ReentrantLock支持两种获取锁的方式，一种是公平模型，一种是非公平模型。

公平锁模型：一个线程在获取了锁之后，再次去获取了同一个锁，这时候仅仅是把状态值进行累加。如果线程A释放了一次锁，仅仅是把状态值减了，只有线程A把此锁全部释放了，状态值减到0了，其他线程才有机会获取锁。当A把锁完全释放后，state恢复为0，然后会通知队列唤醒B线程节点，使B可以再次竞争锁。

非公平锁模型：当线程A执行完之后，要唤醒线程B是需要时间的，而且线程B醒来后还要再次竞争锁，

所以如果在切换过程当中，来了一个线程C，那么线程C是有可能获取到锁的，如果C获取到了锁，B就只能继续乖乖休眠了

CAS，Compare and Swap即比较并交换，设计并发算法时常用到的一种技术，java.util.concurrent包全完建立在CAS之上。ReentrantLock是基于AQS实现的，这在下面会讲到，AQS的基础又是CAS。

CAS有三个操作数：内存值V、旧的预期值A、要修改的值B，当且仅当预期值A和内存值V相同时，将内存值修改为B并返回true，否则什么都不做并返回false。