VISHWAKARMA INSTITUTE OF TECHNOLOGY

BRANCH - AI&DS DIVISION- C BATCH- B2

NAME- Pushkraj Shahane ROLL NO- 22 PRN NO- 12110725

OS LAB - ASSIGNMENT 5

Implementation of Producer Consumer problem using Threads and semaphore.

```
import java.util.concurrent.Semaphore;
public class Assignment5{
   public static void main(String[] args){

                        Semaphore semaphoreProducer=new Semaphore(1);
                        Semaphore semaphoreConsumer=new Semaphore(0);
      Producer producer=new Producer(semaphoreProducer,semaphoreConsumer);
      Consumer consumer=new Consumer(semaphoreConsumer,semaphoreProducer);
       Thread producerThread = new Thread(producer, "ProducerThread");
       Thread consumerThread = new Thread(consumer, "ConsumerThread");

       producerThread.start();
       consumerThread.start();
   }
}
class Producer implements Runnable{
   Semaphore semaphoreProducer;
   Semaphore semaphoreConsumer;
   public Producer(Semaphore semaphoreProducer,Semaphore semaphoreConsumer) {
         this.semaphoreProducer=semaphoreProducer;
         this.semaphoreConsumer=semaphoreConsumer;
   }
   public void run() {
         for(int i=1;i<=5;i++){
               try {
                  semaphoreProducer.acquire();
                  System.out.println("Produced : "+i);
                  System.out.println("Consumer is waiting ");

                                                Thread.sleep(1000);
                  semaphoreConsumer.release();

               } catch (InterruptedException e) {
                    e.printStackTrace();
               }
         }
   }
}
class Consumer implements Runnable{
```

```
    Semaphore semaphoreConsumer;
    Semaphore semaphoreProducer;
    public Consumer(Semaphore semaphoreConsumer,Semaphore semaphoreProducer) {
        this.semaphoreConsumer=semaphoreConsumer;
        this.semaphoreProducer=semaphoreProducer;
    }
    public void run() {
        for(int i=1;i<=5;i++){
            try {
               semaphoreConsumer.acquire();
               System.out.println("Consumed : "+i);
                    System.out.println("Producer is waiting ");

                                                Thread.sleep(1000);
               semaphoreProducer.release();
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
        }
    }
}
```

Output –

```
Produced : 1
Consumer is Waiting
Consumed : 1
Producer is waiting
Produced : 2
Consumer is Waiting
Consumed : 2
Producer is waiting
Produced : 3
Consumer is Waiting
Consumed : 3
Producer is waiting
Produced : 4
Consumer is Waiting
Consumed : 4
Producer is waiting
Produced : 5
Consumer is Waiting
Consumed : 5
Producer is waiting
PS D:\VIT\Third year\OS>
```