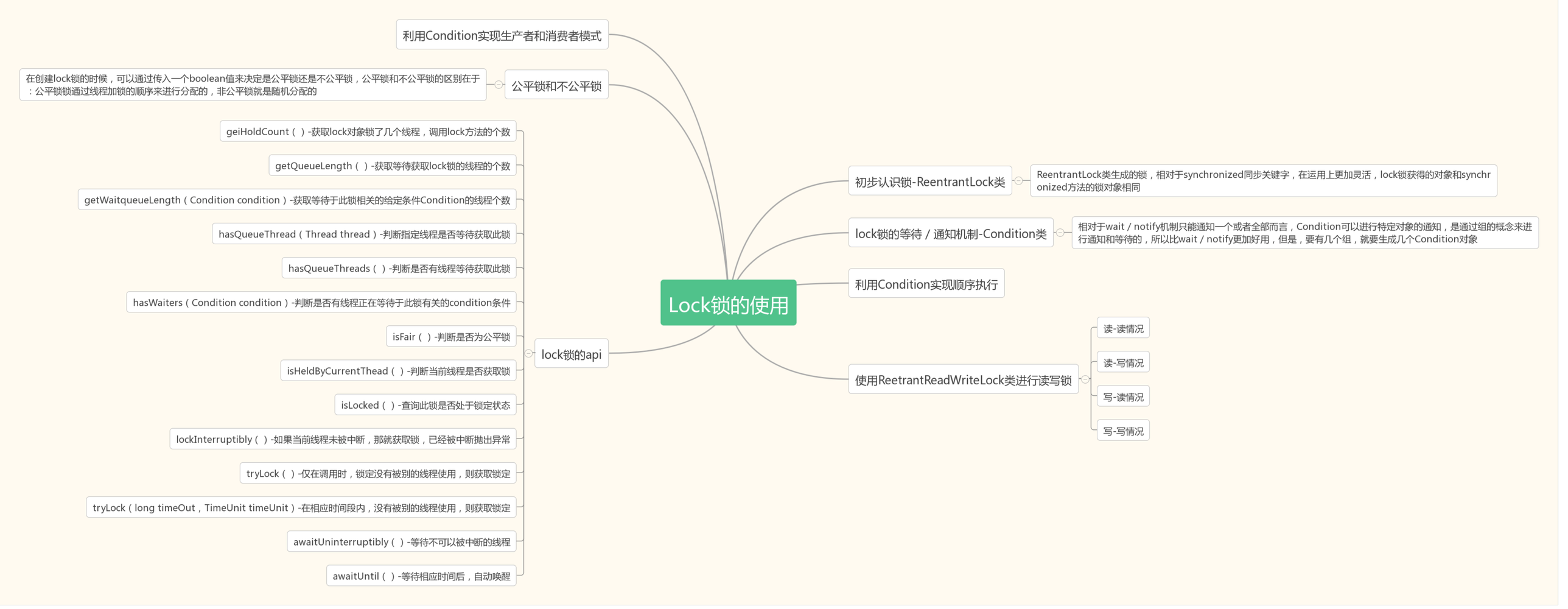
Lock锁的使用
利用Condition实现生产者和消费者模式
公平锁和不公平锁
在创建lock锁的时候，可以通过传入一个boolean值来决定是公平锁还是不公平锁，公平锁和不公平锁的区别在于：公平锁锁通过线程加锁的顺序来进行分配的，非公平锁就是随机分配的
lock锁的api
geiHoldCount（）-获取lock对象锁了几个线程，调用lock方法的个数
getQueueLength（）-获取等待获取lock锁的线程的个数
getWaitqueueLength（Condition condition）-获取等待于此锁相关的给定条件Condition的线程个数
hasQueueThread（Thread thread）-判断指定线程是否等待获取此锁
hasQueueThreads（）-判断是否有线程等待获取此锁
hasWaiters（Condition condition）-判断是否有线程正在等待于此锁有关的condition条件
isFair（）-判断是否为公平锁
isHeldByCurrentThead（）-判断当前线程是否获取锁
isLocked（）-查询此锁是否处于锁定状态
lockInterruptibly（）-如果当前线程未被中断，那就获取锁，已经被中断抛出异常
tryLock（）-仅在调用时，锁定没有被别的线程使用，则获取锁定
tryLock（long timeOut，TimeUnit timeUnit）-在相应时间段内，没有被别的线程使用，则获取锁定
awaitUninterruptibly（）-等待不可以被中断的线程
awaitUntil（）-等待相应时间后，自动唤醒
初步认识锁-ReentrantLock类
ReentrantLock类生成的锁，相对于synchronized同步关键字，在运用上更加灵活，lock锁获得的对象和synchronized方法的锁对象相同
lock锁的等待 / 通知机制-Condition类
相对于wait / notify机制只能通知一个或者全部而言，Condition可以进行特定对象的通知，是通过组的概念来进行通知和等待的，所以比wait / notify更加好用，但是，要有几个组，就要生成几个Condition对象
利用Condition实现顺序执行
使用ReetrantReadWriteLock类进行读写锁
读-读情况
读-写情况
写-读情况
写-写情况