EXPERIMENT 10

Aim :

Write a program to implement Producer-Consumer problem using Threads.

Code :

```
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 Screenshot :