Experiment 4

Student Name: Ujjwal Kumar **UID:22BCS14185**
Branch:CSE **Section:640`B**
Semester:6th **DOP:17/02/25**
Subject:Java **SubjectCode:22CSH-359**

Aim: Develop a ticket booking system with synchronized threads to ensure no double booking of seats. Use thread priorities to simulate VIP bookings being processed first.

Objective:
- Ensure that no two users can book the same seat simultaneously.
- Use synchronized methods to prevent race conditions.
- Use thread priorities to simulate VIP customers' bookings being processed first.

Algorithm:
- We'll use a `TicketBooking` class with synchronized methods to book tickets and display seat availability.
- Use multiple threads: Regular and VIP customers will book seats, and VIP customers will be given higher priority.
- The system will simulate booking a limited number of seats, and no seat will be booked twice.

Code:

```
class TicketBooking {
    private int availableSeats;

    TicketBooking(int totalSeats) {
        this.availableSeats = totalSeats;
    }

    // Synchronized method to book a seat
    synchronized void bookSeat(String customerName, boolean isVIP) {
        // Simulate a VIP customer by checking thread priority
        if (availableSeats > 0) {
            availableSeats--;
            System.out.println(customerName + " booked a seat. Seats remaining: " + availableSeats);
        } else {
            System.out.println("No available seats for " + customerName);
        }
    }

    // Method to check the available seats (not synchronized, for informational purposes)
    int getAvailableSeats() {
        return availableSeats;
    }
}

class CustomerThread extends Thread {
    private TicketBooking ticketBooking;
```

```
    private String customerName;
    private boolean isVIP;

    CustomerThread(TicketBooking ticketBooking, String customerName, boolean isVIP) {
        this.ticketBooking = ticketBooking;
        this.customerName = customerName;
        this.isVIP = isVIP;
    }

    @Override
    public void run() {
        // Simulate booking process
        try {
            // VIP customers are processed first due to their higher priority
            ticketBooking.bookSeat(customerName, isVIP);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    }
}

public class TicketBookingSystem {
    public static void main(String[] args) {
        // Create a TicketBooking object with 5 available seats
        TicketBooking ticketBooking = new TicketBooking(5);

        // Create threads for regular and VIP customers
        CustomerThread vipCustomer1 = new CustomerThread(ticketBooking, "VIP Customer 1", true);
        CustomerThread vipCustomer2 = new CustomerThread(ticketBooking, "VIP Customer 2", true);
        CustomerThread regularCustomer1 = new CustomerThread(ticketBooking, "Regular Customer 1", false);
        CustomerThread regularCustomer2 = new CustomerThread(ticketBooking, "Regular Customer 2", false);
        CustomerThread regularCustomer3 = new CustomerThread(ticketBooking, "Regular Customer 3", false);

        // Set thread priorities: VIP customers have higher priority
        vipCustomer1.setPriority(Thread.MAX_PRIORITY);  // Highest priority
        vipCustomer2.setPriority(Thread.MAX_PRIORITY);  // Highest priority
        regularCustomer1.setPriority(Thread.NORM_PRIORITY);  // Normal priority
        regularCustomer2.setPriority(Thread.NORM_PRIORITY);  // Normal priority
        regularCustomer3.setPriority(Thread.NORM_PRIORITY);  // Normal priority

        // Start all customer threads
        vipCustomer1.start();
        vipCustomer2.start();
        regularCustomer1.start();
        regularCustomer2.start();
        regularCustomer3.start();
    }
}
```

OUTPUT:

```
VIP Customer 1 booked a seat. Seats remaining: 4
VIP Customer 2 booked a seat. Seats remaining: 3
Regular Customer 1 booked a seat. Seats remaining: 2
Regular Customer 2 booked a seat. Seats remaining: 1
Regular Customer 3 booked a seat. Seats remaining: 0
```

Learning Outcomes:

- **Synchronized Methods**:
  - The `bookSeat()` method is synchronized to ensure that only one thread (customer) can book a seat at a time, avoiding double booking.
- **Thread Priorities**:
  - VIP customers are processed before regular customers by setting their thread priority to the maximum (`Thread.MAX_PRIORITY`).
- **Thread Safety**:
  - The use of synchronization ensures thread safety in a multi-threaded environment, preventing issues like double booking and race conditions.