[https://mp.weixin.qq.com/s?\_\_biz=MzA5MTkxMDQ4MQ==&mid=2648933130&idx=1&sn=cecc6bd906e79a86510c1fbb0e66cd21&chksm=88621b34bf159222042da8ed4b633e94ca04a614d290d54a952a668459a339ebec0c754d562d&token=702505185&lang=zh\_CN&scene=21#wechat\_redirect](https://mp.weixin.qq.com/s?__biz=MzA5MTkxMDQ4MQ==&mid=2648933130&idx=1&sn=cecc6bd906e79a86510c1fbb0e66cd21&chksm=88621b34bf159222042da8ed4b633e94ca04a614d290d54a952a668459a339ebec0c754d562d&token=702505185&lang=zh_CN&scene=21" \l "wechat_redirect)

# JUC中的Semaphore（信号量）

Semaphore（信号量）为多线程协作提供了更为强大的控制方法，前面的文章中我们学了synchronized和重入锁ReentrantLock，这2种锁一次都只能允许一个线程访问一个资源，而**信号量可以控制有多少个线程可以访问特定的资源**。

## Semaphore常用场景：限流

举个例子：

比如有个停车场，有5个空位，门口有个门卫，手中5把钥匙分别对应5个车位上面的锁，来一辆车，门卫会给司机一把钥匙，然后进去找到对应的车位停下来，出去的时候司机将钥匙归还给门卫。停车场生意比较好，同时来了100两车，门卫手中只有5把钥匙，同时只能放5辆车进入，其他车只能等待，等有人将钥匙归还给门卫之后，才能让其他车辆进入。

上面的例子中门卫就相当于Semaphore，车钥匙就相当于许可证，车就相当于线程。

## Semaphore主要方法

* Semaphore(int permits)：构造方法，参数表示许可证数量，用来创建信号量
* Semaphore(int permits,boolean fair)：构造方法，当fair等于true时，创建具有给定许可数的计数信号量并设置为公平信号量
* void acquire() throws InterruptedException：从此信号量获取1个许可前线程将一直阻塞，相当于一辆车占了一个车位，此方法会响应线程中断，表示调用线程的interrupt方法，会使该方法抛出InterruptedException异常
* void acquire(int permits) throws InterruptedException ：和acquire()方法类似，参数表示需要获取许可的数量；比如一个大卡车要入停车场，由于车比较大，需要申请3个车位才可以停放
* void acquireUninterruptibly(int permits) ：和acquire(int permits) 方法类似，只是不会响应线程中断
* boolean tryAcquire()：尝试获取1个许可，不管是否能够获取成功，都立即返回，true表示获取成功，false表示获取失败
* boolean tryAcquire(int permits)：和tryAcquire()，表示尝试获取permits个许可
* boolean tryAcquire(long timeout, TimeUnit unit) throws InterruptedException：尝试在指定的时间内获取1个许可，获取成功返回true，指定的时间过后还是无法获取许可，返回false
* boolean tryAcquire(int permits, long timeout, TimeUnit unit) throws InterruptedException：和tryAcquire(long timeout, TimeUnit unit)类似，多了一个permits参数，表示尝试获取permits个许可
* void release()：释放一个许可，将其返回给信号量，相当于车从停车场出去时将钥匙归还给门卫
* void release(int n)：释放n个许可
* int availablePermits()：当前可用的许可数

## 示例1：Semaphore简单的使用



输出：



代码中 newSemaphore(2)创建了许可数量为2的信号量，每个线程获取1个许可，同时允许两个线程获取许可，从输出中也可以看出，同时有两个线程可以获取许可，其他线程需要等待已获取许可的线程释放许可之后才能运行。为获取到许可的线程会阻塞在 acquire()方法上，直到获取到许可才能继续。

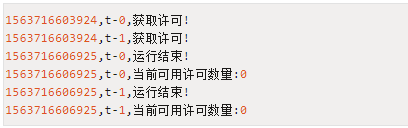
## 示例2：获取许可之后不释放

门卫（Semaphore）有点呆，司机进去的时候给了钥匙，出来的时候不归还，门卫也不会说什么。最终结果就是其他车辆都无法进入了。

如下代码：



输出：



上面程序运行后一直无法结束，观察一下代码，代码中获取许可后，没有释放许可的代码，最终导致，可用许可数量为0，其他线程无法获取许可，会在 semaphore.acquire();处等待，导致程序无法结束。

## 示例3：释放许可正确的姿势

示例1中，在finally里面释放锁，会有问题么？

如果获取锁的过程中发生异常，导致获取锁失败，最后finally里面也释放了许可，最终会怎么样，导致许可数量凭空增长了。



输出：



程序中信号量许可数量为1，创建了3个线程获取许可，线程t1获取成功了，然后休眠100秒。其他两个线程阻塞在 semaphore.acquire();方法处，代码中对线程t2、t3发送中断信号，我们看一下Semaphore中acquire的源码：

public void acquire() throws InterruptedException

这个方法会响应线程中断，主线程中对t2、t3发送中断信号之后， acquire()方法会触发 InterruptedException异常，t2、t3最终没有获取到许可，但是他们都执行了finally中的释放许可的操作，最后导致许可数量变为了2，导致许可数量增加了。所以程序中释放许可的方式有问题。需要改进一下，获取许可成功才去释放锁。

正确的释放锁的方式，如下：



输出：



程序中增加了一个变量 acquireSuccess用来标记获取许可是否成功，在finally中根据这个变量是否为true，来确定是否释放许可。

## 示例4：在规定的时间内希望获取许可

司机来到停车场，发现停车场已经满了，只能在外等待内部的车出来之后才能进去，但是要等多久，他自己也不知道，他希望等10分钟，如果还是无法进去，就不到这里停车了。

Semaphore内部2个方法可以提供超时获取许可的功能：

public boolean tryAcquire(long timeout, TimeUnit unit) throws InterruptedException

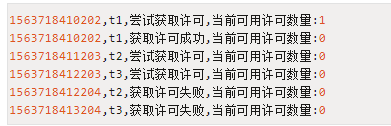
public boolean tryAcquire(int permits, long timeout, TimeUnit unit) throws InterruptedException

在指定的时间内去尝试获取许可，如果能够获取到，返回true，获取不到返回false。

示例代码：



输出：



代码中许可数量为1， semaphore.tryAcquire(1,TimeUnit.SECONDS);：表示尝试在1秒内获取许可，获取成功立即返回true，超过1秒还是获取不到，返回false。线程t1获取许可成功，之后休眠了5秒，从输出中可以看出t2和t3都尝试了1秒，获取失败。

## 其他一些使用说明

1. **Semaphore默认创建的是非公平的信号量**，什么意思呢？这个涉及到公平与非公平。举个例子：5个车位，允许5个车辆进去，来了100辆车，只能进去5辆，其他95在外面排队等着。里面刚好出来了1辆，此时刚好又来了10辆车，这10辆车是直接插队到其他95辆前面去，还是到95辆后面去排队呢？让新来的去排队就表示公平，直接去插队争抢第一个，就表示不公平。对于停车场，排队肯定更好一些。不过对于信号量来说不公平的效率更高一些，所以默认是不公平的。
2. 建议阅读以下Semaphore的源码，对常用的方法有个了解，不需要都记住，用的时候也方便查询就好。
3. 方法中带有 throws InterruptedException声明的，表示这个方法会响应线程中断信号，什么意思？表示调用线程的 interrupt()方法后，会让这些方法触发 InterruptedException异常，即使这些方法处于阻塞状态，也会立即返回，并抛出 InterruptedException异常，线程中断信号也会被清除。