# 概述

同步器AbstractQueuedSynchronizer，简称AQS，主要是为了java的跨平台。C++中在使用信号量时都直接使用操作系统提供的信号量以及信号量自带的同步队列，因此不同操作系统需要使用不同的系统调用。java为了跨平台，则将同步队列抽象出来作为一个抽象类AbstractQueuedSynchronizer。

AQS是一个抽象类，不能直接创建对象，JDK1.8中实现了多个同步器，比如Semaphore、CountDownLatch等都创建了相应的内部类来实现AQS类，AQS的继承关系如下所示。



AQS子类继承关系

为了在不同场景使用同步器，JDK特此提供多种同步器，AQS类是所有这些同步器的核心，各种同步器都只是依赖AQS提供的基本功能做了一层封装。

本文档主要学习JDK1.8中提供的AbstractQueuedSynchronizer类以及其子类的知识点。

# AbstractQueuedSynchronizer

AbstractQueuedSynchronizer同步器，简称AQS。其底层实现依赖于双向链表结构，链表节点来自于其内部类Node。

## 主要成员

|  |  |  |
| --- | --- | --- |
| 名字 | 类型 | 描述 |
| head | Node | 队列头指针 |
| tail | Node | 队列尾指针 |
| state | int | 被同步器同步资源的个数，如果state等于0，获取资源的线程需要排队 |
| headOffset | long | 记录head成员在AQS类中的偏移量 |
| tailOffset | long | 记录tail成员在AQS类中的偏移量 |
| stateOffset | long | 记录state成员在AQS类中的偏移量 |

# 子类

## 信号量Semaphore

信号量提供两种同步器，一种是公平同步器FairSync，一种是非公平同步器NonfairSync，这两个同步器具有很多相似的地方，因此对两种同步器抽象出了一个基类Sync。

Sync、FairSync、NonfairSync都是Semaphore的内部类。

### 构造方法

第一种方式只传递允许同步的线程数permits，默认使用非公平同步器NonfairSync。



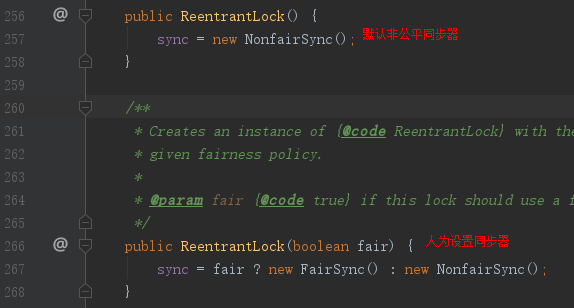
第二种方式传递允许同步的线程数，同时设置同步器类型，如果fair是true创建公平同步器FairSync，否则为非公平同步器。



## 可重入锁ReentrantLock

可重入锁和Semaphore类似，都包含公平同步器好非公平同步器，只不过可重入锁的state变量的值如果为0表示没有线程占有锁，如果为1表示有一个线程占有锁，如果大于1表示该锁被同一个线程多次获取并占有（可以通过多次调用lock操作，调试查看锁中state变量值的变化，每调用一次lock，state自增1，调用一次unlock，state自减）。

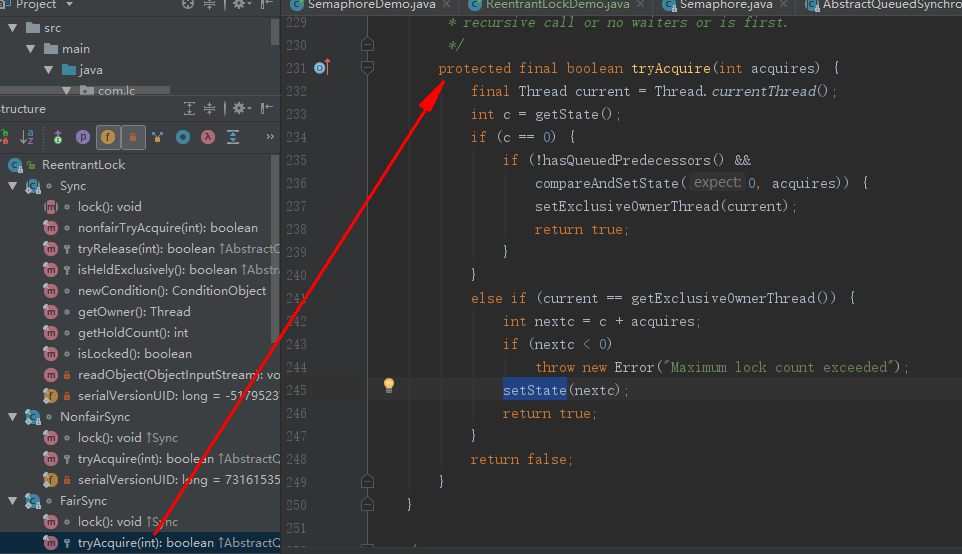
### 构造方法



### 重要操作

#### lock

如果当前的可重入锁使用公平的同步器，调用lock操作时，由于多态会执行FairSync类中的tryAcquire操作。如果此时state==0，表示当前时刻没有线程占有锁，如果同步器队列中有其他线程在排队，则直接返回false，



## 可重入读写锁ReentrantWriteReadLock