Experiment 4.3

Student Name: Somnath **UID: 22BCS12737**
Branch: BE-CSE **Section/Group: IOT-605-A**
Semester: 6th **Date of Performance:19/2/25**
Subject Name: JAVA
Subject Code: 22CSP-359

1.Aim: 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.

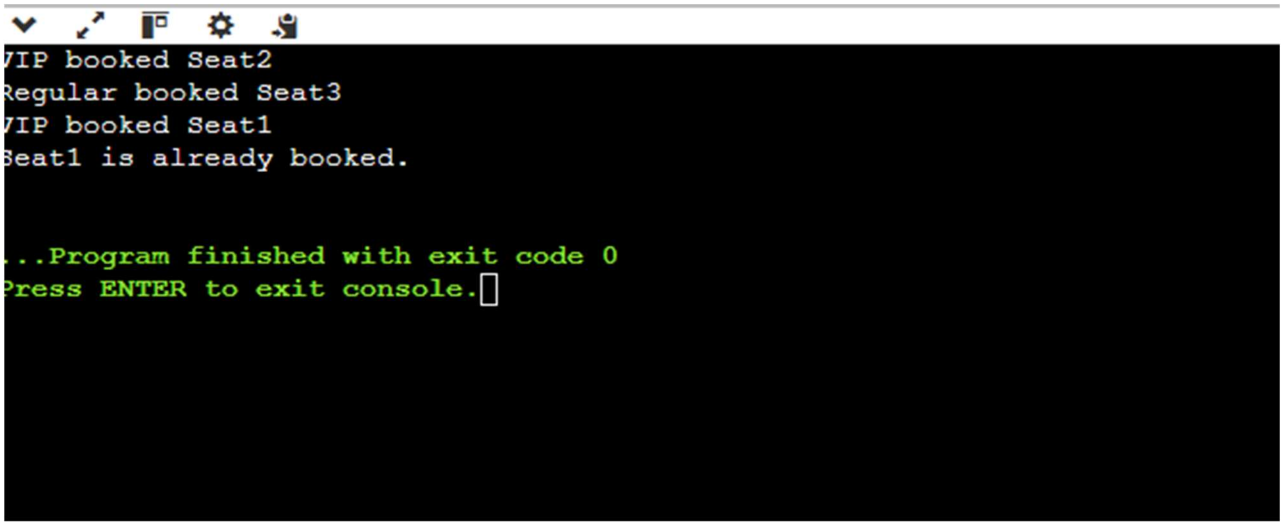
Synchronization: To prevent double booking of seats.

Thread Priorities: To prioritize VIP bookings over regular bookings.

2. Code

```
import java.util.ArrayList;
import java.util.List;

class TicketBookingSystem {
    private final List<String> availableSeats;

    public TicketBookingSystem(int totalSeats) {
        availableSeats = new ArrayList<>();
        for (int i = 1; i <= totalSeats; i++) {
            availableSeats.add("Seat" + i);
        }
    }
    public synchronized boolean bookSeat(String seat, String customerType) {
        if (availableSeats.contains(seat)) {
            System.out.println(customerType + " booked " + seat);
            availableSeats.remove(seat);
            return true;
        } else {
            System.out.println(seat + " is already booked.");
            return false;
        }
    }
}

class BookingThread extends Thread {
    private final TicketBookingSystem bookingSystem;
    private final String seat;
    private final String customerType;

    public BookingThread(TicketBookingSystem bookingSystem, String seat, String
     customerType) {
        this.bookingSystem = bookingSystem;
        this.seat = seat;
        this.customerType = customerType;
    }

    @Override
    public void run() {
        bookingSystem.bookSeat(seat, customerType);
    }
}

public class TicketBookingDemo {
    public static void main(String[] args) {
        TicketBookingSystem bookingSystem = new TicketBookingSystem(10);

        // Create VIP booking threads
        Thread vip1 = new BookingThread(bookingSystem, "Seat1", "VIP");
        Thread vip2 = new BookingThread(bookingSystem, "Seat2", "VIP");

        // Create regular booking threads
        Thread regular1 = new BookingThread(bookingSystem, "Seat1", "Regular");
        Thread regular2 = new BookingThread(bookingSystem, "Seat3", "Regular");

        // Set priorities (higher value means higher priority)
        vip1.setPriority(Thread.MAX_PRIORITY);
        vip2.setPriority(Thread.MAX_PRIORITY);
        regular1.setPriority(Thread.MIN_PRIORITY);
        regular2.setPriority(Thread.MIN_PRIORITY);

        // Start threads
        vip1.start();
        vip2.start();
        regular1.start();
        regular2.start();

        // Wait for all threads to finish
        try {
            vip1.join();
            vip2.join();
            regular1.join();
            regular2.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }
    }
}
```

4.Output:



```
VIP booked Seat2
Regular booked Seat3
VIP booked Seat1
Seat1 is already booked.


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