reentrantLock知识点

一、一些基本概念

1.ReentrantLock是一个可重入的互斥锁。

互斥：ReentrantLock在同一个时间只能被一个线程持有；重入：ReentrantLock可以被单个线程多次获取

ReentrantLock分为“公平锁”和“非公平锁”，ReentrantLock通过一个FIFO的等待队列来管理获得该锁的所有线程。在公平锁的机制下，线程依次排队获得锁；在非公平锁的机制下，如果锁是可获取的，就会从队列中取出，不管是不是在队头。

Condition是需要和Lock联合使用的：通过Condition中的await()方法，能让线程阻塞[类似于wait()]；通过Condition的signal()方法，能让唤醒线程[类似于notify()]

2. AQS，即AbstractQueuedSynchronizer类，他是java中管理“锁”的抽象类，AQS是独占锁和共享锁的公共父类

3.AQS的锁类别：

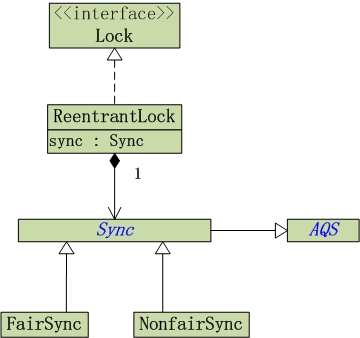
独占锁：锁在一个时间点只能被一个线程占有，根据锁的获得机制，划分为“公平锁”和“非公平锁”。公平锁，获取锁线程是按照FIFO队列依次获取锁；非公平锁，当有线程要获取锁时，它就会直接获取锁。独占锁的典型就是ReentrantLock，ReentrantReadWriteLock.WriteLock也是独占锁。

共享锁：能被多个线程同时拥有，被共享的锁。ReentrantReadWriteLock.ReadLock，CyclicBarrier， CountDownLatch和Semaphore都是共享锁。

4.CLH队列

CLH是AQS"等待锁"的线程队列，存放的是等待获取该锁的线程。他是一种基于链表非阻塞的FIFO队列，并发条件线不会发生阻塞，是通过自旋锁和cas保证节点插入和移除的原子性

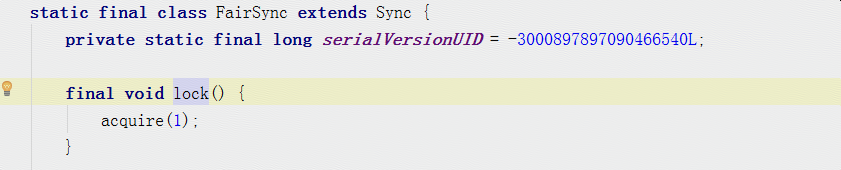
二、ReentrantLock的数据结构



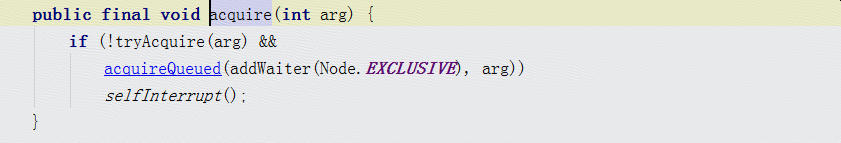
ReentrantLock默认实现的是非公平锁

三、公平锁的获取过程

New ReentrantLock(true)，指定的锁就是公平锁。获取锁时，会调用ReentrantLock的lock()方法，lock()方法调用的是内部抽象类Sync的方法lock()，会调用Sync的子类FairSync中的lock()方法



调用的是AQS类中的acquire()方法：



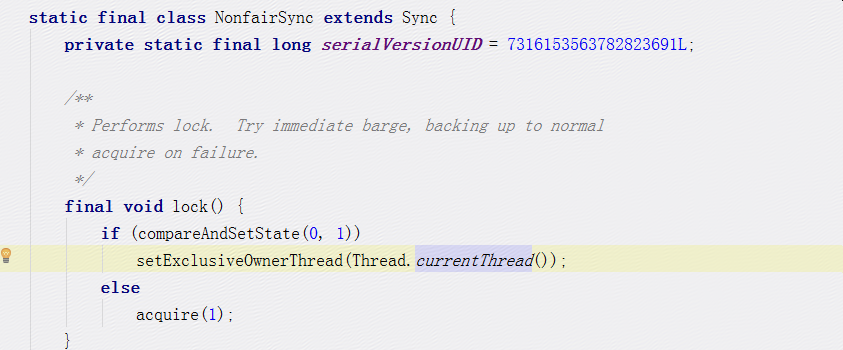
acquire()方法的最终目的就是获得公平锁：具体实现是在子类的tryAcquire()方法中。

(1)先是通过tryAcquire()尝试获得锁，如果成功直接返回，如果失败，再通过acquireQueued()获取锁

(2)尝试失败的情况下，会先通过addWaiter()来将“当前线程”加入到“CLH队列”末尾；然后调用acquireQueued()，在CLH队列中排序等古代获取锁，再次过程中线程处于休眠状态，直到获取锁后才返回。如果在休眠等待过程中被中断过，则调用selfInterrupt()来自己产生中断。

四、非公平锁的获取过程

New ReentrantLock()默认就是使用的非公平锁，获取锁是，会调用NonfairSync类的lock()方法：



公平锁与非公平锁的lock的区别在于

公平锁：会直接调用acquire()方法

非公平锁：先通过cas判断当前锁的状态是否空闲，如果是，直接拿到这个锁。

非公平锁的tryAcquire()方法会尝试获取锁

(1)如果“锁”是空闲的，当前线程直接拿到锁

(2)如果“锁”的持有者是当前线程，更改锁状态即可

(3)上述两条都不成立，认为尝试失败。

公平锁与非公平锁的区别：

公平锁在尝试获取锁时，即使锁没有被任何线程持有，也会判断当前线程是都在CLH队列的表头，如果是才获取锁。

非公平锁在尝试获取锁时，如果锁没有被任何线程持有，不管当前线程在CLH什么位置，直接拿到锁。即使tryAcquire()尝试获取失败，放入了CLH队列，依然是忽略位置，空闲就直接获取锁。