**多线程间协作与通信方式**

说到线程之间通信方式：依据我的理解 主要是有两种吧

1.是通过**共享变量**，线程之间通过该变量进行协作通信；

2.通过**队列**（本质上也是线程间共享同一块内存）来实现消费者和生产者的模式来进行通信；

**1.通过线程之间共享变量的方式**

这个就有必要说下 wait(),notify(),以及notifyAll() 这三个方法

这三个方法都是属于Object的方法；所以所有类都可以继承这三方法；

**wait()方法使得当前线程必须要等待，等到另外一个线程调用notify()或者notifyAll()方法。**

**notify()方法会唤醒一个等待当前对象的锁的线程。而notifyAll(）顾名思义；就是唤醒所有在等待中的方法；**

**wait()和notify()方法要求在调用时线程已经获得了对象的锁，因此对这两个方法的调用需要放在synchronized方法或synchronized块中。**

除了上述在synchronized代码块中使用 wait和 notify之外呢，其实在在java.util.concurrent包中，有两个很特殊的工具类，Condition和ReentrantLock（重入锁）,也可以同样实现线程间的交互协作。

这里要介绍一下condition中的await()和signal() 方法；

**先给A线程加锁，执行完再进入await之后会让线程沉睡，等待signal信号来叫醒，这时A线程解锁后运行B线程；b线程先加锁然后再执行，当到达阈值后也会睡眠的，然后解锁，把锁给A。就这样来进行通信的。**

使用condition可以非常灵活的去控制线程与线程之间的通信。因为在一个类中可以创建多个condition的实例，我们可以通过condition不同的实例的signal和await方法来标识不同的两个线程之间相互通信的标识，而不是统一使用object的notify和wait方法了。

2.通过队列来实现线程的通信

这里用的是java.util.concurrent包中linkedBlockingQueue 来进行线程间交互；

java.util.concurrent.LinkedBlockingQueue 是一个基于单向链表的、范围任意的（其实是有界的）、FIFO 阻塞队列。

**访问与移除操作是在队头进行，添加操作是在队尾进行，并分别使用不同的锁进行保护，只有在可能涉及多个节点的操作才同时对两个锁进行加锁。**

这里通过共享一个队列的信息，实现生产者和消费者