Ticket Booking System Java

Name: Samiksha Sandeep Patil

Superset Id: 5273555

Github Link: https://github.com/samikshapatil07/TicketBookingSystem

JAVA

Task 1:

Conditional Statements

Create a BookingSystem that accepts availableTicket and noOfBookingTicket as input
Use if-else to check if booking is possible
If yes, display "Booking confirmed, remaining tickets: X"
Else, display "Tickets unavailable"

TicketBookingSystem.java

```
package ticketbooking;
import java.util.Scanner;

public class TicketBookingSystem {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.print("Enter number of available tickets: ");
        int availableTickets = scanner.nextInt();

        System.out.print("Enter number of tickets to book: ");
        int noOfBookingTicket = scanner.nextInt();

        if (availableTickets >= noOfBookingTicket) {
            int remainingTickets = availableTickets - noOfBookingTicket;
            System.out.println("Booking Successful!");
            System.out.println("Remaining Tickets: " + remainingTickets);
        } else {
            System.out.println("Ticket Unavailable! Not enough tickets to fulfill your booking.");
        }

        scanner.close();
    }
}
```

Task 2:

Nested Conditional Statements

Simulate ticket booking with category-based pricing
Show ticket options: Silver, Gold, Diamond (e.g., 300, 500, 800)
Ask user for ticket type and number of tickets
Use nested if-else to assign price and calculate total cost
Display ticket type, quantity, and final price.

TicketBookingNested.java

```
package ticketbookingested;
```

```
import java.util.Scanner;

public class TicketBookingNested {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        System.out.println("Available Ticket Categories:");
        System.out.println("1. Silver - ₹300");
        System.out.println("2. Gold   - ₹500");
        System.out.println("3. Diamond - ₹800");

        System.out.print("Enter ticket category (Silver/Gold/Diamond): ");
        String ticketType = scanner.nextLine().toLowerCase();

        System.out.print("Enter number of tickets: ");
        int numberOfTickets = scanner.nextInt();

        int basePrice = 0;

        if (numberOfTickets > 0) {
            if (ticketType.equals("silver")) {
                basePrice = 300;
            } else if (ticketType.equals("gold")) {
                basePrice = 500;
            } else if (ticketType.equals("diamond")) {
                basePrice = 800;
            } else {
                System.out.println("Invalid ticket category selected!");
                return;
            }

            int totalCost = basePrice * numberOfTickets;
            System.out.println("Total cost for " + numberOfTickets + " " + ticketType + " ticket(s): ₹" + totalCost);
        } else {
            System.out.println("Invalid number of tickets.");
        }

        scanner.close();
    }
}
```

Task 3:

Looping

Repeat ticket booking process in a loop
Ask user for input until they type "Exit"
Book tickets and show confirmation inside the loop
Use while or do-while loop to control flow.

TicketBookingWithLoop.java

```
package entity;

import java.util.Scanner;

public class TicketBookingWithLoop {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String userChoice;

        System.out.println("Welcome to the Ticket Booking System!");
        System.out.println("Available Ticket Categories:");
        System.out.println("1. Silver - ₹300");
        System.out.println("2. Gold   - ₹500");
        System.out.println("3. Diamond - ₹800");

        while (true) {
            System.out.print("\nEnter ticket category (Silver/Gold/Diamond) or type 'exit' to quit: ");
            userChoice = scanner.nextLine().toLowerCase();

            if (userChoice.equals("exit")) {
                System.out.println("Thank you for using the Ticket Booking System!");
                break;
            }

            int basePrice;

            if (userChoice.equals("silver")) {
                basePrice = 300;
            } else if (userChoice.equals("gold")) {
                basePrice = 500;
            } else if (userChoice.equals("diamond")) {
                basePrice = 800;
            } else {
                System.out.println("Invalid ticket category. Please try again.");
                continue;
            }

            System.out.print("Enter number of tickets to book: ");
            int numberOfTickets;

            try {
                numberOfTickets = Integer.parseInt(scanner.nextLine());

                if (numberOfTickets <= 0) {
                    System.out.println("Number of tickets must be greater than 0.");
                    continue;
                }

                int totalCost = basePrice * numberOfTickets;
                System.out.println("Booking Successful!");
                System.out.println("Total cost for " + numberOfTickets + " " + userChoice + " ticket(s): ₹" + totalCost);

            } catch (NumberFormatException e) {
                System.out.println("Invalid input for number of tickets. Please enter a number.");
            }
        }

        scanner.close();
    }
}
```

Task 4:

Class & Object

Create Event, Venue, Customer, Booking classes
Define attributes and methods (constructors, getters, setters)
Implement functionality: calculate_total_revenue(), book_tickets(), cancel_booking(), display_event_details()
Use objects of these classes to simulate real-world ticket booking.

Package: /client

BookingSystem .java

```
package client;package client;

import entity.Event;
import entity.Customer;
import entity.Venue;

import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;

public class Booking {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        Venue venue = new Venue("Stadium Arena", "123 Main Street");

        Event event = new Event("Rock Concert", LocalDate.of(2025, 6, 15), LocalTime.of(19, 30), venue.getVenueN

        System.out.print("Enter Your Name: ");
        String name = sc.nextLine();
        System.out.print("Enter Your Email: ");
        String email = sc.nextLine();
        System.out.print("Enter Your Phone Number: ");
        String phone = sc.nextLine();

        Customer customer = new Customer(name, email, phone);

        // Display details
        venue.displayVenueDetails();
        event.displayEventDetails();
        customer.displayCustomerDetails();

        // Book tickets
        System.out.print("\nEnter Number of Tickets to Book: ");
        int numTickets = sc.nextInt();
        event.bookTickets(numTickets);
        event.displayEventDetails();

        // Canceling Tickets
        System.out.print("\nEnter number of tickets to cancel: ");
        int cancelTickets = sc.nextInt();
        event.cancelBooking(cancelTickets);
        event.displayEventDetails();

        // Display revenue
        System.out.println("\nTotal Revenue Generated: Rs." + event.calculateTotalRevenue());

        sc.close();
    }
```

```java
}

import java.util.ArrayList;
import java.util.Scanner;

import entity.BookingSystem;
import entity.Concert;
import entity.Event;
import entity.Movie;
import entity.Sports;

public class TicketBookingSystem extends BookingSystem{

    private final ArrayList<Event> events = new ArrayList<>();
    static TicketBookingSystem system = new TicketBookingSystem();
    static Scanner scan = new Scanner(System.in);

    public static void main(String[] args) {
        String opt;
        do {
            handleMenu();
            System.out.println("Press 'y' to continue:");
            opt = scan.next();
        } while (opt.charAt(0) == 'Y' || opt.charAt(0) == 'y');

        System.out.println("Thank you for using the system!");
    }

    public static void handleMenu() {
        System.out.println("\n--- Ticket Booking System Menu ---");
        System.out.println("1. Create Event");
        System.out.println("2. Book Tickets");
        System.out.println("3. Cancel Tickets");
        System.out.println("4. Get Available Seats");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");

        int menuOpt = scan.nextInt();
        scan.nextLine();
        switch (menuOpt) {
            case 1:
                system.createEvent();
                break;
            case 2:
                system.bookTickets();
                break;
            case 3:
                system.cancelTickets();
                break;
            case 4:
                system.getAvailableSeats();
            case 5:
                system.exit();
                break;
            default:
                System.out.println("Invalid menu option.");
        }
    }

    @Override
```

```
public void createEvent() {
    System.out.println("Enter event type (Movie, Concert, Sports): ");
    String type = scan.nextLine();

    System.out.println("Event Name: ");
    String name = scan.nextLine();
    System.out.println("Date (YYYY-MM-DD): ");
    String date = scan.nextLine();
    System.out.println("Time (HH:MM): ");
    String time = scan.nextLine();
    System.out.println("Venue Name: ");
    String venue = scan.nextLine();
    System.out.println("Total Seats: ");
    int seats = scan.nextInt();
    System.out.println("Ticket Price: ");
    double price = scan.nextDouble();
    scan.nextLine();

    switch (type.toLowerCase()) {
        case "movie":
            System.out.println("Genre: ");
            String genre = scan.nextLine();
            System.out.println("Actor Name: ");
            String actor = scan.nextLine();
            System.out.println("Actress Name: ");
            String actress = scan.nextLine();
            events.add(new Movie(name, date, time, venue, seats, price, genre, actor, actress));
            break;
        case "concert":
            System.out.println("Artist: ");
            String artist = scan.nextLine();
            System.out.println("Type: ");
            String concertType = scan.nextLine();
            events.add(new Concert(name, date, time, venue, seats, price, artist, concertType));
            break;
        case "sports":
            System.out.println("Sport Name: ");
            String sport = scan.nextLine();
            System.out.println("Teams (Team1 vs Team2): ");
            String teams = scan.nextLine();
            events.add(new Sports(name, date, time, venue, seats, price, sport, teams));
            break;
        default:
            System.out.println("Invalid event type.");
    }
}

@Override
public void bookTickets() {
    displayAllEvents();
    System.out.println("Select event index to book tickets: ");
    int index = scan.nextInt();
    System.out.println("Enter number of tickets: ");
    int num = scan.nextInt();
    events.get(index).bookTickets(num);
}

@Override
public void cancelTickets() {
    displayAllEvents();
```

```
        System.out.println("Select event index to cancel tickets: ");
        int index = scan.nextInt();
        System.out.println("Enter number of tickets: ");
        int num = scan.nextInt();
        events.get(index).cancelTickets(num);
    }

    @Override
    public void getAvailableSeats() {
        displayAllEvents();
    }

    public void displayAllEvents() {
        for (int i = 0; i < events.size(); i++) {
            System.out.println("[" + i + "]");
            events.get(i).displayEventDetails();
            System.out.println("---------------");
        }
    }

    public void exit() {
        System.out.println("Exiting the system.");
    }
}
```

Package: /entity

Customer.java

```
package entity;

public class Customer {

    private String customerName;
    private String email;
    private String phoneNumber;

    public Customer() {
        this.customerName = "Unknown";
        this.email = "Unknown";
        this.phoneNumber = "Unknown";
    }

    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;
    }

    // Method to display customer details
    public void displayCustomerDetails() {
        System.out.println("Customer Name: " + customerName);
        System.out.println("Email: " + email);
        System.out.println("Phone Number: " + phoneNumber);
    }
}
```

Event.java

```
package entity;

import java.time.LocalDate;
import java.time.LocalTime;

public class Event {

    private String eventName;
    private LocalDate eventDate;
    private LocalTime eventTime;
    private String venueName;
    private int totalSeats;
    private int availableSeats;
    private double ticketPrice;
    private String eventType;

    public Event() {
        this.eventName = "Unknown Event";
        this.eventDate = LocalDate.now();
        this.eventTime = LocalTime.now();
        this.venueName = "Unknown Venue";
        this.totalSeats = 0;
        this.availableSeats = 0;
        this.ticketPrice = 0.0;
        this.eventType = "Movie";
    }

    public Event(String eventName, LocalDate eventDate, LocalTime eventTime, String venueName, int totalSeats,
        this.eventName = eventName;
        this.eventDate = eventDate;
        this.eventTime = eventTime;
        this.venueName = venueName;
        this.totalSeats = totalSeats;
        this.availableSeats = totalSeats;
        this.ticketPrice = ticketPrice;
        this.eventType = eventType;
    }
```

```
public String getEventName() {
    return eventName;
}

public void setEventName(String eventName) {
    this.eventName = eventName;
}

public LocalDate getEventDate() {
    return eventDate;
}

public void setEventDate(LocalDate 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 String getEventType() {
    return eventType;
}
```

```
    public void setEventType(String eventType) {
        this.eventType = eventType;
    }

    // Method to get booked tickets
    public int getBookedNoOfTickets() {
        return totalSeats - availableSeats;
    }

 // Method to calculate total revenue
    public double calculateTotalRevenue() {
        return getBookedNoOfTickets() * ticketPrice;
    }

    // Method to book tickets
    public boolean bookTickets(int numTickets) {
        if (numTickets <= availableSeats) {
            availableSeats -= numTickets;
            System.out.println(numTickets + " tickets booked successfully!");
            return true;
        } else {
            System.out.println("Not enough tickets available.");
            return false;
        }
    }

    // Method to cancel booking
    public void cancelBooking(int numTickets) {
        if ((availableSeats + numTickets) <= totalSeats) {
            availableSeats += numTickets;
            System.out.println(numTickets + " tickets canceled successfully!");
        } else {
            System.out.println("Invalid cancellation request.");
        }
    }

    public void displayEventDetails() {
        System.out.println("\nEvent Details:");
        System.out.println("Event Name: " + eventName);
        System.out.println("Date: " + eventDate);
        System.out.println("Time: " + eventTime);
        System.out.println("Venue: " + venueName);
        System.out.println("Total Seats: " + totalSeats);
        System.out.println("Available Seats: " + availableSeats);
        System.out.println("Ticket Price: Rs." + ticketPrice);
        System.out.println("Event Type: " + eventType);
    }


}
```

Venue.java

```
package entity;

public class Venue {

    private String venueName;
    private String address;
    public Venue() {
        this.venueName = "Unknown Venue";
        this.address = "Unknown Address";
    }

    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;
    }

    // Display venue details
    public void displayVenueDetails() {
        System.out.println("\nVenue Details:");
        System.out.println("Venue Name: " + venueName);
        System.out.println("Address: " + address);
    }

}
```

Task 5:

Inheritance and Polymorphism

Create subclasses: Movie, Concert, Sports extending Event
Add specific attributes (e.g., genre, artist, teams)
Override display_event_details() in each subclass
Use polymorphism to call display methods on base class reference
Create TicketBookingSystem class with main menu-driven logic for booking and displaying events

Package: /app

TicketBookingSystem .java

```
package app;
import bean.*;
import java.util.Scanner;

import java.time.LocalDate;
import java.time.LocalTime;

public class TicketBookingSystem {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        Event movie = new Movie("Avengers: Endgame", LocalDate.of(2025, 5, 10), LocalTime.of(18, 30),
                            "Cineplex", 100, 15.0, "Action", "Robert Downey Jr.", "Scarlett Johansson");

        Event concert = new Concert("Coldplay Live", LocalDate.of(2025, 6, 20), LocalTime.of(20, 00),
                              "Stadium X", 200, 50.0, "Coldplay", "Rock");

        Event sports = new Sports("Football Match", LocalDate.of(2025, 7, 5), LocalTime.of(19, 00),
                             "National Stadium", 150, 30.0, "Football", "Brazil vs Argentina");

        while (true) {
            System.out.println("\nTICKET BOOKING SYSTEM");
            System.out.println(" 1 .View Events");
            System.out.println(" 2 .Book Tickets");
            System.out.println(" 3 .Cancel Tickets");
            System.out.println(" 4 .Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            if (choice == 1) {
                movie.display_event_details();
                concert.display_event_details();
                sports.display_event_details();
            } else if (choice == 2) {
                System.out.print("Enter event type (Movie/Concert/Sports): ");
                scanner.nextLine();
                String type = scanner.nextLine();
                System.out.print("Enter number of tickets: ");
                int numTickets = scanner.nextInt();

                if (type.equalsIgnoreCase("Movie")) movie.bookTickets(numTickets);
                else if (type.equalsIgnoreCase("Concert")) concert.bookTickets(numTickets);
                else if (type.equalsIgnoreCase("Sports")) sports.bookTickets(numTickets);
            } else if (choice == 3) {
                System.out.print("Enter number of tickets to cancel: ");
                int numTickets = scanner.nextInt();
                movie.cancelBooking(numTickets);
            } else if (choice == 4) {
                System.out.println("Exiting system...");
                break;
            }
        }
        scanner.close();
    }

}
```

Package: /bean

Booking .java

```
package bean;

import java.util.Scanner;
import java.time.LocalDate;
import java.time.LocalTime;

public class Booking {
    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        Event movie = new Movie("Avengers: Endgame", LocalDate.of(2025, 5, 10), LocalTime.of(18, 30),
                            "Cineplex", 100, 15.0, "Action", "Robert Downey Jr.", "Scarlett Johansson");

        Event concert = new Concert("Coldplay Live", LocalDate.of(2025, 6, 20), LocalTime.of(20, 00),
                              "Stadium X", 200, 50.0, "Coldplay", "Rock");

        Event sports = new Sports("Football Match", LocalDate.of(2025, 7, 5), LocalTime.of(19, 00),
                             "National Stadium", 150, 30.0, "Football", "Brazil vs Argentina");

        while (true) {
            System.out.println("\nTICKET BOOKING SYSTEM");
            System.out.println(" 1 .View Events");
            System.out.println(" 2 .Book Tickets");
            System.out.println(" 3 .Cancel Tickets");
            System.out.println(" 4 .Exit");
            System.out.print("Enter your choice: ");
            int choice = scanner.nextInt();

            if (choice == 1) {
                movie.display_event_details();
                concert.display_event_details();
                sports.display_event_details();
            } else if (choice == 2) {
                System.out.print("Enter event type (Movie/Concert/Sports): ");
                scanner.nextLine();
                String type = scanner.nextLine();
                System.out.print("Enter number of tickets: ");
                int numTickets = scanner.nextInt();

                if (type.equalsIgnoreCase("Movie")) movie.bookTickets(numTickets);
                else if (type.equalsIgnoreCase("Concert")) concert.bookTickets(numTickets);
                else if (type.equalsIgnoreCase("Sports")) sports.bookTickets(numTickets);
            } else if (choice == 3) {
                System.out.print("Enter number of tickets to cancel: ");
                int numTickets = scanner.nextInt();
                movie.cancelBooking(numTickets);
            } else if (choice == 4) {
                System.out.println("Application Terminated..");
                break;
            }
        }
        scanner.close();
    }

}
```

Concertjava

```
package bean;

import java.time.LocalDate;
import java.time.LocalTime;

public class Concert extends Event{
    private String artist;
    private String type;

    public Concert() {
        super();
    }

    public Concert(String eventName, LocalDate eventDate, LocalTime eventTime, String venueName,
                   int totalSeats, double ticketPrice, String artist, String type) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, ticketPrice, "Concert");
        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 display_event_details() {
        super.display_event_details();
        System.out.println("Artist: " + artist);
        System.out.println("Type: " + type);
        System.out.println("-----------------------------");
    }
}
```

Event.java

```
package bean;

import java.time.LocalDate;
import java.time.LocalTime;

public class Event {

    private String eventName;
    private LocalDate eventDate;
    private LocalTime eventTime;
    private String venueName;
    private int totalSeats;
    private int availableSeats;
    private double ticketPrice;
    private String eventType;

    public Event() {
        super();
    }
```

```
public Event(String eventName, LocalDate eventDate, LocalTime eventTime, String venueName,
        int totalSeats, double ticketPrice, String eventType) {
    this.eventName = eventName;
    this.eventDate = eventDate;
    this.eventTime = eventTime;
    this.venueName = venueName;
    this.totalSeats = totalSeats;
    this.availableSeats = totalSeats; // Initially, all seats are available
    this.ticketPrice = ticketPrice;
    this.eventType = eventType;
}


public String getEventName() {
    return eventName;
}

public void setEventName(String eventName) {
    this.eventName = eventName;
}

public LocalDate getEventDate() {
    return eventDate;
}

public void setEventDate(LocalDate 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 String getEventType() {
    return eventType;
}

public void setEventType(String eventType) {
    this.eventType = eventType;
}

// Book Tickets
public boolean bookTickets(int numTickets) {
    if (numTickets > availableSeats) {
        System.out.println("Not enough tickets available!");
        return false;
    }
    availableSeats -= numTickets;
    System.out.println(numTickets + " tickets booked successfully!");
    return true;
}

// Calculate Total Revenue
public double calculate_total_revenue() {
    int bookedTickets = getBookedNoOfTickets();
    return bookedTickets * ticketPrice;
}

// Get Number of Booked Tickets
public int getBookedNoOfTickets() {
    return totalSeats - availableSeats;
}

// Cancel Booking
public void cancelBooking(int numTickets) {
    if (numTickets > (totalSeats - availableSeats)) {
        System.out.println("Cannot cancel more tickets than booked!");
        return;
    }
    availableSeats += numTickets;
    System.out.println(numTickets + " tickets canceled successfully!");
}

// Display Event Details
public void display_event_details() {
    System.out.println("Event name: " + eventName );
    System.out.println("Event Type: " + eventType);
    System.out.println("Date: " + eventDate + " | Time: " + eventTime);
    System.out.println("Venue: " + venueName);
    System.out.println("Available Seats: " + availableSeats );
    System.out.println("Ticket Price:Rs." + ticketPrice);
    System.out.println("----------------------------");
    }
}
```

Movie .java

```
package bean;

import java.time.LocalDate;
import java.time.LocalTime;

public class Movie extends Event{

    private String genre;
    private String actorName;
    private String actressName;

    public Movie(String eventName, LocalDate eventDate, LocalTime eventTime, String venueName,
                 int totalSeats, double ticketPrice, String genre, String actorName, String actressName) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, ticketPrice, "Movie");
        this.genre = genre;
        this.actorName = actorName;
        this.actressName = actressName;
    }

    @Override
    public void display_event_details() {
        super.display_event_details();
        System.out.println("Genre: " + genre );
        System.out.println("Actor: " + actorName);
        System.out.println("Actress: " + actressName);
        System.out.println("----------------------------");
    }

}
```

Event .java

```
package bean;
import java.time.LocalDate;
import java.time.LocalTime;

public class Sports extends Event{

    private String sportName;
    private String teamsName;

    public Sports(String eventName, LocalDate eventDate, LocalTime eventTime, String venueName,
                  int totalSeats, double ticketPrice, String sportName, String teamsName) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, ticketPrice, "Sports");
        this.sportName = sportName;
        this.teamsName = teamsName;
    }

    @Override
    public void display_event_details() {
        super.display_event_details();
        System.out.println("Sport: " + sportName);
        System.out.println("Team: " + teamsName);
        System.out.println("----------------------------");
    }
}
```

Task 6:

Abstraction

Create abstract class Event and BookingSystem
Implement abstract methods in Movie, Concert, Sports (event types)
Implement create_event(), book_tickets(), cancel_tickets() in TicketBookingSystem
Allow user interaction using commands like "create_event", "book_tickets", etc.

Package: /client

TicketBookingSystem .java

```
package client;
import java.util.ArrayList;
import java.util.Scanner;

import entity.BookingSystem;
import entity.Concert;
import entity.Event;
import entity.Movie;
import entity.Sports;

public class TicketBookingSystem extends BookingSystem{

    private final ArrayList<Event> events = new ArrayList<>();
    static TicketBookingSystem system = new TicketBookingSystem();
    static Scanner scan = new Scanner(System.in);

    public static void main(String[] args) {
        String opt;
        do {
            handleMenu();
            System.out.println("Press 'y' to continue:");
            opt = scan.next();
        } while (opt.charAt(0) == 'Y' || opt.charAt(0) == 'y');

        System.out.println("Thank you for using the system!");
    }

    public static void handleMenu() {
        System.out.println("\n--- Ticket Booking System Menu ---");
        System.out.println("1. Create Event");
        System.out.println("2. Book Tickets");
        System.out.println("3. Cancel Tickets");
        System.out.println("4. Get Available Seats");
        System.out.println("5. Exit");
        System.out.print("Enter your choice: ");

        int menuOpt = scan.nextInt();
        scan.nextLine();
        switch (menuOpt) {
            case 1:
                system.createEvent();
                break;
            case 2:
                system.bookTickets();
                break;
            case 3:
                system.cancelTickets();
                break;
            case 4:
                system.getAvailableSeats();
            case 5:
                system.exit();
                break;
            default:
                System.out.println("Invalid menu option.");
        }
    }

    @Override
    public void createEvent() {
        System.out.println("Enter event type (Movie, Concert, Sports): ");
        String type = scan.nextLine();

        System.out.println("Event Name: ");
        String name = scan.nextLine();
        System.out.println("Date (YYYY-MM-DD): ");
        String date = scan.nextLine();
        System.out.println("Time (HH:MM): ");
        String time = scan.nextLine();
        System.out.println("Venue Name: ");
        String venue = scan.nextLine();
        System.out.println("Total Seats: ");
        int seats = scan.nextInt();
        System.out.println("Ticket Price: ");
        double price = scan.nextDouble();
        scan.nextLine();

        switch (type.toLowerCase()) {
            case "movie":
                System.out.println("Genre: ");
                String genre = scan.nextLine();
                System.out.println("Actor Name: ");
                String actor = scan.nextLine();
                System.out.println("Actress Name: ");
                String actress = scan.nextLine();
                events.add(new Movie(name, date, time, venue, seats, price, genre, actor, actress));
                break;
            case "concert":
                System.out.println("Artist: ");
                String artist = scan.nextLine();
                System.out.println("Type: ");
                String concertType = scan.nextLine();
                events.add(new Concert(name, date, time, venue, seats, price, artist, concertType));
                break;
            case "sports":
                System.out.println("Sport Name: ");
                String sport = scan.nextLine();
                System.out.println("Teams (Team1 vs Team2): ");
                String teams = scan.nextLine();
                events.add(new Sports(name, date, time, venue, seats, price, sport, teams));
                break;
            default:
                System.out.println("Invalid event type.");
        }
    }

    @Override
    public void bookTickets() {
        displayAllEvents();
        System.out.println("Select event index to book tickets: ");
        int index = scan.nextInt();
        System.out.println("Enter number of tickets: ");
        int num = scan.nextInt();
        events.get(index).bookTickets(num);
    }

    @Override
    public void cancelTickets() {
        displayAllEvents();
        System.out.println("Select event index to cancel tickets: ");
        int index = scan.nextInt();
        System.out.println("Enter number of tickets: ");
        int num = scan.nextInt();
        events.get(index).cancelTickets(num);
    }

    @Override
    public void getAvailableSeats() {
        displayAllEvents();
    }

    public void displayAllEvents() {
        for (int i = 0; i < events.size(); i++) {
            System.out.println("[" + i + "]");
            events.get(i).displayEventDetails();
            System.out.println("---------------");
        }
    }

    public void exit() {
        System.out.println("Exiting the system.");
    }
}
```

Package: /client

BookingSystem .java

```
package entity;

public abstract class BookingSystem {

     public abstract void createEvent();
     public abstract void bookTickets();
     public abstract void cancelTickets();
     public abstract void getAvailableSeats();

}
```

Concert.java

```
package entity;

public class Concert extends Event{
    private String artist;
    private String type;

    public Concert(String eventName, String eventDate, String eventTime, String venueName, int totalSeats, doub
                   String artist, String type) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, ticketPrice, "Concert");
        this.artist = artist;
        this.type = type;
    }

    @Override
    public void displayEventDetails() {
        System.out.println("Concert: " + eventName);
        System.out.println("Artist: " + artist );
        System.out.println("Type: " + type);
        System.out.println("Date: " + eventDate);
        System.out.println("Time: " + eventTime);
        System.out.println("Venue: " + venueName);
        System.out.println("Available Seats: " + availableSeats);
        System.out.println("Ticket Price: " + ticketPrice);
    }

}
```

Event.java

```
package entity;

public abstract class Event {

    protected String eventName;
    protected String eventDate;
    protected String eventTime;
    protected String venueName;
    protected int totalSeats;
    protected int availableSeats;
    protected double ticketPrice;
    protected String eventType;

    public Event() {}

    public Event(String eventName, String eventDate, String eventTime, String venueName, int totalSeats, double
        this.eventName = eventName;
        this.eventDate = eventDate;
        this.eventTime = eventTime;
        this.venueName = venueName;
        this.totalSeats = totalSeats;
        this.availableSeats = totalSeats;
        this.ticketPrice = ticketPrice;
        this.eventType = eventType;
    }

    public abstract void displayEventDetails();

    public void bookTickets(int num) {
        if (availableSeats >= num) {
            availableSeats -= num;
            System.out.println(num + " tickets booked.");
        } else {
            System.out.println("Not enough seats available.");
        }
    }

    public void cancelTickets(int num) {
        availableSeats += num;
        System.out.println(num + " tickets cancelled.");
    }

    public int getAvailableSeats() {
        return availableSeats;
    }

    public double calculateTotalRevenue() {
        return (totalSeats - availableSeats) * ticketPrice;
    }

}
```

Movie.java

```
package entity;

public class Movie extends Event{

    private String genre;
    private String actorName;
    private String actressName;

    public Movie(String eventName, String eventDate, String eventTime, String venueName, int totalSeats, double
                String genre, String actorName, String actressName) {
        super(eventName, eventDate, eventTime, venueName, totalSeats, ticketPrice, "Movie");
        this.genre = genre;
        this.actorName = actorName;
        this.actressName = actressName;
    }

    @Override
    public void displayEventDetails() {
        System.out.println("Movie: " + eventName + " | Genre: " + genre );
        System.out.println("Actor: " + actorName + " | Actress: " + actressName);
        System.out.println("Date: " + eventDate + " | Time: " + eventTime);
        System.out.println("Venue: " + venueName);
        System.out.println("Available Seats: " + availableSeats + " | Ticket Price: " + ticketPrice);
    }

}
```

Sports.java

```
package entity;

public class Sports extends Event{

    private String sportName;
    private String teamsName;

    public Sports(String eventName, String eventDate, String eventTime, String venueName, int totalSeats, double
                 String sportName, String teamsName) {
```

```
        super(eventName, eventDate, eventTime, venueName, totalSeats, ticketPrice, "Sports");
        this.sportName = sportName;
        this.teamsName = teamsName;
    }

    @Override
    public void displayEventDetails() {
        System.out.println("Sport Event: " + eventName);
        System.out.println( "Game: " + sportName + " | Teams: " + teamsName);
        System.out.println("Date: " + eventDate + " | Time: " + eventTime);
        System.out.println( "Venue: " + venueName);
        System.out.println("Available Seats: " + availableSeats + " | Ticket Price: " + ticketPrice);
    }

}
```

Task 7:

Has-A Relationship / Association

Define Venue class and include it as a reference in Event class
Create association between Event and Booking using object references
Implement logic to create events and book tickets with customer details
Maintain array of events and customers in BookingSystem class
Perform bookings, cancellations, and display info with linked classes

Package: /app

Main.java

```
package app;

import bean.*;
import service.BookingSystem;

import java.util.Scanner;

public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        BookingSystem system = new BookingSystem();
        boolean running = true;

        while (running) {
            System.out.println("\n--------------------------------------------------");
            System.out.println("         Welcome to the Ticket Booking System");
            System.out.println("--------------------------------------------------");
            System.out.println("1. Create Event");
            System.out.println("2. Book Tickets");
            System.out.println("3. Cancel Tickets");
            System.out.println("4. Get Available Seats");
            System.out.println("5. Get Event Details");
            System.out.println("6. Get Booking Details");
            System.out.println("7. Exit");
            System.out.print("Enter your choice: ");

            int choice = Integer.parseInt(sc.nextLine());
            switch (choice) {
                case 1:
                    System.out.print("Enter Event Name: ");
                    String name = sc.nextLine();
                    System.out.print("Enter Event Date (YYYY-MM-DD): ");
                    String date = sc.nextLine();
                    System.out.print("Enter Event Time (HH:MM): ");
                    String time = sc.nextLine();
                    System.out.print("Enter Total Seats: ");
                    int seats = Integer.parseInt(sc.nextLine());
                    System.out.print("Enter Ticket Price: ");
                    double price = Double.parseDouble(sc.nextLine());
                    System.out.print("Enter Event Type (movie/concert/sport): ");
                    String type = sc.nextLine();
                    System.out.print("Enter Venue Name: ");
                    String vName = sc.nextLine();
                    System.out.print("Enter Venue Address: ");
                    String vAddr = sc.nextLine();
                    Venue venue = new Venue(vName, vAddr);
                    system.createEvent(name, date, time, seats, price, type, venue);
                    break;

                case 2:
                    System.out.print("Enter Event Name to Book: ");
                    String eName = sc.nextLine();
                    System.out.print("Enter Number of Tickets: ");
                    int num = Integer.parseInt(sc.nextLine());
                    Customer[] customers = new Customer[num];
                    for (int i = 0; i < num; i++) {
                        System.out.println("Customer " + (i + 1) + ":");
                        System.out.print("Name: ");
                        String cname = sc.nextLine();
                        System.out.print("Email: ");
                        String email = sc.nextLine();
                        System.out.print("Phone: ");
                        String phone = sc.nextLine();
                        customers[i] = new Customer(cname, email, phone);
                    }
                    system.bookTickets(eName, customers);
                    break;

                case 3:
                    System.out.print("Enter Booking ID to Cancel: ");
                    int cancelId = Integer.parseInt(sc.nextLine());
                    system.cancelBooking(cancelId);
                    break;

                case 4:
                    System.out.print("Enter Event Name to Check Seats: ");
                    String evName = sc.nextLine();
                    system.getAvailableSeats(evName);
                    break;

                case 5:
                    system.getEventDetails();
                    break;

                case 6:
                    system.getBookingDetails();
                    break;

                case 7:
                    System.out.println("\n👋 Exiting Ticket Booking System. Thank you!");
```

```
                running = false;
                break;

            default:
                System.out.println("Invalid choice. Try again.");
        }
    }
    sc.close();
  }
}
```

Package: /service

BookingSystem.java

```
package service;

import bean.*;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.List;

public class BookingSystem {
    private List<Event> events = new ArrayList<>();
    private List<Booking> bookings = new ArrayList<>();

    public void createEvent(String name, String date, String time, int seats, double price, String type, Venue venue
        Event event = null;
        LocalDate eventDate = LocalDate.parse(date);
        LocalTime eventTime = LocalTime.parse(time);

        switch (type.toLowerCase()) {
            case "movie":
                event = new Movie(name, eventDate, eventTime, venue, seats, price);
                break;
            case "concert":
                event = new Concert(name, eventDate, eventTime, venue, seats, price);
                break;
            case "sport":
                event = new Sport(name, eventDate, eventTime, venue, seats, price);
                break;
            default:
                System.out.println("Invalid event type.");
                return;
        }

        events.add(event);
    }

    public void bookTickets(String eventName, Customer[] customers) {
        for (Event e : events) {
            if (e.getEventName().equalsIgnoreCase(eventName)) {
                Booking booking = new Booking(e, customers);
                bookings.add(booking);
                System.out.println("✅ Booking successful! Booking ID: " + booking.getBookingId() +
                        " | Total Cost: ₹" + booking.getTotalCost());
                return;
            }
        }
        System.out.println("❌ Event not found.");
    }

    public void cancelBooking(int bookingId) {
        for (Booking b : bookings) {
            if (b.getBookingId() == bookingId) {
                b.cancelBooking();
                bookings.remove(b);
                System.out.println("✅ Booking ID " + bookingId + " cancelled successfully.");
                return;
            }
        }
        System.out.println("❌ Booking ID not found.");
    }

    public void getAvailableSeats(String eventName) {
        for (Event e : events) {
            if (e.getEventName().equalsIgnoreCase(eventName)) {
                System.out.println("Available Seats for '" + eventName + "': " + e.getAvailableSeats());
                return;
            }
        }
        System.out.println("❌ Event not found.");
    }

    public void getEventDetails() {
        if (events.isEmpty()) {
            System.out.println("No events created yet.");
            return;
        }
        for (Event e : events) {
            e.displayEventDetails();
        }
    }

    public void getBookingDetails() {
        if (bookings.isEmpty()) {
            System.out.println("No bookings yet.");
            return;
        }
        for (Booking b : bookings) {
            b.displayBookingDetails();
        }
    }
}
```

Package: /bean

Booking.java

```
package bean;

import java.time.LocalDateTime;

public class Booking {
    private static int counter = 1;
    private int bookingId;
    private Customer[] customers;
    private Event event;
    private int numTickets;
    private double totalCost;
    private LocalDateTime bookingDate;

    public Booking(Event event, Customer[] customers) {
        this.bookingId = counter++;
        this.event = event;
        this.customers = customers;
        this.numTickets = customers.length;
        this.totalCost = event.ticketPrice * numTickets;
        this.bookingDate = LocalDateTime.now();
        event.bookTickets(numTickets);
    }

    public int getBookingId() {
        return bookingId;
    }

    public double getTotalCost() {
        return totalCost;
    }

    public void cancelBooking() {
        event.cancelBooking(numTickets);
    }

    public void displayBookingDetails() {
        System.out.println("Booking ID: " + bookingId);
        System.out.println("Event: " + event.getEventName());
        System.out.println("Tickets: " + numTickets);
        System.out.println("Total Cost: ₹" + totalCost);
        System.out.println("Booking Date: " + bookingDate);
        for (Customer c : customers) {
            c.displayCustomerDetails();
        }
    }
}
```

Concert.java

```
package bean;

import java.time.LocalDate;
import java.time.LocalTime;

public class Concert extends Event {
    public Concert(String eventName, LocalDate eventDate, LocalTime eventTime, Venue venue,
                   int totalSeats, double ticketPrice) {
        super(eventName, eventDate, eventTime, venue, totalSeats, ticketPrice, "Concert");
    }

    @Override
    public void displayEventDetails() {
        System.out.println("[Concert] " + eventName + " at " + venue.getVenueName() + " on " + eventDate + " " +
        System.out.println("Seats Available: " + availableSeats + "/" + totalSeats);
        System.out.println("Price: ₹" + ticketPrice);
    }
}
```

Customer.java

```
package bean;

public class Customer {
    private String name;
    private String email;
    private String phone;

    public Customer() {}

    public Customer(String name, String email, String phone) {
        this.name = name;
        this.email = email;
        this.phone = phone;
    }

    public void displayCustomerDetails() {
        System.out.println("Customer: " + name + ", Email: " + email + ", Phone: " + phone);
    }
}
```

Event.java

```
package bean;

import java.time.LocalDate;
import java.time.LocalTime;

public abstract class Event {
    protected String eventName;
    protected LocalDate eventDate;
    protected LocalTime eventTime;
    protected Venue venue;
    protected int totalSeats;
    protected int availableSeats;
    protected double ticketPrice;
    protected String eventType;

    public Event() {}

    public Event(String eventName, LocalDate eventDate, LocalTime eventTime, Venue venue,
                 int totalSeats, double ticketPrice, String eventType) {
        this.eventName = eventName;
        this.eventDate = eventDate;
        this.eventTime = eventTime;
        this.venue = venue;
        this.totalSeats = totalSeats;
        this.availableSeats = totalSeats;
        this.ticketPrice = ticketPrice;
        this.eventType = eventType;
    }

    public abstract void displayEventDetails();

    public int getAvailableSeats() {
        return availableSeats;
    }

    public int getBookedNoOfTickets() {
        return totalSeats - availableSeats;
    }

    public double calculateTotalRevenue() {
        return getBookedNoOfTickets() * ticketPrice;
    }

    public void bookTickets(int numTickets) {
        if (availableSeats >= numTickets) {
            availableSeats -= numTickets;
        } else {
            System.out.println("Not enough seats available.");
        }
    }

    public void cancelBooking(int numTickets) {
        if ((availableSeats + numTickets) <= totalSeats) {
            availableSeats += numTickets;
        } else {
            System.out.println("Cancellation failed. Exceeds total seats.");
        }
    }

    public String getEventName() {
        return eventName;
    }
}
```

Movie.java

```
package bean;

import java.time.LocalDate;
import java.time.LocalTime;

public class Movie extends Event {
    public Movie(String eventName, LocalDate eventDate, LocalTime eventTime, Venue venue,
                 int totalSeats, double ticketPrice) {
        super(eventName, eventDate, eventTime, venue, totalSeats, ticketPrice, "Movie");
    }

    @Override
    public void displayEventDetails() {
        System.out.println("[Movie] " + eventName + " at " + venue.getVenueName() + " on " + eventDate + " " + e
        System.out.println("Seats Available: " + availableSeats + "/" + totalSeats);
        System.out.println("Price: ₹" + ticketPrice);
    }
}
```

Sport.java

```
package bean;

import java.time.LocalDate;
import java.time.LocalTime;

public class Sport extends Event {
    public Sport(String eventName, LocalDate eventDate, LocalTime eventTime, Venue venue,
                 int totalSeats, double ticketPrice) {
        super(eventName, eventDate, eventTime, venue, totalSeats, ticketPrice, "Sports");
    }
    @Override
    public void displayEventDetails() {
        System.out.println("[Sport] " + eventName + " at " + venue.getVenueName() + " on " + eventDate + " " + ev
        System.out.println("Seats Available: " + availableSeats + "/" + totalSeats);
        System.out.println("Price: ₹" + ticketPrice);
    }
}
```

Venue.java

```
package bean;

public class Venue {
    private String venueName;
    private String address;

    public Venue() {
        this.venueName = "Unknown";
        this.address = "Unknown";
    }

    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("Address: " + address);
    }
}
```

Task 8:

Interface, Single Inheritance, Static Variable

Create interfaces: IEventServiceProvider, IBookingSystemServiceProvider
Implement interfaces in EventServiceProviderImpl and BookingSystemServiceProviderImpl
Use inheritance and static variables for ID generation or counters
Implement all required methods: create_event(), getAvailableNoOfTickets(), etc.
Maintain data in arrays/lists/maps as per design

Package: /app

TicketBookingSystem.java

```
package client;

import entity.Event;
import entity.Cupackage app;
    import java.util.Scanner;

    import bean.Customer;
    import bean.Event;
    import bean.Venue;
    import service.BookingSystemServiceProviderImpl;

    public class TicketBookingSystem {

        public static void main(String[] args) {
            Scanner sc = new Scanner(System.in);
            BookingSystemServiceProviderImpl system = new BookingSystemServiceProviderImpl();

            while (true) {
                System.out.println("\n----- Ticket Booking System Menu -----");
                System.out.println("1. Create Event");
                System.out.println("2. Book Tickets");
                System.out.println("3. Cancel Tickets");
                System.out.println("4. Get Available Seats");
                System.out.println("5. Get Event Details");
                System.out.println("6. Get Booking Details");
                System.out.println("7. Exit");
                System.out.print("Enter your choice: ");

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

                switch (choice) {

                    case 1:
                        System.out.print("Enter event name: ");
                        String eventName = sc.nextLine();

                        System.out.print("Enter event date (YYYY-MM-DD): ");
                        String date = sc.nextLine();

                        System.out.print("Enter event time (HH:MM): ");
                        String time = sc.nextLine();

                        System.out.print("Enter venue name: ");
                        String venueName = sc.nextLine();

                        System.out.print("Enter venue address: ");
                        String address = sc.nextLine();

                        System.out.print("Enter total seats: ");
                        int totalSeats = sc.nextInt();

                        System.out.print("Enter ticket price: ");
                        double ticketPrice = sc.nextDouble();

                        sc.nextLine(); // consume newline
                        System.out.print("Enter event type (Movie/Sports/Concert): ");
                        String eventType = sc.nextLine();

                        Venue venue = new Venue(venueName, address);
                        system.createEvent(eventName, date, time, totalSeats, ticketPrice, eventType, venue);
                        break;

                    case 2:
                        System.out.print("Enter event name to book: ");
                        String bookEvent = sc.nextLine();

                        System.out.print("Enter number of tickets: ");
                        int numTickets = sc.nextInt();
                        sc.nextLine(); // consume newline

                        Customer[] customers = new Customer[numTickets];
                        for (int i = 0; i < numTickets; i++) {
                            System.out.println("Enter details for Customer " + (i + 1));
                            System.out.print("Name: ");
                            String name = sc.nextLine();
                            System.out.print("Email: ");
                            String email = sc.nextLine();
                            System.out.print("Phone: ");
                            String phone = sc.nextLine();
                            customers[i] = new Customer(name, email, phone);
                        }

                        system.bookTickets(bookEvent, numTickets, customers);
                        break;

                    case 3:
                        System.out.print("Enter booking ID to cancel: ");
                        int bookingId = sc.nextInt();
                        system.cancelBooking(bookingId);
                        break;

                    case 4:
                        int availableSeats = system.getAvailableNoOfTickets();
                        System.out.println("Total Available Tickets: " + availableSeats);
                        break;

                    case 5:
                        bean.Event[] events = system.getEventDetails();
                        if (events.length == 0) {
                            System.out.println("No events found.");
                        } else {
                            for (Event e : events) {
                                e.displayEventDetails();
                            }
                        }
                        break;

                    case 6:
                        System.out.print("Enter booking ID: ");
                        int id = sc.nextInt();
                        system.getBookingDetails(id);
                        break;

                    case 7:
                        System.out.println("Exiting Ticket Booking System. Thank you!");
```

```
                    sc.close();
                    return;

                default:
                    System.out.println("Invalid choice. Please try again.");
            }
        }
    }

}
stomer;
import entity.Venue;

import java.time.LocalDate;
import java.time.LocalTime;
import java.util.Scanner;

public class Booking {

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        Venue venue = new Venue("Stadium Arena", "123 Main Street");

        Event event = new Event("Rock Concert", LocalDate.of(2025, 6, 15), LocalTime.of(19, 30), venue.getVenueN

        System.out.print("Enter Your Name: ");
        String name = sc.nextLine();
        System.out.print("Enter Your Email: ");
        String email = sc.nextLine();
        System.out.print("Enter Your Phone Number: ");
        String phone = sc.nextLine();

        Customer customer = new Customer(name, email, phone);

        // Display details
        venue.displayVenueDetails();
        event.displayEventDetails();
        customer.displayCustomerDetails();

        // Book tickets
        System.out.print("\nEnter Number of Tickets to Book: ");
        int numTickets = sc.nextInt();
        event.bookTickets(numTickets);
        event.displayEventDetails();

        // Canceling Tickets
        System.out.print("\nEnter number of tickets to cancel: ");
        int cancelTickets = sc.nextInt();
        event.cancelBooking(cancelTickets);
        event.displayEventDetails();

        // Display revenue
        System.out.println("\nTotal Revenue Generated: Rs." + event.calculateTotalRevenue());

        sc.close();
    }

}
```

Package: /service

BookingSystemServiceProviderImpl.java

```
package service;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.List;

import bean.Booking;
import bean.Customer;
import bean.Event;

public class BookingSystemServiceProviderImpl extends EventServiceProviderImpl implements IBookingSystem

    private static List<Booking> bookings = new ArrayList<>();

    @Override
    public double calculateBookingCost(int numTickets, double ticketPrice) {
        return numTickets * ticketPrice;
    }

    @Override
    public void bookTickets(String eventName, int numTickets, Customer[] customers) {
        Event selectedEvent = null;
        for (Event e : events) {
            if (e.getEventName().equalsIgnoreCase(eventName)) {
                selectedEvent = e;
                break;
            }
        }

        if (selectedEvent == null) {
            System.out.println("Event not found.");
            return;
        }

        if (selectedEvent.getAvailableSeats() < numTickets) {
            System.out.println("Not enough seats available!");
            return;
        }

        selectedEvent.bookTickets(numTickets);
        double totalCost = calculateBookingCost(numTickets, selectedEvent.getTicketPrice());
        Booking booking = new Booking(customers, selectedEvent, numTickets, totalCost, LocalDateTime.now());
        bookings.add(booking);

        System.out.println("Booking successful! Booking ID: " + booking.getBookingId());
    }

    @Override
    public void cancelBooking(int bookingId) {
        Booking toCancel = null;
        for (Booking b : bookings) {
            if (b.getBookingId() == bookingId) {
                toCancel = b;
                break;
            }
        }
        if (toCancel == null) {
            System.out.println("Booking ID not found.");
            return;
        }

        toCancel.getEvent().cancelBooking(toCancel.getNumTickets());
        bookings.remove(toCancel);
        System.out.println("Booking cancelled successfully.");
    }

    @Override
    public void getBookingDetails(int bookingId) {
        for (Booking b : bookings) {
            if (b.getBookingId() == bookingId) {
                b.displayBookingDetails();
                return;
            }
        }
        System.out.println("Booking ID not found.");
    }

}
```

EventServiceProviderImpl.java

```
package service;

import java.util.ArrayList;
import java.util.List;

import bean.Event;
import bean.Sport;
import bean.Venue;
import bean.Movie;
import bean.Concert;

public class EventServiceProviderImpl implements IEventServiceProvider{

    protected static List<Event> events = new ArrayList<>();

    @Override
    public Event createEvent(String eventName, String date, String time, int totalSeats, double ticketPrice, String e
        Event event = null;

        switch (eventType.toLowerCase()) {
            case "movie":
                event = new Movie(eventName, date, time, venue, totalSeats, ticketPrice, "Movie", "Action", "Shah Rukl
                break;
            case "concert":
                event = new Concert(eventName, date, time, venue, totalSeats, ticketPrice, "Concert", "Arijit Singh", "R
                break;
            case "sports":
                event = new Sport(eventName, date, time, venue, totalSeats, ticketPrice, "Sports", "Cricket", "India vs /
                break;
            default:
                System.out.println("Invalid event type.");
        }

        if (event != null) {
            events.add(event);
            System.out.println("Event created successfully!");
        }

        return event;
    }

    @Override
    public Event[] getEventDetails() {
        return events.toArray(new Event[0]);
    }

    @Override
    public int getAvailableNoOfTickets() {
        int totalAvailable = 0;
        for (Event e : events) {
            totalAvailable += e.getAvailableSeats();
        }
        return totalAvailable;
    }}
```

IBookingSystemServiceProvider.java

```
package service;
import bean.Customer;

public interface IBookingSystemServiceProvider {

        double calculateBookingCost(int numTickets, double ticketPrice);
        void bookTickets(String eventName, int numTickets, Customer[] customers);
        void cancelBooking(int bookingId);
        void getBookingDetails(int bookingId);

}
```

IEventServiceProvider.java

```
package service;
import bean.Event;
import bean.Venue;

public interface IEventServiceProvider {

    Event createEvent(String eventName, String date, String time, int totalSeats, double ticketPrice, String eventTy
    Event[] getEventDetails();
    int getAvailableNoOfTickets();

}
```

Package: /bean

Booking.java

```
package bean;
import java.time.LocalDateTime;

public class Booking {

        private static int bookingCounter = 1000;

        private int bookingId;
```

```
    private Customer[] customers;
    private Event event;
    private int numTickets;
    private double totalCost;
    private LocalDateTime bookingDate;

    public Booking() {
        this.bookingId = bookingCounter++;
        this.bookingDate = LocalDateTime.now();
    }

    public Booking(Customer[] customers, Event event, int numTickets, double totalCost) {
        this();
        this.customers = customers;
        this.event = event;
        this.numTickets = numTickets;
        this.totalCost = totalCost;
    }

    public Booking(Customer[] customers, Event event, int numTickets, double totalCost, LocalDateTime bookin
        this.bookingId = ++Booking.bookingCounter;
        this.customers = customers;
        this.event = event;
        this.numTickets = numTickets;
        this.totalCost = totalCost;
        this.bookingDate = bookingDate;
    }


    public int getBookingId() {
        return bookingId;
    }

    public Customer[] getCustomers() {
        return customers;
    }

    public Event getEvent() {
        return event;
    }

    public int getNumTickets() {
        return numTickets;
    }

    public double getTotalCost() {
        return totalCost;
    }

    public LocalDateTime getBookingDate() {
        return bookingDate;
    }

    public void setCustomers(Customer[] customers) {
        this.customers = customers;
    }

    public void setEvent(Event event) {
        this.event = event;
    }
    public void setNumTickets(int numTickets) {
        this.numTickets = numTickets;
    }

    public void setTotalCost(double totalCost) {
        this.totalCost = totalCost;
    }

    public void displayBookingDetails() {
        System.out.println("Booking ID: " + bookingId);
        System.out.println("Booking Date: " + bookingDate);
        System.out.println("Number of Tickets: " + numTickets);
        System.out.println("Total Cost: ₹" + totalCost);
        System.out.println("Event Booked:");
        event.displayEventDetails();
        System.out.println("Customer Details:");
        for (Customer customer : customers) {
            customer.displayCustomerDetails();
            System.out.println("---");
        }}}
```

Concert.java

```
package bean;

public class Concert extends Event{

    private String artist;
    private String type;
    public Concert() {}

    public Concert(String eventName, String eventDate, String eventTime, Venue venue, int totalSeats, double ti
        super(eventName, eventDate, eventTime, venue, totalSeats, ticketPrice, eventType);
        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 displayEventDetails() {
        System.out.println("Concert: " + getEventName() + " | Type: " + type + ", Artist: " + artist);
        System.out.println("Date: " + getEventDate() + ", Time: " + getEventTime());
        getVenue().displayVenueDetails();
        System.out.println("Tickets: " + getAvailableSeats() + "/" + getTotalSeats());
        }

}
```

Customer.java

```
package bean;

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 String getEmail() {
            return email;
        }

        public String getPhoneNumber() {
            return phoneNumber;
        }

        public void setCustomerName(String customerName) {
            this.customerName = customerName;
        }

        public void setEmail(String email) {
            this.email = email;
        }

        public void setPhoneNumber(String phoneNumber) {
            this.phoneNumber = phoneNumber;
        }

        public void displayCustomerDetails() {
            System.out.println("Customer Name: " + customerName);
            System.out.println("Email: " + email);
            System.out.println("Phone Number: " + phoneNumber);
        }}
```

Event.java

```
package bean;

public abstract class Event {

    private String eventName;
```

```
  private String eventDate;
  private String eventTime;
  private Venue venue;
  private int totalSeats;
  private int availableSeats;
  private double ticketPrice;
  private String eventType;

    public Event() {}

    public Event(String eventName, String eventDate, String eventTime, Venue venue, int totalSeats, double tick
      this.eventName = eventName;
      this.eventDate = eventDate;
      this.eventTime = eventTime;
      this.venue = venue;
      this.totalSeats = totalSeats;
      this.availableSeats = totalSeats; // available = total at start
      this.ticketPrice = ticketPrice;
      this.eventType = eventType;
    }

    public String getEventDate() {
      return eventDate;
    }

    public void setEventDate(String eventDate) {
      this.eventDate = eventDate;
    }

    public String getEventTime() {
      return eventTime;
    }

    public void setEventTime(String eventTime) {
      this.eventTime = eventTime;
    }

    public Venue getVenue() {
      return venue;
    }

    public void setVenue(Venue venue) {
      this.venue = venue;
    }

    public int getTotalSeats() {
      return totalSeats;
    }

    public void setTotalSeats(int totalSeats) {
      this.totalSeats = totalSeats;
    }

    public double getTicketPrice() {
      return ticketPrice;
    }

    public void setTicketPrice(double ticketPrice) {
      this.ticketPrice = ticketPrice;
    }
    public String getEventType() {
        return eventType;
    }

    public void setEventType(String eventType) {
        this.eventType = eventType;
    }

    public void setEventName(String eventName) {
        this.eventName = eventName;
    }

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

    public int getAvailableSeats() { return availableSeats; }

    public void bookTickets(int num) {
        if (availableSeats >= num) availableSeats -= num;
        else System.out.println("Not enough seats.");
    }

    public void cancelBooking(int num) {
        availableSeats += num;
    }

    public double calculateTotalRevenue() {
        return (totalSeats - availableSeats) * ticketPrice;
    }

    public int getBookedNoOfTickets() {
        return totalSeats - availableSeats;
    }

    public abstract void displayEventDetails();

    public String getEventName() {
        return eventName;
    }
}
```

Movie.java

```
package bean;

public class Movie extends Event{

    private String genre;
    private String actorName;
    private String actressName;

    public Movie() {}

    public Movie(String eventName, String eventDate, String eventTime, Venue venue, int totalSeats, double tick
        super(eventName, eventDate, eventTime, venue, totalSeats, ticketPrice, eventType);
        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 displayEventDetails() {
        System.out.println("Movie: " + getEventName() + " | Genre: " + genre);
        System.out.println("Actor: " + actorName + ", Actress: " + actressName);
        getVenue().displayVenueDetails();
    }


}
```

Sport.java

```
package bean;

public class Sport extends Event{

    private String sportName;
    private String teamsName;

    public Sport() {}

    public Sport(String eventName, String eventDate, String eventTime, Venue venue, int totalSeats, double tick
        super(eventName, eventDate, eventTime, venue, totalSeats, ticketPrice, eventType);
        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;
    }

    @Override
    public void displayEventDetails() {
        System.out.println("Sports Event: " + getEventName() + " | Sport: " + sportName + " | Match: " + teamsNa
        System.out.println("Date: " + getEventDate() + ", Time: " + getEventTime());
        getVenue().displayVenueDetails();
        System.out.println("Tickets: " + getAvailableSeats() + "/" + getTotalSeats());
    }
}
```

Venue.java

```
package bean;

public class Venue {

    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: " + venueName + ", Address: " + address);
    }

}
```

Task 9:

Exception Handling

Create custom exceptions: EventNotFoundException, InvalidBookingIDException
Throw and handle exceptions in TicketBookingSystem class methods
Handle NullPointerException and SQLException gracefully in main method

Package: /app

TicketBookingSystem.java

```
package app;
```

```java
import dao.BookingSystemService;
import bean.Booking;
import bean.Event;
import exception.EventNotFoundException;
import exception.InvalidBookingIDException;

import java.util.List;
import java.util.Scanner;

public class TicketBookingSystem {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        BookingSystemService service = new BookingSystemService();

        while (true) {
            displayMenu();
            try {
                int choice = Integer.parseInt(scanner.nextLine());
                switch (choice) {
                    case 1 -> handleBooking(scanner, service);
                    case 2 -> handleCancellation(scanner, service);
                    case 3 -> showEvents(service.getAllEvents());
                    case 4 -> showBookings(service.getAllBookings());
                    case 5 -> {
                        System.out.println("Exiting system. Goodbye!");
                        return;
                    }
                    default -> System.out.println("Invalid option.");
                }
            } catch (EventNotFoundException | InvalidBookingIDException e) {
                System.out.println(e.getMessage());
            } catch (IllegalArgumentException e) {
                System.out.println(e.getMessage());
            } catch (Exception e) {
                System.out.println("Unexpected Error: " + e.getMessage());
            }
        }
    }

    private static void displayMenu() {
        System.out.println("\n--- Ticket Booking System ---");
        System.out.println("1. Book Tickets");
        System.out.println("2. Cancel Booking");
        System.out.println("3. Show Events");
        System.out.println("4. Show All Bookings");
        System.out.println("5. Exit");
        System.out.print("Enter choice: ");
    }

    private static void handleBooking(Scanner scanner, BookingSystemService service) throws EventNotFoundExc
        System.out.print("Enter event name: ");
        String eventName = scanner.nextLine();

        System.out.print("Enter number of tickets: ");
        int numTickets = Integer.parseInt(scanner.nextLine());

        Booking booking = service.bookTickets(eventName, numTickets);
        System.out.println("Booking successful!");
        booking.displayBooking();
    }
    private static void handleCancellation(Scanner scanner, BookingSystemService service) throws InvalidBooking
        System.out.print("Enter booking ID: ");
        int bookingId = Integer.parseInt(scanner.nextLine());

        service.cancelBooking(bookingId);
        System.out.println(" Booking cancelled successfully.");
    }

    private static void showEvents(List<Event> events) {
        System.out.println("\nAvailable Events:");
        events.forEach(Event::displayEventDetails);
    }

    private static void showBookings(List<Booking> bookings) {
        System.out.println("\nAll Bookings:");
        if (bookings.isEmpty()) {
            System.out.println("No bookings yet.");
        } else {
            bookings.forEach(Booking::displayBooking);
        }
    }
}
```

Package: /bean

Booking.java

```
package bean;

public class Booking {
    private int bookingId;
    private String eventName;
    private int numTickets;

    public Booking(int bookingId, String eventName, int numTickets) {
        this.bookingId = bookingId;
        this.eventName = eventName;
        this.numTickets = numTickets;
    }

    public int getBookingId() {
        return bookingId;
    }

    public String getEventName() {
        return eventName;
    }

    public int getNumTickets() {
        return numTickets;
    }

    public void displayBooking() {
        System.out.println("Booking ID: " + bookingId + ", Event: " + eventName + ", Tickets: " + numTickets);
    }
}
```

Event.java

```java
package bean;

public class Event {
    private String eventName;
    private int availableTickets;

    public Event(String eventName, int availableTickets) {
        this.eventName = eventName;
        this.availableTickets = availableTickets;
    }

    public String getEventName() {
        return eventName;
    }

    public int getAvailableTickets() {
        return availableTickets;
    }

    public void reduceTickets(int count) {
        this.availableTickets -= count;
    }

    public void displayEventDetails() {
        System.out.println("Event: " + eventName + ", Tickets Left: " + availableTickets);
    }
}
```

Package: /dao

BookingSystemService.java

```java
package dao;

import bean.Booking;
import bean.Event;
import exception.EventNotFoundException;
import exception.InvalidBookingIDException;

import java.util.ArrayList;
import java.util.List;
import java.util.Optional;

public class BookingSystemService {
    private final List<Event> events = new ArrayList<>();
    private final List<Booking> bookings = new ArrayList<>();
    private int bookingIdCounter = 1;

    public BookingSystemService() {
        events.add(new Event("Concert Night", 50));
        events.add(new Event("Comedy Show", 30));
    }

    public List<Event> getAllEvents() {
        return events;
    }

    public Booking bookTickets(String eventName, int numTickets) throws EventNotFoundException {
        Optional<Event> optionalEvent = events.stream()
                .filter(e → e.getEventName().equalsIgnoreCase(eventName))
                .findFirst();

        if (optionalEvent.isEmpty()) {
            throw new EventNotFoundException("Event not found: " + eventName);
        }

        Event event = optionalEvent.get();
        if (numTickets > event.getAvailableTickets()) {
            throw new IllegalArgumentException("Not enough tickets. Only " + event.getAvailableTickets() + " availab
        }

        event.reduceTickets(numTickets);
        Booking booking = new Booking(bookingIdCounter++, eventName, numTickets);
        bookings.add(booking);
        return booking;
    }

    public void cancelBooking(int bookingId) throws InvalidBookingIDException {
        Optional<Booking> optionalBooking = bookings.stream()
                .filter(b → b.getBookingId() == bookingId)
                .findFirst();

        if (optionalBooking.isEmpty()) {
            throw new InvalidBookingIDException("Booking ID not found: " + bookingId);
        }

        Booking booking = optionalBooking.get();
        events.stream()
                .filter(e → e.getEventName().equalsIgnoreCase(booking.getEventName()))
                .findFirst()
                .ifPresent(e → e.reduceTickets(-booking.getNumTickets())); // Add tickets back

        bookings.remove(booking);
    }

    public List<Booking> getAllBookings() {
        return bookings;
    }
}
```

Package: /exception

EventNotFoundException.java

```
package exception;

public class EventNotFoundException extends Exception {
    public EventNotFoundException(String message) {
        super(message);
    }
}
```

InvalidBookingIDException.java

```
package exception;

public class InvalidBookingIDException extends Exception {
    public InvalidBookingIDException(String message) {
        super(message);
    }
}
```

Task 10:

Collections
Convert arrays to List, Set, Map for events and bookings
Prevent duplicates using Set, sort using Comparator
Store bookings in Map with bookingId as key.

Package: /app

MainApp.java

```
package app;

import entity.Customer;
import entity.Venue;
import serviceImpl.BookingServiceImpl;

import java.util.ArrayList;
import java.util.List;
import java.util.Scanner;

public class MainApp {
    public static void main(String[] args) {
        BookingServiceImpl system = new BookingServiceImpl();
        Scanner sc = new Scanner(System.in);

        while (true) {
            System.out.println("\n1. Create Event\n2. Book Tickets\n3. Cancel Tickets\n4. Show Events\n5. Exit");
            System.out.print("Enter option: ");
            int choice = sc.nextInt();
            sc.nextLine();

            switch (choice) {
                case 1:
                    System.out.print("Event Name: ");
                    String name = sc.nextLine();
                    System.out.print("Date (YYYY-MM-DD): ");
                    String date = sc.nextLine();
                    System.out.print("Time (HH:MM): ");
                    String time = sc.nextLine();
                    System.out.print("Total Seats: ");
                    int seats = sc.nextInt();
                    System.out.print("Ticket Price: ");
                    double price = sc.nextDouble();
                    sc.nextLine();
                    System.out.print("Type (Movie/Sports/Concert): ");
                    String type = sc.nextLine();
                    System.out.print("Venue Name: ");
                    String vname = sc.nextLine();
                    System.out.print("Venue Address: ");
                    String address = sc.nextLine();

                    Venue venue = new Venue(vname, address);
                    system.createEvent(name, date, time, seats, price, type, venue);
                    break;
                case 2:
                    System.out.print("Enter Event Name: ");
                    String ename = sc.nextLine();
                    System.out.print("Number of Tickets: ");
                    int tickets = sc.nextInt();
                    sc.nextLine();
                    List<Customer> customers = new ArrayList<>();
                    for (int i = 0; i < tickets; i++) {
                        System.out.print("Customer " + (i + 1) + " Name: ");
                        String cname = sc.nextLine();
                        System.out.print("Email: ");
                        String email = sc.nextLine();
                        System.out.print("Phone: ");
                        String phone = sc.nextLine();
                        customers.add(new Customer(cname, email, phone));
                    }
                    system.bookTickets(ename, tickets, customers);
                    break;

                case 3:
                    System.out.print("Enter Booking ID: ");
                    int id = sc.nextInt();
                    sc.nextLine();
                    system.cancelTickets(id);
                    break;

                case 4:
                    system.displayEventDetails();
                    break;

                case 5:
                    System.out.println("Exiting.");
                    return;

                default:
                    System.out.println("Invalid option.");
            }
        }
    }
}
```

Package: /entity

Booking.java

```
package entity;

import java.time.LocalDateTime;
import java.util.List;

import service.IBookingService;

public class Booking {
    private static int bookingCounter = 1;
    private int bookingId;
    private List<Customer> customers;
    private Event event;
    private int numTickets;
    private double totalCost;
    private LocalDateTime bookingDate;
    public Booking(List<Customer> customers, Event event, int numTickets) {
        this.bookingId = bookingCounter++;
        this.customers = customers;
        this.event = event;
        this.numTickets = numTickets;
        this.totalCost = event.getTicketPrice() * numTickets;
        this.bookingDate = LocalDateTime.now();
    }

    public int getBookingId() {
        return bookingId;
    }

    public void displayBookingDetails() {
        System.out.println("Booking ID: " + bookingId);
        event.displayEventDetails();
        for (Customer c : customers) {
            c.displayCustomerDetails();
        }
        System.out.println("Tickets: " + numTickets + ", Total Cost: ₹" + totalCost + ", Date: " + bookingDate);
    }

    public IBookingService getEvent() {
        // TODO Auto-generated method stub
        return null;
    }
}
```

Customer.java

```
package entity;

public class Customer {
    private String name;
    private String email;
    private String phone;

    public Customer(String name, String email, String phone) {
        this.name = name;
        this.email = email;
        this.phone = phone;
    }

    public void displayCustomerDetails() {
        System.out.println("Name: " + name + ", Email: " + email + ", Phone: " + phone);
    }
}
```

Event.java

```
package entity;

public class Event {
    private String eventName;
    private String eventDate;
    private String eventTime;
    private Venue venue;
    private int totalSeats;
    private int availableSeats;
    private double ticketPrice;
    private String eventType;

    public Event(String eventName, String eventDate, String eventTime, Venue venue, int totalSeats, double ticketI
        this.eventName = eventName;
        this.eventDate = eventDate;
        this.eventTime = eventTime;
        this.venue = venue;
        this.totalSeats = totalSeats;
        this.availableSeats = totalSeats;
        this.ticketPrice = ticketPrice;
        this.eventType = eventType;
    }

    public String getEventName() {
        return eventName;
    }

    public int getAvailableSeats() {
        return availableSeats;
    }

    public double getTicketPrice() {
        return ticketPrice;
    }

    public Venue getVenue() {
        return venue;
    }

    public void bookTickets(int count) {
        availableSeats -= count;
    }

    public void cancelTickets(int count) {
        availableSeats += count;
    }

    public void displayEventDetails() {
        System.out.println("Event: " + eventName + ", Type: " + eventType + ", Date: " + eventDate + ", Time: " + ev
        System.out.println("Venue: " + venue.getVenueName());
        System.out.println("Tickets: " + availableSeats + "/" + totalSeats + ", Price: ₹" + ticketPrice);
    }
}
```

Venue.java

```
package entity;

public class Venue {
    private String venueName;
    private String address;

    public Venue(String venueName, String address) {
        this.venueName = venueName;
        this.address = address;
    }

    public String getVenueName() {
        return venueName;
    }

    public String getAddress() {
        return address;
    }

    public void displayVenueDetails() {
        System.out.println("Venue: " + venueName + ", Address: " + address);
    }
}
```

Package: /service

IBookingService.java

```
package service;

import entity.Customer;
import entity.Event;
import entity.Venue;

import java.util.List;

public interface IBookingService {
    Event createEvent(String name, String date, String time, int seats, double price, String type, Venue venue);
    void bookTickets(String eventName, int ticketCount, List<Customer> customers);
    void cancelTickets(int bookingId);
    void displayEventDetails();
}
```

Package: /serviceImpl

BookingServiceImpl.java

```
package serviceImpl;

import entity.*;
import service.IBookingService;

import java.util.*;

public class BookingServiceImpl implements IBookingService {
    private Map<String, Event> eventMap = new HashMap<>();
    private Map<Integer, Booking> bookingMap = new HashMap<>();

    @Override
    public Event createEvent(String name, String date, String time, int seats, double price, String type, Venue venu
        Event event = new Event(name, date, time, venue, seats, price, type);
        eventMap.put(name.toLowerCase(), event);
        return event;
    }

    @Override
    public void bookTickets(String eventName, int ticketCount, List<Customer> customers) {
        Event event = eventMap.get(eventName.toLowerCase());
        if (event != null && event.getAvailableSeats() >= ticketCount) {
            event.bookTickets(ticketCount);
            Booking booking = new Booking(customers, event, ticketCount);
            bookingMap.put(booking.getBookingId(), booking);
            System.out.println("Booking successful. ID: " + booking.getBookingId());
        } else {
            System.out.println("Event not found or not enough tickets.");
        }
    }

    @Override
    public void cancelTickets(int bookingId) {
        Booking booking = bookingMap.remove(bookingId);
        if (booking != null) {
            booking.getEvent().cancelTickets(booking.getBookingId());
            System.out.println("Booking cancelled.");
        } else {
            System.out.println("Invalid booking ID.");
        }
    }

    @Override
    public void displayEventDetails() {
        for (Event e : eventMap.values()) {
            e.displayEventDetails();
        }
    }
}
```

Task 11:

Database Connectivity

Use JDBC to connect to DB
Implement repository interfaces for CRUD operations
Store and retrieve event, customer, and booking data from DB
Use utility classes for DB connection and property loading
Handle exceptions and implement main method UI with menu-driven commands

Package: /app

MainApp.java

```
package app;

import dao.EventDAO;
import daoImpl.EventDAOImpl;
import model.Event;

import java.sql.Date;
import java.sql.Time;
import java.util.Scanner;

public class MainApp {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        System.out.println("Enter Event Name:");
        String name = sc.nextLine();

        System.out.println("Enter Event Date (yyyy-mm-dd):");
        String dateStr = sc.nextLine();
        Date date = Date.valueOf(dateStr);

        System.out.println("Enter Event Time (hh:mm:ss):");
        String timeStr = sc.nextLine();
        Time time = Time.valueOf(timeStr);

        System.out.println("Enter Venue ID:");
        int venueId = sc.nextInt();

        System.out.println("Enter Total Seats:");
        int totalSeats = sc.nextInt();

        System.out.println("Enter Available Seats:");
        int availableSeats = sc.nextInt();

        System.out.println("Enter Ticket Price:");
        double price = sc.nextDouble();

        sc.nextLine();
        System.out.println("Enter Event Type (Movie/Sports/Concert):");
        String type = sc.nextLine();

        System.out.println("Enter Booking ID:");
        int bookingId = sc.nextInt();

        Event event = new Event(name, date, time, venueId, totalSeats, availableSeats, price, type, bookingId);
        EventDAO dao = new EventDAOImpl();
        dao.saveEvent(event);
    }
}
```

Package: /dao

EventDAO.java

```
package dao;

import model.Event;

public interface EventDAO {
    void saveEvent(Event event);
}
```

Package: /daoImpl

BookingSystem .java

```
package daoImpl;

import dao.EventDAO;
import model.Event;
import util.DBConnUtil;

import java.sql.Connection;
import java.sql.PreparedStatement;

public class EventDAOImpl implements EventDAO {

    @Override
    public void saveEvent(Event event) {
        try (Connection conn = DBConnUtil.getConnection()) {
            String query = "INSERT INTO event (event_name, event_date, event_time, venue_id, total_seats, available
                        + "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)";
            PreparedStatement ps = conn.prepareStatement(query);
            ps.setString(1, event.getEventName());
            ps.setDate(2, event.getEventDate());
            ps.setTime(3, event.getEventTime());
            ps.setInt(4, event.getVenueId());
            ps.setInt(5, event.getTotalSeats());
            ps.setInt(6, event.getAvailableSeats());
            ps.setDouble(7, event.getTicketPrice());
            ps.setString(8, event.getEventType());
            ps.setInt(9, event.getBookingId());

            int rows = ps.executeUpdate();
            if (rows > 0) {
                System.out.println("Event saved successfully!");
            }
        } catch (Exception e) {
            e.printStackTrace();
            System.out.println("Error while saving event: " + e.getMessage());
        }
    }
}
```

Package: /model

Event.java

```
package model;

import java.sql.Date;
import java.sql.Time;

public class Event {
    private int eventId;
    private String eventName;
    private Date eventDate;
    private Time eventTime;
    private int venueId;
    private int totalSeats;
    private int availableSeats;
    private double ticketPrice;
    private String eventType;
    private int bookingId;

    public Event(String eventName, Date eventDate, Time eventTime, int venueId,
                 int totalSeats, int availableSeats, double ticketPrice,
                 String eventType, int bookingId) {
        this.eventName = eventName;
        this.eventDate = eventDate;
        this.eventTime = eventTime;
        this.venueId = venueId;
        this.totalSeats = totalSeats;
        this.availableSeats = availableSeats;
        this.ticketPrice = ticketPrice;
        this.eventType = eventType;
        this.bookingId = bookingId;
    }

    // Getters
    public String getEventName() { return eventName; }
    public Date getEventDate() { return eventDate; }
    public Time getEventTime() { return eventTime; }
    public int getVenueId() { return venueId; }
    public int getTotalSeats() { return totalSeats; }
    public int getAvailableSeats() { return availableSeats; }
    public double getTicketPrice() { return ticketPrice; }
    public String getEventType() { return eventType; }
    public int getBookingId() { return bookingId; }
}
```

Package: /util

DBConnUtil.java

```
package util;
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DBConnUtil {

    private static final String fileName="db.properties";
    public static Connection getConnection() {
        Connection con=null;
        String connString=null;
        try {
        connString=DBPropertyUtil.getConnection(fileName);
        }catch (IOException e) {
            System.out.println("Connection String Creation Failed");
            e.printStackTrace();
        }
        if(connString!=null) {
            try {
                con=DriverManager.getConnection(connString);
            }catch (SQLException e) {
                System.out.println("Error While Establishing DBConnection........");
                e.printStackTrace();
            }
        }
        return con;
    }
}
```

DBPropertyUtil.java

```
package util;
import java.io.FileInputStream;
import java.io.IOException;
import java.util.Properties;
public class DBPropertyUtil {

    public static String getConnection(String fileName)throws IOException {
        //fileName="db.properties"
        String connStr=null;
        Properties props=new Properties();
        FileInputStream fis=new FileInputStream(fileName);
        props.load(fis);
        String user=props.getProperty("user");
        String password=props.getProperty("password");
        String protocol=props.getProperty("protocol");
        String system=props.getProperty("system");
        String database=props.getProperty("database");
        String port=props.getProperty("port");
        connStr=protocol+"//"+system+":"+port+"/"+database+"?user="+user+"&password="+password;
        return  connStr;
    }
}
```