Java学习day12

注意：本章中的生产者与消费者这一章教会了我们如何看官方文档的一些例子，但是自己还不是特别会看理解，有空的时候尝试看一下视频的回顾。

1. 多线程 （线程间通信）：

学习到了wait()和notify()两个函数。

线程操作其实就是多个线程在操作一个资源，只是操作的动作不同。

将这个事例想象成运煤，共有output与input就是运进来与运进去两个类。

可以发现的问题是出现名字与性别不匹配，

原因：input与output顺序的不协调

一旦出现了安全问题那么解决方法就是同步。

这个例子的类名不能用Object和this，因为object要写在两个地方，并且this会指向input和output两个类。

所以可以用到其他四个字节码文件。 其中用r最简单，因为它指向Res这个类的资源

观察一下输出，为什么会有连着一大片打印的这种情况呢？

因为cpu还未来得及切换，就有很多output输出，造成有一大堆重名的数据

卧槽，为啥wait这个东西不管用啊。。。

因为自己没有给它一个类名对象的调用，就相当于开关同样一个锁。如果不加锁，就不知道是哪个类的线程。

为什么这些操作线程的方法要定义在object类中呢？

因为这些方法在操作同步线程时，都必须要表示他们所操作线程的只有锁。

只有同一个锁上的被等待线程，可以被同一个锁上的notify唤醒。

不可以对不同所中的线程进行唤醒。

也就是说，等待和唤醒必须是同一个锁。

而锁可以是任意对象，所以可以被任意对象调用的方法定义在object类中。

1. 对上一个1中联系用到的代码的优化；

做一个代码优化 ， 将方法封装到Res类中，这样不必像外界暴露 这个代码的优化还可以练一下。。。因为感觉自己在类的封装这一块好像不太熟悉。

总结一个编程技巧：类似于数学中的分配率：a(b+c) = ab+ac

if(条件a)

{

if(条件b1)

else

}

等价于

if(条件b1)

{

if(条件a)

}

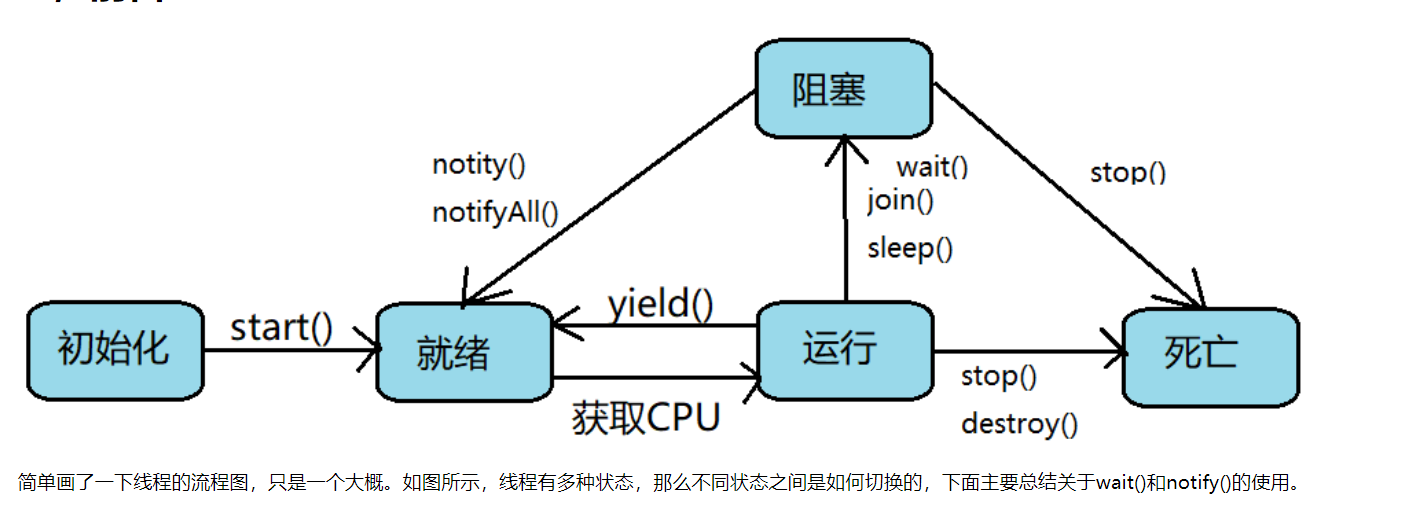
else

{

if(条件a)

}

1. 知识的回顾：线程的流程图：



4.生产者与消费者

在1.0版本中出现生产者生产2次的原因是：第一个sell（）函数中生产者的唤醒是直接从try后面的语句唤醒的，

也就是说t1，t2在获得执行权之后，t1唤醒了t2后不需要通过条件语句判断，就直接从等待状态开始运行，从而使count+1；

运行的流程为：两个生产者t1，t2。T1有执行权，标志位符合在t1运行好后，放弃资格，唤醒t2，t2有执行权，t2标志位不符合进入阻塞状态。唤醒t3，t3有执行权，并且符合标志位，t3运行一次后，放弃资格，唤醒t4，t4有执行权，但是不符合标志位，唤醒t1，此时注意唤醒后的t1不需要判断标志位即可执行，同理再运行一次后如果唤醒的是t2，那么同样也不需要在判断标志位即可执行。这样就出现了生产者同时生产两次的情况。

错误的关键原因是没有判断标记！！！

解决的方法：从if判断语句改成while循环，这样就能够避免从try后面的语句开始执行的问题。但是这样又会引入一个新的问题：就是死锁的问题，原因分析一下：因为notify只唤醒一个对象，如果不慎唤醒了自己方的一个对象，那么程序就会锁住，解决的方法是：将notify改为notifyAll()这样的话将所有的对象全部唤醒，总有能够满足条件的对象。

5.多线程之间的通信(生产着消费者JDK5.0升级版)

学习到了condition， await函数,signal函数以及lock与unlock()函数; 其中await和signal可以类比wait和notify

JDK1.5新特性 提供了多线程升级解决方案

将synchronized 用lock锁来代替执行

将wait()和notify()/notify()用 condition对象中的await()以及signal来代替执行

condition对象可以通过lock锁来进行获取。

本例中实现了只唤醒对方的操作。

lock与synchronized 的区别：lock这一个锁上可以有多个相关的condition，而synchronized这个关键字里面只能对应有一组相关的wait和notify

1. 多线程(停止线程):

stop方法已经过时，

如何停止线程？只有一种，run方法结束。

开启多线程运行，运行代码通常是循环结构。

只要控制住循环就可以让run方法结束，也就是线程结束，

多线程：停止线程

具体程序的实现： 创建一个StopThread 用来继承Runnable类，并且在里面重写run方法，StopThread里面有的私有变量有 flag（boolean类），

定义一个控制标识符真假的方法

Interrupt方法将处于冻结的线程强制恢复到运行状态上。

Thread提供 interrupt()方法。

特殊情况：当线程处于冻结状态，就不会读取到标记，线程就不会结束。

1. 守护线程

注意两点：1).当正在运行的线程都是守护线程的时候，java虚拟机退出；

1. .该方法必须在启动线程之前调用。

可以把守护线程理解为后台线程，然后和主线程之间的关系比作雅典娜和圣斗士的关系。当setDaemon()这一个函数里面的参数是true的时候启动守护线程。

8..join方法：当A线程执行时碰到B线程的.join方法时，就会让出执行权，等B线程都执行完，才会让A线程执行。

join可以临时加入线程运行。

这个程序中的main函数会交出执行权，在t1运行完后才会得到执行权。

9.多线程优先级以及yield()方法。

.join方法：当A线程执行时碰到B线程的.join方法时，就会让出执行权，等B线程都执行完，才会让A线程执行。

join可以临时加入线程运行。

.yield()方法，这一个方法能够使当前的线程放弃对cpu的控制权，是的循环能够执行。

setPriority() 能够设置优先级的优先权

这个程序中的main函数会交出执行权，在t1运行完后才会得到执行权。

for(int x=0; x<100; x++)

{

System.out.println(Thread.currentThread().getName().toString()+"-----run---"+x);

}

如果这样的代码有三轮循环，如此下去运行三次，如果其中的x基数比较大的话，那么就必须运行好这个代码块后才能接下运行第二个代码块，

多线程的好处是能够是代码块并发运行。

10.什么时候会用到线程呢？

答：当程序需要并行运行时，需要把每个线程封装起来，然后一起并行运行。

这个例子中用三种方法实现 线程的封装

并且温习了一下Thread和Runnable之间的关系以及区别。继承和实现