Experiment 4.3

Student Name: Anshu Kumar **UID: 22BCS16672**
Branch: BE-CSE **Section/Group: 641/A**
Semester: 6th **Date of Performance: 14/02/2025**
Subject Name: Project Based Learning in Java with Lab **Subject Code: 22CSH-359**

1. **Aim:** Write a program to develop a **ticket booking system** using synchronized threads to ensure that no double booking of seats occurs. The program will also use **thread priorities** to simulate VIP bookings being processed first.
2. **Objective:** The objective of this program is to **implement a multi-threaded ticket booking system** where multiple users attempt to book seats concurrently. To prevent race conditions and ensure **thread safety**, the program will synchronize the booking process. Additionally, **thread priorities** will be used so that VIP bookings are handled before regular users. This program will demonstrate how to effectively manage concurrency using Java's **thread synchronization** and **priority-based execution**.
3. **Implementation/Code:**

```
import java.util.HashSet;
import java.util.Random;

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

    // Constructor to set total seats
    public TicketBookingSystem(int totalSeats) {
        this.totalSeats = totalSeats;
    }

    // Synchronized method to book a seat
    public synchronized boolean bookSeat(int seatNumber, String user) {
```

```
        if (bookedSeats.contains(seatNumber)) {
            System.out.println(user + " tried to book Seat " + seatNumber + " but it was
already booked.");
            return false;
        } else if (seatNumber < 1 || seatNumber > totalSeats) {
            System.out.println(user + " tried to book an invalid seat: " + seatNumber);
            return false;
        }
        bookedSeats.add(seatNumber);
        System.out.println(user + " successfully booked Seat " + seatNumber);
        return true;
    }
}

// Booking thread class
class BookingThread extends Thread {
    private final TicketBookingSystem system;
    private final String user;
    private final Random random = new Random();

    public BookingThread(TicketBookingSystem system, String user, int priority) {
        this.system = system;
        this.user = user;
        setPriority(priority); // Set thread priority
    }

    @Override
    public void run() {
        int seatNumber = random.nextInt(10) + 1; // Generate a seat number between 1-
10
        system.bookSeat(seatNumber, user);
    }
}
```

```
// Main class to run the booking system
public class TicketBookingMain {
    public static void main(String[] args) {
        TicketBookingSystem system = new TicketBookingSystem(10);
         System.out.println("Anshu Kumar 22BCS16672");

        // Creating users with different priorities (VIPs get higher priority)
        BookingThread vip1 = new BookingThread(system, "VIP_User_1",
Thread.MAX_PRIORITY);
        BookingThread vip2 = new BookingThread(system, "VIP_User_2",
Thread.MAX_PRIORITY);
        BookingThread user1 = new BookingThread(system, "Regular_User_1",
Thread.NORM_PRIORITY);
        BookingThread user2 = new BookingThread(system, "Regular_User_2",
Thread.NORM_PRIORITY);
        BookingThread user3 = new BookingThread(system, "Regular_User_3",
Thread.NORM_PRIORITY);
        BookingThread user4 = new BookingThread(system, "Regular_User_4",
Thread.MIN_PRIORITY);

        // Start threads
        vip1.start();
        vip2.start();
        user1.start();
        user2.start();
        user3.start();
        user4.start();

        // Wait for all threads to finish
        try {
            vip1.join();
            vip2.join();
            user1.join();
            user2.join();
```

```
            user3.join();
            user4.join();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        System.out.println("All bookings completed!");
    }
}
```

4. **Output:**

```
Anshu Kumar 22BCS16672
VIP_User_1 successfully booked Seat 8
Regular_User_4 successfully booked Seat 6
Regular_User_3 successfully booked Seat 7
Regular_User_2 successfully booked Seat 1
Regular_User_1 tried to book Seat 1 but it was already booked.
VIP_User_2 successfully booked Seat 10
All bookings completed!
```

5. **Learning Outcomes:**
   1. **Understand thread synchronization** using the synchronized keyword to avoid concurrency issues.
   2. **Learn how to manage multiple threads** in Java to handle concurrent booking requests.
   3. **Implement thread priorities** to ensure VIP customers are served first.
   4. **Practice safe data handling** using HashSet to avoid duplicate seat bookings.
   5. **Gain familiarity with real-world booking system logic** involving **multi-threading and concurrency control**.