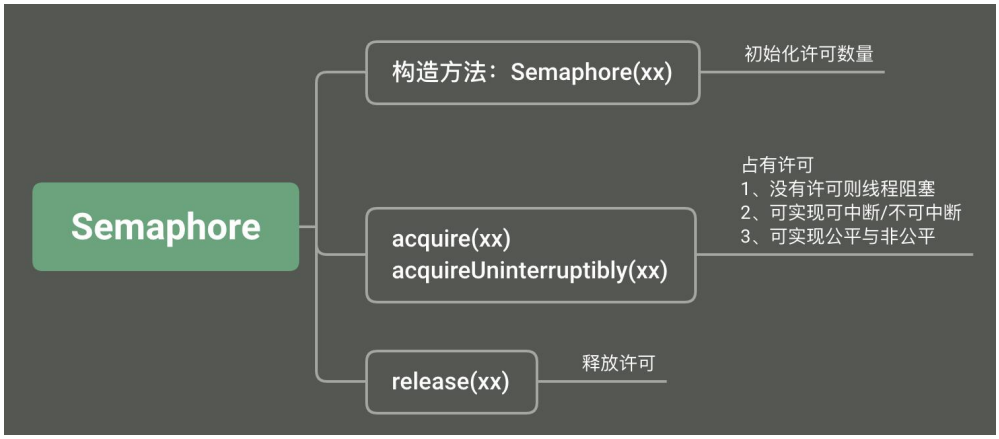
Semaphore/CountDownLatch/CycBarrier

Semaphore

共享锁可以实现线程同步功能，Semaphore就是利用state来实现线程同步控制的同步工具类，通过预先设置
state的大小，线程获取到锁则state+1，释放锁则state-1，也将这同步模式称为准许permit申请/释放，常
用在对资源的限制访问，如数据库连接池，限流等；Semaphore并没有实现Lock工具类，因此自实现了acquire/
release等方法供外界调用.



Semaphore实现策略与之前的读锁类似，调用acquire(permit)，会首先判断申请参数是否合法permit>=0，然后调用sync.acquireSharedInterruptibly，进行state>0判断，如果有资源则去CAS竞争，否则返回失败，进入共享锁相同的挂起等待流程.也实现了公平/非公平、可中断/不可中断模式，其中默认是非公平可中断的模式.
使用时，需要先调用semaphore.acuire，退出前调用semaphore.release.

CountDownLatch

CountDownLatch是为了解决**两类线程**之间的同步关系的同步工具，例如A线程要等待B/C线程执行完某个任务后才能进行下一步操作；实现了tryAcquireShared方法，每次只访问state判断当前state是否为0，并不修改state的值，通过await调用tryAcquireShared，将线程阻塞等待state为0被唤醒；实现了tryReleaseShared方法cas将state-1，只有state=0时才调用doReleaseShared唤醒阻塞线程.

CyclicBarrier

CyclicBarrier是为了解决线程之间同步到达后再进行下一轮操作的同步关系的同步工具，例如A、B、C线程都完成了准备工作后再进入下一阶段；初始时指定同步线程的数量count，实现了await函数，利用ReentrantLock对count进行修改独占count-1修改，判断count是否为0，不为0则阻塞在ReentrantLock生成的Condition条件trip下，直到被最后一个调用await的线程调用signalAll唤醒.

- **参考资料**
  - Java Semaphore/CountDownLatch/CyclicBarrier 深入解析(原理篇) - 简书 (bingj.com)