****独占锁是一种悲观锁，synchronized就是一种独占锁，会导致其它所有需要锁的线程挂起，等待持有锁的线程释放锁。而另一个更加有效的锁就是乐观锁。****

****所谓乐观锁就是，每次不加锁而是假设没有冲突而去完成某项操作，如果因为冲突失败就重试，直到成功为止。乐观锁用到的机制就是CAS，Compare and Swap。****

通常将 CAS 用于同步的方式是从地址 V 读取值 A，执行多步计算来获得新 值 B，然后使用 CAS 将 V 的值从 A 改为 B。如果 V 处的值尚未同时更改，则 CAS 操作成功。

****谈到并发，不得不谈ReentrantLock；而谈到ReentrantLock，不得不谈AbstractQueuedSynchronizer（AQS）！****

类如其名，抽象的队列式的同步器，AQS定义了一套多线程访问共享资源的同步器框架，许多同步类实现都依赖于它，如常用的ReentrantLock/Semaphore/CountDown

Latch...。

# **实现Lock接口的基本思想**

          需要实现锁的功能，两个必备元素，一个是表示（锁）状态的变量（我们假设0表示没有线程获取锁，1表示已有线程占有锁），另一个是队列，队列中的节点表示因未能获取锁而阻塞的线程。为了解决多核处理器下多线程缓存不一致的问题，表示状态的变量必须声明为voaltile类型，并且对表示状态的变量和队列的某些操作要保证原子性和可见性。原子性和可见性的操作主要通过Atomic包中的方法实现。

# **Lock与synchronized的区别**

          1. Lock的加锁和解锁都是由java代码配合native方法（调用操作系统的相关方法）实现的，而synchronize的加锁和解锁的过程是由JVM管理的

          2. 当一个线程使用synchronize获取锁时，若锁被其他线程占用着，那么当前只能被阻塞，直到成功获取锁。而Lock则提供超时锁和可中断等更加灵活的方式，在未能获取锁的     条件下提供一种退出的机制。

          3. 一个锁内部可以有多个Condition实例，即有多路条件队列，而synchronize只有一路条件队列；同样Condition也提供灵活的阻塞方式，在未获得通知之前可以通过中断线程以及设置等待时限等方式退出条件队列。

         4. synchronize对线程的同步仅提供独占模式，而Lock即可以提供独占模式，也可以提供共享模式