QUESTION:

Event Ticketing System – Specification Document

Problem Statement

Develop a console app to manage events, venues, seats, customers, bookings, and cancellations.

Class Requirements

1. Event
2. Venue
3. Seat
4. Customer
5. Booking
6. BookingItem
7. Cancellation

Business Rules

1. Seats can be booked only if available.
2. Seat availability updates on booking/cancellation.
3. Cancellations may include a fee based on policy and timing.
4. Each booking item links directly to a seat and event.
5. Prevent double-booking of the same seat.

Console Interface Requirements

• Add Event / Add Venue & Seats / Book Seats / Cancel Booking / Display Events & Availability / Exit

Department of Computer Science

and Engineering

Expected Output Behavior

• Booking confirmation with seat map summary; cancellation receipt; updated availability.

Questions for Students:

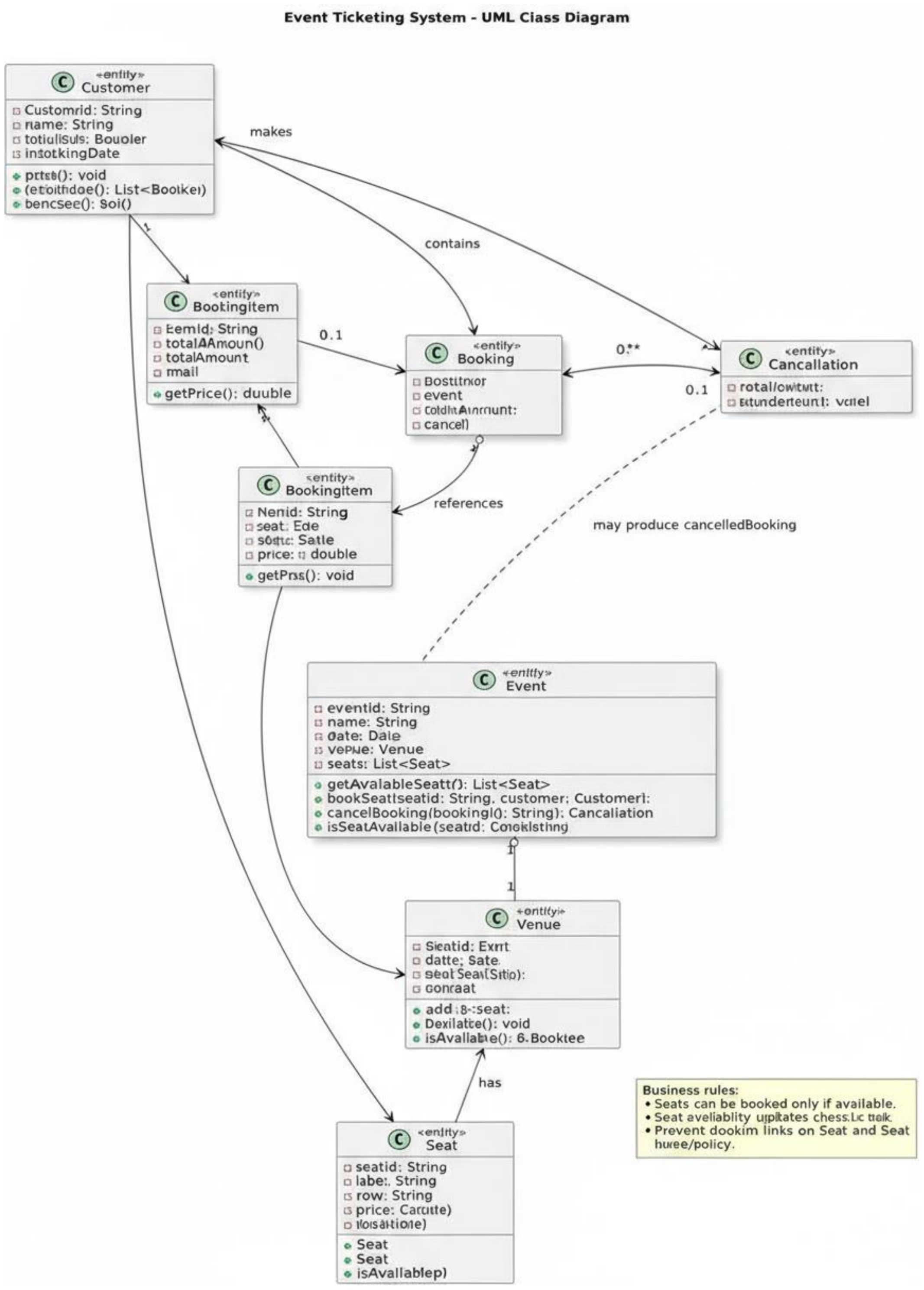
1. Draw the UML Class Diagram for the system.
2. Implement the classes with the necessary data members and methods for

system functionality and business rules.

3. Use encapsulation, inheritance, aggregation, and polymorphism wherever required.

4. Implement the main method for a menu-driven system.

UML DIAGRAM:


Event Ticketing System - UML Class Diagram
«entity» Customer
Customrid: String
name: String
totiulisuls: Bouoler
instotkingDate
prtst(): void
(etoithdoe(): List<Bootkel)
bencsee(): Soi()
makes
contains
«entity» BookingItem
Eemld: String
totalAAmoun()
totalAmount
mail
getPrice(): duuble
0.1
«entity» Booking
Bostitnor
event
coldltAinrunt:
cancel)
0.**
0.1
«entity» Cancallation
rotal/owtwtt:
Etunderteunl: vatel
«entity» BookingItem
Nenid: String
seat: Ede
söttc: Satle
price: double
getPrss(): void
references
may produce cancelledBooking
«entity» Event
eventld: String
name: String
date: Dale
venue: Venue
seats: List<Seat>
getAvalableSeatt(): List<Seat>
bookSeatIseatid: String, customer: Customerl:
cancelBooking(bookingl(): String): Cancaliation
isSeatAvallable(seatid: Conokisthng
1
1
«entity» Venue
Sieatid: Exrrt
datte: Sate
seat Sean(Stip):
conraat
add 8-:seat:
Dexilatce(): void
isAvallable(): 6-Booktee
has
«entity» Seat
seatid: String
label: String
row: String
price: Cacutte)
lloisaitione)
Seat
Seat
isAvallablep)
Business rules:
• Seats can be booked only if available.
• Seat aveliablity upltates chess.Lc ttalk.
• Prevent dookim links on Seat and Seat huwee/policy.

SOURCE CODE:

```
package EventTicketingSystem;

import java.time.*;
import java.time.format.DateTimeFormatter;
import java.util.*;
import java.util.concurrent.atomic.AtomicInteger;

public class EventTicketing {

    private static AtomicInteger venueCounter = new AtomicInteger(1);
    private static AtomicInteger eventCounter = new AtomicInteger(1);
    private static AtomicInteger customerCounter = new AtomicInteger(1);
    private static AtomicInteger bookingCounter = new AtomicInteger(1);
    private static AtomicInteger cancellationCounter = new AtomicInteger(1);

    private Map<String, Venue> venues = new HashMap<>();
    private Map<String, Event> events = new HashMap<>();
    private Map<String, Customer> customers = new HashMap<>();
    private Map<String, Booking> bookings = new HashMap<>();
    private Map<String, Cancellation> cancellations = new HashMap<>();

    private Scanner scanner = new Scanner(System.in);
    private DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm");

    public static void main(String[] args) {
        EventTicketing app = new EventTicketing();
        app.loadSampleData();
        app.menuLoop();
    }

    private void menuLoop() {
        boolean running = true;
        while (running) {
            System.out.println("\n=== Event Ticketing System ===");
            System.out.println("1) Add Venue & Seats");
            System.out.println("2) Add Event");
            System.out.println("3) Register Customer");
            System.out.println("4) Book Seats");
            System.out.println("5) Cancel Booking");
            System.out.println("6) Display Events & Availability");
            System.out.println("7) Display Bookings");
            System.out.println("8) Exit");
            System.out.print("Choose option: ");

            String choice = scanner.nextLine().trim();
            switch (choice) {
                case "1": addVenueAndSeats(); break;
                case "2": addEvent(); break;
case "3": registerCustomer(); break;
                case "4": bookSeats(); break;
                case "5": cancelBooking(); break;
                case "6": displayEventsAndAvailability(); break;
                case "7": displayAllBookings(); break;
                case "8": running = false; System.out.println("Goodbye!"); break;
                default: System.out.println("Invalid option");
            }
        }
    }

private void loadSampleData() {
        Venue v = new Venue("Venue A", "City Center");
        for (char r = 'A'; r <= 'C'; r++) {
            for (int c = 1; c <= 5; c++) {
                double price = switch (r) {
                    case 'A' -> 1700;
                    case 'B' -> 1000;
                    default -> 600;
                };
                v.addSeat(new Seat(r + String.valueOf(c), "Row " + r + " Seat " + c, price));
            }
        }
        venues.put(v.getId(), v);

        LocalDateTime start =
LocalDateTime.now().plusDays(10).withHour(19).withMinute(0);
        LocalDateTime end = start.plusHours(2);
        Event e = new Event("Rock Night", v, start, end);
        events.put(e.getId(), e);

Customer c1 = new Customer("Alice", "alice@example.com", "9999999999");
        customers.put(c1.getId(), c1);
    }

    private void addVenueAndSeats() {
        System.out.print("Venue name: ");
        String name = scanner.nextLine().trim();
        System.out.print("Venue location: ");
        String location = scanner.nextLine().trim();

        Venue v = new Venue(name, location);

        System.out.print("Rows (e.g. A-C): ");
        String rowsInput = scanner.nextLine().trim();
        System.out.print("Columns (e.g. 1-10): ");
        String colsInput = scanner.nextLine().trim();

        char startRow = 'A', endRow = 'A';
        int startCol = 1, endCol = 5;
        try {
            String[] rs = rowsInput.split("-");
            startRow = rs[0].charAt(0);
```

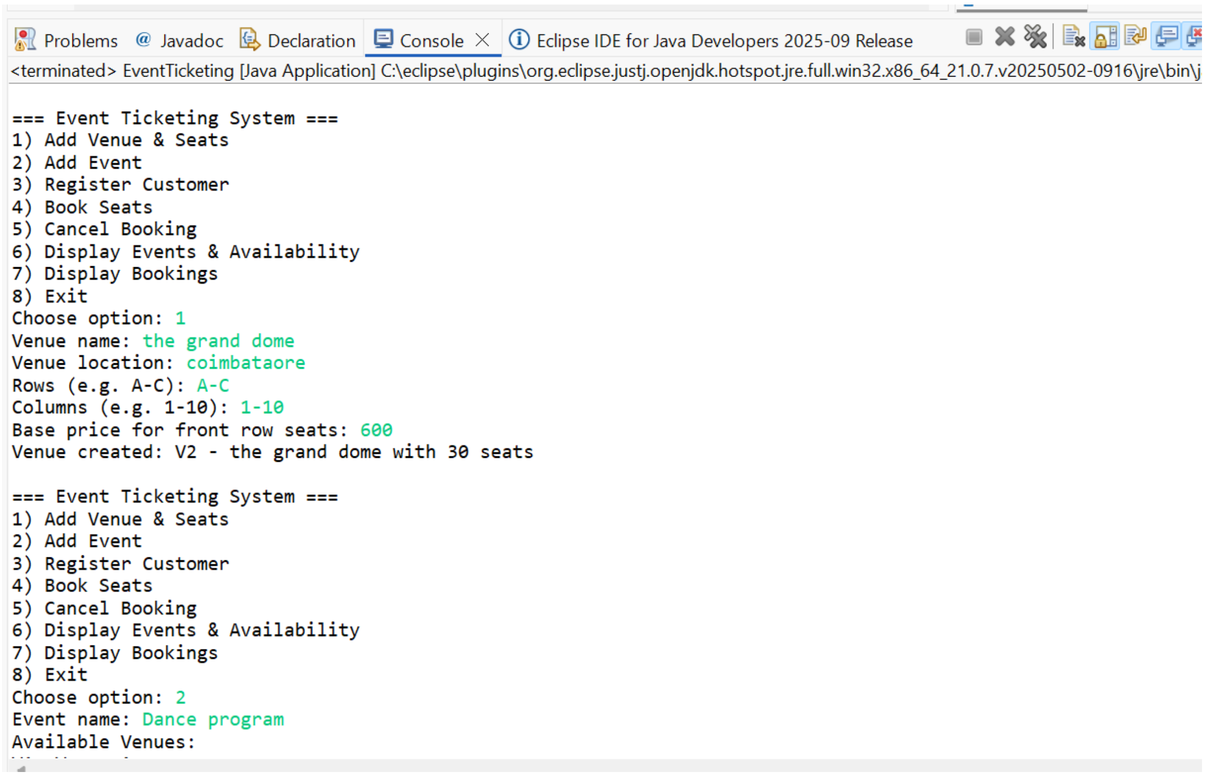
```
startRow = rs[0].charAt(0);
            endRow = rs[1].charAt(0);
            String[] cs = colsInput.split("-");
            startCol = Integer.parseInt(cs[0]);
            endCol = Integer.parseInt(cs[1]);
        } catch (Exception ex) {
            System.out.println("Invalid format, using default A-1..A-5.");
        }

        System.out.print("Base price for front row seats: ");
        double frontPrice = readDoubleOrDefault(1000);

        for (char r = startRow; r <= endRow; r++) {
            for (int c = startCol; c <= endCol; c++) {
                double price = frontPrice - (r - startRow) * 200;
                if (price < 100) price = 100;
                v.addSeat(new Seat(r + String.valueOf(c), "Row " + r + " Seat " + c, price));
            }
        }
        venues.put(v.getId(), v);
        System.out.println("Venue created: " + v.getId() + " - " + v.getName() + " with " +
v.getSeats().size() + " seats");
    }
private void addEvent() {
        System.out.print("Event name: ");
        String name = scanner.nextLine().trim();
        System.out.println("Available Venues:");
        venues.values().forEach(v -> System.out.println(v.getId() + ": " + v.getName()));
        System.out.print("Enter venue ID: ");
        String venueId = scanner.nextLine().trim();

        Venue venue = venues.get(venueId);
        if (venue == null) {
            System.out.println("Invalid venue ID");
            return;
        }
   System.out.print("Start DateTime (yyyy-MM-dd HH:mm): ");
        LocalDateTime start = LocalDateTime.parse(scanner.nextLine().trim(), dtf);
        System.out.print("End DateTime (yyyy-MM-dd HH:mm): ");
        LocalDateTime end = LocalDateTime.parse(scanner.nextLine().trim(), dtf);

        Event e = new Event(name, venue, start, end);
        events.put(e.getId(), e);
        System.out.println("Event created: " + e.getId());
    }

    private void registerCustomer() {
        System.out.print("Name: ");
        String name = scanner.nextLine().trim();
        System.out.print("Email: ");
        String email = scanner.nextLine().trim();
```

```
System.out.print("Phone: ");
        String phone = scanner.nextLine().trim();
        Customer c = new Customer(name, email, phone);
        customers.put(c.getId(), c);
        System.out.println("Customer registered: " + c.getId());
    }

    private void bookSeats() {
        System.out.println("Available Events:");
        events.values().forEach(e -> System.out.println(e.getId() + ": " + e.getName()));
        System.out.print("Enter event ID: ");
        String eventId = scanner.nextLine().trim();
        Event e = events.get(eventId);
        if (e == null) {
            System.out.println("Invalid event ID");
            return;
        }

        System.out.println("Available Customers:");
        customers.values().forEach(c -> System.out.println(c.getId() + ": " + c.getName()));
        System.out.print("Enter customer ID: ");
        String customerId = scanner.nextLine().trim();
        Customer cust = customers.get(customerId);
        if (cust == null) {
            System.out.println("Invalid customer ID");
            return;
        }
System.out.println("Available seats:");
        e.getVenue().getSeats().values().forEach(s -> System.out.println(s.getSeatId() + " - " +
s.getDescription() + " - ₹" + s.getPrice()));

        System.out.print("Enter seat IDs (comma separated): ");
        String[] seatIds = scanner.nextLine().split(",");
        Booking booking = new Booking(cust, e);
        for (String sid : seatIds) {
            Seat seat = e.getVenue().getSeats().get(sid.trim());
            if (seat != null) booking.addSeat(seat);
        }
        bookings.put(booking.getId(), booking);
        System.out.println("Booking successful! ID: " + booking.getId());
    }
private void cancelBooking() {
        System.out.print("Enter booking ID to cancel: ");
        String id = scanner.nextLine().trim();
        Booking b = bookings.get(id);
```

```
if (b == null) {
            System.out.println("Invalid booking ID");
            return;
        }
        Cancellation c = new Cancellation(b);
        cancellations.put(c.getId(), c);
        bookings.remove(id);
        System.out.println("Booking cancelled. Cancellation ID: " + c.getId());
    }

    private void displayEventsAndAvailability() {
        for (Event e : events.values()) {
            System.out.println(e);
        }
    }
private void displayAllBookings() {
        for (Booking b : bookings.values()) {
            System.out.println(b);
        }
    }

    private double readDoubleOrDefault(double def) {
        try {
            return Double.parseDouble(scanner.nextLine().trim());
        } catch (Exception e) {
            return def;
        }
    }
}
class Venue {
    private static AtomicInteger counter = new AtomicInteger(1);
    private String id;
    private String name;
    private String location;
    private Map<String, Seat> seats = new HashMap<>();

    public Venue(String name, String location) {
        this.id = "V" + counter.getAndIncrement();
        this.name = name;
        this.location = location;
    }

    public void addSeat(Seat seat) {
        seats.put(seat.getSeatId(), seat);
    }

    public String getId() { return id; }
    public String getName() { return name; }
    public Map<String, Seat> getSeats() { return seats; }

    @Override
    public String toString() {
        return id + " - " + name + " (" + location + ")";
    }
}

class Seat {
    private String seatId;
    private String description;
    private double price;

    public Seat(String seatId, String description, double price) {
        this.seatId = seatId;
        this.description = description;
        this.price = price;
    }
public String getSeatId() { return seatId; }
    public String getDescription() { return description; }
    public double getPrice() { return price; }

    @Override
    public String toString() {
        return seatId + ": " + description + " (₹" + price + ")";
    }
}

class Event {
    private static AtomicInteger counter = new AtomicInteger(1);
    private String id;
    private String name;
    private Venue venue;
    private LocalDateTime start;
    private LocalDateTime end;

    public Event(String name, Venue venue, LocalDateTime start, LocalDateTime end) {
        this.id = "E" + counter.getAndIncrement();
        this.name = name;
        this.venue = venue;
        this.start = start;
        this.end = end;
    } public String getId() { return id; }
    public String getName() { return name; }
    public Venue getVenue() { return venue; }

    @Override
    public String toString() {
        return id + ": " + name + " at " + venue.getName() + " [" + start + " - " + end + "]";
    }
}

class Customer {
    private static AtomicInteger counter = new AtomicInteger(1);
    private String id;
    private String name;
    private String email;
    private String phone;

    public Customer(String name, String email, String phone) {
        this.id = "C" + counter.getAndIncrement();
```

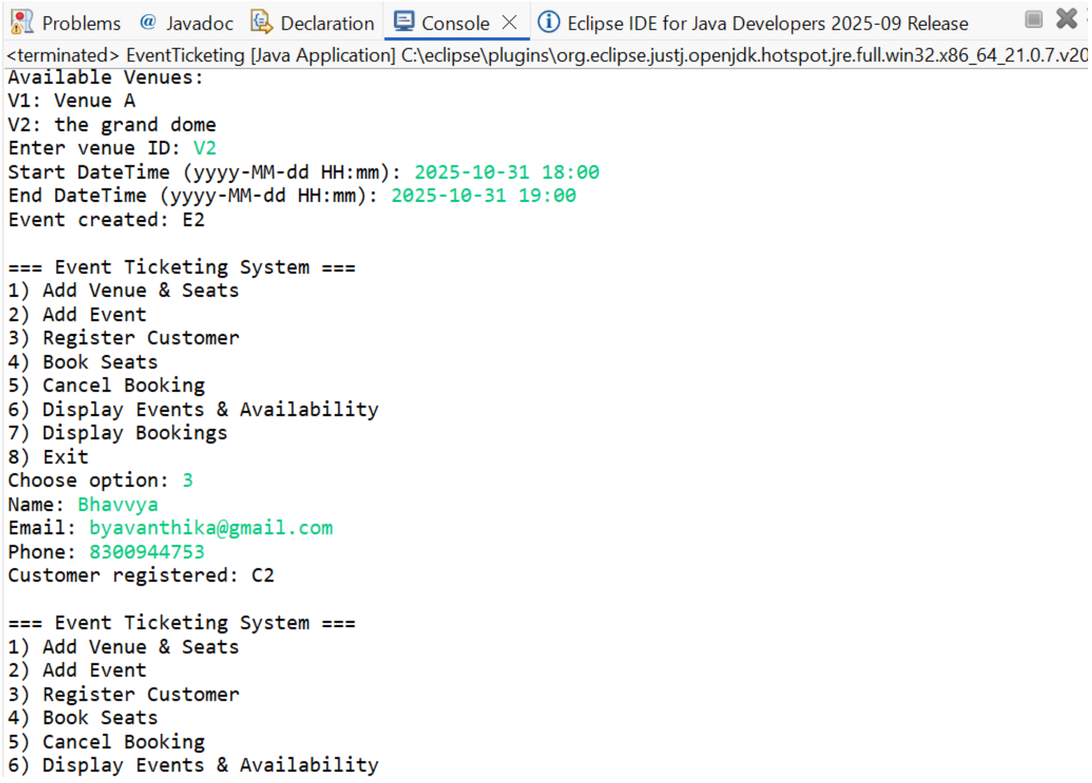
```
            this.name = name;
            this.email = email;
            this.phone = phone;
    }
        public String getId() { return id; }
          public String getName() { return name; }

          @Override
          public String toString() {
            return id + ": " + name + " (" + email + ")";
          }
        }

        class Booking {
          private static AtomicInteger counter = new AtomicInteger(1);
          private String id;
          private Customer customer;
          private Event event;
          private List<Seat> bookedSeats = new ArrayList<>();

          public Booking(Customer customer, Event event) {
            this.id = "B" + counter.getAndIncrement();
            this.customer = customer;
            this.event = event;
          }

          public void addSeat(Seat s) {
            bookedSeats.add(s);
          }

          public String getId() { return id; }

          @Override
          public String toString() {
            return id + " - " + customer.getName() + " booked " + bookedSeats.size() + " seats for
        " + event.getName();
          }
        } class Cancellation {
          private static AtomicInteger counter = new AtomicInteger(1);
          private String id;
          private Booking booking;
          private LocalDateTime cancelledOn;

          public Cancellation(Booking booking) {
            this.id = "X" + counter.getAndIncrement();
            this.booking = booking;
            this.cancelledOn = LocalDateTime.now();
          }

          public String getId() { return id; }

          @Override
          public String toString() {
            return id + ": Cancelled booking " + booking.getId() + " on " + cancelledOn;
          }
}
```

OUTPUT:



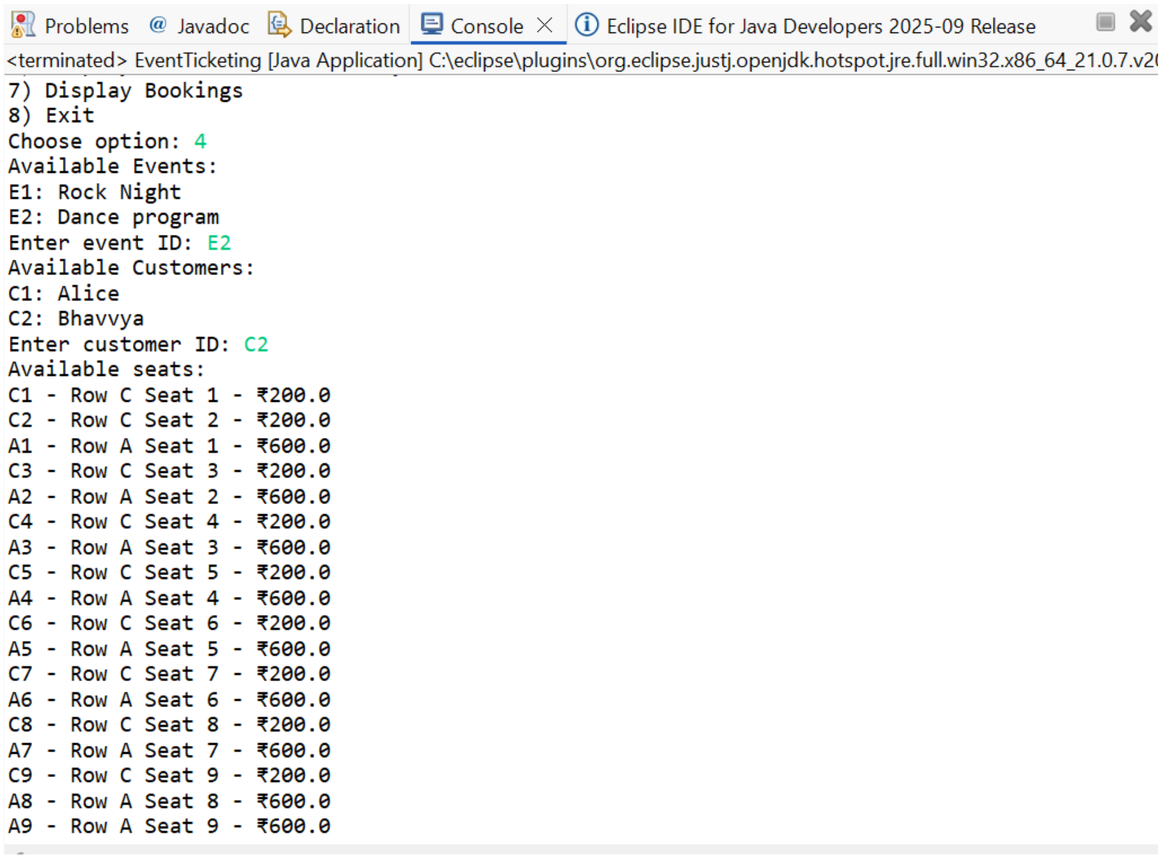
Problems @ Javadoc Declaration Console Eclipse IDE for Java Developers 2025-09 Release

```
<terminated> EventTicketing [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20250502-0916\jre\bin\j

=== Event Ticketing System ===
1) Add Venue & Seats
2) Add Event
3) Register Customer
4) Book Seats
5) Cancel Booking
6) Display Events & Availability
7) Display Bookings
8) Exit
Choose option: 1
Venue name: the grand dome
Venue location: coimbataore
Rows (e.g. A-C): A-C
Columns (e.g. 1-10): 1-10
Base price for front row seats: 600
Venue created: V2 - the grand dome with 30 seats

=== Event Ticketing System ===
1) Add Venue & Seats
2) Add Event
3) Register Customer
4) Book Seats
5) Cancel Booking
6) Display Events & Availability
7) Display Bookings
8) Exit
Choose option: 2
Event name: Dance program
Available Venues:
```



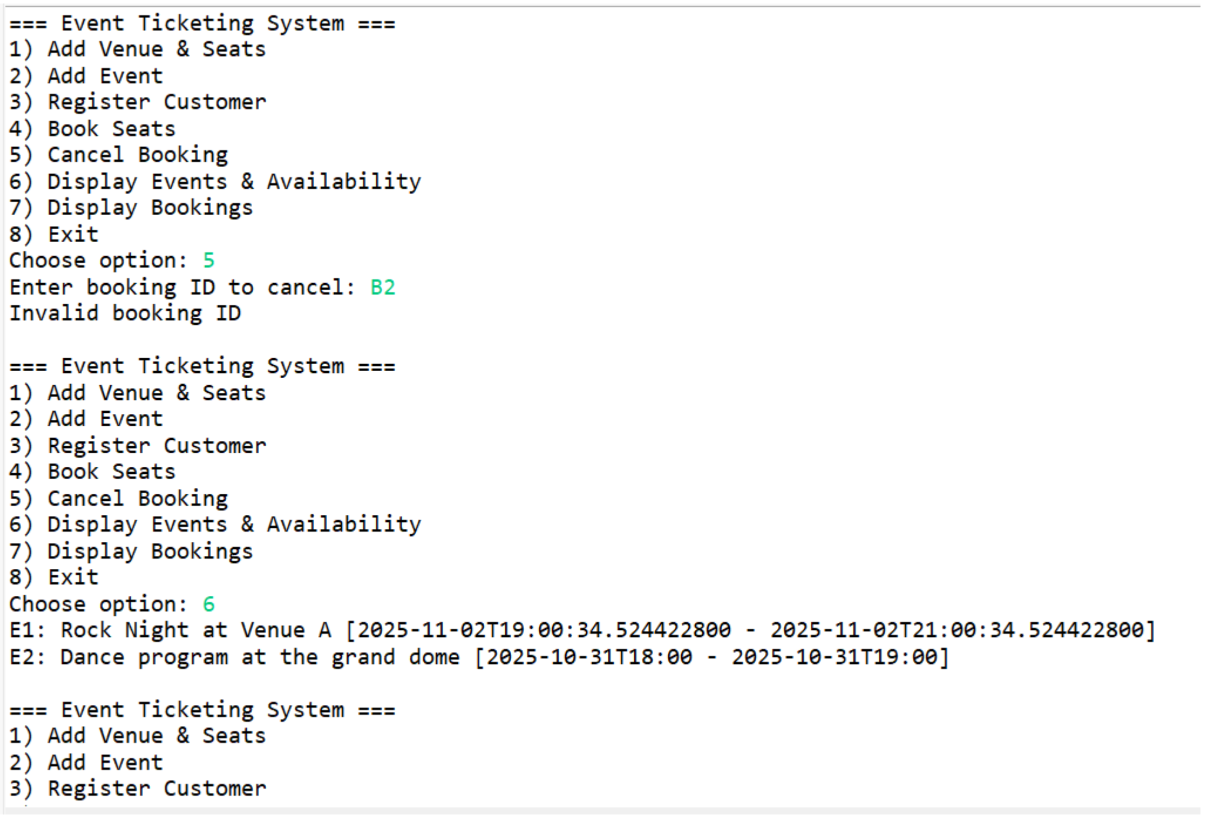
Problems @ Javadoc Declaration Console Eclipse IDE for Java Developers 2025-09 Release

```
<terminated> EventTicketing [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v20
Available Venues:
V1: Venue A
V2: the grand dome
Enter venue ID: V2
Start DateTime (yyyy-MM-dd HH:mm): 2025-10-31 18:00
End DateTime (yyyy-MM-dd HH:mm): 2025-10-31 19:00
Event created: E2

=== Event Ticketing System ===
1) Add Venue & Seats
2) Add Event
3) Register Customer
4) Book Seats
5) Cancel Booking
6) Display Events & Availability
7) Display Bookings
8) Exit
Choose option: 3
Name: Bhavvya
Email: byavanthika@gmail.com
Phone: 8300944753
Customer registered: C2

=== Event Ticketing System ===
1) Add Venue & Seats
2) Add Event
3) Register Customer
4) Book Seats
5) Cancel Booking
6) Display Events & Availability
```

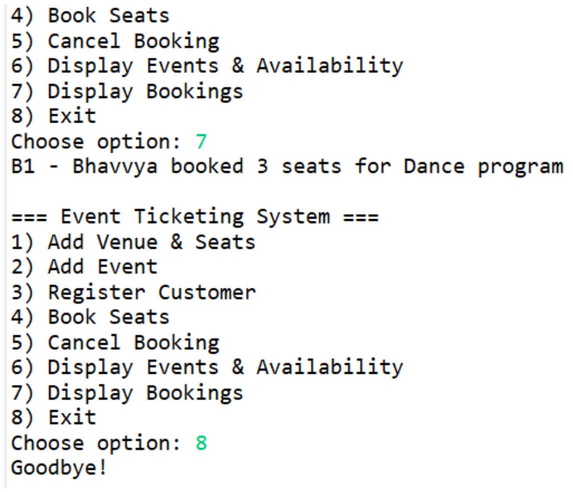


<terminated> EventTicketing [Java Application] C:\eclipse\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_21.0.7.v2

```
7) Display Bookings
8) Exit
Choose option: 4
Available Events:
E1: Rock Night
E2: Dance program
Enter event ID: E2
Available Customers:
C1: Alice
C2: Bhavvya
Enter customer ID: C2
Available seats:
C1 - Row C Seat 1 - ₹200.0
C2 - Row C Seat 2 - ₹200.0
A1 - Row A Seat 1 - ₹600.0
C3 - Row C Seat 3 - ₹200.0
A2 - Row A Seat 2 - ₹600.0
C4 - Row C Seat 4 - ₹200.0
A3 - Row A Seat 3 - ₹600.0
C5 - Row C Seat 5 - ₹200.0
A4 - Row A Seat 4 - ₹600.0
C6 - Row C Seat 6 - ₹200.0
A5 - Row A Seat 5 - ₹600.0
C7 - Row C Seat 7 - ₹200.0
A6 - Row A Seat 6 - ₹600.0
C8 - Row C Seat 8 - ₹200.0
A7 - Row A Seat 7 - ₹600.0
C9 - Row C Seat 9 - ₹200.0
A8 - Row A Seat 8 - ₹600.0
A9 - Row A Seat 9 - ₹600.0
```



```
=== Event Ticketing System ===
1) Add Venue & Seats
2) Add Event
3) Register Customer
4) Book Seats
5) Cancel Booking
6) Display Events & Availability
7) Display Bookings
8) Exit
Choose option: 5
Enter booking ID to cancel: B2
Invalid booking ID

=== Event Ticketing System ===
1) Add Venue & Seats
2) Add Event
3) Register Customer
4) Book Seats
5) Cancel Booking
6) Display Events & Availability
7) Display Bookings
8) Exit
Choose option: 6
E1: Rock Night at Venue A [2025-11-02T19:00:34.524422800 - 2025-11-02T21:00:34.524422800]
E2: Dance program at the grand dome [2025-10-31T18:00 - 2025-10-31T19:00]

=== Event Ticketing System ===
1) Add Venue & Seats
2) Add Event
3) Register Customer
```



```
4) Book Seats
5) Cancel Booking
6) Display Events & Availability
7) Display Bookings
8) Exit
Choose option: 7
B1 - Bhavvya booked 3 seats for Dance program

=== Event Ticketing System ===
1) Add Venue & Seats
2) Add Event
3) Register Customer
4) Book Seats
5) Cancel Booking
6) Display Events & Availability
7) Display Bookings
8) Exit
Choose option: 8
Goodbye!
```

GITHUB LINK:

https://github.com/