Solve Consumer Producer problem by using BlockingQueue in multithreading in j - Detailed explanation with full program

In this thread concurrency tutorial we will learn how to Solve Consumer Producer problem by using BlockingQueue and LinkedBlockingQueue i multithreading in java using example and program.

Now it's time to gear up to face question which is most probably going to be followed up by previous question i.e. how to solve consumer produ problem using wait() and notify() method in java. Generally you might wonder why interviewer's are so much interested in asking this question, a is they want to know how strong knowledge you have about java concurrent Api's, this Api use consumer producer pattern in very optimized ma BlockingQueue is designed is such a manner that it offer us the best performance in java.
BlockingQueue is a interface and we will use its **implementation class LinkedBlockingQueue**. In one my post i have mentioned how to implement custom BlockingQueue and solving Producer Consumer pattern using Custom implementation of BlockingQueue interface.

Key methods in BlockingQueue Api for solving consumer producer in java pattern are >

```
put(i);      //used by producer to put/produce in sharedQueue.
take(); //used by consumer to take/consume from sharedQueue.
```

Example/ Full Program/sourceCode to solve consumer producer problem using custom BlockingQueue in java >

```
import java.util.concurrent.BlockingQueue;
import java.util.concurrent.LinkedBlockingQueue;

/**
 * Producer Class in java.
 */
class Producer implements Runnable {

    private final BlockingQueue<Integer> sharedQueue;

    public Producer(BlockingQueue<Integer> sharedQueue) {
        this.sharedQueue = sharedQueue;
    }

    @Override
    public void run() {
        for(int i=1; i<=10; i++){
         try {
             System.out.println("Produced : " + i);
             //put/produce into sharedQueue.
             sharedQueue.put(i);
         } catch (InterruptedException ex) {

         }
        }
    }

}

/**
 * Consumer Class in java.
 */
class Consumer implements Runnable{

    private BlockingQueue<Integer> sharedQueue;

    public Consumer (BlockingQueue<Integer> sharedQueue) {
        this.sharedQueue = sharedQueue;
    }

    @Override
    public void run() {
        while(true){
         try {
           //take/consume from sharedQueue.
             System.out.println("CONSUMED : "+ sharedQueue.take());
         } catch (InterruptedException ex) {

         }
        }
    }
```

```
}

/** Copyright (c), AnkitMittal JavaMadeSoEasy.com */
public class ProducerConsumerBlockingQueue {

    public static void main(String args[]){

     //Creating shared object
     BlockingQueue<Integer> sharedQueue = new LinkedBlockingQueue<Integer>();

     Producer producer=new Producer(sharedQueue);
     Consumer consumer=new Consumer(sharedQueue);

     Thread producerThread = new Thread(producer, "ProducerThread");
     Thread consumerThread = new Thread(consumer, "ConsumerThread");
     producerThread.start();
     consumerThread.start();

    }

}

/*OUTPUT

Produced : 1
Produced : 2
CONSUMED : 1
Produced : 3
CONSUMED : 2
Produced : 4
CONSUMED : 3
Produced : 5
CONSUMED : 4
Produced : 6
CONSUMED : 5
Produced : 7
CONSUMED : 6
Produced : 8
CONSUMED : 7
Produced : 9
CONSUMED : 8
Produced : 10
CONSUMED : 9
CONSUMED : 10

*/
```

SUMMARY >

So in this thread concurrency tutorial we learned how to **Solve Consumer Producer problem by using BlockingQueue and LinkedBlockingQueue** in multithreading in java.

Having any doubt? or you you liked the tutorial! Please comment in below section.
*Please express your love by liking **JavaMadeSoEasy.com** (**JMSE**) on **facebook**, following on **google+** or Twitter.*