

Java Assignment 4 Hard Level

Problem Statement :

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.

How to Run  Navigate to the Hard/ folder.

Compile and run the TicketBookingSystem.java file.

Observe how VIP bookings are prioritized and how synchronization prevents double booking.

Code :

```
import java.util.*;

class TicketBookingSystem {
    private final int totalSeats;
    private final boolean[] seats;
    public TicketBookingSystem(int totalSeats) {
        this.totalSeats = totalSeats;
        this.seats = new boolean[totalSeats];
    }
    public synchronized boolean bookSeat(int seatNumber) {
        if (seatNumber < 0 || seatNumber >= totalSeats) {
```

Name : Ayush Anand
UID : 22BCS10841

```
        System.out.println(Thread.currentThread().getName() + " - Invalid seat  
number.");
```

```
        return false;
```

```
    }
```

```
    if (!seats[seatNumber]) {
```

```
        seats[seatNumber] = true;
```

```
        System.out.println(Thread.currentThread().getName() + " - Successfully  
booked seat " + seatNumber);
```

```
        return true;
```

```
    } else {
```

```
        System.out.println(Thread.currentThread().getName() + " - Seat " +  
seatNumber + " is already booked.");
```

```
        return false;
```

```
    }}}
```

```
class BookingThread extends Thread {
```

```
    private final TicketBookingSystem system;
```

```
    private final int seatNumber;
```

```
    public BookingThread(TicketBookingSystem system, int seatNumber, String  
name, int priority) {
```

```
        super(name);
```

```
        this.system = system;
```

```
        this.seatNumber = seatNumber;
```

```
        setPriority(priority);}
```

```
    @Override
```

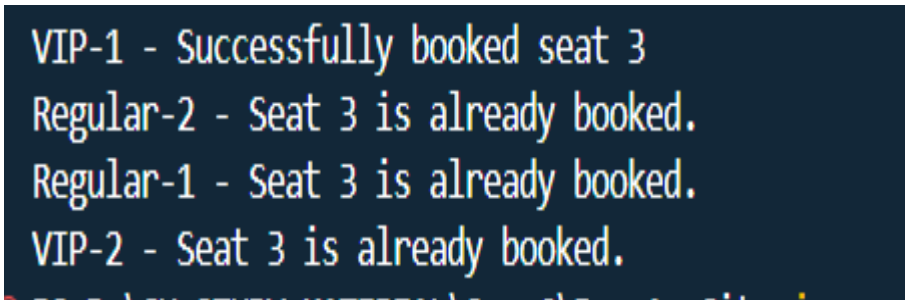
```
    public void run() {
```

```
        system.bookSeat(seatNumber);
```

Name : Ayush Anand
UID : 22BCS10841

```
    }}  
  
    public class TicketBookingMain {  
        public static void main(String[] args) {  
            TicketBookingSystem system = new TicketBookingSystem(10);  
            BookingThread vip1 = new BookingThread(system, 3, "VIP-1",  
Thread.MAX_PRIORITY);  
            BookingThread vip2 = new BookingThread(system, 3, "VIP-2",  
Thread.MAX_PRIORITY);  
            BookingThread regular1 = new BookingThread(system, 3, "Regular-1",  
Thread.NORM_PRIORITY);  
            BookingThread regular2 = new BookingThread(system, 3, "Regular-2",  
Thread.NORM_PRIORITY);  
            vip1.start();  
            vip2.start();  
            regular1.start();  
            regular2.start();  
        }}  
    }
```

OUTPUT:

A screenshot of a terminal window with a dark background and light-colored text. It displays the output of a Java program where four threads attempt to book seat 3. The output shows that the first thread (VIP-1) successfully books the seat, while the subsequent three threads (Regular-2, Regular-1, and VIP-2) fail because the seat is already booked.

```
VIP-1 - Successfully booked seat 3  
Regular-2 - Seat 3 is already booked.  
Regular-1 - Seat 3 is already booked.  
VIP-2 - Seat 3 is already booked.
```