

```

1  -----ProducerTask.java-----
2  package com.ameya.tasks;
3
4  import java.util.concurrent.BlockingQueue;
5
6  public class ProducerTask implements Runnable{
7      BlockingQueue<Integer> obj;
8      public ProducerTask(BlockingQueue<Integer> obj) {
9          this.obj=obj;
10     }
11
12     @Override
13     public void run() {
14         for(int i=1;i<=4;i++) {
15             try {
16                 obj.put(i);
17                 System.out.println("PRODUCED => "+i);
18             }catch(InterruptedException e) {
19                 e.printStackTrace();
20             }
21         }
22     }
23 }
24
25 }
26 -----ConsumerTask.java-----
27 package com.ameya.tasks;
28
29 import java.util.concurrent.BlockingQueue;
30
31 public class ConsumerTask implements Runnable {
32     BlockingQueue<Integer> obj;
33     private int taken=-1;
34     public ConsumerTask(BlockingQueue<Integer> obj) {
35         this.obj=obj;
36     }
37
38     @Override
39     public void run() {
40         while(taken !=4) {
41             try {

```

```

42         taken=obj.take();
43         System.out.println("CONSUMED => "+taken);
44     }catch(InterruptedException e) {
45         e.printStackTrace();
46     }
47 }
48
49 }
50
51 }
52 -----TestBlockingQueueProducerConsumer.java-----
53 package com.ameya.test;
54
55 import java.util.concurrent.ArrayBlockingQueue;
56 import java.util.concurrent.BlockingQueue;
57
58 import com.ameya.tasks.ConsumerTask;
59 import com.ameya.tasks.ProducerTask;
60
61 public class TestBlockingQueueProducerConsumer {
62
63     public static void main(String[] args) {
64         BlockingQueue<Integer> q=new ArrayBlockingQueue<Integer>(4);
65         ProducerTask p1=new ProducerTask(q);
66         ConsumerTask c1=new ConsumerTask(q);
67         Thread pThread=new Thread(p1);
68         Thread cThread=new Thread(c1);
69         pThread.start();
70         cThread.start();
71         //Try using Executor Framework for thread pooling
72     }
73
74 }
75

```