Experiment 4.3

Student Name: Akshat Srivastava
UID: 22BCS11740
Branch: BE CSE
Section/Group: 22BCS_IOT_618_A
Semester: 6th
DoP: 18/02/2025
Subject Name: PBLJ Lab
Subject Code: 22CSH-359

1. **Aim:** To develop a multi-threaded Ticket Booking System in Java that ensures synchronized seat allocation while handling priority-based booking for VIP users.

2. **Objective:**
   - Implement thread synchronization to prevent double booking.
   - Use thread priorities to ensure VIP users get booking preference.
   - Handle concurrent seat booking requests efficiently.

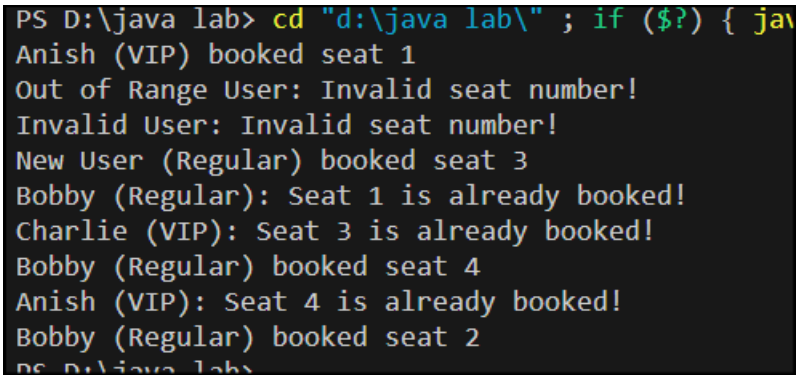
3. **Implementation/Code:**

```
class TicketBookingSystem {
    private final boolean[] seats;
    public TicketBookingSystem(int totalSeats) {
        seats = new boolean[totalSeats];
    }
    public synchronized void bookSeat(int seatNumber, String user, boolean isVIP) {
        if (seatNumber < 1 || seatNumber > seats.length) {
            System.out.println(user + ": Invalid seat number!");
            return;
        }
        if (!seats[seatNumber - 1]) {
```

```
            seats[seatNumber - 1] = true;
            System.out.println(user + " booked seat " + seatNumber);
        } else {
            System.out.println(user + ": Seat " + seatNumber + " is already booked!");
        }}}
class User extends Thread {
    private final TicketBookingSystem system;
    private final int seatNumber;
    private final String userName;
    private final boolean isVIP;
    public User(TicketBookingSystem system, int seatNumber, String userName, boolean isVIP) {
        this.system = system;
        this.seatNumber = seatNumber;
        this.userName = userName;
        this.isVIP = isVIP;
    }
    public void run() {
        system.bookSeat(seatNumber, userName, isVIP);
    }}
public class TicketMain {
    public static void main(String[] args) {
        TicketBookingSystem system = new TicketBookingSystem(5);
        User u1 = new User(system, 1, "Anish (VIP)", true);
        User u2 = new User(system, 2, "Bobby (Regular)", false);
```

```
        User u3 = new User(system, 3, "Charlie (VIP)", true);
        User u4 = new User(system, 4, "Bobby (Regular)", false);
        User u5 = new User(system, 4, "Anish (VIP)", true);
        User u6 = new User(system, 1, "Bobby (Regular)", false);
        User u7 = new User(system, 3, "New User (Regular)", false);
        User u8 = new User(system, 0, "Invalid User", false);
        User u9 = new User(system, 6, "Out of Range User", false);
        u1.setPriority(Thread.MAX_PRIORITY);
        u3.setPriority(Thread.MAX_PRIORITY);
        u5.setPriority(Thread.MAX_PRIORITY);
        u1.start();
        u2.start();
        u3.start();
        u4.start();
        u5.start();
        u6.start();
        u7.start();
        u8.start();
        u9.start();
    }}
```

4. Output



5. Learning Outcome:

- Understanding Java Multithreading and Thread Synchronization.
- Implementing synchronized methods to handle shared resources safely.
- Using thread priorities to manage execution order.