How to solve Consumer Producer problem without using wait() and notify() methods, where consumer can consume only when production is over in java.

How to solve **Consumer Producer** problem **without using wait() and notify()** methods, where **consumer can consume only when production is over.**

Producer will allow consumer to consume only when 10 products have been produced (i.e. when production is over).

We will approach by keeping one boolean variable **productionInProcess** and initially setting it to **true**, and later when production will be over we will set it to **false**.

**In program producer thread will start() and it will start producing and called sleep(1000) in between, which will give consumer thread chance to execute. consumer checks whether productionInProcess is true or not, if it's true,
consumer will sleep(4000) and wake up after specified time and again check whether productionInProcess is true or false. process will repeat till productionInProcess is true. Meanwhile, producer thread will complete production and ultimately make productionInProcess to false. Once productionInProcess is false, consumer will consume.**

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


/**
 * Producer Class in java, Producer will allow consumer to consume only
 * when 10 products have been produced (i.e. when production is over).
 */
class Producer implements Runnable{

    boolean productionInProcess;
    List<Integer> list;

    Producer(){
           //initially Producer will be producing, so make this productionInProcess true.
```

```
            productionInProcess=true;
            list=new LinkedList<Integer>();
    }

    @Override
    public void run(){

            for(int i=1;i<=10;i++){ //Producer will produce 10 products
                    list.add(i);
                    System.out.println("Producer is still Producing, Produced : "+i);

                    try{
                          Thread.sleep(1000);
                    }catch(InterruptedException e){e.printStackTrace();}

            }

            /* Once production is over, make this productionInProcess false.
             * Production is over, consumer can consume.
             */
            productionInProcess=false;

    }

}

/**
 * Consumer Class.
 */
class Consumer extends Thread{
    Producer prod;

    Consumer(Producer obj){
     prod=obj;
    }

    public void run(){
            /*
             * consumer checks whether productionInProcess is true or not,
             * if it's true, consumer will sleep and wake up after certain time
             * and again check whether productionInProcess is true or false.
             * process will repeat till productionInProcess is true.
             * Once productionInProcess is false we'll exit below while loop.
             */
            while(this.prod.productionInProcess){
               System.out.println("Consumer waiting for production to get over.");
               try{
                   Thread.sleep(4000);
               }catch(InterruptedException e){e.printStackTrace();}

            }


            /*productionInProcess is false means production is over,
             * consumer will start consuming. */
            System.out.println("Production is over, consumer can consume.");
            int productSize=this.prod.list.size();
            for(int i=0;i<productSize;i++)
                    System.out.println("CONSUMED : "+ this.prod.list.remove(0) +" ");

    }
```

```
}

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class ProducerConsumerWithoutWaitNotify {
    public static void main(String args[]){

            Producer prod=new Producer();
            Consumer cons=new Consumer(prod);

            Thread prodThread=new Thread(prod,"prodThread");
            Thread consThread=new Thread(cons,"consThread");

            prodThread.start();     //start producer thread.
            consThread.start();     //start consumer thread.


    }

}
/*OUTPUT

Consumer waiting for production to get over.
Producer is still Producing, Produced : 1
Producer is still Producing, Produced : 2
Producer is still Producing, Produced : 3
Producer is still Producing, Produced : 4
Consumer waiting for production to get over.
Producer is still Producing, Produced : 5
Producer is still Producing, Produced : 6
Producer is still Producing, Produced : 7
Producer is still Producing, Produced : 8
Consumer waiting for production to get over.
Producer is still Producing, Produced : 9
Producer is still Producing, Produced : 10
Production is over, consumer can consume.
CONSUMED : 1
CONSUMED : 2
CONSUMED : 3
CONSUMED : 4
CONSUMED : 5
CONSUMED : 6
CONSUMED : 7
CONSUMED : 8
CONSUMED : 9
CONSUMED : 10


*/
```