生产者和消费者

⭐ 所谓的生产者消费者模型，是通过一个容器来解决生产者和消费者的强耦合问题。通俗的讲，就是生产者在不断的生产，消费者也在不断的消费，可是消费者消费的产品是生产者生产的，这就必然存在一个中间容器，我们可以把这个容器想象成是一个货架，当货架空的时候，生产者要生产产品，此时消费者在等待生产者往货架上生产产品，而当货架有货物的时候，消费者可以从货架上拿走商品，生产者此时等待货架出现空位，进而补货，这样不断的循环。

通过多线程编程，来模拟一个餐厅的2个厨师和3个顾客，假设厨师炒出一个菜的时间为3秒，顾客吃掉菜品的时间为4秒。

```
import java.util.LinkedList;
import java.util.Objects;
import java.util.Queue;

public class Main {

    //创建一个队列模拟送餐窗口
    private static final Queue<Object> queue = new LinkedList<>();

    public static void main(String[] args) {
        //创建生产者
        Thread p1 = producer("p1");
        Thread p2 = producer("p2");
        p1.start();
        p2.start();

        //创建消费者
        Thread c1 = consumer("c1");
        Thread c2 = consumer("c2");
        Thread c3 = consumer("c3");
        c1.start();
        c2.start();
        c3.start();



    }

    /**
     * 创建生产者
     * @param name 生产者名
     * @return 一个生产者线程
     */
    public static Thread producer(String name) {
        return new Thread(() -> {
            while (true) {
                try {
```

```
                    Thread.sleep(3000); //生产者需要3秒生产，不能放在锁内部，如果放在锁内部就会导致有可能一个生产者一直占据着锁，因为是while循环执行完成一个循环立马会执行下一个
                    synchronized (queue) { //同步锁，避免两个生产者同时操作这个数组\
                        System.out.println(name+"生产完成");
                        queue.offer(new Object());
                        queue.notifyAll();
                    }
                } catch (InterruptedException e) {
                    e.printStackTrace();
                }
            }
        }, name);
    }
    /**
     * 创建消费者
     * @param name 消费者名
     * @return 一个消费者线程
     */

    public static Thread consumer(String name){
        return  new Thread(() -> {
           while (true) {
               try {
                   synchronized (queue) {
                       while (queue.isEmpty())
                           queue.wait(); //这里使用while而不是if因为如果顾客被唤醒，发现餐被其他顾客已经拿走了，那就完蛋了，所有要使用while循环判断
                       queue.poll();
                       System.out.println(name + "拿到资源，并开始消费");
                   }
                   Thread.sleep(4000); //消费者需要4s才可以将资源消费，sleep不能放在锁里面，如果放在锁里面，就会占着锁4s,导致厨师和其他消费者都不能拿到这个队列
                   System.out.println(name + "消费完毕");
               } catch (InterruptedException e) {
                   throw new RuntimeException(e);
               }
           }
        });

    }


}

```

执行:

```
p1生产完成
c3拿到资源，并开始消费
p2生产完成
c1拿到资源，并开始消费
```

```
p1生产完成
c2拿到资源，并开始消费
p2生产完成
c3消费完毕
c3拿到资源，并开始消费
c1消费完毕
p1生产完成
c1拿到资源，并开始消费
p2生产完成
c2消费完毕
c2拿到资源，并开始消费
c3消费完毕
p2生产完成
c3拿到资源，并开始消费
p1生产完成
c1消费完毕
c1拿到资源，并开始消费
c2消费完毕
p2生产完成
c2拿到资源，并开始消费
p1生产完成
c3消费完毕
c3拿到资源，并开始消费
c1消费完毕
p2生产完成
c1拿到资源，并开始消费
p1生产完成
c2消费完毕
c2拿到资源，并开始消费
```