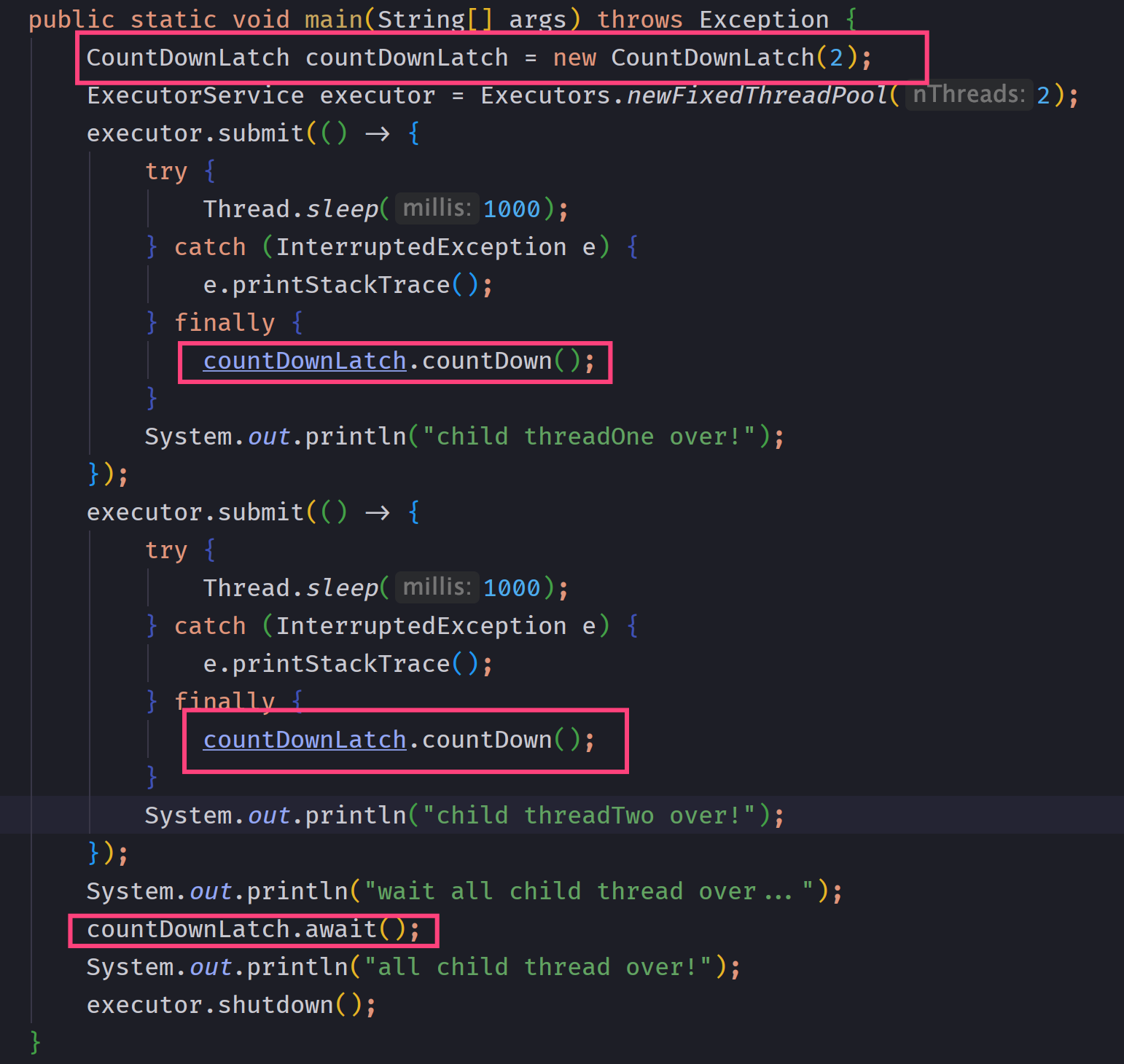
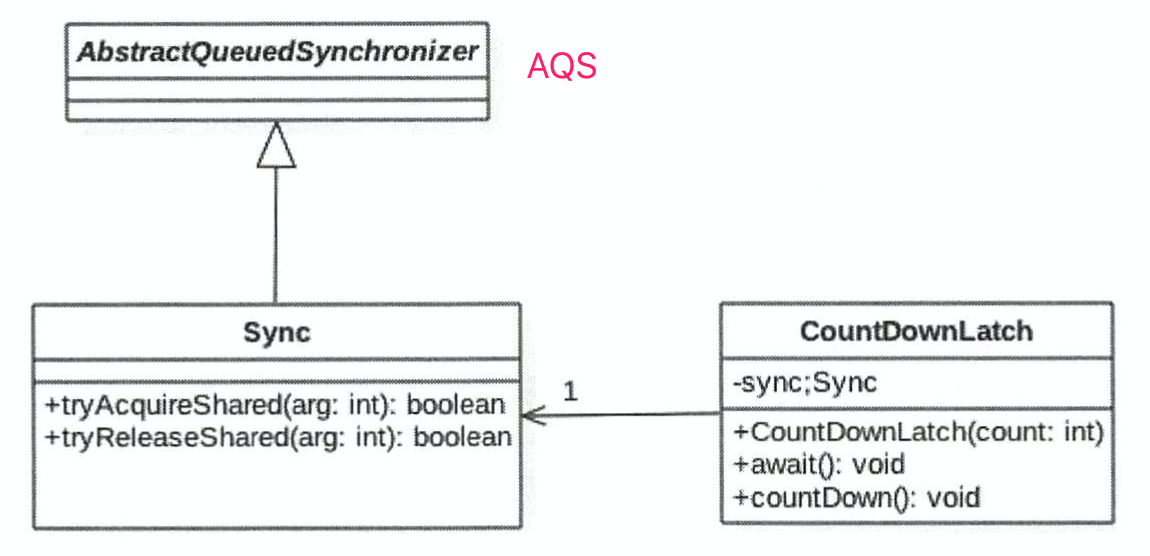
## CountDownLatch



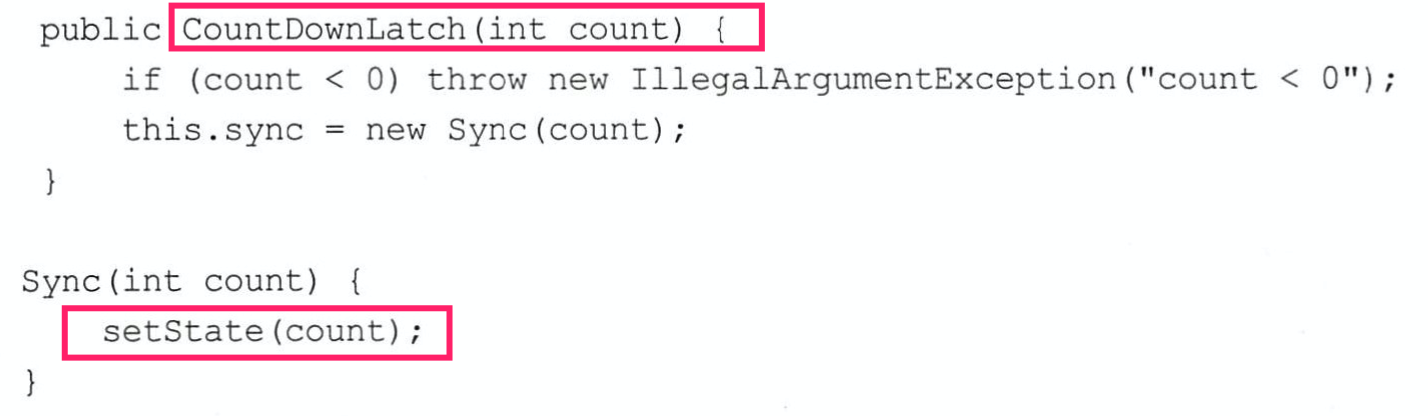
1. CountDownLatch简单实例



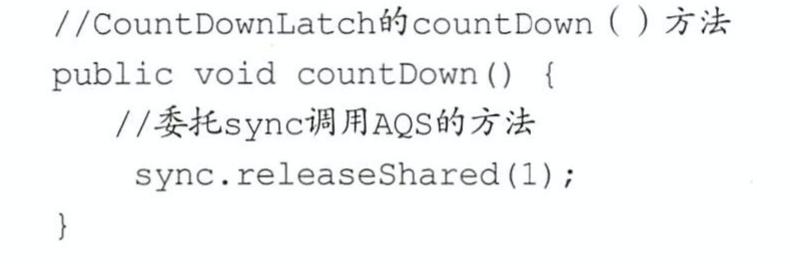
1. CountDownLatch与join的区别
   1. CountDownLatch比join灵活。调用一个子线程的join后，该线程会一直被阻塞直到子线程运行完毕；CountDownLatch可以在运行中任何时刻递减计数器，不一定要等线程结束
   2. 若使用线程池管理线程Runnable，没有办法调用线程的join方法
2. 实现原理

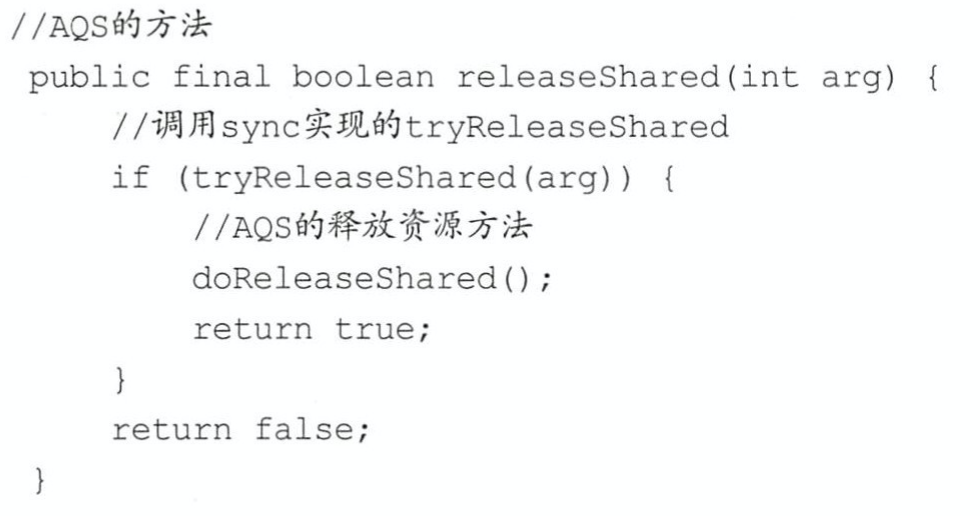


CountDownLatch是使用AQS实现的，把计数器的值赋值给了AQS的state，也就是AQS的状态值表示计数器值



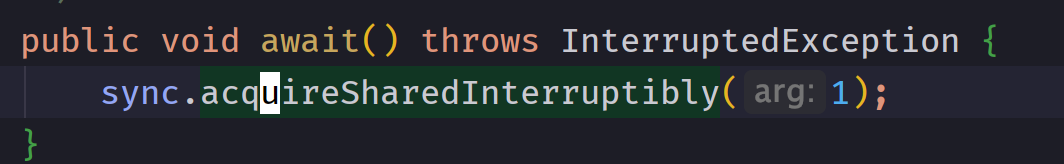
* 1. countDown()，线程调用该方法后，计数器的值递减，递减后如果计数器的值为0则唤醒所有因调用await方法阻塞的线程。





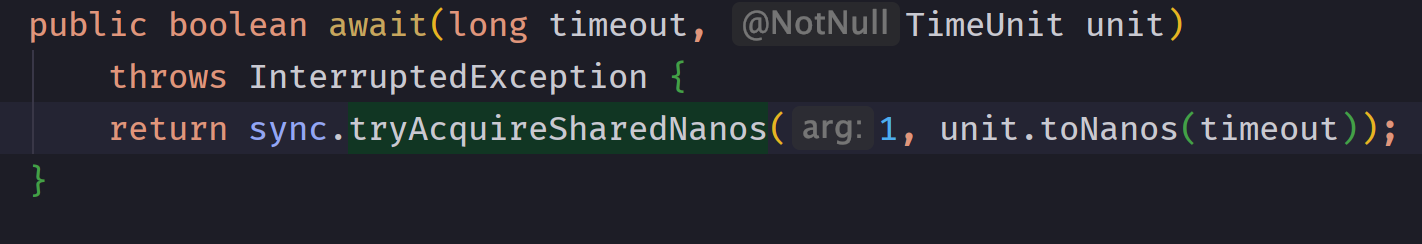


* 1. await()方法，当线程调用CountDownLatch的await方法后，当前线程会被阻塞，直到以下情况之一发生：CountDownLatch的计数器值为0时，即所有线程调用了countDown方法；其它线程调用当前线程的interrupt()方法中断当前线程，当前线程抛出InterruptedException 异常后返回

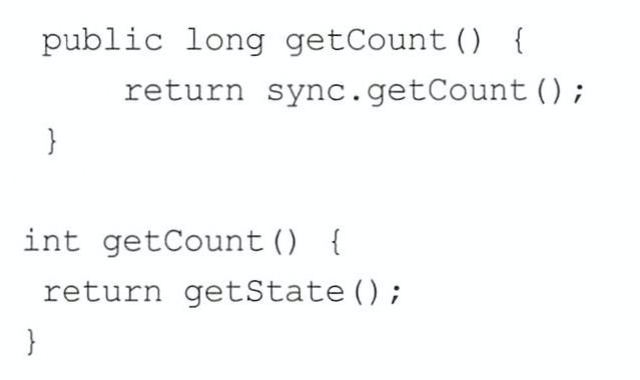




* 1. boolean await(long timeout, TimeUnit unit)方法，当线程调用CountDownLatch的await方法后，当前线程会被阻塞，直到以下情况之一发生：CountDownLatch的计数器值为0时，即所有线程调用了countDown方法，这时候返回true；当设置的timeout超时时间到了，这时候返回false；其它线程调用当前线程的interrupt()方法中断当前线程，当前线程抛出InterruptedException 异常后返回

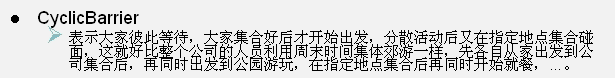


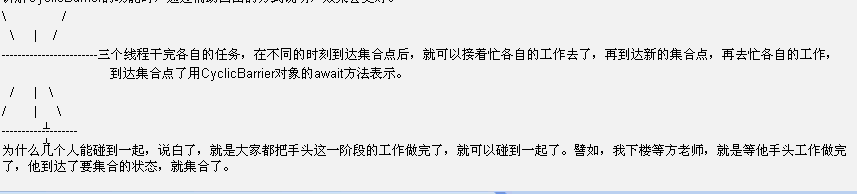
* 1. long getCount()，获取当前计数器的值，也就是AQS计数器的值，一般在测试的时候用



## CyclicBarrier

CountDownLatch的计数器是**一次性**的，也就是计数器到0之后，再调用await和countDown方法都会立刻返回，起不到线程同步的作用了。为了满足**计数器可以重置**的需求，提供了CyclicBarrier。CyclicBarrier回环屏障，可以让一组线程全部到达一个状态后再全部**同时**执行。回环，当所有等待线程执行完毕并重置CyclicBarrier的状态后可以被重用；屏障，线程调用await方法后会被阻塞，这个阻塞点叫做屏障点，等所有线程都调用了await后，线程们就可以冲破屏障，继续向下运行

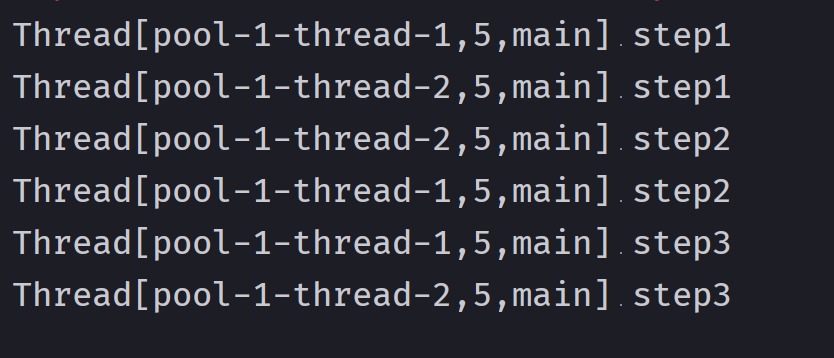




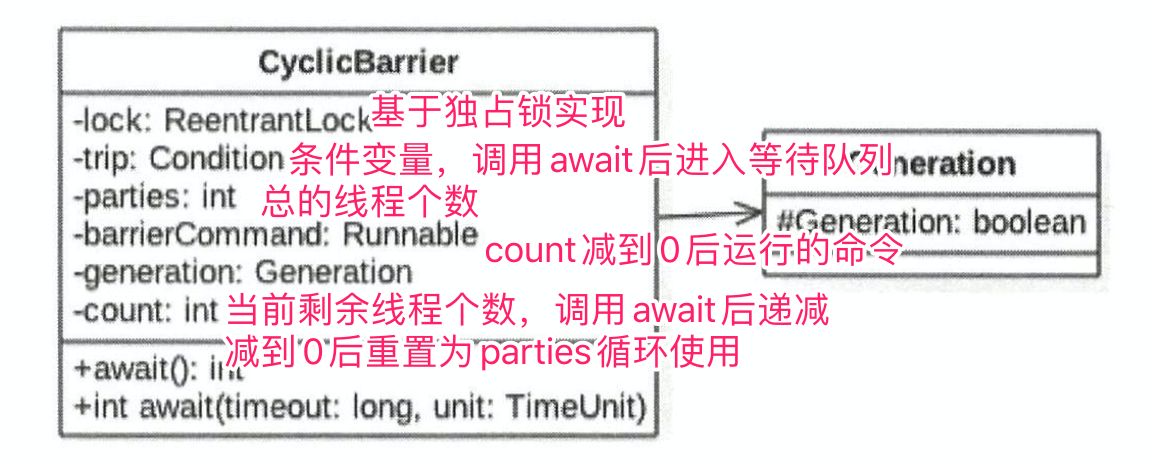
1. CyclicBarrier简单实例，一个任务由阶段1、阶段2和阶段3组成，每个线程要串行地执行阶段1、阶段2和阶段3，当多个线程执行该任务时，必须保证所有线程阶段1全部完成后才能进入阶段2，当所有线程的阶段2全部完成后才能进入阶段3执行

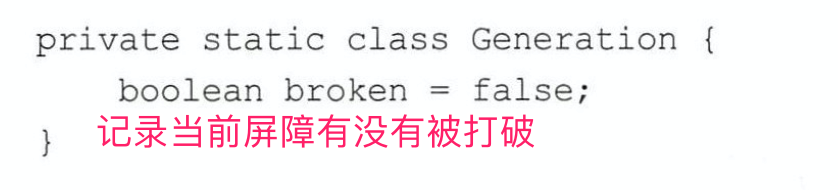


输出结果



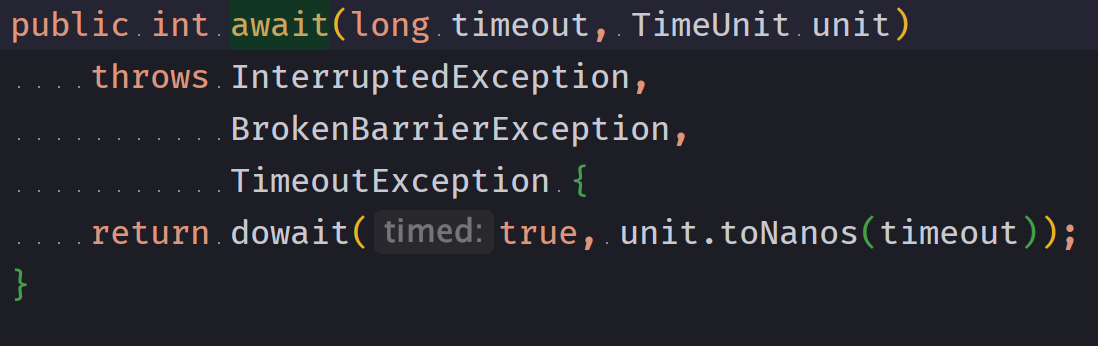
1. 实现原理



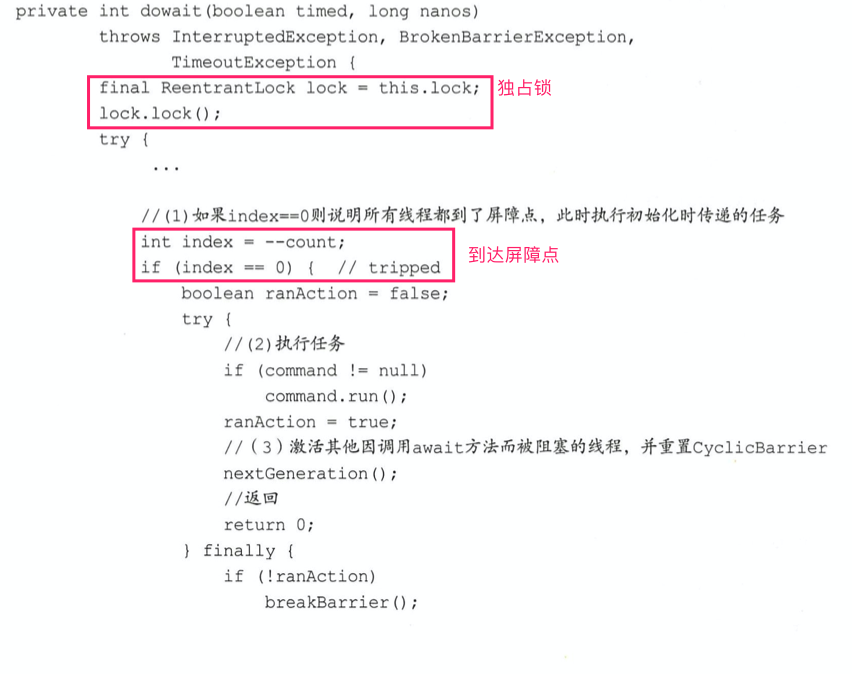


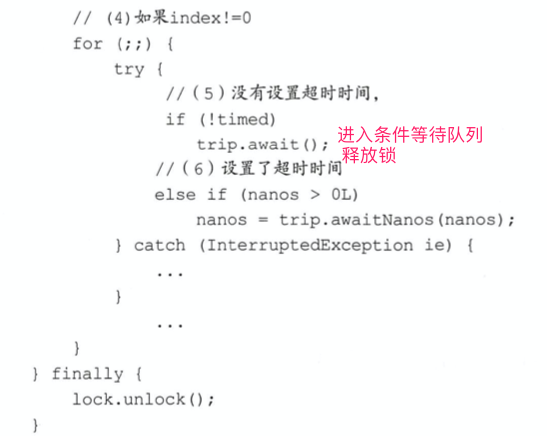
* 1. await()方法，当前线程调用该方法时会被阻塞，直到满足下面条件之一才会返回：parties个线程都调用了await()方法，即所有线程都到达了屏障点；设置的超时时间到了后返回false；其它线程调用了当前线程的interrupt()方法中断了当前线程，则当前线程抛出InterruptedException异常而返回；与当前屏障点关联的Generation对象的broken标识被设置为true时，会抛出BrokenBarrierException异常而返回

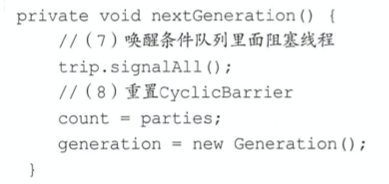




* 1. dowait()方法，该方法实现了CyclicBarrier的核心功能

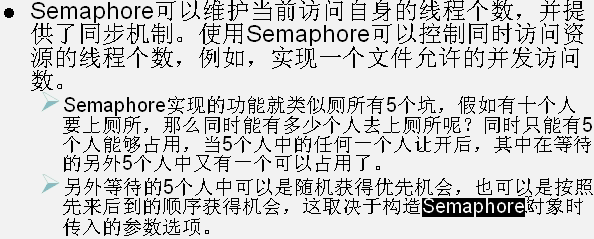


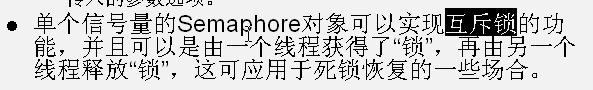




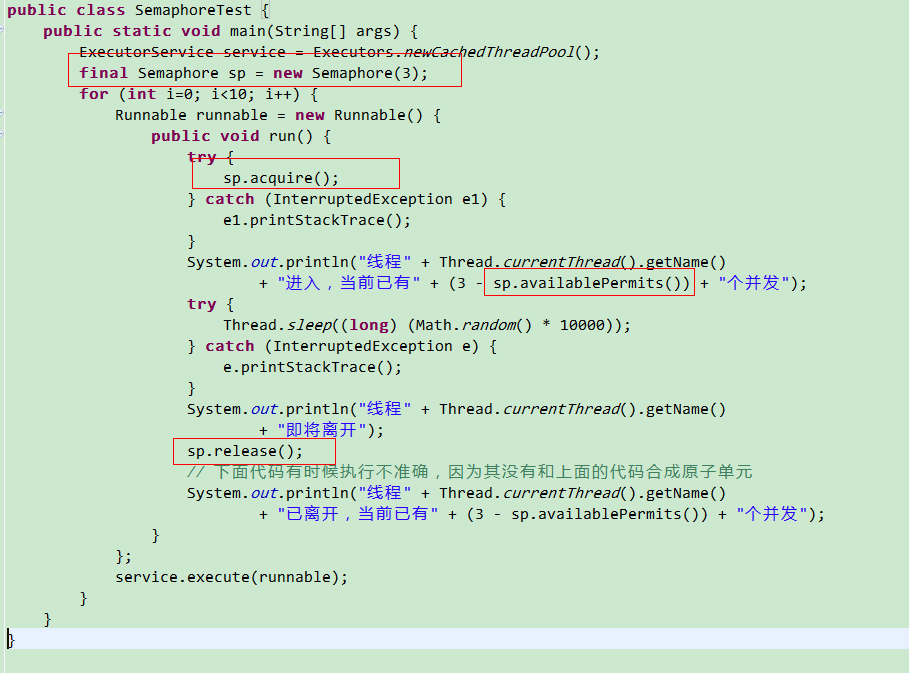
1. 总结，相对于CountDownLatch，CyclicBarrier是可以复用的，并且适合分段任务有序执行的场景。CyclicBarrier通过ReentrantLock实现计数器的原子性更新，并使用条件变量队列实现线程同步

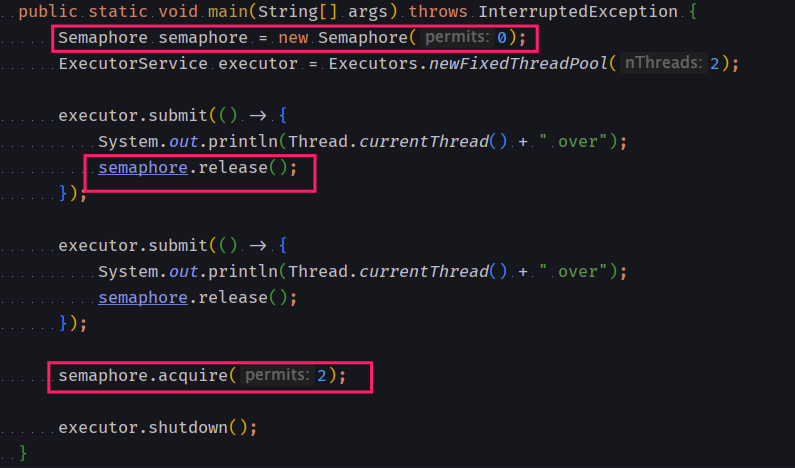
## semaphore信号量



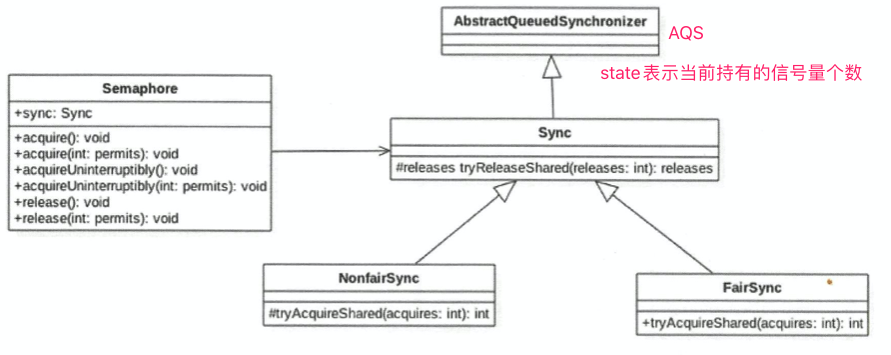


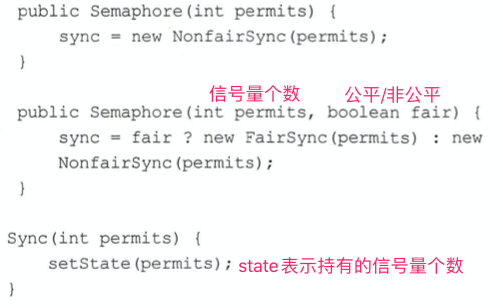
1. semaphore信号量简单实例



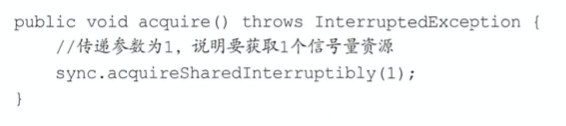


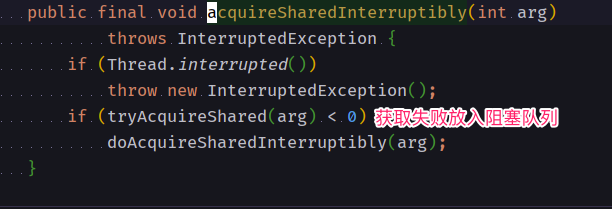
1. 实现原理，基于AQS实现

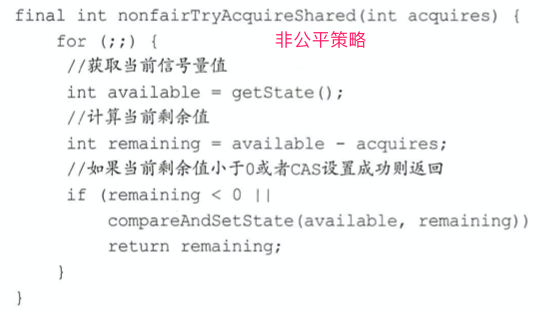


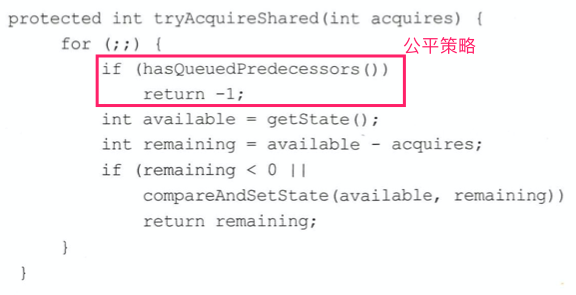


* 1. void acquire()方法

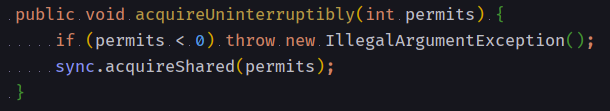






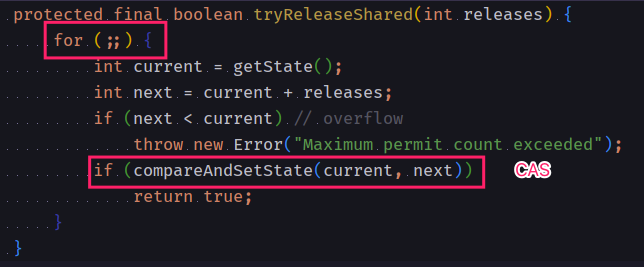


* 1. void acquire(int permits)方法，获取permits个信号量
  2. void acquireUninterruptibly()方法，对中断不响应



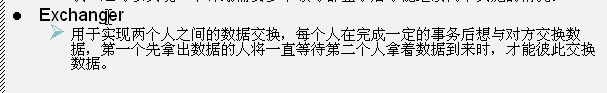
* 1. void release()方法





* 1. void acquire()方法

## Exchanger



1. Exchanger简单实例

