Name: Akshay Subhash Gadhave

Gr No. : 21920090

Roll No : 221076

SY COMP A3

Assignment 4

Aim: Implement Producer/Consumer problem using java threads.

Theory:

Multithreading in Java is a process of executing multiple threads simultaneously.

A thread is a lightweight sub-process, the smallest unit of processing. Multiprocessing and multithreading, both are used to achieve multitasking.

However, we use multithreading than multiprocessing because threads use a shared memory area. They don't allocate separate memory area so saves memory, and context-switching between the threads takes less time than process.

Java Multithreading is mostly used in games, animation, etc.

Advantages of Java Multithreading

1) It doesn't block the user because threads are independent and you can perform multiple operations at the same time.

2) You can perform many operations together, so it saves time.

3) Threads are independent, so it doesn't affect other threads if an exception occurs in a single thread.

A thread can be in one of the five states. According to sun, there is only 4 states

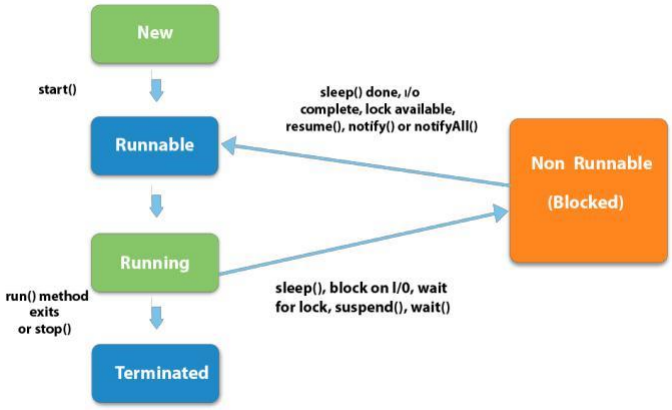
in **thread life cycle in java** new, runnable, non-runnable and terminated. There is no running state.

But for better understanding the threads, we are explaining it in the 5 states.

The life cycle of the thread in java is controlled by JVM. The java thread states are as follows:

1. New
2. Runnable
3. Running

4. Non-Runnable (Blocked)
5. Terminated



1) New

The thread is in new state if you create an instance of Thread class but before the invocation of start() method.

2) Runnable

The thread is in runnable state after invocation of start() method, but the thread scheduler has not selected it to be the running thread.

3) Running

The thread is in running state if the thread scheduler has selected it.

4) Non-Runnable (Blocked)

This is the state when the thread is still alive, but is currently not eligible to run.

5) Terminated

A thread is in terminated or dead state when its run() method exits.

Code:

1.ThreadExample.java file

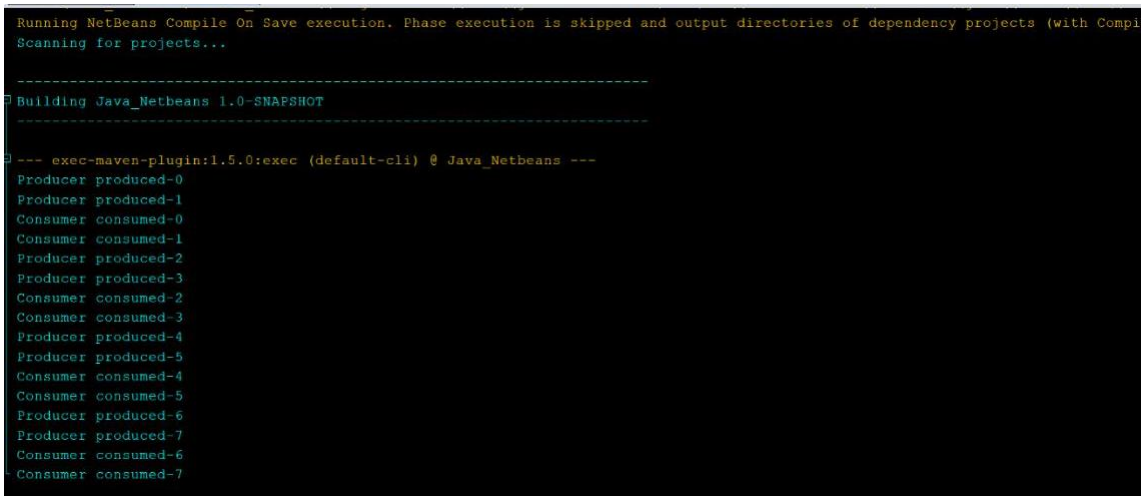
```
import java.util.LinkedList;
```

```
public class Threadexample {
    public static void main(String[] args)
        throws InterruptedException
    {
        final PC pc = new PC();

        Thread t1 = new Thread(new Runnable() {
            @Override
            public void run()
            {
                try {
                    pc.produce();
                }
                catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        });

        Thread t2 = new Thread(new Runnable() {
            @Override
            public void run()
            {
                try {
                    pc.consume();
                }
                catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
```

```
    });

    t1.start();
    t2.start();

    t1.join();
    t2.join();
}

public static class PC {

    LinkedList<Integer> list = new
    LinkedList<>(); int capacity = 2;

    public void produce() throws
    InterruptedException {
        int value = 0;
        while (true) {
            synchronized (this)
            {
                while (list.size() == capacity)
                    wait();

                System.out.println("Producer produced-"
                                   + value);

                list.add(value++);

                notify();
```

```
                Thread.sleep(1000);
            }
        }
    }

    public void consume() throws
    InterruptedException {
        while (true) {
            synchronized (this)
            {
                while (list.size() == 0)
                    wait();

                int val = list.removeFirst();

                System.out.println("Consumer consumed-"
                                + val);

                notify();

                Thread.sleep(1000);
            }
        }
    }
  }
}
```

Output:


Running NetBeans Compile On Save execution. Phase execution is skipped and output directories of dependency projects (with Compi
Scanning for projects...
Building Java_Netbeans 1.0-SNAPSHOT
--- exec-maven-plugin:1.5.0:exec (default-cli) @ Java_Netbeans ---
Producer produced-0
Producer produced-1
Consumer consumed-0
Consumer consumed-1
Producer produced-2
Producer produced-3
Consumer consumed-2
Consumer consumed-3
Producer produced-4
Producer produced-5
Consumer consumed-4
Consumer consumed-5
Producer produced-6
Producer produced-7
Consumer consumed-6
Consumer consumed-7