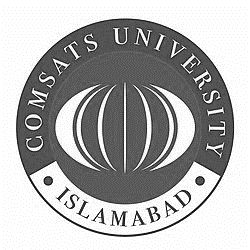
**Lab 5 — Cinema Management System**

By

**ARHAAM MUSTAFA (SP25-BCS-016)**

****

**Submitted to:** Muhammad Shahid Bhatti

**Subject:** OBJECT ORIENTED PROGRAMMING

**Date:** 01/10/2025

**DEPARTMENT OF COMPUTER SCIENCE**

**COMMISSION ON SCIENCE AND TECHNOLOGY**

**FOR SUSTAINABLE DEVELOPMENT IN THE SOUTH**

**LAHORE, PAKISTAN**

# 1. Objective

To design and implement a Cinema Management System in Java using object-oriented programming concepts such as classes, objects, arrays, enumerations, and encapsulation.

# 2. Source Code Overview

## Class: Seat

Handles individual seat details (type, price, ID, availability) and supports booking and cancellation.

public class Seat {

String seatType;

double price;

String id;

boolean isAvailable;

Seat(){}

Seat(String id, String type, double price, boolean isAvailable){

this.id=id;

this.seatType=type;

this.price=price;

this.isAvailable=isAvailable;

}

public boolean bookseat(){

this.isAvailable=false;

return true;

}

public boolean cancelbooking(){

isAvailable=true;

return this.isAvailable;

}

public String toString(){

return String.format("[%s:%c]",id,getAvaialability());

}

}

## Class: Screen

Represents a screen inside a cinema and maintains a 2D array of Seat objects.

public void DisplayLayout(){

System.out.println("=== Screen-"+name+" | Layout ===");

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

for(int j=0;j<seats[i].length;j++){

System.out.print(seats[i][j].toString()+" ");

}

System.out.println();

}

System.out.println("Total: "+gettotalseatcount()+", Available: "+getavailableseatcount());

}

## Class: Cinema

Contains multiple screens and can display their layouts.

public void DisplayCinemaLayout(){

System.out.println("=== CINEMA: "+cinema\_name+" | Layouts ===");

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

screens[i].DisplayLayout();

}

}

## Class: CityCinema

Represents a city containing multiple cinemas.

public void DisplayCityLayout(){

System.out.println("=== CITY: "+city\_name+" | All Cinema Layouts ===");

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

cinemas[i].DisplayCinemaLayout();

}

}

## Enum: SeatType

Defines seat categories and their prices.

public enum SeatType{

Premium("Premium",1500),VIP("VIP",1000),Regular("Regular",500);

}

## Class: ScreenDemo

Main driver class that runs the system.

public class ScreenDemo{

public static void main(String[] args){

CityCinema c1=new CityCinema("Lahore",2);

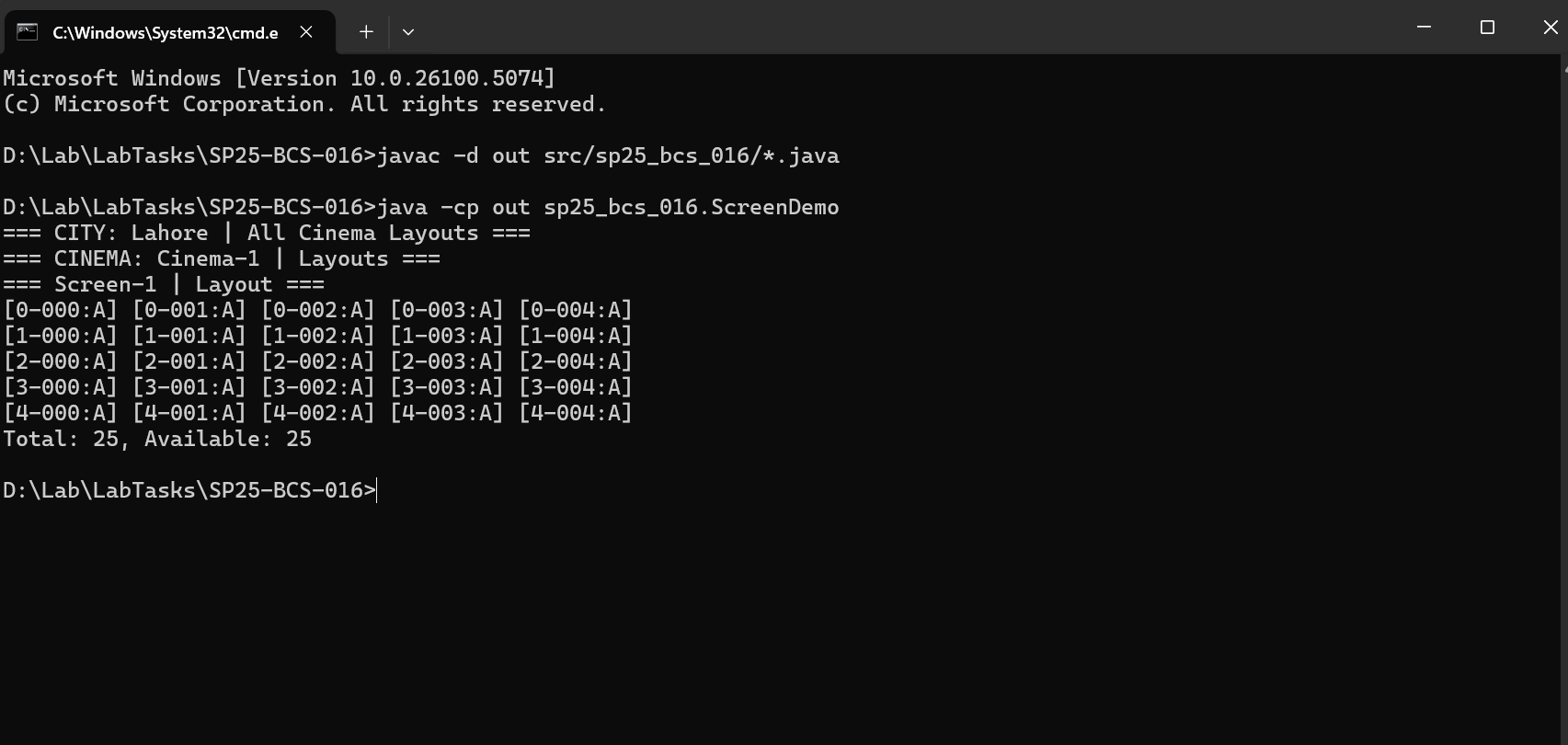
c1.cinemas[0].screens[0].book("0-001");

c1.cinemas[0].screens[0].DisplayLayout();

}

}

# 3. Sample Output



# 4. Concepts Used

Classes and Objects: Modular structure with Seat, Screen, Cinema, and CityCinema.

Arrays: 2D seat grid for each screen.

Enumerations (Enums): Used for SeatType for better readability and control.

Encapsulation: Access to attributes via getters/setters.

Method Overloading: Multiple versions of booking/canceling by seat ID or index.

# 5. Conclusion

This lab demonstrates the use of object-oriented design in building a modular Java application. The Cinema Management System efficiently handles seat management, booking operations, and layout visualization.