CS5541 - Computer Systems
Petersons Algorithm
Akshay Kumar

Implementation:

- I have initially created two threads in the program.
- When the execution of the thread begins, it creates a method where the constructors are called and the control goes there.
- When the constructors are called, objects are created.
- The program executes run () function and checks for the critical section.
- If the critical section is true, then the thread that is the critical section is being executed and the other thread is on wait. This state is in Mutual Exclusion of Petersons Algorithm and the threads are executed one after another till the critical section is true.
- If the critical section is false, then both the threads are in wait condition and the program will not execute.

```
public void run()
 {
do {
criticalSection.indicator[1] = true;
criticalSection.turn = 0;
while (criticalSection.indicator[0] == true && criticalSection.turn == 0) {
}
for (int i = 0; i < 5; i++) {
criticalSection.count++;
criticalSection.totalRun++;
System.out.print(character);
if (criticalSection.count == 30) {
System.out.print(" Thread 1");
criticalSection.count = 0;
System.out.println(" Value = " + criticalSection.turn);
}
}
criticalSection.indicator[1] = false;
```

```
} while (criticalSection.totalRun < 12000);
}
}
```

public class CriticalSectionTest

```
{
public int count;
public int turn;
public boolean[] indicator;
public int totalRun;
public CriticalSectionTest() {
indicator = new boolean[2];
indicator[0] = false;
indicator[1] = false;
totalRun = 0;
}
}
```

1. Mutual exclusion must be enforced: Only one process at a time is allowed into its critical section, among all processes that have critical sections for the same resource or shared object.

Ans: When a process is in critical section no other process will be executing in critical section. In a critical section, more than one process cannot execute at same time. Critical section requires mutual exclusion of access. In mutual exclusion one thread of execution never enter its critical section at the same time while other thread of execution enters its critical section.

```
for (int i = 0; i < 5; i++) {
criticalSection.count++;
criticalSection.totalRun++;
System.out.print(character);
```

```
if (criticalSection.count == 30) {
System.out.print(" Thread 1");
criticalSection.count = 0;
System.out.println(" Value = " + criticalSection.turn);
}
}
```

2. A process that halts in its noncritical section must do so without interfering with other processes.

Ans: Assuming that one process is in critical section and the other is outside critical section then the process that is outside critical section will have to wait till the process inside critical section is executed. After the process is executed the process waiting outside will enter the critical section and gets executed if it has to run in multiples of 30. Suppose if it has to run say like 100 times the process will be in critical section till it executes 90 times and after executing the remaining 10 times the condition will become false and process will come out of the loop and the program stops.

```
do {
criticalSection.indicator[1] = true;
criticalSection.turn = 0;
while (criticalSection.indicator[0] == true && criticalSection.turn == 0) {
}

public CriticalSectionTest() {
indicator = new boolean[2];
indicator[0] = false;
indicator[1] = false;
totalRun = 0;
```

3. It must not be possible for a process requiring access to a critical section to be delayed indefinitely: no deadlock or starvation.

Ans: Process will enter the critical section only if the flag is true. Once the process enters the critical section means that it has get executed till the flag is true. Once the flag is false control has to come out of the loop and the execution of the process has to stop. If the flag is false, the execution of process in critical section should not stay inside the critical section and go for deadlock or starvation rather the execution should stop and come out of the loop.

criticalSection.indicator[1] = false;

4. When no process is in a critical section, any process that requests entry to its critical section must be permitted to enter without delay.

Ans: A process can enter critical section only if the flag is true. If the flag is false, then the process cannot enter critical section. If no process is in critical section, then the process that requests to enter critical section will be given priority and would check if the flag is true and enters the section without any delay.

5. No assumptions are made about relative process speeds or number of processors.

Ans: I think we cannot control the speed of execution of the process that enter critical section but we can only make sure that it executes all instances associated with that process successfully.

6. A process remains inside its critical section for a finite time only.

Ans: A critical section usually terminate in finite time without going into deadlock state and a thread should later wait a fixed time to enter bound waiting state. We can also use a synchronized way of giving an input thread to critical section where one executes one after each other.