```
------------ProducerTask.java--------
package com.ameya.tasks;

import java.util.concurrent.BlockingQueue;

public class ProducerTask implements Runnable{
    BlockingQueue<Integer> obj;
    public ProducerTask(BlockingQueue<Integer> obj) {
        this.obj=obj;
    }

    @Override
    public void run() {
    for(int i=1;i<=4;i++) {
        try {
            obj.put(i);
            System.out.println("PRODUCED => "+i);
        }catch(InterruptedException e) {
            e.printStackTrace();
        }
    }

    }

}
----------------ConsumerTask.java--------
package com.ameya.tasks;

import java.util.concurrent.BlockingQueue;

public class ConsumerTask implements Runnable {
    BlockingQueue<Integer> obj;
    private int taken=-1;
    public ConsumerTask(BlockingQueue<Integer> obj) {
        this.obj=obj;
    }

    @Override
    public void run() {
        while(taken !=4) {
            try {
                taken=obj.take();
                System.out.println("CONSUMED => "+taken);
            }catch(InterruptedException e) {
                e.printStackTrace();
            }
        }

    }

}
-------------------TestBlockingQueueProducerConsumer.java--------
package com.ameya.test;

import java.util.concurrent.ArrayBlockingQueue;
import java.util.concurrent.BlockingQueue;

import com.ameya.tasks.ConsumerTask;
import com.ameya.tasks.ProducerTask;

public class TestBlockingQueueProducerConsumer {

    public static void main(String[] args) {
        BlockingQueue<Integer> q=new ArrayBlockingQueue<Integer>(4);
        ProducerTask p1=new ProducerTask(q);
        ConsumerTask c1=new ConsumerTask(q);
        Thread pThread=new Thread(p1);
        Thread cThread=new Thread(c1);
        pThread.start();
        cThread.start();
        //Try using Executor Framework for thread pooling
    }

}

```