线程的管理---生产者与消费者问题

1、需要锁

2、通过标记来控制生产和消费的行为

3、Object.wait();

if方法：

class Storage {

// 存放生产的 商品------集合

ArrayList list = new ArrayList();

int i = 0;// 商品的标号

// 定义标记---生产的标记 标记为true 表示可以生产，否则只能消费

boolean isProduct = true;

// 生产的行为

public void product() {

while (true) {

synchronized (list) {// 一般使用共享资源 list是锁

if (!isProduct) {// 如果不能生产，就等待

try {

System.out.println(Thread.currentThread().getName()+"---生产者在等待");

list.wait();// 当前线程处于等待状态，放弃锁。

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

// 将生产的商品放到集合

System.out.println(Thread.currentThread().getName() + "--正在生产第" + i + "号商品");

list.add(new Integer(i));

// 生产完了改变标记，等到消费完这个商品才能继续生产商品

isProduct = false;

//唤醒所有等待的线程

list.notifyAll();

}

}

}

**其中一个生产者开始生产，将商品标号上锁，在锁没有打开的时候，其他生产者抢到运行权时由于没有商品标号则无法完全运行。只有等到消费者重新唤醒之后才能重新抢占cpu资源运行。**

while (true) {

synchronized (list) {

if (isProduct) {// 如果不能消费 就等待

try {

System.out.println(Thread.currentThread().getName()+"---消费者在等待");

list.wait();

} catch (InterruptedException e) {

e.printStackTrace();

}

}

try {

Thread.sleep(100);

} catch (InterruptedException e) {

e.printStackTrace();

}

// 将生产的商品消费掉

System.out.println(Thread.currentThread().getName() + "--正在消费第" + i + "号商品");

list.remove(new Integer(i));

i++;

// 消费完了改变标记，等到生产下一个才能继续消费

isProduct = true;

//唤醒一个等待的线程

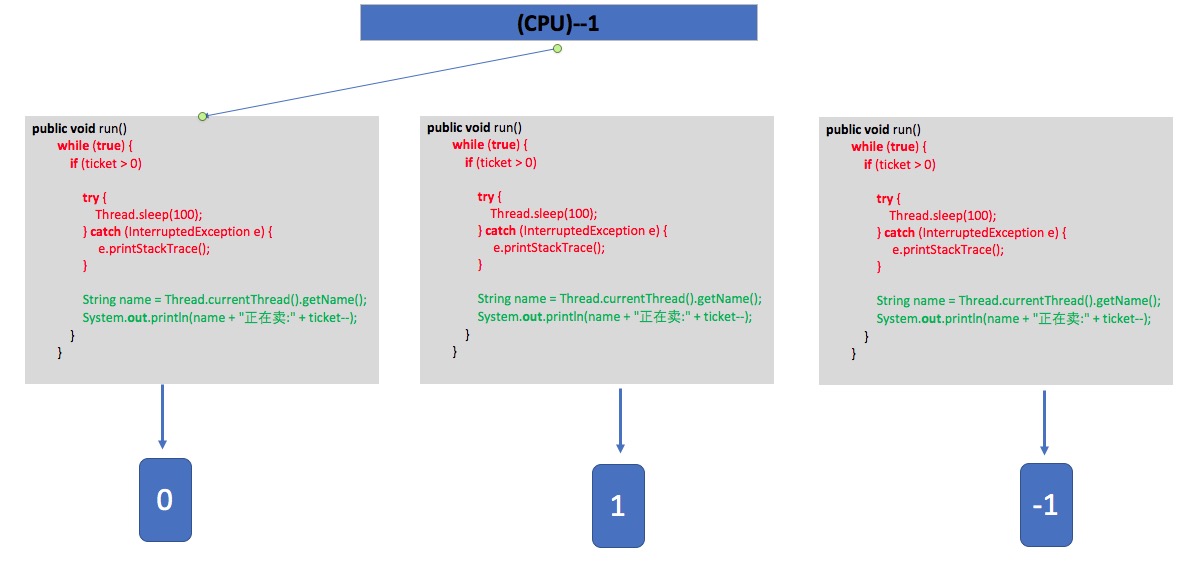
list.notifyAll();

}

}

**当生产者生产完成之后，消费者抢到cpu资源才能运行，否则只会到缓存集合中等待，生产者生产完成之后重新执行。**

**if判断是：被唤醒之后不会重新判断，所以从被上锁的商标开始运行，所以会产生多个相同的商品，在正常情况下是不允许出现这种情况的。**



**while方法：**

**// 生产的行为**

**public void product() {**

**while (true) {**

**synchronized (list) {// 一般使用共享资源 list是锁**

**while (!isProduct) {// 如果不能生产，就等待**

**try {**

**System.out.println(Thread.currentThread().getName()+"---生产者在等待");**

**list.wait();// 当前线程处于等待状态，放弃锁。**

**} catch (InterruptedException e) {**

**// TODO Auto-generated catch block**

**e.printStackTrace();**

**}**

**}**

**try {**

**Thread.sleep(100);**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**// 将生产的商品放到集合**

**System.out.println(Thread.currentThread().getName() + "--正在生产第" + i + "号商品");**

**list.add(new Integer(i));**

**// 生产完了改变标记，等到消费完这个商品才能继续生产商品**

**isProduct = false;**

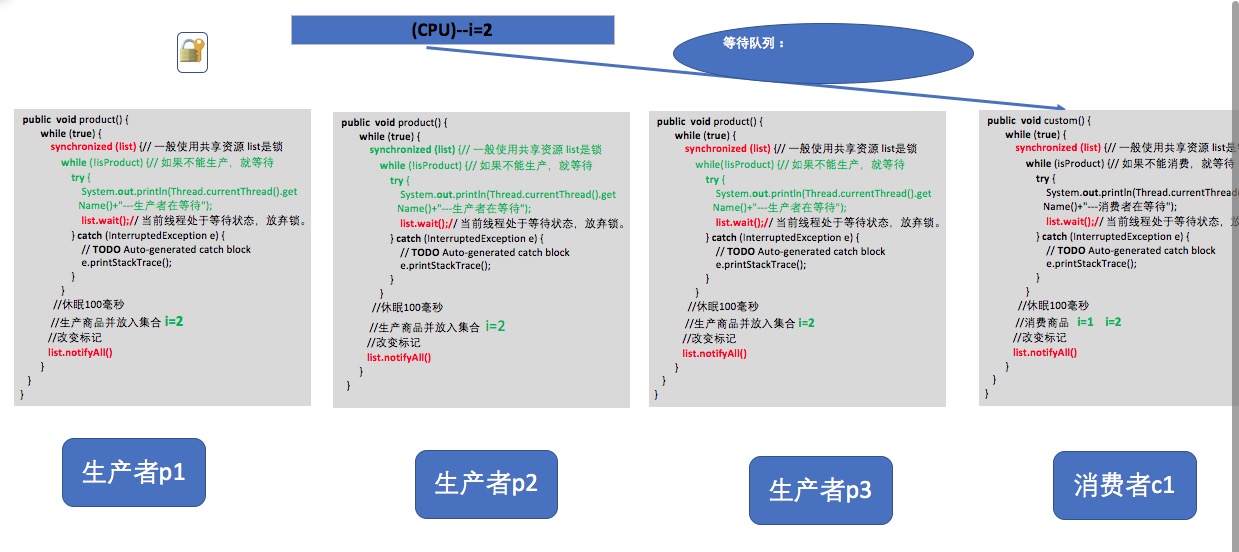
**//唤醒所有等待的线程**

**list.notifyAll();**

**}**

**}**

**}**

**一个生产者开始生产时，将商品标号上锁，其他生产者由于没有商品标号使用，抢占到cpu资源也无法完成生产。所以到判断是否上锁，如果已经上锁则当前线程处于等待状态，放弃锁，等待重新唤醒。**

**// 消费行为**

**public void custom() {**

**while (true) {**

**synchronized (list) {**

**while (isProduct) {// 如果不能消费 就等待**

**try {**

**System.out.println(Thread.currentThread().getName()+"---消费者在等待");**

**list.wait();**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**}**

**try {**

**Thread.sleep(100);**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**// 将生产的商品消费掉**

**System.out.println(Thread.currentThread().getName() + "--正在消费第" + i + "号商品");**

**list.remove(new Integer(i));**

**i++;**

**// 消费完了改变标记，等到生产下一个才能继续消费**

**isProduct = true;**

**//唤醒一个等待的线程**

**list.notifyAll();**

**}**

**}**

**}**

**当消费者判断有商品生产完成之后，就调用消费者消费商品。消费完成之后重新换新等待状态下事务。**

**while形式是被唤醒之后重新判断商品标号是否上锁，如果已经上锁则当前线程处于等待状态，放弃锁，等待重新唤醒。如果没有上锁，则进行生产。**

# 线程的生命周期：

