Hard Level: Ticket Booking System with Multithreading Problem Statement 📝
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.

Key Concepts Used 🛠️ Multithreading: To handle multiple booking requests simultaneously.

Synchronization: To prevent double booking of seats.

Thread Priorities: To prioritize VIP bookings over regular bookings.

Code:

```
import java.util.*;

class TicketBookingSystem {
    private final int totalSeats;
    private final Set<Integer> bookedSeats = new HashSet<>();

    public TicketBookingSystem(int totalSeats) {
        this.totalSeats = totalSeats;
    }
    public synchronized boolean bookSeat(int seatNumber, String
customerType) {
        if (seatNumber < 1 || seatNumber > totalSeats) {
            System.out.println(customerType + " Booking Failed: Invalid
seat number " + seatNumber);
            return false;
        }
        if (!bookedSeats.contains(seatNumber)) {
            bookedSeats.add(seatNumber);
            System.out.println(customerType + " Booking Successful: Seat "
+ seatNumber);
            return true;
        } else {
            System.out.println(customerType + " Booking Failed: Seat " +
seatNumber + " is already booked.");
            return false;
        }
    }
}
class Customer extends Thread {
    private final TicketBookingSystem system;
    private final int seatNumber;
    private final String customerType;
    public Customer(TicketBookingSystem system, int seatNumber, String
customerType, int priority) {
        this.system = system;
        this.seatNumber = seatNumber;
        this.customerType = customerType;
        this.setPriority(priority);
    }
    @Override
    public void run() {
        system.bookSeat(seatNumber, customerType);
    }
}
public class TicketBookingApp {
```

```java
    public static void main(String[] args) {
        TicketBookingSystem system = new TicketBookingSystem(10); // 10
seats available
        List<Customer> customers = new ArrayList<>();
        customers.add(new Customer(system, 1, "VIP", Thread.MAX_PRIORITY));
        customers.add(new Customer(system, 2, "Regular",
Thread.NORM_PRIORITY));
        customers.add(new Customer(system, 1, "Regular",
Thread.NORM_PRIORITY)); // Double booking attempt
        customers.add(new Customer(system, 3, "VIP", Thread.MAX_PRIORITY));
        customers.add(new Customer(system, 4, "Regular",
Thread.MIN_PRIORITY));
        for (Customer customer : customers) {
            customer.start();
        }
    }
}
```

Output:

```
Run    TicketBookingApp

"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaag
VIP Booking Successful: Seat 1
Regular Booking Successful: Seat 4
VIP Booking Successful: Seat 3
Regular Booking Failed: Seat 1 is already booked.
Regular Booking Successful: Seat 2

Process finished with exit code 0
```