



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

## Experiment - 4

**Student Name:** Ronak Jain

**Branch:** BE-CSE

**Semester:** 5<sup>th</sup>

**Subject Name:** Project Based Learning in Java

**Subject Code:** 23CSH-304

**UID:** 23BCS10225

**Section/Group:** KRG-2B

**Date of Performance:** 23/9/25

**Aim:** To 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:** To understand multithreading, thread synchronization, and thread priorities in Java.

**Input Used:** Thread, synchronized method, setPriority(), ticket counter simulation.

### **Procedure:**

1. Create a TicketBooking class with synchronized bookTicket() method.
2. Use a Thread class to simulate customers (normal and VIP).
3. Create threads with different priorities.
4. Start threads and observe how VIPs are handled first due to higher priority.
5. Ensure no 2 threads can book the same seat using synchronization.

### **Sample Input -**

Thread 1: Normal User - Booking Seat 1

Thread 2: VIP User - Booking Seat 1

### **Sample Output -**

VIP Thread booked Seat 1

Normal Thread could not book. Seat already booked.

### **Code -**

```
class TicketBooking {
    private boolean isBooked = false;

    public synchronized void bookTicket(String userType, String threadName) {
        if (!isBooked) {
            System.out.println(userType + " " + threadName + " booked the seat.");
            isBooked = true;
        } else {
            System.out.println(userType + " " + threadName + " could not book. Seat
already booked.");
        }
    }
}
```



# DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

Discover. Learn. Empower.

```
class Customer extends Thread {
    private TicketBooking bookingSystem;
    private String userType;

    public Customer(TicketBooking bookingSystem, String userType) {
        this.bookingSystem = bookingSystem;
        this.userType = userType;
    }

    public void run() {
        bookingSystem.bookTicket(userType, Thread.currentThread().getName());
    }
}

public class Main {
    public static void main(String[] args) {
        TicketBooking booking = new TicketBooking();

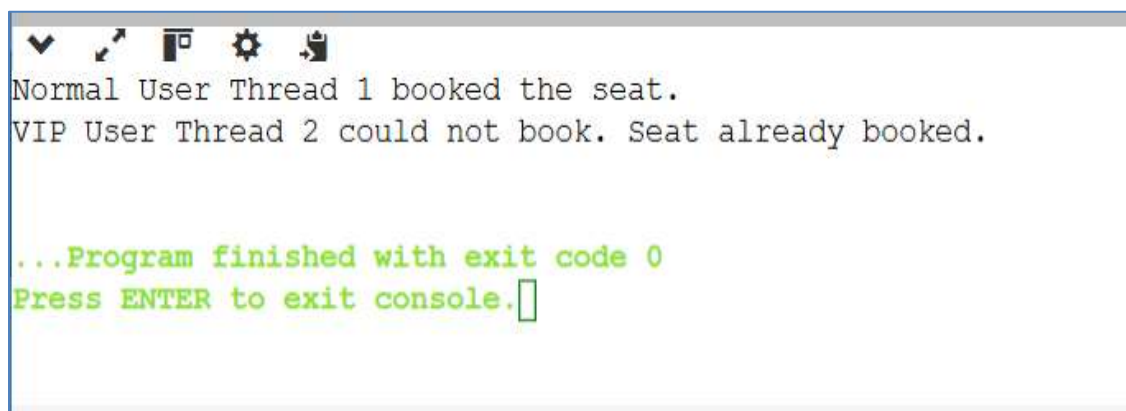
        Customer normalUser = new Customer(booking, "Normal User");
        normalUser.setName("Thread 1");

        Customer vipUser = new Customer(booking, "VIP User");
        vipUser.setName("Thread 2");

        normalUser.setPriority(Thread.MIN_PRIORITY);
        vipUser.setPriority(Thread.MAX_PRIORITY);

        normalUser.start();
        vipUser.start();
    }
}
```

## Output –



```
Normal User Thread 1 booked the seat.
VIP User Thread 2 could not book. Seat already booked.

...Program finished with exit code 0
Press ENTER to exit console.
```