# Java多线程知识点总结

Thread类中，start()用来启动新创建的线程，其内部调用了run()方法。

与直接调用run()方法不一样，run()是在原来的线程中调用，没有新的线程启动。而start()则会启动新的线程。

Volatile变量可保证多线程对成员变量的操作对其他线程是透明的。

多线程环境下，当线程访问某个变量的时候，会从主存中将值load到线程内存中，建一个副本，线程直接操作副本。

而volatile则强制操作（读写）主存，保证读取的数据都是最新的。

但保证不了自增这一类操作的原子性，因为这些操作就不是原子性。

而volatile又禁止了指令重排，多以能在一定程度上保证有序性。

volatile多了一个lock前缀指令，形成内存屏障，从而：

1. 确保指令重排时，不会把后面的指令排前面。
2. 强制将对缓存的修改立即写入主存。
3. 写操作会导致其他线程对应的缓存无效。

可以使用volatile布尔变量来退出run()方法的循环或取消任务。

如果存在阻塞方法，且能响应中断，用Thread.interput()。

否则只能通过关闭底层资源，让代码抛出异常的方式来结束线程。

多线程共享数据。如果任务相同，则将数据放入Runnable实现类中，在run()方法中注意加同步锁。此数据就是共享的。

如果任务不用，则将数据放入一个公共类里，操作时加同步锁。

设计一个典型的场景，来阐述wait()、notify()、notifyAll()，生产消费者。

有生产者、消费者、缓冲队列（size=10）。

生产者不停地向缓冲队列里放数据，一旦队列满了，则wait()，并notify()消费者进行处理。

消费者不停地从缓冲队列里取数据，一旦队列空了，则wait()，并notify()生产者继续放数据。

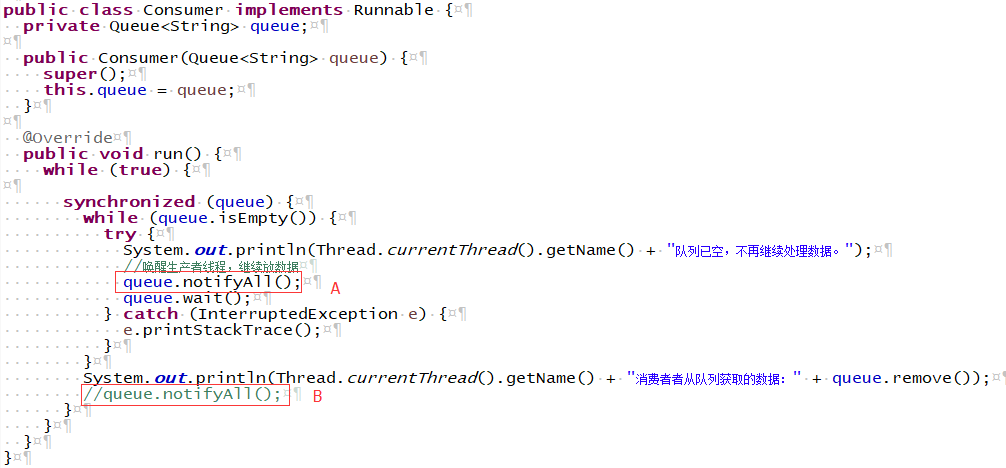
生产者：



notifyAll()放在A处，代表缓冲队列满了，再提醒消费者处理。

放在B处，代表只要缓冲队列里有数据，就提醒消费者处理。

消费者：



notifyAll()放在A处，代表缓冲队列空了，再提醒生产者放数据。

放在B处，代表只要处理了缓冲队列里的一个数据，就提醒生产者放数据。

测试类：



先解释一下notify()和notifyAll()。从字面意义上也很好理解，这两个都是用来唤醒wait()的线程，如果有多个线程wait()，notify()只唤醒其中一个，是哪一个就看JVM了。notifyAll()则唤醒所有wait()的线程，至于哪一个能拿到锁并执行，就看JVM了。

举个例子，有10个要吃饭的人都在睡觉（wait），我是送饭的人，我随机叫醒其中一个来吃饭，这是notify。我大喊一声“吃饭啦”，10个人全起来抢饭，这是notifyAll，至于是哪一个能抢到，就看命了。

使用wait、notify、notifyAll要注意的点：

1. 必须在synchronized里使用。
2. 必须在while而非if内使用wait。这是为了线程在被唤醒的前后都会去检查条件是否被满足。以刚才吃饭为例，有两个要吃饭的人，没饭吃就睡觉，有饭吃就被唤醒，然后都去领饭吃。但是在唤醒之后，吃之前要检查一下是否真的有饭，多线程情况下，可能唯一一碗饭被别人抢先拿走了，这时候本应去睡觉，但如果不做检查的话，就会一直去要吃的，从而产生错误。
3. 在多线程要处理的公共对象（数据）那里用wait，在本文，就是缓冲队列。
4. 如果是真多线程，比如多个生产者，多个消费者，用notifyAll更合适。