DISTRIBUTED SYSTEMS

LAB ASSIGNMENT-4

SUBMITTED TO

Professor ARUNKUMAR.T

NAME:SINDUMANI.M

REGNO:19MIC0002

COURSECODE:

SLOT:L15+L16

1.Write a program to stimulate the concept of mutual exclusion over lamport's algorithm for n number of jobs waiting in the queue to be serviced.Implement the phenomenon of Casual Ordering and Total Ordering

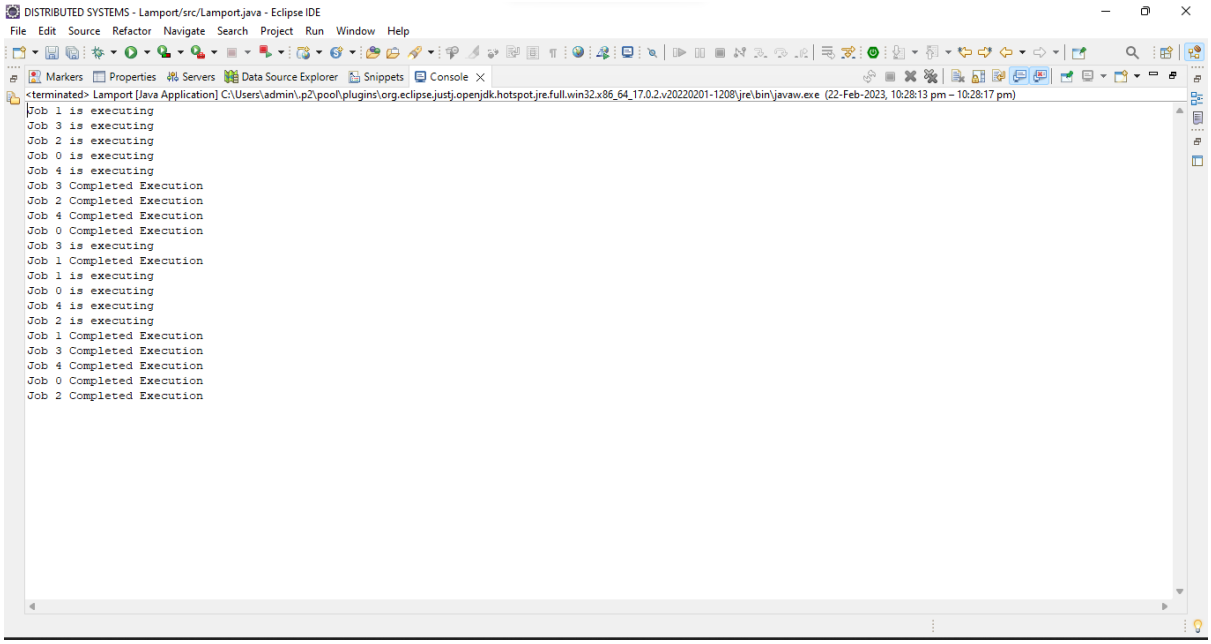
CODE:

```
import java.util.*;
import java.util.concurrent.*;
import java.util.concurrent.atomic.*;
class Job {
int id;
AtomicInteger timestamp;
public Job(int id, int timestamp) {
this.id = id;
this.timestamp = new AtomicInteger(timestamp);
}
}
public class Lamport {
int n;
ExecutorService executor;
List<Job> jobs;
AtomicInteger counter;
AtomicIntegerArray queue;
Map<Integer, Integer> timestamps;
boolean[] isExecuting;
public Lamport(int n) {
this.n = n;
this.executor = Executors.newFixedThreadPool(n);
this.jobs = new ArrayList<>();
this.counter = new AtomicInteger(0);
this.queue = new AtomicIntegerArray(n);
this.timestamps = new ConcurrentHashMap<>();
this.isExecuting = new boolean[n];
}
public void execute() {
for (int i = 0; i < n; i++) {
int finalI = i;
executor.execute(() -> {
while (true) {
Job job = jobs.get(finalI);
int id = job.id;
int timestamp = job.timestamp.getAndIncrement();
```

```
timestamps.put(id, timestamp);
for (int j = 0; j < n; j++) {
if (j == finalI) continue;
while (timestamps.containsKey(j) && timestamps.get(j) < timestamp) {}
if (queue.get(j) == finalI && timestamps.get(j) == timestamp) {
isExecuting[finalI] = true;
break;
}
}
System.out.println("Job " + id + " is executing");
try {
Thread.sleep((long) (Math.random() * 1000));
} catch (InterruptedException e) {
e.printStackTrace();
}
System.out.println("Job " + id + " Completed Execution");
isExecuting[finalI] = false;
for (int j = 0; j < n; j++) {
if (j == finalI) continue;
if (queue.get(j) == finalI) {
queue.set(j, -1);
}
}
// Move to the next job in the queue
int next = counter.getAndIncrement();
if (next >= n) {
break;
}
queue.set(finalI, next);
}
});
}
executor.shutdown();
}
public static void main(String[] args) {
Lamport mutex = new Lamport(5);
mutex.jobs.add(new Job(4, 0));
mutex.jobs.add(new Job(3, 0));
mutex.jobs.add(new Job(1, 0));
mutex.jobs.add(new Job(2, 0));
mutex.jobs.add(new Job(0, 0));
mutex.execute();
}
```

}

OUTPUT:


DISTRIBUTED SYSTEMS - Lamport/src/Lamport.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Markers Properties Servers Data Source Explorer Snippets Console
<terminated> Lamport [Java Application] C:\Users\admin\.p2\pool\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.2.v20220201-1208\jre\bin\javaw.exe (22-Feb-2023, 10:28:13 pm – 10:28:17 pm)
Job 1 is executing
Job 3 is executing
Job 2 is executing
Job 0 is executing
Job 4 is executing
Job 3 Completed Execution
Job 2 Completed Execution
Job 4 Completed Execution
Job 0 Completed Execution
Job 3 is executing
Job 1 Completed Execution
Job 1 is executing
Job 0 is executing
Job 4 is executing
Job 2 is executing
Job 1 Completed Execution
Job 3 Completed Execution
Job 4 Completed Execution
Job 0 Completed Execution
Job 2 Completed Execution