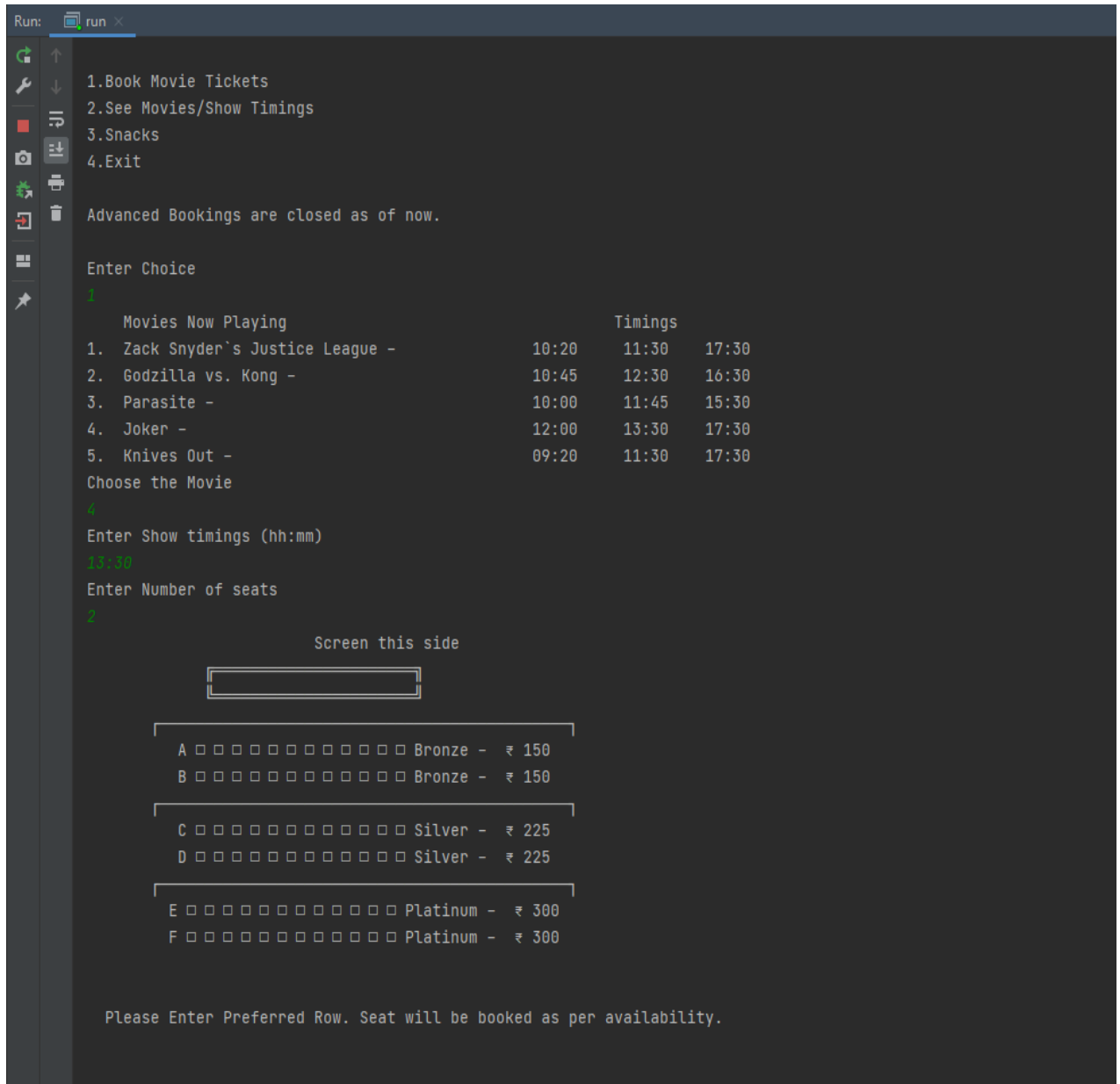


Fig 6.1



## 6.2 Let's user choose the movie, select timings and number of seats.



```

Run: run
1.Book Movie Tickets
2.See Movies/Show Timings
3.Snacks
4.Exit

Advanced Bookings are closed as of now.

Enter Choice
1

Movies Now Playing
1. Zack Snyder's Justice League - 10:20 11:30 17:30
2. Godzilla vs. Kong - 10:45 12:30 16:30
3. Parasite - 10:00 11:45 15:30
4. Joker - 12:00 13:30 17:30
5. Knives Out - 09:20 11:30 17:30

Choose the Movie
4

Enter Show timings (hh:mm)
13:30

Enter Number of seats
2

Screen this side

A □ □ □ □ □ □ □ □ □ □ Bronze - ₹ 150
B □ □ □ □ □ □ □ □ □ □ Bronze - ₹ 150

C □ □ □ □ □ □ □ □ □ □ Silver - ₹ 225
D □ □ □ □ □ □ □ □ □ □ Silver - ₹ 225

E □ □ □ □ □ □ □ □ □ □ Platinum - ₹ 300
F □ □ □ □ □ □ □ □ □ □ Platinum - ₹ 300

Please Enter Preferred Row. Seat will be booked as per availability.
  
```

Fig 6.2

### 6.3 Let's user choose the seat numbers according to the seat type (Bronze, Silver, Platinum)



Fig 6.3

## 6.4 Shows the tickets with all the details like movie name, time, seat number, date, time and the price for your ticket.

```

Run: run x
Please Enter Preferred Row. Seat will be booked as per availability.
1 2 3 4 5 6 7 9 10 11 12
9
#####REV Cinemas#####
      MOVIE BOOKING
Movie: Joker
Time: 13:30      Seat: E8
Category: Platinum      Screen: 3
Date: Thu Jul 01 20:28:04 IST 2021
Price: ₹300
Thank You! Hope to see you again soon!

#####REV Cinemas#####
      MOVIE BOOKING
Movie: Joker
Time: 13:30      Seat: E9
Category: Platinum      Screen: 3
Date: Thu Jul 01 20:28:04 IST 2021
Price: ₹300
Thank You! Hope to see you again soon!

      Movie Ticket Booking System

1.Book Movie Tickets
2.See Movies/Show Timings
3.Snacks
4.Exit

Advanced Bookings are closed as of now.

Enter Choice

```

Fig 6.4

## 6.5 CHOICE 2:

**Shows all the movies running in the theatre currently.**

```

run x
MOVIE BOOKING
Movie: Joker
Time: 13:30      Seat: E9
Category: Platinum      Screen: 3
Date: Thu Jul 01 20:28:04 IST 2021
Price: ₹300
Thank You! Hope to see you again soon!

Movie Ticket Booking System

1.Book Movie Tickets
2.See Movies/Show Timings
3.Snacks
4.Exit

Advanced Bookings are closed as of now.

Enter Choice
2

Movies Now Playing      Timings
1. Zack Snyder's Justice League -      10:20      11:30      17:30
2. Godzilla vs. Kong -      10:45      12:30      16:30
3. Parasite -      10:00      11:45      15:30
4. Joker -      12:00      13:30      17:30
5. Knives Out -      09:20      11:30      17:30

Movie Ticket Booking System

1.Book Movie Tickets
2.See Movies/Show Timings
3.Snacks
4.Exit

Advanced Bookings are closed as of now.

Enter Choice

```

Fig. 6.5

## 6.6 CHOICE 3:

**Shows the available brands of snacks in the canteen (Popcorns, Soft drinks and chips).**

```
Movie Ticket Booking System

1.Book Movie Tickets
2.See Movies/Show Timings
3.Snacks
4.Exit

Advanced Bookings are closed as of now.

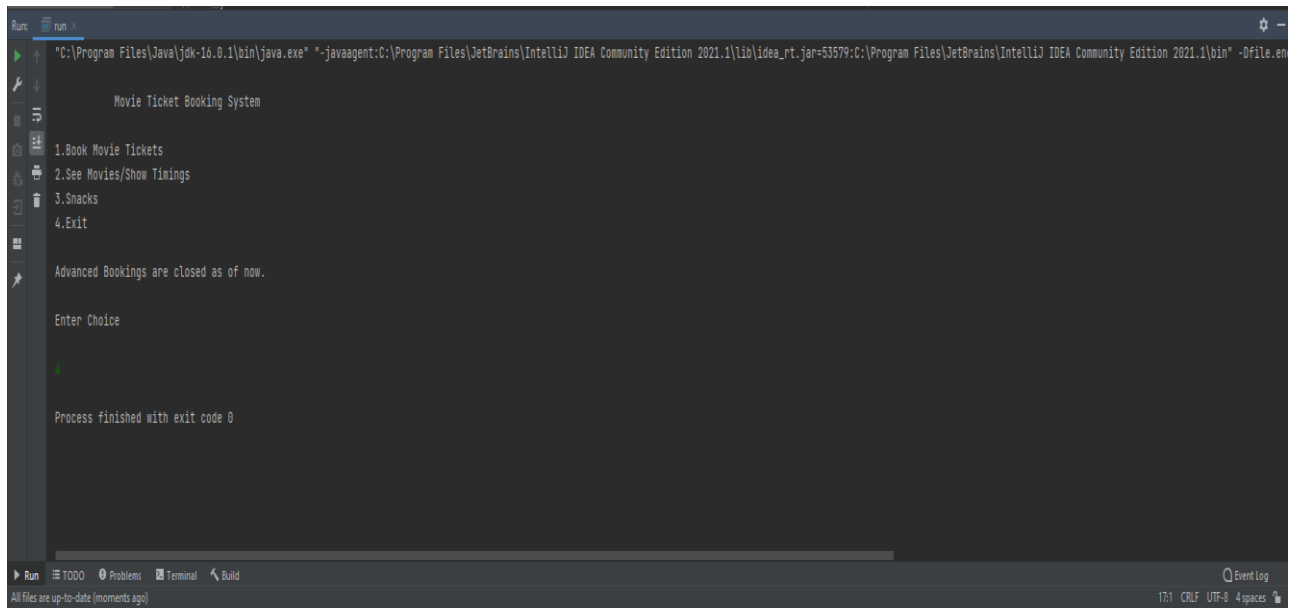
Enter Choice
3
Don't forget to get some snacks for the movies at our canteen!
AVAILABLE POPCORN BRANDS: Act II, Black Jewell, Jolly Time, Pop Secret
AVAILABLE DRINKS BRANDS: Coke, Pepsi, 7UP, Maaza
AVAILABLE CHIPS BRANDS: Lays, Bingo, Yumitos, Kurkure

Process finished with exit code 0
|
```

Fig 6.6

## 6.7 CHOICE 4:

**Exits from the ticket booking system.**



```
"C:\Program Files\Java\jdk-16.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1\lib\idea_rt.jar=53579:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2021.1\bin" -Dfile.encoding=UTF-8

Movie Ticket Booking System

1.Book Movie Tickets
2.See Movies/Show Timings
3.Snacks
4.Exit

Advanced Bookings are closed as of now.

Enter Choice

4

Process finished with exit code 0
```

Fig 6.7

## CHAPTER 7

### CONCLUSION

The main intention of this project was to replace the manual, time consuming, slow process with computerized, time saving and a faster process. A movie booking platform facilitates selection, reservation and purchase of tickets for the movies. Waiting in queue for hours in long queues to get tickets booked is definitely something which we want to avoid and overcome. The project provides complete information regarding currently running movies on all the screens with details of show timings and available seats. This project makes an entire process hassle free and less time consuming than the traditional methods of movie booking.

## REFERENCES

- [1] <https://www.programiz.com/>
- [2] <https://www.geeksforgeeks.org/>
- [3] <https://www.javatpoint.com/>
- [4] <https://www.tutorialspoint.com/>
- [5] <https://www.tutorialsteacher.com/>