**TICKET BOOKING SYSTEM**

**-Vaishnavi Surikutchi**

**TASK 1: Conditional Statements**

**In a BookingSystem, you have been given the task is to create a program to book tickets. if available tickets more than noOfTicket to book then display the remaining tickets or ticket unavailable:**

**Tasks:**

**1. Write a program that takes the availableTicket and noOfBookingTicket as input.**

**2. Use conditional statements (if-else) to determine if the ticket is available or not.**

**3. Display an appropriate message based on ticket availability.**

**SOURCE CODE :**

**package** com.hexaware.entity;

**import** java.util.Scanner;

**public** **class** Task1 {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

System.***out***.print("Enter the available tickets: ");

**int** availableTickets = scanner.nextInt();

System.***out***.print("Enter the number of tickets to book: ");

**int** numberOfTicketsToBook = scanner.nextInt();

**if** (numberOfTicketsToBook > 0 && numberOfTicketsToBook <= availableTickets) {

availableTickets -= numberOfTicketsToBook; // Update available tickets after booking

System.***out***.println("Tickets booked successfully! Remaining tickets: " + availableTickets);

} **else** {

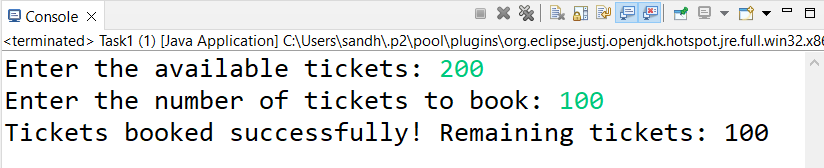
System.***out***.println("Sorry, tickets unavailable or invalid input.");

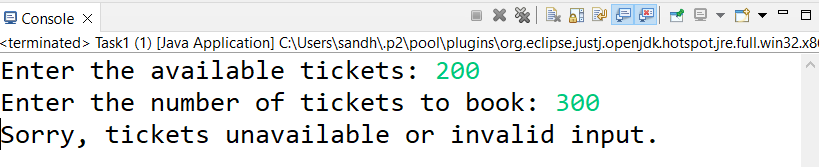
}

}

}

**OUTPUT :**





**TASK 2: Nested Conditional Statements**

**Create a program that simulates a Ticket booking and calculating cost of tickets. Display tickets options such as "Silver", "Gold", "Dimond". Based on ticket category fix the base ticket price and get the user input for ticket type and no of tickets need and calculate the total cost of tickets booked.**

**TASK 3: Looping**

**From the above task book the tickets for repeatedly until user type "Exit"**

**SOURCE CODE TASK 2 AND 3 :**

**package** com.hexaware.entity;

**import** java.util.Scanner;

**public** **class** Task2and3 {

**public** **static** **void** main(String[] args) {

Scanner scanner = **new** Scanner(System.***in***);

**boolean** exitRequested = **false**;

**do** {

System.***out***.println("Ticket Options:");

System.***out***.println("1. Silver");

System.***out***.println("2. Gold");

System.***out***.println("3. Diamond");

System.***out***.print("Enter the ticket type number (or type 'Exit' to quit): ");

**if** (scanner.hasNextInt()) {

**int** ticketTypeNumber = scanner.nextInt();

**if** (ticketTypeNumber >= 1 && ticketTypeNumber <= 3) {

**int** baseTicketPrice;

**switch** (ticketTypeNumber) {

**case** 1:

baseTicketPrice = 50;

**break**;

**case** 2:

baseTicketPrice = 100;

**break**;

**case** 3:

baseTicketPrice = 200;

**break**;

**default**:

System.***out***.println("Invalid ticket type number. Please enter a valid option.");

**continue**;

}

System.***out***.println("Selected Ticket Type: " + *getTicketTypeName*(ticketTypeNumber));

System.***out***.println("Base Ticket Price: $" + baseTicketPrice);

System.***out***.print("Enter the number of tickets to book: ");

**int** numberOfTickets = scanner.nextInt();

**int** totalCost = baseTicketPrice \* numberOfTickets;

System.***out***.println("Tickets booked successfully!");

System.***out***.println("Number of Tickets: " + numberOfTickets);

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

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

} **else** **if** (ticketTypeNumber == 0) {

exitRequested = **true**;

} **else** {

System.***out***.println("Invalid ticket type number. Please enter a valid option.");

}

} **else** {

String userInput = scanner.next();

**if** (userInput.equalsIgnoreCase("exit")) {

exitRequested = **true**;

} **else** {

System.***out***.println("Invalid input. Please enter a valid option.");

}

}

scanner.nextLine();

} **while** (!exitRequested);

System.***out***.println("Exiting the Ticket Booking System. Thank you!");

scanner.close();

}

**private** **static** String getTicketTypeName(**int** ticketTypeNumber) {

**switch** (ticketTypeNumber) {

**case** 1:

**return** "Silver";

**case** 2:

**return** "Gold";

**case** 3:

**return** "Diamond";

**default**:

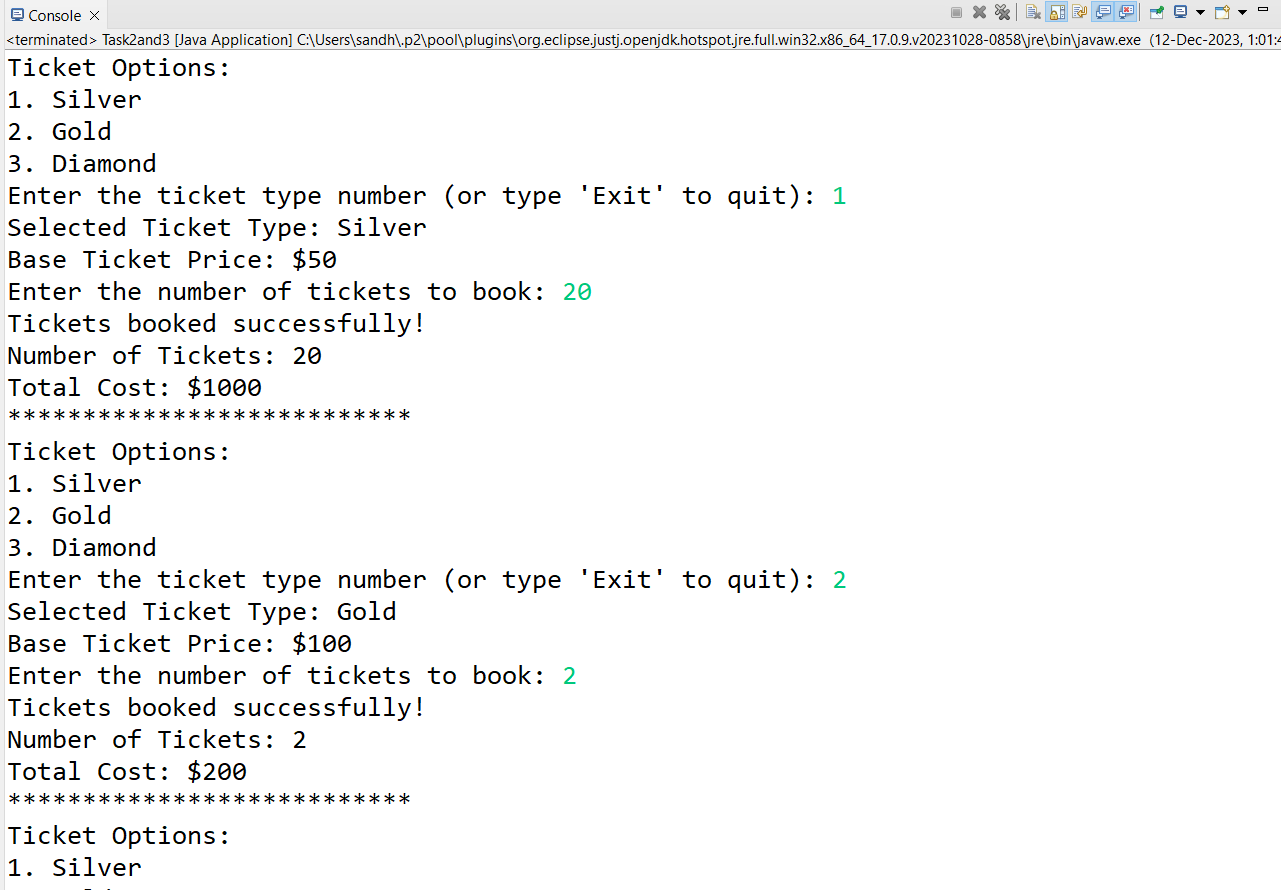
**return** "Unknown";

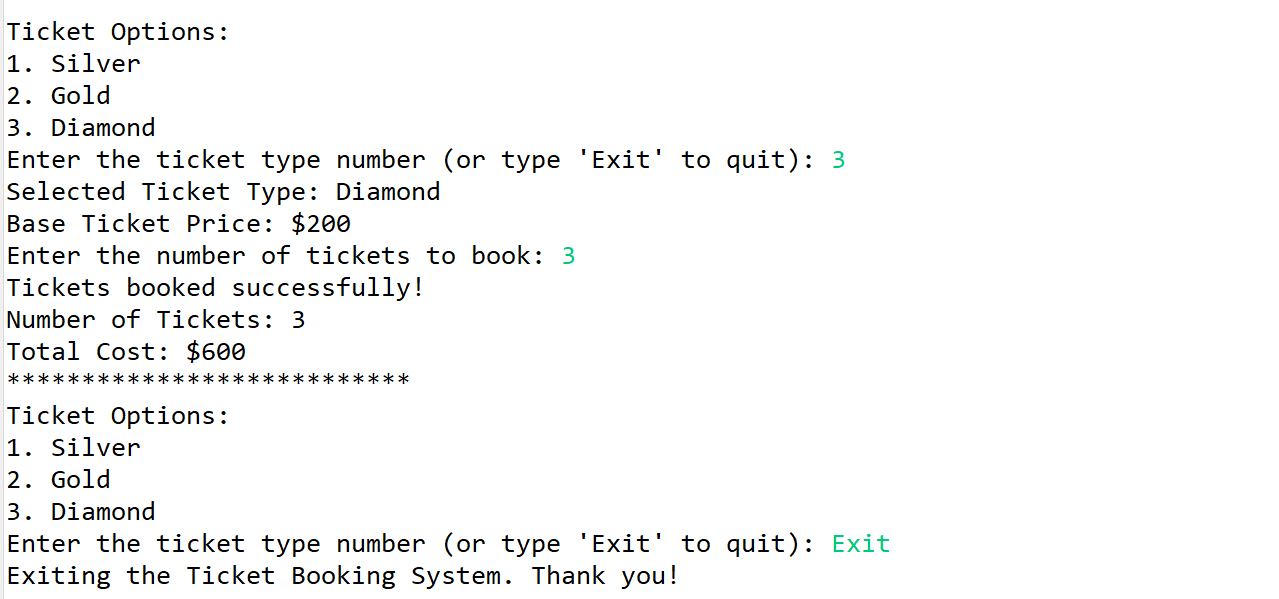
}

}

}

**OUTPUT :**

****



**TASK 4: Class & Object**

**Create a Following classes with the following attributes and methods:**

**Event Class:**

**• Attributes:**

**o event\_name,**

**o event\_date DATE,**

**o event\_time TIME,**

**o venue\_name,**

**o total\_seats,**

**o available\_seats,**

**o ticket\_price DECIMAL,**

**o event\_type ENUM('Movie', 'Sports', 'Concert')**

**• Methods and Constuctors:**

**o Implement default constructors and overload the constructor with Customer**

**attributes, generate getter and setter, (print all information of attribute) methods for**

**the attributes.**

**o calculate\_total\_revenue(): Calculate and return the total revenue based on the**

**number of tickets sold.**

**o getBookedNoOfTickets(): return the total booked tickets**

**o book\_tickets(num\_tickets): Book a specified number of tickets for an event. Initially**

**available seats are equal to the total seats when tickets are booked available seats**

**number should be reduced.**

**o cancel\_booking(num\_tickets): Cancel the booking and update the available seats.**

**o display\_event\_details(): Display event details, including event name, date time seat**

**availability.**

**1. EVENT CLASS :**

**package** com.hexaware.entity;

**import** java.util.Date;

**import** java.time.LocalTime;

**public** **class** Event {

**private** String eventName;

**private** Date eventDate;

**private** LocalTime eventTime;

**private** String venueName;

**private** **int** totalSeats;

**private** **int** availableSeats;

**private** **static** **double** *ticketPrice*;

**private** EventType eventType;

**private** **int** numTicketsSold;

**public** **enum** EventType {

***MOVIE***, ***SPORTS***, ***CONCERT***

}

**public** Event() {

System.***out***.println("From Event Constructor");

}

**public** Event(String eventName, Date eventDate, LocalTime eventTime, String venueName, **int** totalSeats,

**int** availableSeats, EventType eventType, **int** numTicketsSold) {

**super**();

**this**.eventName = eventName;

**this**.eventDate = eventDate;

**this**.eventTime = eventTime;

**this**.venueName = venueName;

**this**.totalSeats = totalSeats;

**this**.availableSeats = availableSeats;

**this**.eventType = eventType;

**this**.numTicketsSold = numTicketsSold;

}

**public** String getEventName() {

**return** eventName;

}

**public** **void** setEventName(String eventName) {

**this**.eventName = eventName;

}

**public** Date getEventDate() {

**return** eventDate;

}

**public** **void** setEventDate(Date eventDate) {

**this**.eventDate = eventDate;

}

**public** LocalTime getEventTime()

{

**return** eventTime;

}

**public** **void** setEventTime(LocalTime eventTime) {

**this**.eventTime = eventTime;

}

**public** String getVenueName() {

**return** venueName;

}

**public** **void** setVenueName(String venueName) {

**this**.venueName = venueName;

}

**public** **int** getTotalSeats() {

**return** totalSeats;

}

**public** **void** setTotalSeats(**int** totalSeats) {

**this**.totalSeats = totalSeats;

}

**public** **int** getAvailableSeats() {

**return** availableSeats;

}

**public** **void** setAvailableSeats(**int** availableSeats) {

**this**.availableSeats = availableSeats;

}

**public** **double** getTicketPrice() {

**return** *ticketPrice*;

}

**public** **void** setTicketPrice(**double** ticketPrice) {

**this**.*ticketPrice* = ticketPrice;

}

**public** EventType getEventType() {

**return** eventType;

}

**public** **void** setEventType(EventType eventType) {

**this**.eventType = eventType;

}

**public** **void** printEventDetails() {

System.***out***.println("Event Name: " + eventName);

System.***out***.println("Event Date: " + eventDate);

System.***out***.println("Event Time: " + eventTime);

System.***out***.println("Venue Name: " + venueName);

System.***out***.println("Total Seats: " + totalSeats);

System.***out***.println("Available Seats: " + availableSeats);

System.***out***.println("Ticket Price: $" + *ticketPrice*);

System.***out***.println("Number of Tickets Booked: " + num);

System.***out***.println("Price of Tickets Booked: " + calculateTotalRevenue());

}

**private** **int** num = 0;

**public** **int** getBookedNoOfTickets() {

**return** num;

}

**public** **int** getNumTicketsSold() {

**return** totalSeats-getBookedNoOfTickets();

}

**public** **double** calculateTotalRevenue() {

**return** num \* *ticketPrice*;

}

**public** **void** bookTickets(**int** numTickets) {

**if** (numTickets > 0 && numTickets <= availableSeats) {

availableSeats = availableSeats-numTickets;

num += numTickets;

System.***out***.println(numTickets + " tickets booked successfully!");

} **else** {

System.***out***.println("Invalid number of tickets to book or insufficient seats available.");

}

}

**public** **void** cancelBooking(**int** numTickets) {

**if** (numTickets > 0 && numTickets <= (totalSeats - availableSeats)) {

availableSeats -= numTickets;

System.***out***.println(numTickets + " tickets canceled successfully!");

} **else** {

System.***out***.println("Invalid number of tickets to cancel or insufficient booked tickets.");

}

}

@Override

**public** String toString() {

**return** "Event [eventName=" + eventName + ", eventDate=" + eventDate + ", eventTime=" + eventTime

+ ", venueName=" + venueName + ", totalSeats=" + totalSeats + ", availableSeats=" + availableSeats

+ ", ticketPrice=" + *ticketPrice* + ", eventType=" + eventType + "]";

}

}

**2. VENUE CLASS :**

**• Attributes:**

**o venue\_name,**

**o address**

**• Methods and Constuctors:**

**o display\_venue\_details(): Display venue details.**

**o Implement default constructors and overload the constructor with Customer**

**attributes, generate getter and setter methods.**

**package** com.hexaware.entity;

**public** **class** Venue {

@Override

**public** String toString() {

**return** "Venue [venueName=" + venueName + ", address=" + address + "]";

}

**private** String venueName;

**private** String address;

**public** Venue() {

}

**public** Venue(String venueName, String address) {

**this**.venueName = venueName;

**this**.address = address;

}

**public** String getVenueName() {

**return** venueName;

}

**public** **void** setVenueName(String venueName) {

**this**.venueName = venueName;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** **void** displayVenueDetails() {

System.***out***.println("Venue Name: " + venueName);

System.***out***.println("Venue Address: " + address);

}

}

**3. CUSTOMER CLASS :**

**• Attributes:**

**o customer\_name,**

**o email,**

**o phone\_number,**

**• Methods and Constuctors:**

**o Implement default constructors and overload the constructor with Customer**

**attributes, generate getter and setter methods.**

**o display\_customer\_details(): Display customer details.**

**Customer.java**

**package** com.hexaware.entity;

**public** **class** Customer {

**private** String customerName;

**private** String email;

**private** String phoneNumber;

**public** Customer() {

}

**public** Customer(String customerName, String email, String phoneNumber) {

**this**.customerName = customerName;

**this**.email = email;

**this**.phoneNumber = phoneNumber;

}

**public** String getCustomerName() {

**return** customerName;

}

**public** **void** setCustomerName(String customerName) {

**this**.customerName = customerName;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPhoneNumber() {

**return** phoneNumber;

}

**public** **void** setPhoneNumber(String phoneNumber) {

**this**.phoneNumber = phoneNumber;

}

**public** **void** display\_customer\_details() {

System.***out***.println("Customer Name" + customerName);

System.***out***.println("Customer Email" + email);

System.***out***.println("Customer Phone Number" + phoneNumber);

}

@Override

**public** String toString() {

**return** "Customer [customerName=" + customerName + ", email=" + email + ", phoneNumber=" + phoneNumber + "]";

}

}

**4. BOOKING CLASS :**

**to represent the Tiket booking system. Perform the following operation in main**

**method. Note:- Use Event class object for the following operation.**

**• Methods and Constuctors:**

**o calculate\_booking\_cost(num\_tickets): Calculate and set the total cost of the**

**booking.**

**o book\_tickets(num\_tickets): Book a specified number of tickets for an event.**

**o cancel\_booking(num\_tickets): Cancel the booking and update the available seats.**

**o getAvailableNoOfTickets(): return the total available tickets**

**o getEventDetails(): return event details from the event class**

**SOURCE CODE :**

**Main.java**

**package** com.hexaware.entity;

**import** com.hexaware.entity.Booking;

**import** com.hexaware.entity.\*;

**import** java.text.ParseException;

**import** java.util.Scanner;

**public** **class** Main{

**public** **static** **void** main(String [] args) **throws** ParseException

{

Scanner sc = **new** Scanner(System.***in***);

Booking booking = **new** Booking();

String ch = **null**;

**try** {

booking.addEvent();

} **catch** (ParseException e1) {

e1.printStackTrace();

}

**do**

{

System.***out***.println("Welcome to ticket booking system");

System.***out***.println("Choose from the options to continue: ");

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

System.***out***.println("2. Calculate total revenue");

System.***out***.println("3. Cancel Booking");

System.***out***.println("4. Display Event Details");

**int** choice = sc.nextInt();

**switch**(choice)

{

**case** 1:{

booking.book\_tickets();

**break**;

}

**case** 2:{

booking.revenue();

**break**;

}

**case** 3:

{

booking.cancel\_booking();

**break**;

}

**case** 4:

{

booking.getEventDetails();

**break**;

}

**default** :{

System.***out***.println("Invalid choice, choose again");

**break**;

}

}

System.***out***.println("Do you want to continue? Y | n");

ch = sc.next();

}

**while** (ch.equals("Y") || ch.equals("y"));

System.***out***.println("Thanks for using our system");

}

}

**Booking.java**

**package** com.hexaware.entity;

**import** java.text.ParseException;

**import** java.text.SimpleDateFormat;

**import** java.time.LocalDate;

**import** java.time.LocalTime;

**import** java.util.Date;

**import** java.util.Scanner;

**public** **class** Booking {

Event event = **new** Event();

**private** Scanner scanner = **new** Scanner(System.***in***);

**public** **void** addEvent() **throws** ParseException {

System.***out***.println("Enter the Event name");

String eventName = scanner.nextLine();

event.setEventName(eventName);

System.***out***.println("Enter Event date (dd/MM/yyyy): ");

String dateStr = scanner.next();

SimpleDateFormat sdf = **new** SimpleDateFormat("dd/MM/yyyy");

**try** {

Date eventDate = sdf.parse(dateStr);

event.setEventDate(eventDate);

} **catch** (ParseException e) {

System.***out***.println("Invalid date format, enter in dd/MM/yyyy format");

e.printStackTrace();

**return**;

}

System.***out***.println("Enter Event time: ");

String eventTime = scanner.next();

LocalTime localTime = LocalTime.*parse*(eventTime);

event.setEventTime(localTime);

System.***out***.println("Enter the Event Venue:");

String venueName = scanner.next();

event.setVenueName(venueName);

System.***out***.println("Enter total no. of seats: ");

**int** totalSeats = scanner.nextInt();

event.setTotalSeats(totalSeats);

System.***out***.println("Enter the Available seats: ");

**int** availableSeats = scanner.nextInt();

event.setAvailableSeats(availableSeats);

System.***out***.println("Enter the Ticket Price:");

**double** ticketPrice = scanner.nextDouble();

event.setTicketPrice(ticketPrice);

}

**public** **void** revenue() {

**double** totalRevenue = event.calculateTotalRevenue();

System.***out***.println("Total Revenue: $" + totalRevenue);

}

**public** **void** book\_tickets() {

System.***out***.println("Enter the no. of tickets to book: ");

**int** num = scanner.nextInt();

event.bookTickets(num);

}

**public** **void** cancel\_booking() {

System.***out***.println("Enter the no. of tickets to be cancelled : ");

**int** num = scanner.nextInt();

event.cancelBooking(num);

}

**public** **int** getAvailableNoOfTickets(**int** num\_tickets) {

**return** event.getAvailableSeats();

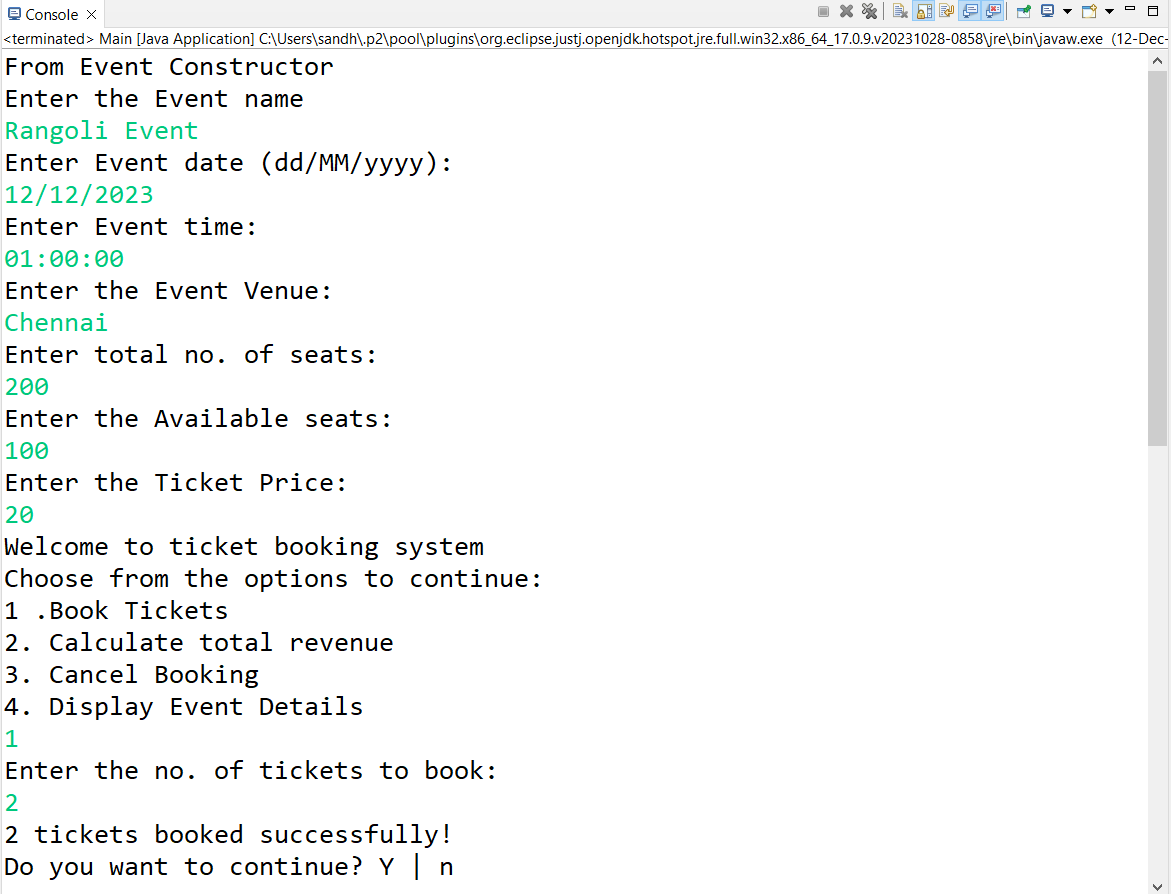
}

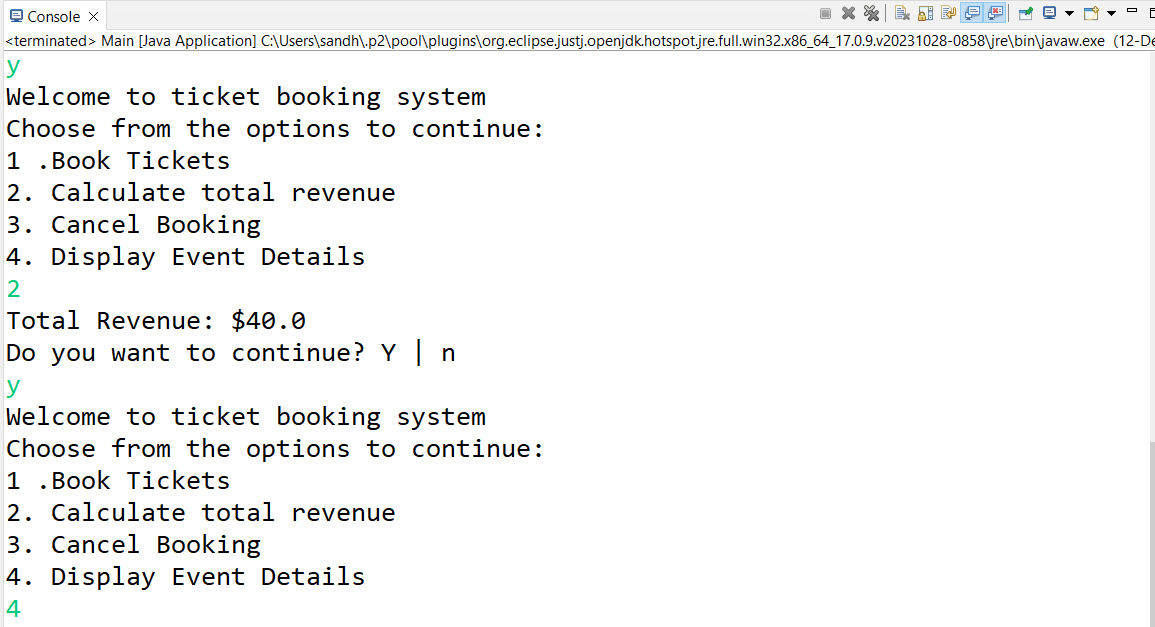
**public** **void** getEventDetails() {

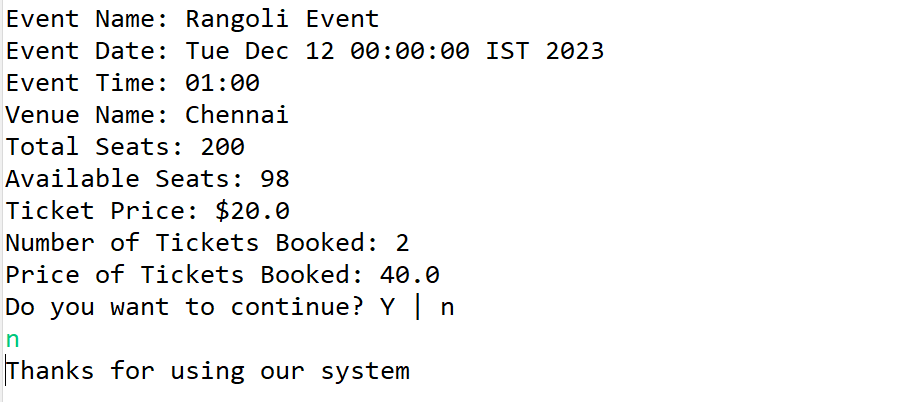
event.printEventDetails();

}}

**OUTPUT :**

****

****

****

**Task 5: Inheritance and polymorphism**

**1. Inheritance**

**• Create a subclass Movie that inherits from Event. Add the following attributes and methods:**

**o Attributes:**

**1. genre: Genre of the movie (e.g., Action, Comedy, Horror).**

**2. ActorName**

**3. ActresName**

**o Methods:**

**1. Implement default constructors and overload the constructor with Customer**

**attributes, generate getter and setter methods.**

**2. display\_event\_details(): Display movie details, including genre.**

**• Create another subclass Concert that inherits from Event. Add the following attributes and**

**methods:**

**o Attributes:**

**1. artist: Name of the performing artist or band.**

**2. type: (Theatrical, Classical, Rock, Recital)**

**o Methods:**

**1. Implement default constructors and overload the constructor with Customer**

**attributes, generate getter and setter methods.**

**2. display\_concert\_details(): Display concert details, including the artist.**

**• Create another subclass Sports that inherits from Event. Add the following attributes and**

**methods:**

**o Attributes:**

**1. sportName: Name of the game.**

**2. teamsName: (India vs Pakistan)**

**o Methods:**

**1. Implement default constructors and overload the constructor with Customer**

**attributes, generate getter and setter methods.**

**2. display\_sport\_details(): Display concert details, including the artist.**

**• Create a class TicketBookingSystem with the following methods:**

**o create\_event(event\_name: str, date:str, time:str, total\_seats: int, ticket\_price:**

**float, event\_type: str, venu\_name:str): Create a new event with the specified details**

**and event type (movie, sport or concert) and return event object.**

**o display\_event\_details(event: Event): Accepts an event object and calls its**

**display\_event\_details() method to display event details.**

**o book\_tickets(event: Event, num\_tickets: int):**

**1. Accepts an event object and the number of tickets to be booked.**

**2. Checks if there are enough available seats for the booking.**

**3. If seats are available, updates the available seats and returns the total cost**

**of the booking.**

**4. If seats are not available, displays a message indicating that the event is sold**

**out.**

**o cancel\_tickets(event: Event, num\_tickets): cancel a specified number of tickets for**

**an event.**

**o main(): simulates the ticket booking system**

**1. User can book tickets and view the event details as per their choice in menu**

**(movies, sports, concerts).**

**2. Display event details using the display\_event\_details() method without**

**knowing the specific event type (demonstrate polymorphism).**

**3. Make bookings using the book\_tickets() and cancel tickets cancel\_tickets() method.**

**SOURCE CODE :**

**Movie.java**

**package** com.hexaware.entity;

**import** java.time.LocalTime;

**public** **class** Movie **extends** Event {

**private** String genre;

**private** String actorName;

**private** String actressName;

**public** Movie(String eventName, java.util.Date eventDate, LocalTime eventTime, String venueName, **int** totalSeats,

**int** availableSeats, EventType eventType, **int** numTicketsSold, String genre, String actorName,

String actressName) {

**super**(eventName, eventDate, eventTime, venueName, totalSeats, availableSeats, eventType, numTicketsSold);

**this**.genre = genre;

**this**.actorName = actorName;

**this**.actressName = actressName;

}

**public** String getGenre() {

**return** genre;

}

**public** **void** setGenre(String genre) {

**this**.genre = genre;

}

**public** String getActorName() {

**return** actorName;

}

**public** **void** setActorName(String actorName) {

**this**.actorName = actorName;

}

**public** String getActressName() {

**return** actressName;

}

**public** **void** setActressName(String actressName) {

**this**.actressName = actressName;

}

@Override

**public** **void** printEventDetails() {

**super**.printEventDetails();

System.***out***.println("Genre: " + genre);

System.***out***.println("Actor Name: " + actorName);

System.***out***.println("Actress Name: " + actressName);

}

@Override

**public** String toString() {

**return** "Movie [genre=" + genre + ", actorName=" + actorName + ", actressName=" + actressName + "]";

}

}

**Concert.java**

**package** com.hexaware.entity;

**import** java.sql.Date;

**import** java.time.LocalTime;

**import** com.hexaware.entity.Event.EventType;

**public** **class** Concert **extends** Event {

**private** String type;

**private** String artist;

**public** Concert(String eventName, java.util.Date eventDate, LocalTime eventTime, String venueName, **int** totalSeats,

**int** availableSeats, EventType eventType, **int** numTicketsSold,String artist,String type) {

**super**(eventName, eventDate, eventTime, venueName, totalSeats, availableSeats, eventType, numTicketsSold);

**this**.artist = artist;

**this**.type = type;

}

**public** String getArtist() {

**return** artist;

}

**public** **void** setArtist(String artist) {

**this**.artist = artist;

}

**public** String getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

@Override

**public** **void** printEventDetails() {

**super**.printEventDetails();

System.***out***.println("Concert Type: " + type);

System.***out***.println("Artist Name: " + artist);

}

@Override

**public** String toString() {

**return** "Concert [artist=" + artist + ", type=" + type + "]";

}

}

**Sports.java**

**package** com.hexaware.entity;

**import** java.sql.Date;

**import** java.time.LocalTime;

**import** com.hexaware.entity.Event.EventType;

**public** **class** Sports **extends** Event{

**private** String sportName;

**private** String teamsName;

**public** Sports(String eventName, java.util.Date eventDate, LocalTime eventTime, String venueName, **int** totalSeats,

**int** availableSeats, EventType eventType, **int** numTicketsSold, String sportName, String teamsName) {

**super**(eventName, eventDate, eventTime, venueName, totalSeats, availableSeats, eventType, numTicketsSold);

**this**.sportName = sportName;

**this**.teamsName = teamsName;

}

**public** String getSportName() {

**return** sportName;

}

**public** **void** setSportName(String sportName) {

**this**.sportName = sportName;

}

**public** String getTeamsName() {

**return** teamsName;

}

**public** **void** setTeamsName(String teamsName) {

**this**.teamsName = teamsName;

}

**public** **void** display\_event\_details() {

**super**.printEventDetails();

System.***out***.println("Sport: " + sportName);

}

@Override

**public** **void** printEventDetails() {

**super**.printEventDetails();

System.***out***.println("Sport Name: " + sportName);

System.***out***.println("Teams Name: " + teamsName);

}

@Override

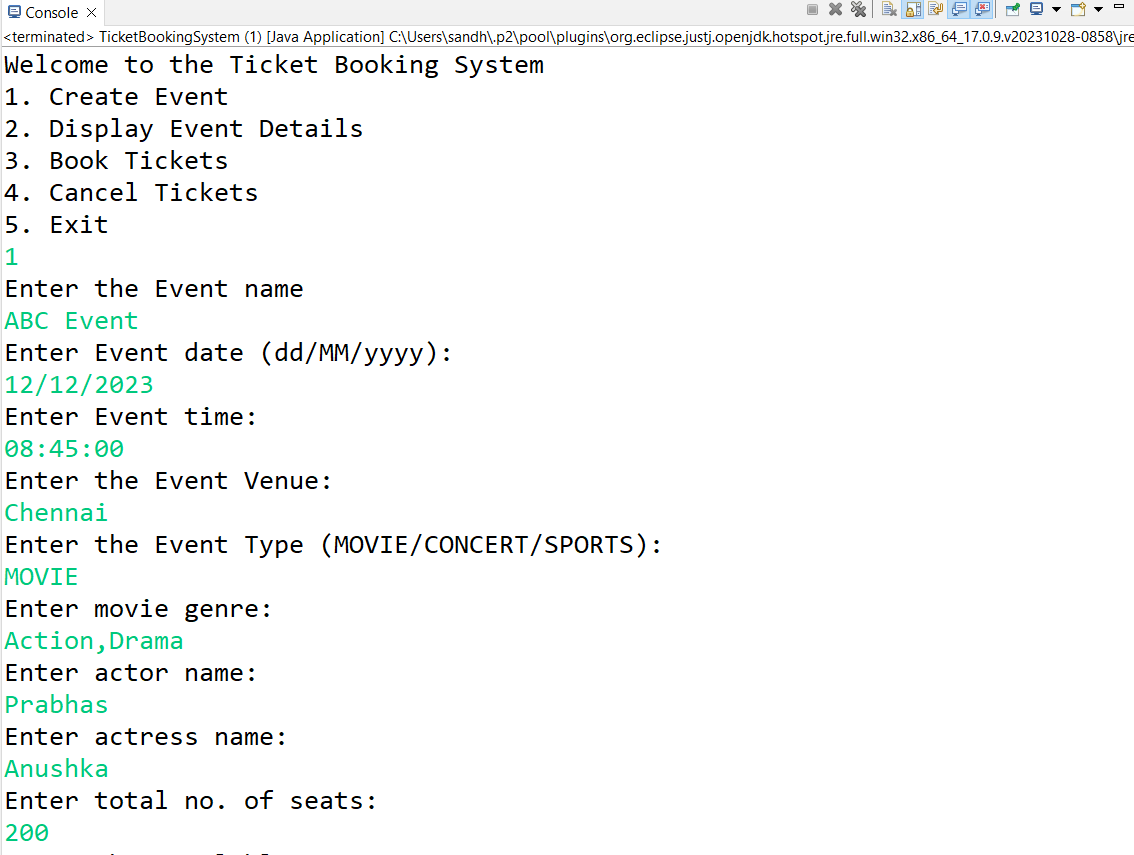
**public** String toString() {

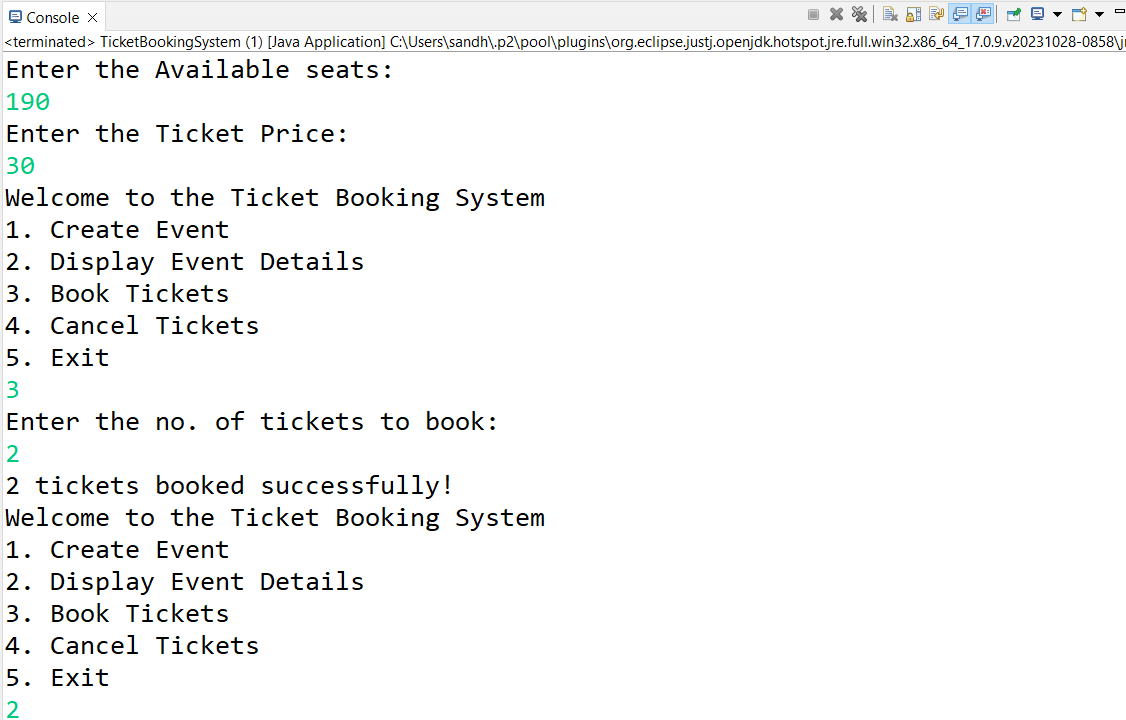
**return** "Sports [sportName=" + sportName + ", teamsName=" + teamsName + "]";

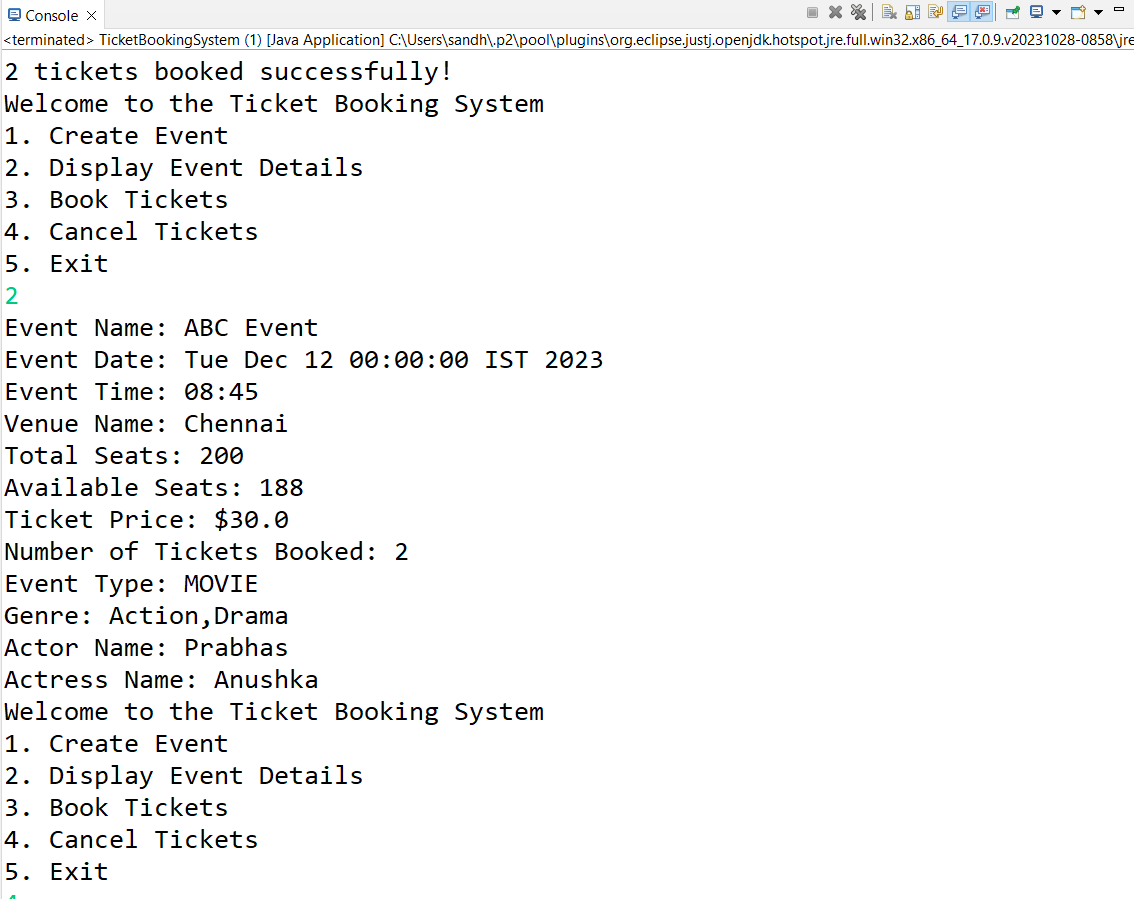
}

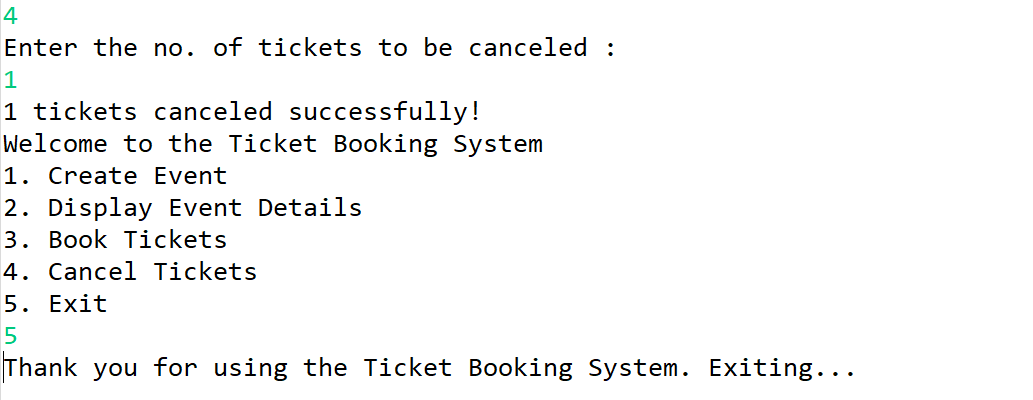
}

**MOVIE :**

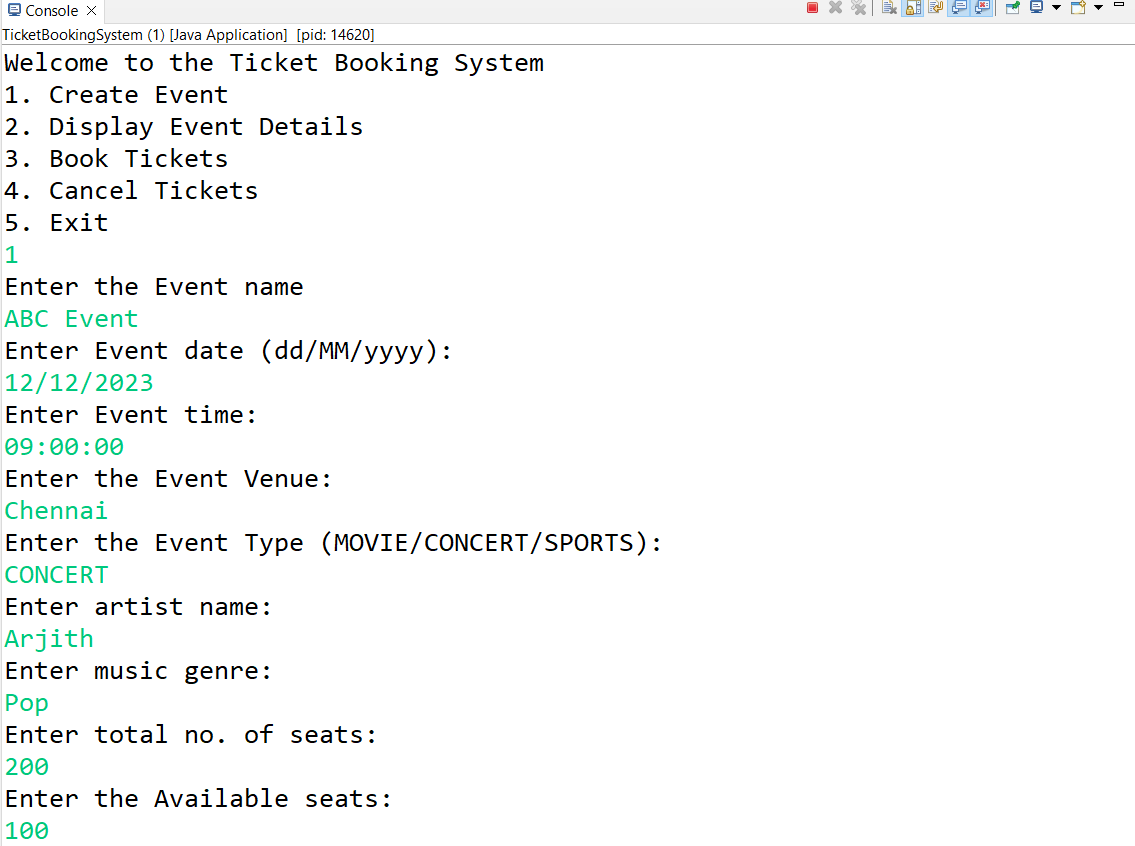
****

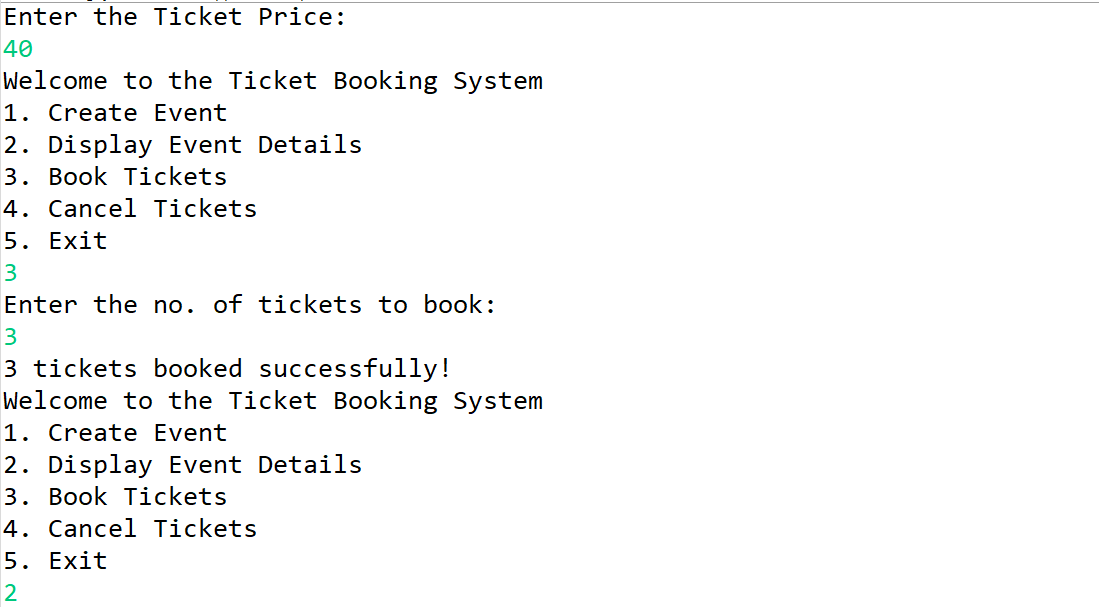
****

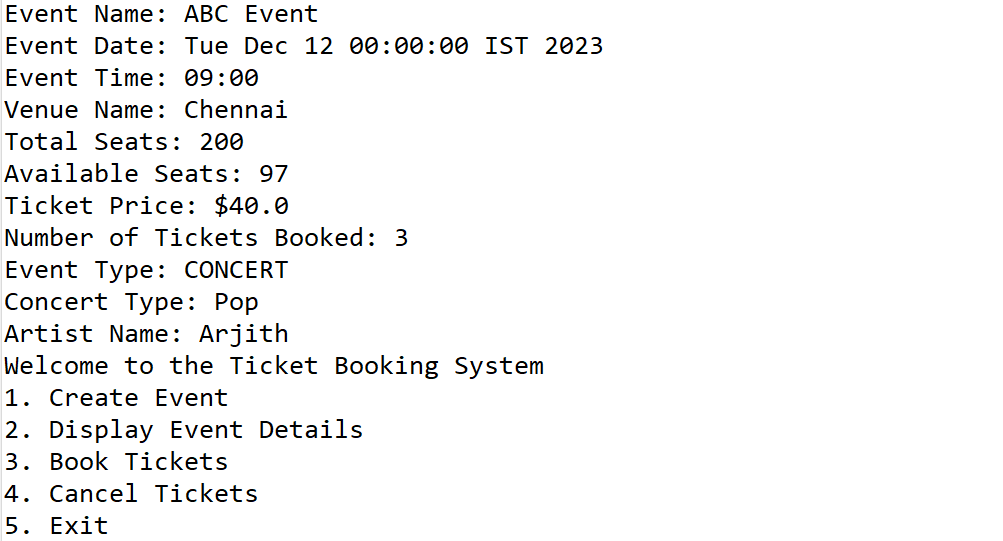
****

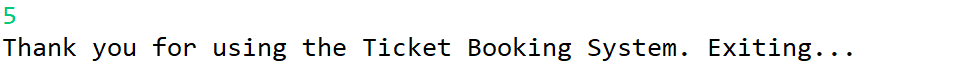
****

**CONCERT :**

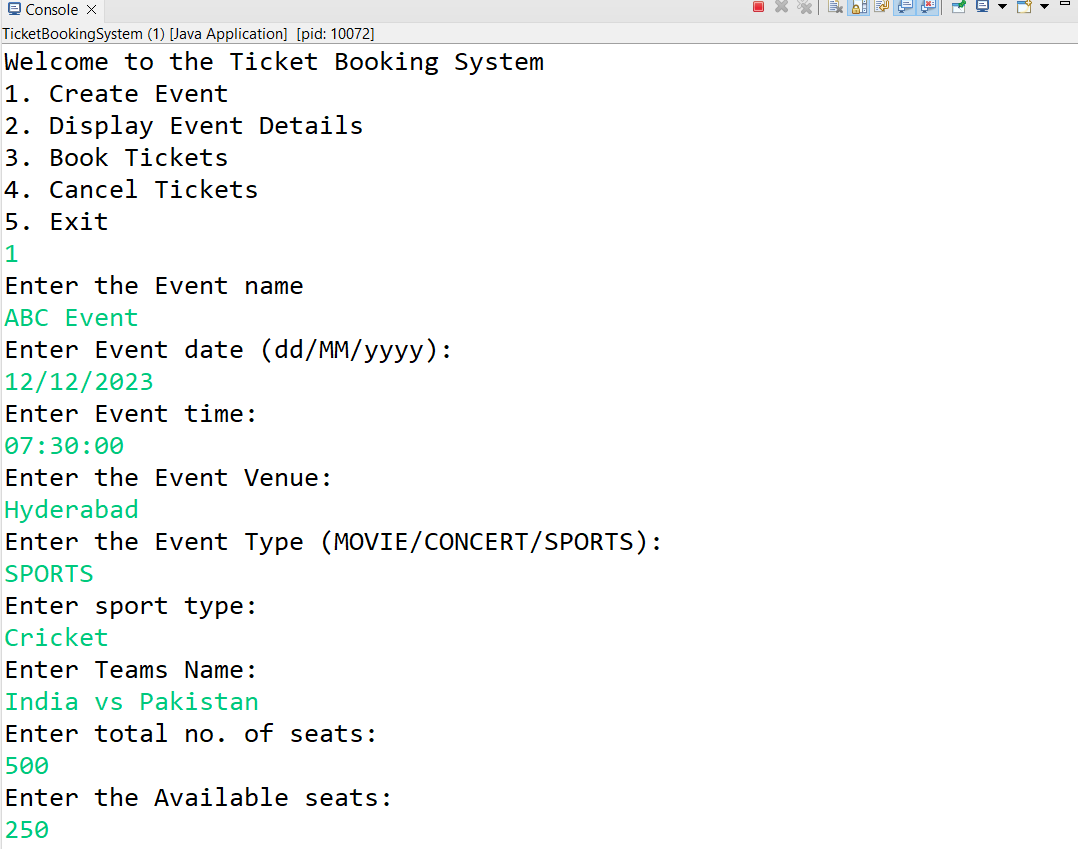
****

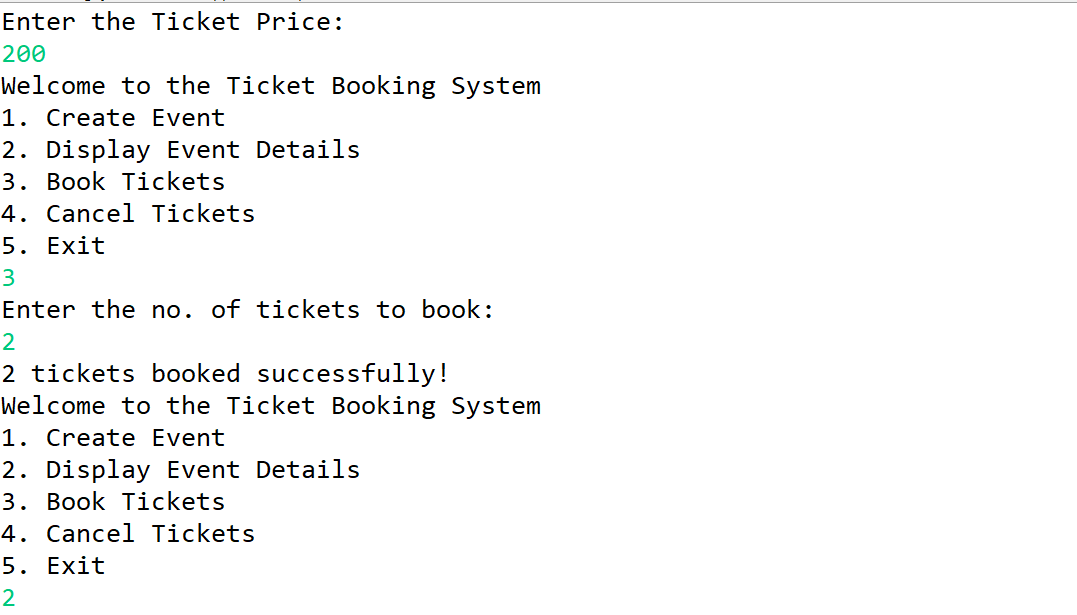
****

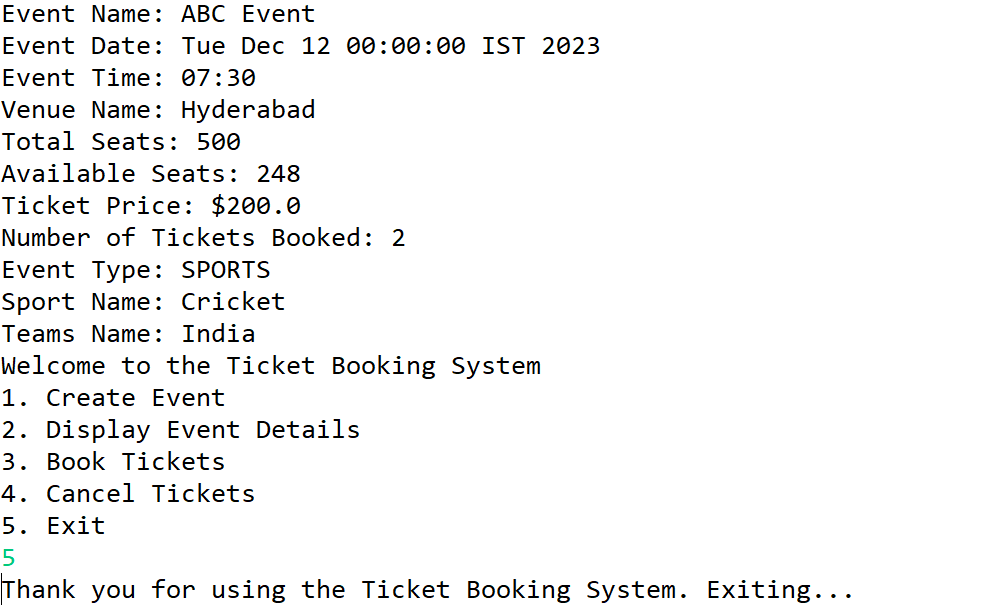
****

****

**SPORTS :**

****

****

****

**TASK 6 to 10:**

**Abstraction**

**Requirements:**

**1. Event Abstraction:**

**• Create an abstract class Event that represents a generic event. It should include the**

**following attributes and methods as mentioned in TASK 1:**

**2. Concrete Event Classes:**

**• Create three concrete classes that inherit from Event abstract class and override abstract**

**methods in concrete class should declare the variables as mentioned in above Task 2:**

**• Movie.**

**• Concert.**

**• Sport.**

**3. BookingSystem Abstraction:**

**• Create an abstract class BookingSystem that represents the ticket booking system. It should**

**include the methods of TASK 2 TicketBookingSystem:**

**4. Concrete TicketBookingSystem Class:**

**• Create a concrete class TicketBookingSystem that inherits from BookingSystem:**

**• TicketBookingSystem: Implement the abstract methods to create events, book**

**tickets, and retrieve available seats. Maintain an array of events in this class.**

**• Create a simple user interface in a main method that allows users to interact with the ticket**

**booking system by entering commands such as "create\_event", "book\_tickets",**

**"cancel\_tickets", "get\_available\_seats," and "exit."**

**Create a Following classes with the following attributes and methods:**

**1. Venue Class**

**• Attributes:**

**o venue\_name,**

**o address**

**• Methods and Constuctors:**

**o display\_venue\_details(): Display venue details.**

**o Implement default constructors and overload the constructor with Customer**

**attributes, generate getter and setter methods.**

**2. Event Class:**

**• Attributes:**

**o event\_name,**

**o event\_date DATE,**

**o event\_time TIME,**

**o venue (reference of class Venu),**

**o total\_seats,**

**o available\_seats,**

**o ticket\_price DECIMAL,**

**o event\_type ENUM('Movie', 'Sports', 'Concert')**

**• Methods and Constuctors:**

**o Implement default constructors and overload the constructor with Customer**

**attributes, generate getter and setter, (print all information of attribute) methods**

**for the attributes.**

**o calculate\_total\_revenue(): Calculate and return the total revenue based on the**

**number of tickets sold.**

**o getBookedNoOfTickets(): return the total booked tickets**

**o book\_tickets(num\_tickets): Book a specified number of tickets for an event. Initially**

**available seats are equal to total seats when tickets are booked available seats**

**number should be reduced.**

**o cancel\_booking(num\_tickets): Cancel the booking and update the available seats.**

**o display\_event\_details(): Display event details, including event name, date time seat**

**availability.**

**3. Event sub classes:**

**• Create three sub classes that inherit from Event abstract class and override abstract**

**methods in concrete class should declare the variables as mentioned in above Task 2:**

**o Movie.**

**o Concert.**

**o Sport.**

**4. Customer Class**

**• Attributes:**

**o customer\_name,**

**o email,**

**o phone\_number,**

**• Methods and Constuctors:**

**o Implement default constructors and overload the constructor with Customer**

**attributes, generate getter and setter methods.**

**o display\_customer\_details(): Display customer details.**

**5. Create a class Booking with the following attributes:**

**• bookingId (should be incremented for each booking)**

**• array of customer (reference to the customer who made the booking)**

**• event (reference to the event booked)**

**• num\_tickets(no of tickets and array of customer must equal)**

**• total\_cost**

**• booking\_date (timestamp of when the booking was made)**

**• Methods and Constuctors:**

**o Implement default constructors and overload the constructor with Customer**

**attributes, generate getter and setter methods.**

**o display\_booking\_details(): Display customer details.**

**6. BookingSystem Class to represent the Ticket booking system. Perform the following operation in**

**main method. Note: - Use Event class object for the following operation.**

**• Attributes**

**o array of events**

**• Methods and Constuctors:**

**o create\_event(event\_name: str, date:str, time:str, total\_seats: int, ticket\_price:**

**float, event\_type: str, venu:Venu): Create a new event with the specified details and**

**event type (movie, sport or concert) and return event object.**

**o calculate\_booking\_cost(num\_tickets): Calculate and set the total cost of the**

**booking.**

**o book\_tickets(eventname:str, num\_tickets, arrayOfCustomer): Book a specified**

**number of tickets for an event. for each tickets customer object should be created**

**and stored in array also should update the attributes of Booking class.**

**o cancel\_booking(booking\_id): Cancel the booking and update the available seats.**

**o getAvailableNoOfTickets(): return the total available tickets**

**o getEventDetails(): return event details from the event class**

**o Create a simple user interface in a main method that allows users to interact with**

**the ticket booking system by entering commands such as "create\_event",**

**"book\_tickets", "cancel\_tickets", "get\_available\_seats,", "get\_event\_details," and**

**"exit."**

**SOURCE CODE :**

**Event.java**

**package** com.hexaware.entity;

**import** java.util.Date;

**import** com.hexaware.dao.Bookingdao;

**import** java.time.LocalTime;

**public** **public** **abstract** **class**Event {

**private** **int** eventId;

**private** String eventName;

**private** Date eventDate;

**private** LocalTime eventTime;

**private** **int** venueId;

**private** **int** totalSeats;

**private** **int** availableSeats;

**private** **static** **double** *ticketPrice*;

**private** EventType eventType;

**private** **int** numTicketsSold;

**private** Bookingdao bookingdao = **new** Bookingdao();

**public** **enum** EventType {

***MOVIE***, ***SPORTS***, ***CONCERT***

}

**public** Event() {

System.***out***.println("From Event Constructor");

}

**public** Event(**int** eventId,String eventName, Date eventDate, LocalTime eventTime, **int** venueId,**int** totalSeats,

**int** availableSeats, EventType eventType, **int** numTicketsSold) {

**super**();

**this**.eventId=eventId;

**this**.eventName = eventName;

**this**.eventDate = eventDate;

**this**.eventTime = eventTime;

**this**.venueId = venueId;

**this**.totalSeats = totalSeats;

**this**.availableSeats = availableSeats;

**this**.eventType = eventType;

**this**.numTicketsSold = numTicketsSold;

}

**public** **int** getEventId() {

**return** eventId;

}

**public** **void** setEventId(**int** eventId) {

**this**.eventId = eventId;

}

**public** String getEventName() {

**return** eventName;

}

**public** **void** setEventName(String eventName) {

**this**.eventName = eventName;

}

**public** Date getEventDate() {

**return** eventDate;

}

**public** **void** setEventDate(Date eventDate) {

**this**.eventDate = eventDate;

}

**public** LocalTime getEventTime() {

**return** eventTime;

}

**public** **void** setEventTime(LocalTime eventTime) {

**this**.eventTime = eventTime;

}

**public** **int** getVenueId() {

**return** venueId;

}

**public** **void** setVenueId(**int** venueId) {

**this**.venueId = venueId;

}

**public** **int** getTotalSeats() {

**return** totalSeats;

}

**public** **void** setTotalSeats(**int** totalSeats) {

**this**.totalSeats = totalSeats;

}

**public** **int** getAvailableSeats() {

**return** availableSeats;

}

**public** **void** setAvailableSeats(**int** availableSeats) {

**this**.availableSeats = availableSeats;

}

**public** **double** getTicketPrice() {

**return** *ticketPrice*;

}

**public** **void** setTicketPrice(**double** ticketPrice) {

**this**.*ticketPrice* = ticketPrice;

}

**public** EventType getEventType() {

**return** eventType;

}

**public** **void** setEventType(EventType eventType) {

**this**.eventType = eventType;

}

**public** **void** printEventDetails() {

System.***out***.println("Event Id: " + eventId);

System.***out***.println("Event Name: " + eventName);

System.***out***.println("Event Date: " + eventDate);

System.***out***.println("Event Time: " + eventTime);

System.***out***.println("Venue Id: " + venueId);

System.***out***.println("Total Seats: " + totalSeats);

System.***out***.println("Available Seats: " + availableSeats);

System.***out***.println("Ticket Price: $" + *ticketPrice*);

//System.out.println("Number of Tickets Booked: " + num);

System.***out***.println("Event Type: " + eventType);

System.***out***.println("Event Type: " + eventType);

}

**private** **int** num = 0;

**public** **int** getBookedNoOfTickets() {

**return** num;

}

**public** **int** getNumTicketsSold() {

**return** totalSeats - getBookedNoOfTickets();

}

**public** **double** calculateTotalRevenue(**double** booked) {

**return** booked \* *ticketPrice*;

}

**public** **double** bookTickets(**int** numTickets,**int** eventId) {

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

availableSeats=bookingdao.getAvailableTickets(eventId);

**if** (numTickets > 0 && numTickets <= availableSeats) {

availableSeats = availableSeats - numTickets;

num += numTickets;

System.***out***.println(numTickets + " tickets booked successfully!");

} **else** {

System.***out***.println("Invalid number of tickets to book or insufficient seats available.");

}

**return** numTickets;

}

**public** **void** cancelBooking(**int** numTickets) {

**if** (numTickets > 0 && numTickets <= (totalSeats - availableSeats)) {

availableSeats -= numTickets;

System.***out***.println(numTickets + " tickets canceled successfully!");

} **else** {

System.***out***.println("Invalid number of tickets to cancel or insufficient booked tickets.");

}

}

@Override

**public** String toString() {

**return** "Event [eventId=" + eventId + ", eventName=" + eventName + ", eventDate=" + eventDate + ", eventTime="

+ eventTime + ", venueId=" + venueId + ", totalSeats=" + totalSeats + ", availableSeats="

+ availableSeats + ", eventType=" + eventType + ", numTicketsSold=" + numTicketsSold + ", num=" + num

+ "]";

}

}

**Movie.java**

**package** com.hexaware.entity;

**import** java.time.LocalTime;

**public** **class** Movie **extends** Event {

**private** String genre;

**private** String actorName;

**private** String actressName;

**public** Movie(**int** eventId,String eventName, java.util.Date eventDate, LocalTime eventTime,**int** venueId, **int** totalSeats,

**int** availableSeats, EventType eventType, **int** numTicketsSold, String genre, String actorName,

String actressName) {

**super**(eventId,eventName, eventDate, eventTime,venueId, totalSeats, availableSeats, eventType, numTicketsSold);

**this**.genre = genre;

**this**.actorName = actorName;

**this**.actressName = actressName;

}

**public** String getGenre() {

**return** genre;

}

**public** **void** setGenre(String genre) {

**this**.genre = genre;

}

**public** String getActorName() {

**return** actorName;

}

**public** **void** setActorName(String actorName) {

**this**.actorName = actorName;

}

**public** String getActressName() {

**return** actressName;

}

**public** **void** setActressName(String actressName) {

**this**.actressName = actressName;

}

@Override

**public** **void** printEventDetails() {

**super**.printEventDetails();

System.***out***.println("Genre: " + genre);

System.***out***.println("Actor Name: " + actorName);

System.***out***.println("Actress Name: " + actressName);

}

@Override

**public** String toString() {

**return** "Movie [genre=" + genre + ", actorName=" + actorName + ", actressName=" + actressName + "]";

}

}

**Concert.java**

**package** com.hexaware.entity;

**import** java.time.LocalTime;

**public** **class** Concert **extends** Event {

**private** String type;

**private** String artist;

**public** Concert(**int** eventId,String eventName, java.util.Date eventDate, LocalTime eventTime,**int** venueId, **int** totalSeats,

**int** availableSeats, EventType eventType, **int** numTicketsSold,String artist,String type) {

**super**(eventId,eventName, eventDate, eventTime,venueId, totalSeats, availableSeats, eventType, numTicketsSold);

**this**.artist = artist;

**this**.type = type;

}

**public** String getArtist() {

**return** artist;

}

**public** **void** setArtist(String artist) {

**this**.artist = artist;

}

**public** String getType() {

**return** type;

}

**public** **void** setType(String type) {

**this**.type = type;

}

@Override

**public** **void** printEventDetails() {

**super**.printEventDetails();

System.***out***.println("Concert Type: " + type);

System.***out***.println("Artist Name: " + artist);

}

@Override

**public** String toString() {

**return** "Concert [artist=" + artist + ", type=" + type + "]";

}

}

**Sports.java**

**package** com.hexaware.entity;

**import** java.time.LocalTime;

**public** **class** Sports **extends** Event{

**private** String sportName;

**private** String teamsName;

**public** Sports(**int** eventId,String eventName, java.util.Date eventDate, LocalTime eventTime,**int** venueId,**int** totalSeats,

**int** availableSeats, EventType eventType, **int** numTicketsSold, String sportName, String teamsName) {

**super**(eventId,eventName, eventDate, eventTime,venueId, totalSeats, availableSeats, eventType, numTicketsSold);

**this**.sportName = sportName;

**this**.teamsName = teamsName;

}

**public** String getSportName() {

**return** sportName;

}

**public** **void** setSportName(String sportName) {

**this**.sportName = sportName;

}

**public** String getTeamsName() {

**return** teamsName;

}

**public** **void** setTeamsName(String teamsName) {

**this**.teamsName = teamsName;

}

**public** **void** display\_event\_details() {

**super**.printEventDetails();

System.***out***.println("Sport: " + sportName);

}

@Override

**public** **void** printEventDetails() {

**super**.printEventDetails();

System.***out***.println("Sport Name: " + sportName);

System.***out***.println("Teams Name: " + teamsName);

}

@Override

**public** String toString() {

**return** "Sports [sportName=" + sportName + ", teamsName=" + teamsName + "]";

}

}

**Venue.java**

**package** com.hexaware.entity;

**import** com.hexaware.controller.BookingController;

**public** **class** Venue {

**private** **int** venueId;

**private** String venueName;

**private** String address;

**public** Venue() {

}

**public** Venue(**int** venueId,String venueName, String address) {

**this**.venueId = venueId;

**this**.venueName = venueName;

**this**.address = address;

}

**public** **int** getVenueId() {

**return** venueId;

}

**public** **void** setVenueId(**int** venueId) {

**this**.venueId = venueId;

}

**public** String getVenueName() {

**return** venueName;

}

**public** **void** setVenueName(String venueName) {

**this**.venueName = venueName;

}

**public** String getAddress() {

**return** address;

}

**public** **void** setAddress(String address) {

**this**.address = address;

}

**public** **void** displayVenueDetails() {

System.***out***.println("Venue Id: " + venueId);

System.***out***.println("Venue Name: " + venueName);

System.***out***.println("Venue Address: " + address);

}

@Override

**public** String toString() {

**return** "Venue [venueId=" + venueId + ", venueName=" + venueName + ", address=" + address + "]";

}

}

**Customer.java**

**package** com.hexaware.entity;

**public** **class** Customer {

**private** String customerName;

**private** String email;

**private** String phoneNumber;

**private** **int** customerId;

**public** Customer() {

}

**public** Customer(String customerName, String email, String phoneNumber,**int** customerId) {

**this**.customerName = customerName;

**this**.email = email;

**this**.phoneNumber = phoneNumber;

**this**.customerId=customerId;

}

**public** String getCustomerName() {

**return** customerName;

}

**public** **void** setCustomerName(String customerName) {

**this**.customerName = customerName;

}

**public** String getEmail() {

**return** email;

}

**public** **void** setEmail(String email) {

**this**.email = email;

}

**public** String getPhoneNumber() {

**return** phoneNumber;

}

**public** **void** setPhoneNumber(String phoneNumber) {

**this**.phoneNumber = phoneNumber;

}

**public** **int** getCustomerId() {

**return** customerId;

}

**public** **void** setCustomerId(**int** customerId) {

**this**.customerId = customerId;

}

**public** **void** display\_customer\_details() {

System.***out***.println("Customer Name" + customerName);

System.***out***.println("Customer Email" + email);

System.***out***.println("Customer Phone Number" + phoneNumber);

}

@Override

**public** String toString() {

**return** "Customer [customerName=" + customerName + ", email=" + email + ", phoneNumber=" + phoneNumber

+ ", customerId=" + customerId + "]";

}

}

**TicketBookingSystem.java**

**package** com.hexaware.main;

**import** java.text.ParseException;

**import** java.util.Scanner;

**import** com.hexaware.controller.BookingController;

**import** com.hexaware.entity.Event;

**public** **class** TicketBookingSystem {

BookingController booking = **new** BookingController();

**private** **static** Scanner *scanner* = **new** Scanner(System.***in***);

**public** **void** displayEventDetails(Event event) {

event.printEventDetails();

}

**public** **double** bookTickets(Event event, **int** numTickets,**int** eventId) {

**double** totalCost = event.bookTickets(numTickets,eventId);

**if** (totalCost > 0) {

System.***out***.println(numTickets + " tickets booked successfully!");

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

} **else** {

System.***out***.println("Booking failed. Insufficient seats available.");

}

**return** totalCost;

}

**public** **void** cancelTickets(Event event, **int** numTickets) {

event.cancelBooking(numTickets);

System.***out***.println(numTickets + " tickets canceled successfully!");

}

**public** **static** **void** main(String[] args) **throws** ParseException {

TicketBookingSystem ticketBookingSystem = **new** TicketBookingSystem();

**while** (**true**) {

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

System.***out***.println("1. Create Event");

System.***out***.println("2. Display Event Details");

System.***out***.println("3. Book Tickets");

System.***out***.println("4. Cancel Tickets");

System.***out***.println("5. Create Customer");

System.***out***.println("6. Create Venue");

System.***out***.println("7. Exit");

**int** choice = *scanner*.nextInt();

*scanner*.nextLine();

**switch** (choice) {

**case** 1:

ticketBookingSystem.createEvent();

**break**;

**case** 2:

ticketBookingSystem.displayEventDetails();

**break**;

**case** 3:

ticketBookingSystem.bookTickets();

**break**;

**case** 4:

ticketBookingSystem.cancelTickets();

**break**;

**case** 5:

ticketBookingSystem.registercustomer();

**break**;

**case** 6:

ticketBookingSystem.createVenue();

**break**;

**case** 7:

System.***out***.println("Thank you for using the Ticket Booking System. Exiting...");

System.*exit*(0);

**break**;

**default**:

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

}

}

}

**private** **void** createEvent() **throws** ParseException {

booking.addEvent();

}

**private** **void** createVenue() **throws** ParseException {

booking.createVenueDetails();

}

**private** **void** displayEventDetails() {

booking.displayBookingDetails();

}

**private** **void** bookTickets() {

booking.book\_tickets();

}

**private** **void** cancelTickets() {

booking.cancel\_booking();

}

**private** **void** registercustomer() {

booking.createCustomer();

}

/\*private void convertToDate(String dateStr) throws ParseException {

}

private void convertToTime(String timeStr) {

}\*/

}

package com.hexaware.controller;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.time.LocalDate;

import java.time.LocalTime;

import java.util.Date;

import java.util.Scanner;

import com.hexaware.entity.Concert;

import com.hexaware.entity.Customer;

import com.hexaware.entity.Event;

import com.hexaware.controller.\*;

import com.hexaware.entity.Movie;

import com.hexaware.entity.Sports;

import com.hexaware.entity.Venue;

import com.hexaware.dao.\*;

public class BookingController extends Exception implements BookingInterface{

private Event event = new Event();

private Customer customer;

private Venue venue = new Venue();

private Scanner scanner = new Scanner(System.in);

private Bookingdao bookingdao = new Bookingdao();

public void addEvent() throws ParseException {

System.out.println("Enter the Event Id");

int eventId = scanner.nextInt();

System.out.println("Enter the Event name");

String eventName = scanner.next();

System.out.println("Enter Event date (dd/MM/yyyy): ");

String dateStr = scanner.next();

SimpleDateFormat sdf = new SimpleDateFormat("dd/MM/yyyy");

Date eventDate = sdf.parse(dateStr);

System.out.println("Enter Event time: ");

String eventTime = scanner.next();

LocalTime localTime = LocalTime.parse(eventTime);

System.out.println("Enter Venue Id: ");

int venueId = scanner.nextInt();

System.out.println("Enter the Event Type (MOVIE/CONCERT/SPORTS):");

String eventTypeStr = scanner.next();

Event.EventType eventType = Event.EventType.valueOf(eventTypeStr.toUpperCase());

switch (eventType) {

case MOVIE:

event = createMovie(eventId,eventName, eventDate, localTime,venueId);

break;

case CONCERT:

event = createConcert(eventId,eventName, eventDate, localTime,venueId);

break;

case SPORTS:

event = createSports(eventId,eventName, eventDate, localTime,venueId);

break;

default:

System.out.println("Unsupported event type: " + eventTypeStr);

return;

}

System.out.println("Enter total no. of seats: ");

int totalSeats = scanner.nextInt();

event.setTotalSeats(totalSeats);

System.out.println("Enter the Available seats: ");

int availableSeats = scanner.nextInt();

event.setAvailableSeats(availableSeats);

System.out.println("Enter the Ticket Price:");

double ticketPrice = scanner.nextDouble();

event.setTicketPrice(ticketPrice);

bookingdao.addEvent(event);

}

private Movie createMovie(int eventId,String eventName, Date eventDate, LocalTime eventTime,int venueId) {

System.out.println("Enter movie genre: ");

String genre = scanner.next();

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

String actorName = scanner.next();

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

String actressName = scanner.next();

return new Movie(eventId,eventName, eventDate, eventTime,venueId, 0, 0, Event.EventType.MOVIE, 0, genre, actorName, actressName);

}

private Concert createConcert(int eventId,String eventName, Date eventDate, LocalTime eventTime,int venueId) {

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

String artistName = scanner.next();

System.out.println("Enter music genre: ");

String musicGenre = scanner.next();

return new Concert(eventId,eventName, eventDate, eventTime,venueId, 0, 0, Event.EventType.CONCERT, 0, artistName, musicGenre);

}

private Sports createSports(int eventId,String eventName, Date eventDate, LocalTime eventTime,int venueId) {

System.out.println("Enter sport type: ");

String sportType = scanner.next();

System.out.println("Enter Teams Name: ");

String matchDescription = scanner.next();

scanner.nextLine();

return new Sports(eventId,eventName, eventDate, eventTime,venueId, 0, 0, Event.EventType.SPORTS, 0, sportType, matchDescription);

}

public Customer createCustomer() {

System.out.println("Enter Customer Name: ");

String customerName = scanner.next();

System.out.println("Enter Customer Email: ");

String email = scanner.next();

System.out.println("Enter Customer Phone Number: ");

String phoneNumber = scanner.next();

System.out.println("Enter Customer Id: ");

int customerId = scanner.nextInt();

Customer customer = new Customer(customerName, email, phoneNumber, customerId);

bookingdao.addCustomer(customer);

return customer;

}

public Venue createVenueDetails() {

System.out.println("Enter the Venue Name:");

String venueName = scanner.next();

System.out.println("Enter the Venue Address:");

String address = scanner.next();

System.out.println("Enter the Venue Id");

int venueId = scanner.nextInt();

scanner.nextLine();

Venue newVenue = new Venue(venueId, venueName, address);

System.out.println("Venue ID to be inserted: " + newVenue.getVenueId());

bookingdao.addVenue(newVenue);

return newVenue;

}

public void book\_tickets() {

try {

System.out.println("Enter the Event Id: ");

int eventId = scanner.nextInt();

// Check if the event with the given ID exists

if (!bookingdao.eventExists(eventId)) {

System.out.println("Event with ID " + eventId + " does not exist.");

return;

}

System.out.println("Enter the Customer Id: ");

int customerId = scanner.nextInt();

if (!bookingdao.customerExists(customerId)) {

System.out.println("Customer with ID " + customerId + " does not exist.");

return;

}

int availableTickets = bookingdao.getAvailableTickets(eventId);

System.out.println("Available Tickets for Event ID " + eventId + ": " + availableTickets);

System.out.println("Enter the no. of tickets to book: ");

int num = scanner.nextInt();

double booked = event.bookTickets(num,eventId);

double cost = event.calculateTotalRevenue(booked);

bookingdao.bookTickets(eventId,customerId,booked,cost);

} catch (Exception e) {

e.printStackTrace();

}

}

public void cancel\_booking() {

System.out.println("Enter the no. of tickets to be canceled : ");

int num = scanner.nextInt();

event.cancelBooking(num);

}

public int getAvailableNoOfTickets() {

return event.getAvailableSeats();

}

public void getEventDetails() {

event.printEventDetails();

}

private static int bookingIdCounter = 1;

private int bookingId;

private Customer[] customers;

private LocalDate bookingDate;

private double totalCost;

public BookingController() {

this.bookingId = bookingIdCounter++;

}

public BookingController(Event event, double totalCost, LocalDate bookingDate, Customer[] customers) {

this.bookingId = bookingIdCounter++;

this.event = event;

this.totalCost = totalCost;

this.bookingDate = bookingDate;

this.customers = customers;

}

public int getBookingId() {

return bookingId;

}

public void setBookingId(int bookingId) {

this.bookingId = bookingId;

}

public Customer[] getCustomers() {

return customers;

}

public void setCustomers(Customer[] customers) {

this.customers = customers;

}

public Event getEvent() {

return event;

}

public void setEvent(Event event) {

this.event = event;

}

public double getTotalCost() {

return totalCost;

}

public void setTotalCost(double totalCost) {

this.totalCost = totalCost;

}

public LocalDate getBookingDate() {

return bookingDate;

}

public void setBookingDate(LocalDate bookingDate) {

this.bookingDate = bookingDate;

}

public void displayBookingDetails() {

try {

System.out.println("Booking ID: " + bookingId);

System.out.println("Event Details:");

event.printEventDetails();

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

System.out.println("Booking Date: " + bookingDate);

System.out.println("Customer Details:");

if (customers != null) {

for (Customer customer : customers) {

System.out.println("Customer ID: " + customer.getCustomerId());

System.out.println("Customer Name: " + customer.getCustomerName());

System.out.println("Customer Email: " + customer.getEmail());

}

} else {

System.out.println("No customer information available for this booking.");

}

} catch (NullPointerException e) {

System.out.println("Error: An unexpected null pointer exception occurred.");

e.printStackTrace(); // Optionally, you can print more details about the exception if needed.

}

}

}

**TASK 11: Database Connectivity.**

**1. Create Venue, Event, Customer and Booking class as mentioned above Task 5.**

**2. Create Event sub classes as mentioned in above Task 4.**

**3. Create interface/abstract class IEventServiceProvider, IBookingSystemServiceProvider and its**

**implementation classes as mentioned in above Task 5.**

**4. Create IBookingSystemRepository interface/abstract class which include following methods to**

**interact with database.**

**• create\_event(event\_name: str, date:str, time:str, total\_seats: int, ticket\_price: float,**

**event\_type: str, venu: Venu): Create a new event with the specified details and event type**

**(movie, sport or concert) and return event object and should store in database.**

**• getEventDetails(): return array of event details from the database.**

**• getAvailableNoOfTickets(): return the total available tickets from the database.**

**• calculate\_booking\_cost(num\_tickets): Calculate and set the total cost of the booking.**

**• book\_tickets(eventname:str, num\_tickets, listOfCustomer): Book a specified number of**

**tickets for an event. for each tickets customer object should be created and stored in array**

**also should update the attributes of Booking class and stored in database.**

**• cancel\_booking(booking\_id): Cancel the booking and update the available seats and stored**

**in database.**

**• get\_booking\_details(booking\_id): get the booking details from database.**

**5. Create BookingSystemRepositoryImpl interface/abstract class which implements**

**IBookingSystemRepository interface/abstract class and provide implementation of all methods and**

**perform the database operations.**

**6. Create DBUtil class and add the following method.**

**• static getDBConn():Connection Establish a connection to the database and return**

**Connection reference**

**7. Place the interface/abstract class in service package and interface implementation class, concrete**

**class in bean package and TicketBookingSystemRepository class in app package.**

**8. Should throw appropriate exception as mentioned in above task along with handle SQLException.**

**9. Create TicketBookingSystem class and perform following operations:**

**• Create a simple user interface in a main method that allows users to interact with the ticket**

**booking system by entering commands such as "create\_event", "book\_tickets",**

**"cancel\_tickets", "get\_available\_seats,", "get\_event\_details," and "exit."**

**MyDBConnection :**

package com.hexaware.util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class MyDBConnection {

static Connection con;

public static Connection getMyDbConnection() {

try {

con=DriverManager.getConnection("jdbc:mysql://localhost:3306/TicketBookingSystem", "root", "mulio@05");

} catch (SQLException e) {

e.printStackTrace();

}

return con;

}

public static void main(String[] args) {

System.out.println(getMyDbConnection());

}

**Booking.dao**

**package com.hexaware.dao;**

import java.sql.Connection;

import java.sql.Date;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.time.LocalDate;

import com.hexaware.entity.Customer;

import com.hexaware.entity.Event;

import com.hexaware.entity.Venue;

import com.hexaware.util.MyDBConnection;

public class Bookingdao {

Connection connection;

Statement statement;

ResultSet rs;

public void addEvent(Event event) {

try {

connection = MyDBConnection.getMyDbConnection();

String query = "INSERT INTO event (event\_id, event\_name, event\_date, event\_time, venue\_id, total\_seats, available\_seats, ticket\_price, event\_type) " +

"VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";

try (PreparedStatement ps = connection.prepareStatement(query)) {

ps.setInt(1, event.getEventId());

ps.setString(2, event.getEventName());

java.sql.Date sqlEventDate = new java.sql.Date(event.getEventDate().getTime());

ps.setDate(3, sqlEventDate);

ps.setTime(4, java.sql.Time.valueOf(event.getEventTime()));

ps.setInt(5, event.getVenueId());

ps.setInt(6, event.getTotalSeats());

ps.setInt(7, event.getAvailableSeats());

ps.setDouble(8, event.getTicketPrice());

ps.setString(9, event.getEventType().name());

int rowsAffected = ps.executeUpdate();

if (rowsAffected > 0) {

System.out.println("Event added to the database successfully.");

} else {

System.out.println("Failed to add event to the database.");

}

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public void addVenue(Venue venue) {

try {

connection = MyDBConnection.getMyDbConnection();

String query = "INSERT INTO venue (venue\_id, venue\_name, address) " +

"VALUES (?, ?, ?)";

try (PreparedStatement ps = connection.prepareStatement(query)) {

ps.setInt(1, venue.getVenueId());

ps.setString(2, venue.getVenueName());

ps.setString(3, venue.getAddress());

int rowsAffected = ps.executeUpdate();

if (rowsAffected > 0) {

System.out.println("Venue added to the database successfully.");

} else {

System.out.println("Failed to add venue to the database.");

}

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public void addCustomer(Customer customer) {

try {

connection = MyDBConnection.getMyDbConnection();

String query = "INSERT INTO customer (customer\_name, email, phone\_number, customer\_id) " +

"VALUES (?, ?, ?, ?)";

try (PreparedStatement ps = connection.prepareStatement(query)) {

ps.setString(1, customer.getCustomerName());

ps.setString(2, customer.getEmail());

ps.setString(3, customer.getPhoneNumber());

ps.setInt(4, customer.getCustomerId());

int rowsAffected = ps.executeUpdate();

if (rowsAffected > 0) {

System.out.println("Customer added to the database successfully.");

} else {

System.out.println("Failed to add customer to the database.");

}

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public boolean eventExists(int eventId) {

boolean exists = false;

try {

Connection connection = MyDBConnection.getMyDbConnection();

String query = "SELECT \* FROM event WHERE event\_id = ?";

try (PreparedStatement ps = connection.prepareStatement(query)) {

ps.setInt(1, eventId);

try (ResultSet rs = ps.executeQuery()) {

if (rs.next()) {

exists = true;

}

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return exists;

}

public boolean customerExists(int customerId) {

boolean exists = false;

try {

Connection connection = MyDBConnection.getMyDbConnection();

String query = "SELECT \* FROM customer WHERE customer\_id = ?";

try (PreparedStatement ps = connection.prepareStatement(query)) {

ps.setInt(1, customerId);

try (ResultSet rs = ps.executeQuery()) {

if (rs.next()) {

exists = true;

}

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return exists;

}

public int getAvailableTickets(int eventId) {

try {

connection = MyDBConnection.getMyDbConnection();

String query = "SELECT available\_seats FROM event WHERE event\_id = ?";

try (PreparedStatement ps = connection.prepareStatement(query)) {

ps.setInt(1, eventId);

try (ResultSet rs = ps.executeQuery()) {

if (rs.next()) {

return rs.getInt("available\_seats");

}

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return 0;

}

public void bookTickets(int eventId, int customerId, double booked,double cost) {

try {

connection = MyDBConnection.getMyDbConnection();

int availableTickets = getAvailableTickets(eventId);

if (booked > 0 && booked <= availableTickets) {

int bookingId = generateUniqueBookingId();

LocalDate currentDate = LocalDate.now();

String bookingQuery = "INSERT INTO booking (booking\_id, event\_id, customer\_id, num\_tickets,total\_cost,booking\_date) VALUES (?, ?, ?, ?,?,?)";

try (PreparedStatement bookingPs = connection.prepareStatement(bookingQuery)) {

bookingPs.setInt(1, bookingId);

bookingPs.setInt(2, eventId);

bookingPs.setInt(3, customerId);

bookingPs.setDouble(4, booked);

bookingPs.setDouble(5, cost);

java.sql.Date sqlDate = java.sql.Date.valueOf(currentDate);

bookingPs.setDate(6, sqlDate);

bookingPs.executeUpdate();

}

String updateEventQuery = "UPDATE event SET available\_seats = available\_seats - ?, booking\_id = ? WHERE event\_id = ?";

try (PreparedStatement updateEventPs = connection.prepareStatement(updateEventQuery)) {

updateEventPs.setDouble(1, booked);

updateEventPs.setInt(2, bookingId);

updateEventPs.setInt(3, eventId);

updateEventPs.executeUpdate();

}

// Update booking ID in the customer table

String updateCustomerQuery = "UPDATE customer SET booking\_id = ? WHERE customer\_id = ?";

try (PreparedStatement updateCustomerPs = connection.prepareStatement(updateCustomerQuery)) {

updateCustomerPs.setInt(1, bookingId);

updateCustomerPs.setInt(2, customerId);

updateCustomerPs.executeUpdate();

}

// Display the unique booking ID

System.out.println("Booking ID: " + bookingId);

} else {

System.out.println("Invalid number of tickets to book or insufficient seats available.");

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private int generateUniqueBookingId() throws SQLException {

String maxBookingIdQuery = "SELECT MAX(booking\_id) FROM booking";

try (PreparedStatement maxBookingIdPs = connection.prepareStatement(maxBookingIdQuery);

ResultSet resultSet = maxBookingIdPs.executeQuery()) {

if (resultSet.next()) {

int maxBookingId = resultSet.getInt(1);

return maxBookingId + 1;

} else {

return 1000;

}

}

}

public void cancelTickets(int eventId, int customerId, int numTickets) {

try {

connection = MyDBConnection.getMyDbConnection();

int bookingId = getBookingIdForCustomer(customerId);

if (bookingId > 0) {

String updateEventQuery = "UPDATE event SET available\_seats = available\_seats + ?, booking\_id = NULL WHERE event\_id = ?";

try (PreparedStatement updateEventPs = connection.prepareStatement(updateEventQuery)) {

updateEventPs.setInt(1, numTickets);

updateEventPs.setInt(2, eventId);

updateEventPs.executeUpdate();

}

String deleteBookingQuery = "DELETE FROM booking WHERE booking\_id = ?";

try (PreparedStatement deleteBookingPs = connection.prepareStatement(deleteBookingQuery)) {

deleteBookingPs.setInt(1, bookingId);

deleteBookingPs.executeUpdate();

}

String updateCustomerQuery = "UPDATE customer SET booking\_id = NULL WHERE customer\_id = ?";

try (PreparedStatement updateCustomerPs = connection.prepareStatement(updateCustomerQuery)) {

updateCustomerPs.setInt(1, customerId);

updateCustomerPs.executeUpdate();

}

System.out.println("Booking canceled successfully!");

} else {

System.out.println("No valid booking found for the customer.");

}

} catch (SQLException e) {

e.printStackTrace();

}

}

private int getBookingIdForCustomer(int customerId) {

try {

String query = "SELECT booking\_id FROM customer WHERE customer\_id = ?";

try (PreparedStatement ps = connection.prepareStatement(query)) {

ps.setInt(1, customerId);

try (ResultSet rs = ps.executeQuery()) {

if (rs.next()) {

return rs.getInt("booking\_id");

}

}

}

} catch (SQLException e) {

e.printStackTrace();

}

return 0;

}

public void displayBookingDetails(int eventId, int customerId) {

try {

connection = MyDBConnection.getMyDbConnection();

String query = "SELECT \* FROM booking b JOIN event e ON b.event\_id = e.event\_id " +

"JOIN venue v ON e.venue\_id = v.venue\_id " +

"WHERE b.event\_id = ? AND b.customer\_id = ?";

try (PreparedStatement ps = connection.prepareStatement(query)) {

ps.setInt(1, eventId);

ps.setInt(2, customerId);

try (ResultSet rs = ps.executeQuery()) {

if (rs.next()) {

int bookingId = rs.getInt("booking\_id");

int numTickets = rs.getInt("num\_tickets");

double totalCost = rs.getDouble("total\_cost");

LocalDate bookingDate = rs.getDate("booking\_date").toLocalDate();

int venueId = rs.getInt("venue\_id");

String venueName = rs.getString("venue\_name");

String venueAddress = rs.getString("address");

System.out.println("Booking ID: " + bookingId);

System.out.println("Event ID: " + eventId);

System.out.println("Customer ID: " + customerId);

System.out.println("Number of Tickets: " + numTickets);

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

System.out.println("Booking Date: " + bookingDate);

System.out.println("Venue ID: " + venueId);

System.out.println("Venue Name: " + venueName);

System.out.println("Venue Address: " + venueAddress);

} else {

System.out.println("No booking found for the given event and customer.");

}

}

}

} catch (SQLException e) {

e.printStackTrace();

}

}

public Event getEventDetails(int eventId) {

Event event = null;

try {

connection = MyDBConnection.getMyDbConnection();

String query = "SELECT \* FROM event WHERE event\_id = ?";

try (PreparedStatement ps = connection.prepareStatement(query)) {

ps.setInt(1, eventId);

try (ResultSet rs = ps.executeQuery()) {

if (rs.next()) {

event = new Event();

event.setEventId(rs.getInt("event\_id"));

event.setEventName(rs.getString("event\_name"));

event.setEventDate(rs.getDate("event\_date"));

event.setEventTime(rs.getTime("event\_time").toLocalTime());

event.setVenueId(rs.getInt("venue\_id"));

event.setTotalSeats(rs.getInt("total\_seats"));

event.setAvailableSeats(rs.getInt("available\_seats"));

event.setTicketPrice(rs.getDouble("ticket\_price"));

event.setEventType(Event.EventType.valueOf(rs.getString("event\_type").toUpperCase()));

}

}

}

} catch (SQLException e) {

e.printStackTrace();

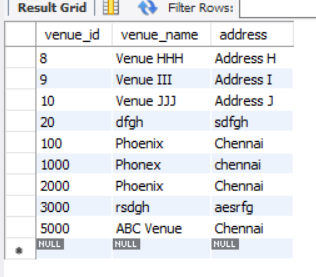
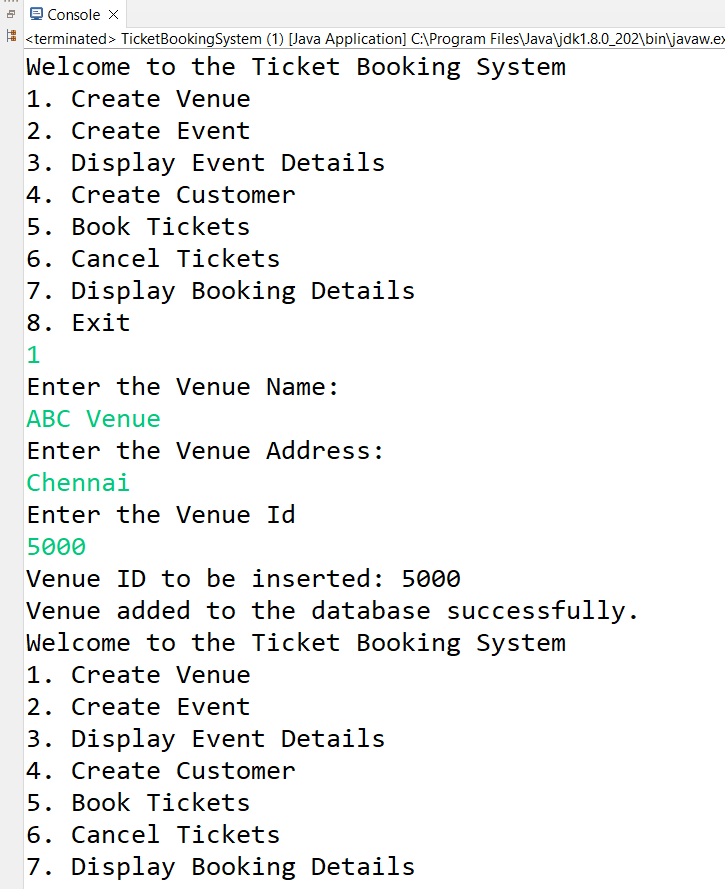
}

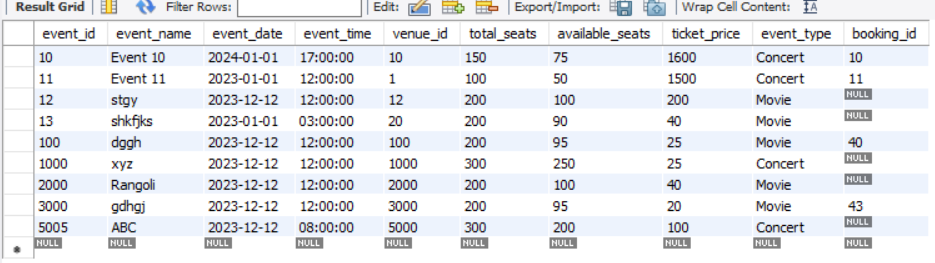
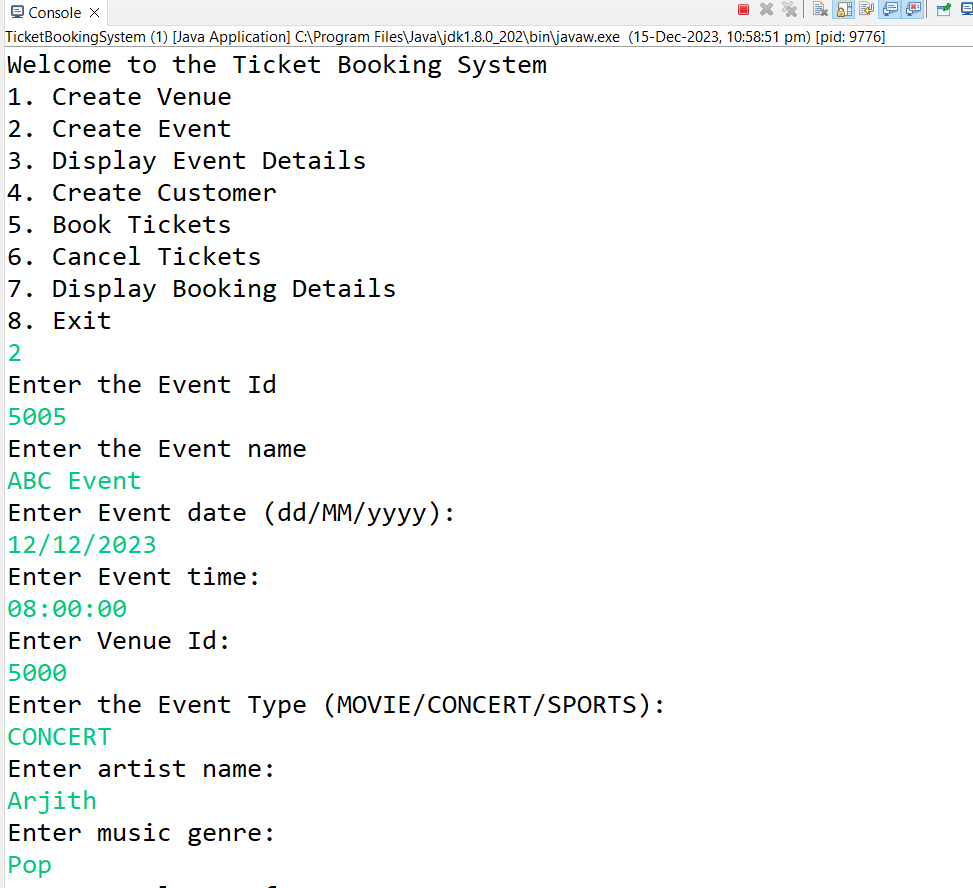
return event;

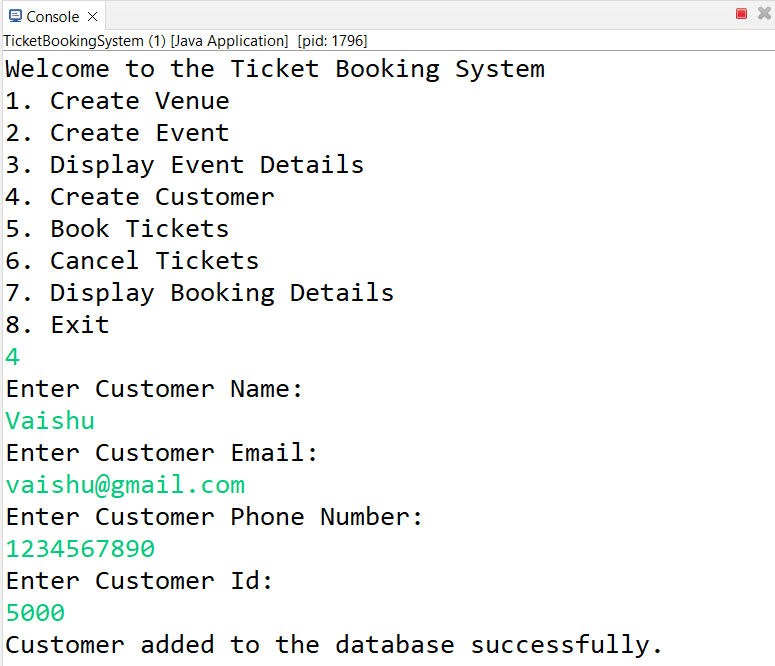
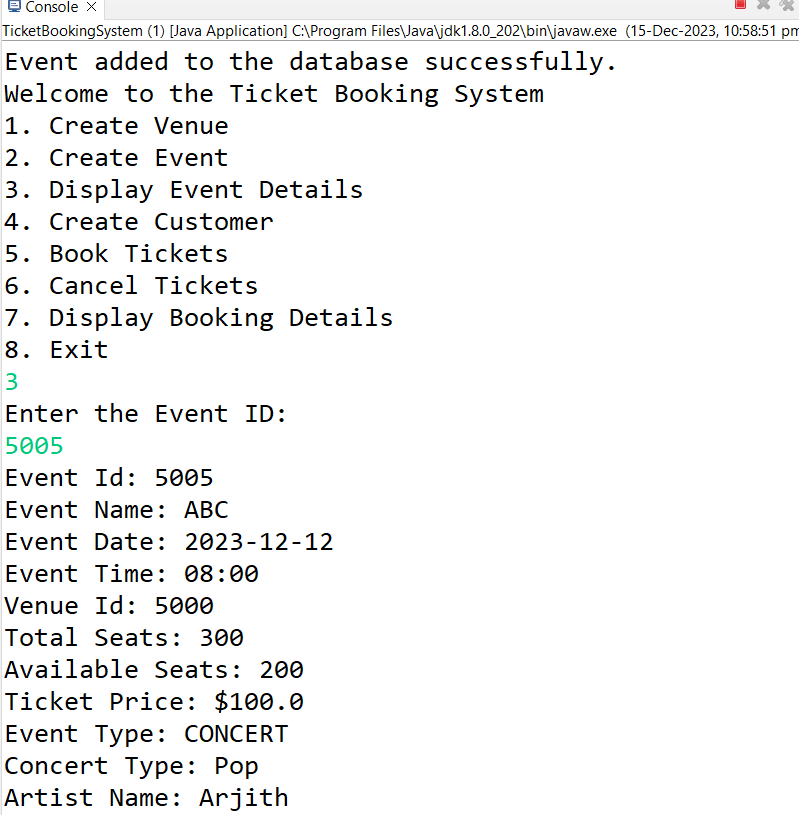
}

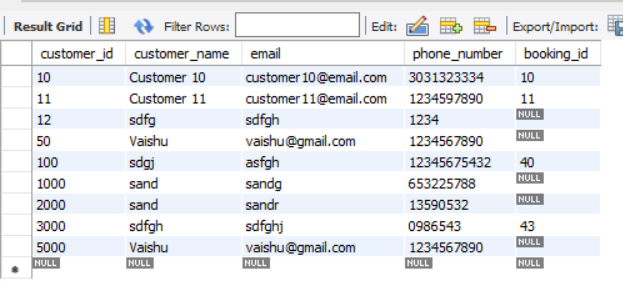
}

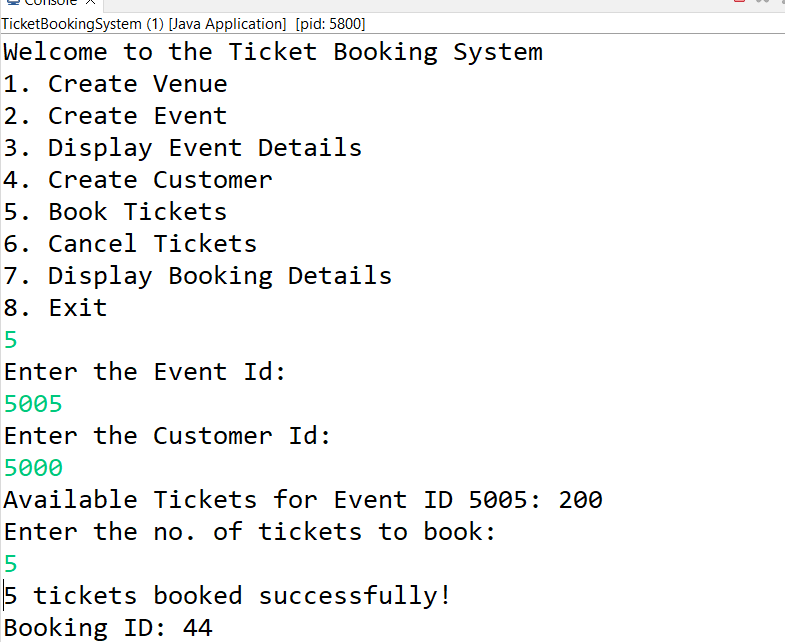
**OUTPUT :**

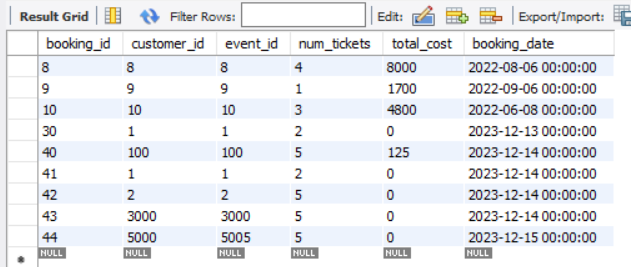
****

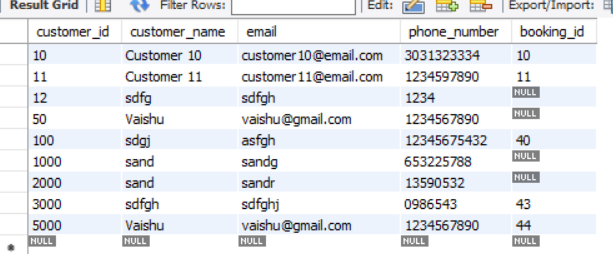
****

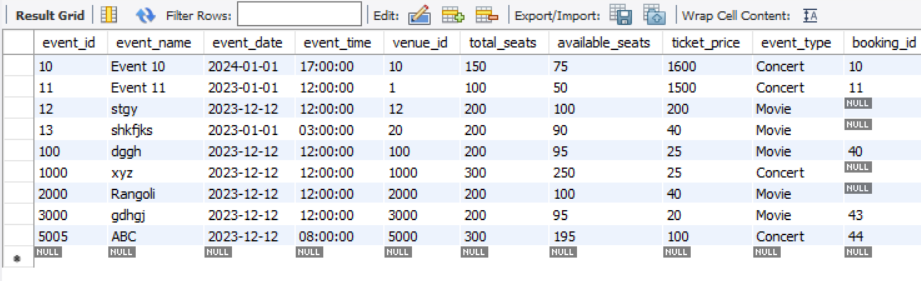
****

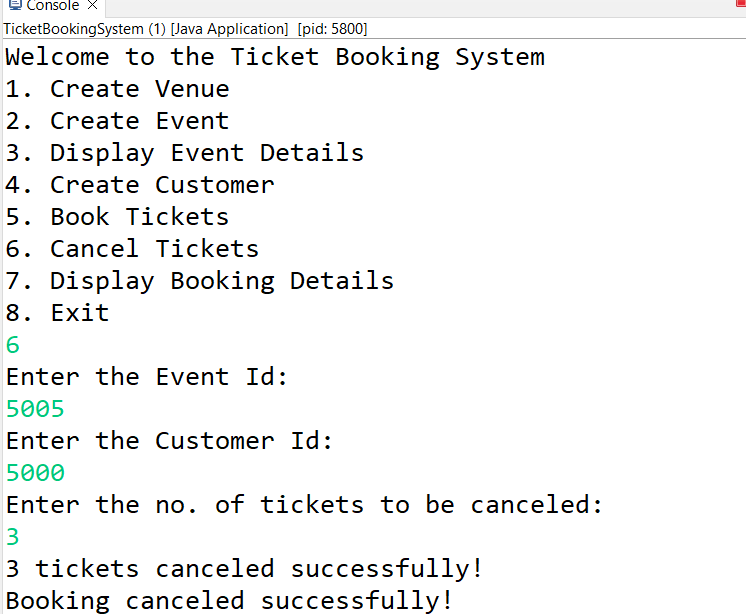
****

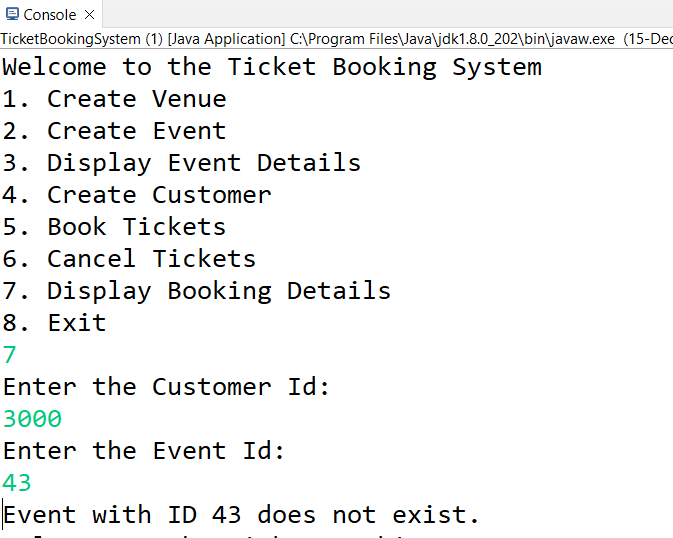
****

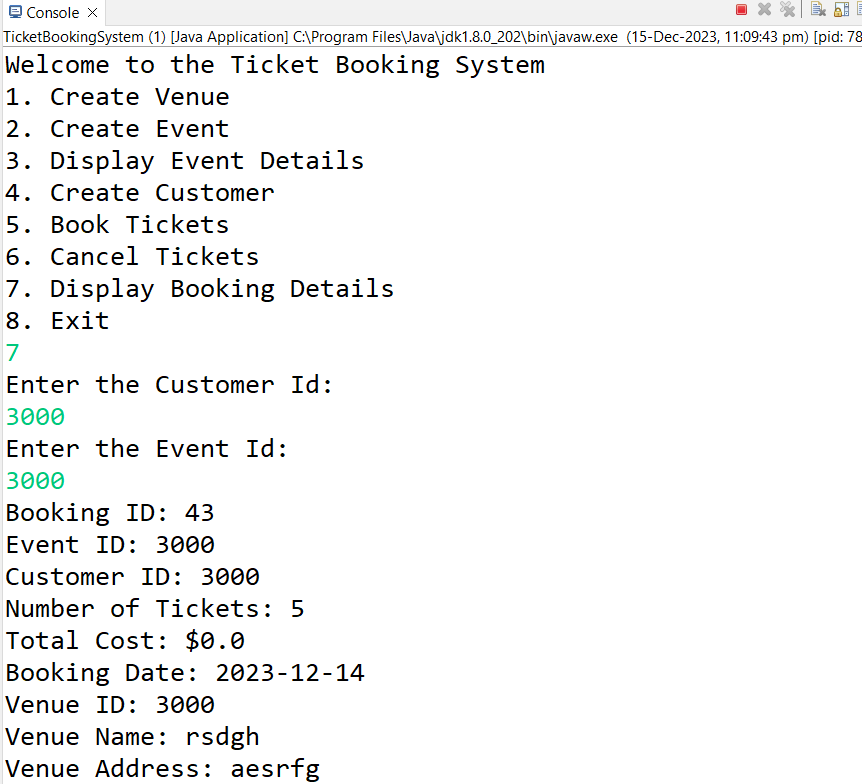
****

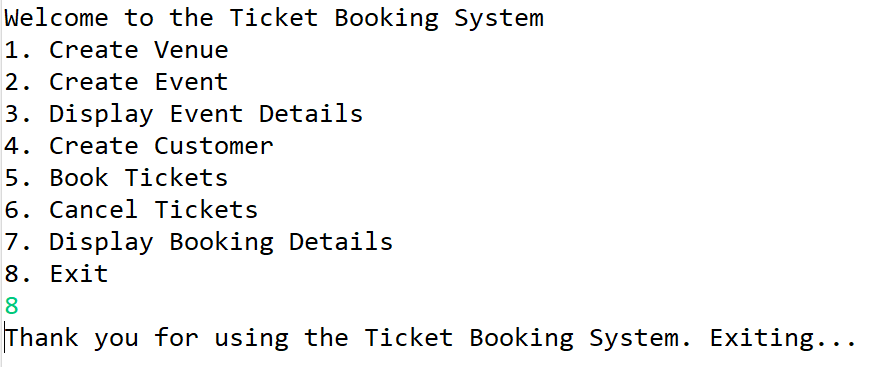
****

****

****

****

****

****