Operating Systems

Experiment 3

Name: Kartik Jolapara **SAP ID: 60004200107**

Div.: B1 **Branch: Computer Engineering**

Aim -

Building multi-threaded and multi-process applications

Problem Statement -

1) Build an instance of bus ticket reservation system using multithreading for the following scenario.

   ABC Bus service has only two seats left for reservations. Two users are trying to book the ticket at the same time.
2) Create separate thread per user. Show how this code is leading to inconsistency
3) Improve the code by applying synchronization.
4) Conclude the experiment by stating the importance of synchronization in multiprocess and multithread application

Theory -

Multithreading

The concept of Multithreading consists of two terms – a process and a thread. A process is a program being executed. A process can be further divided into independent units known as threads. A thread is like a small light-weight process within a process. Or we can say a collection of threads is what is known as a process.

Therefore, by the above information, it is understood that Multithreading means that you have multiple threads of execution inside the same application. A thread is like a separate CPU executing your application. Thus, a multithreaded application is like an application that has multiple CPUs executing different parts of the code at the same time

Runnable Interface

Java runnable is an interface used to execute code on a concurrent thread. It is an interface which is implemented by any class if we want that the instances of that class should be executed by a thread.

The runnable interface has an undefined method run() with void as return type, and it takes in no arguments. The runnable interface provides a standard set of rules for the instances of classes which wish to execute code when they are active. The most common use case of the Runnable interface is when we want only to override the run method. When a thread is started by the object of any class which is implementing Runnable, then it invokes the run method in the separately executing thread.

Extends Thread

One way to create a thread is to create a new class that extends Thread, and then to create an instance of that class. The extending class must override the run() method, which is the entry point for the new thread. It must also call start() to begin execution of the new thread.

Code -

Using Multithreading (NO Synchronization)

```
class Passenger extends Thread {
    int seats_req;
    String name;

    Passenger(int s, String n, Reservation r) {
        super(r);
        seats_req = s;
        name = n;
```

```
    }
}

class Reservation implements Runnable {
    int seats_aval = 2;

    public void run() {
        Passenger p = (Passenger) Thread.currentThread();
        book(p.seats_req, p.name);
    }

    void book(int req, String n) {
        System.out.println(n);
        if (req <= seats_aval) {
            System.out.println("seats avaliable: " + seats_aval);
            seats_aval -= req;
            System.out.println(req + " tickets booked for " + n);
        } else {
            System.out.println("seats avaliable: " + seats_aval);
            System.out.println("Tickets not avaliable!!");
        }
    }
}

public class MultiThreading {
    public static void main(String[] args) {
        Reservation r = new Reservation();
        Passenger p1 = new Passenger(2, "Kartik", r);
        Passenger p2 = new Passenger(2, "Meet", r);
        p2.start();
        p1.start();
    }
}
```

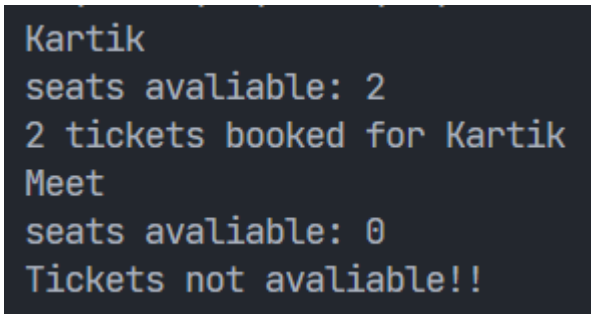
Output (NO Synchronization)

```
Kartik
Meet
seats avaliable: 2
seats avaliable: 2
2 tickets booked for Meet
2 tickets booked for Kartik
```

Using Multithreading (With Synchronization)

```
class Passenger extends Thread {
    int seats_req;
    String name;

    Passenger(int s, String n, Reservation r) {
        super(r);
        seats_req = s;
        name = n;
    }
}

class Reservation implements Runnable {
    int seats_aval = 2;

    public void run() {
        Passenger p = (Passenger) Thread.currentThread();
        book(p.seats_req, p.name);
    }

    // for synchronization change below function to synchronized
    synchronized void book(int req, String n) {
        System.out.println(n);
        if (req <= seats_aval) {
            System.out.println("seats avaliable: " + seats_aval);
            seats_aval -= req;
            System.out.println(req + " tickets booked for " + n);
        } else {
            System.out.println("seats avaliable: " + seats_aval);
            System.out.println("Tickets not avaliable!!");
        }
    }
}

public class MultiThreading {
    public static void main(String[] args) {
        Reservation r = new Reservation();
        Passenger p1 = new Passenger(2, "Kartik", r);
        Passenger p2 = new Passenger(2, "Meet", r);
        p1.start();
        p2.start();
    }
}
```

Output (With Synchronization)



Conclusion –

The concept of multithreading was executed, using it in the application of bus reservation system.