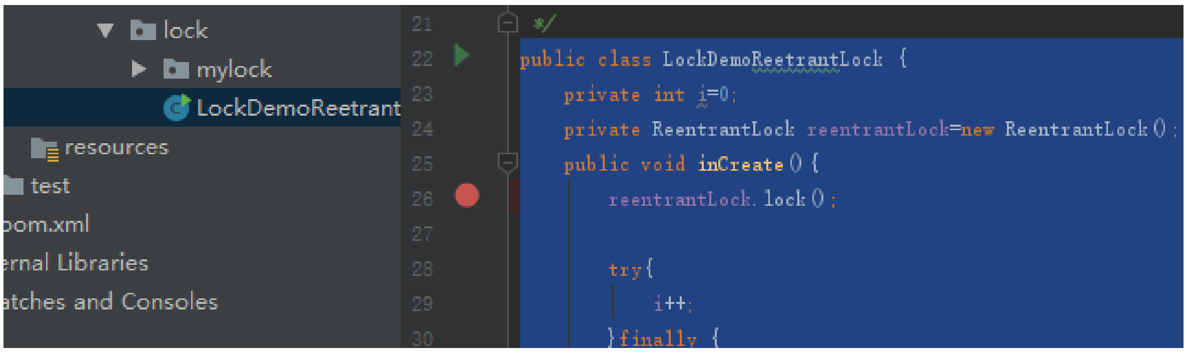
idea多线程Debug

多线程的开发，就需要多线程调试， 如何进行多线程调试呢？以下是一个小Demo来演示多线程的Debug调试：

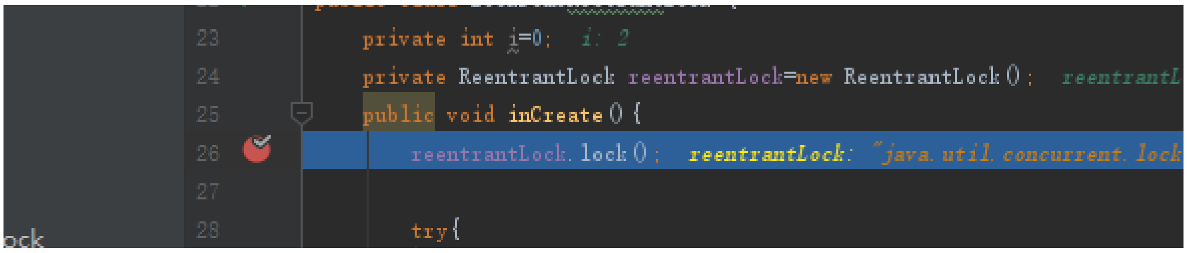
```
public class LockDemoReetrantLock{
     private int i=0;
     private ReentrantLock reentrantLock=new ReentrantLock();
     public void inCreate(){
     // 断点
         reentrantLock.lock();
         try{
             i++;
     }finally {
            //注意：一般的释放锁的操作都放到finally中，
            // 多线程可能会出错而停止运行，如果不释放锁其他线程都不会拿到该锁
            reentrantLock.unlock();
        }
    }
    public static void main(String[] args){
        ReentrantLock lock = new ReentrantLock();
        lock.lock();
        LockDemoReetrantLock lockDemoReetrantLock = new LockDemoReetrantLock();
        for(int i=0;i<3;i++){
            new Thread(()->{
                    lockDemoReetrantLock.inCreate();
                }).start();
        }
    }
}
```

普通的调试



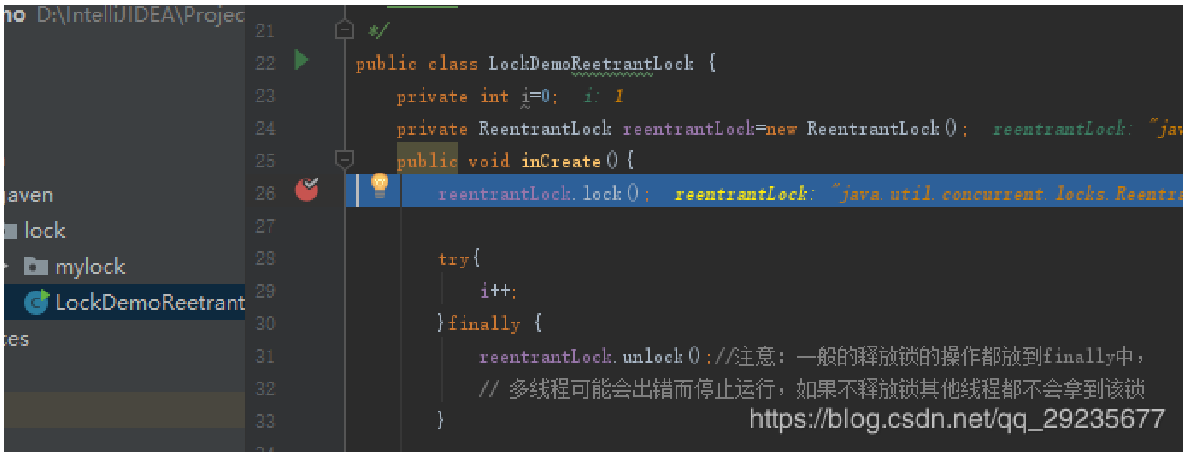
```
*/
public class LockDemoReetrantLock {
    private int i=0;
    private ReentrantLock reentrantLock=new ReentrantLock();
    public void inCreate() {
        reentrantLock.lock();

        try{
            i++;
        }finally {
```

开始刚一执行此时，i=2



```
    private int i=0;   i: 2
    private ReentrantLock reentrantLock=new ReentrantLock();   reentrantL
    public void inCreate() {
        reentrantLock.lock();   reentrantLock: "java.util.concurrent.lock

        try{
```

接着下一步下一步，程序直接跳出 看不到ReentrantLock的排队操作，再次运行，在进行一次调试此时i=1



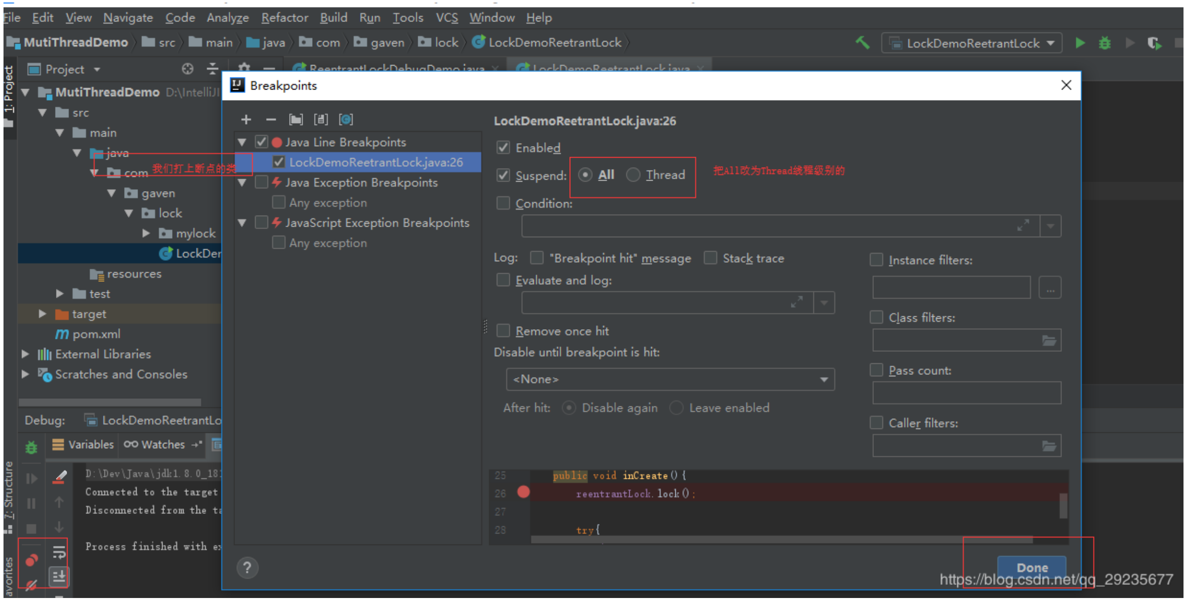
```
*/
public class LockDemoReetrantLock {
    private int i=0;   i: 1
    private ReentrantLock reentrantLock=new ReentrantLock();  reentrantLock: "java
    public void inCreate(){
        reentrantLock.lock();  reentrantLock: "java.util.concurrent.locks.Reentra

        try{
            i++;
        }finally {
            reentrantLock.unlock();//注意：一般的释放锁的操作都放到finally中，
            // 多线程可能会出错而停止运行，如果不释放锁其他线程都不会拿到该锁
        }
```

同样看不到排队操作，不是我们想要的结果！！ 达不到想要的效果！！！

多线程调试

关键关键关键来了！我们需要将断点阻塞设置成针对线程的, 而非全局的, 有两种方式可以设置, 推荐第②种，更加方便!

①.运行Debug, 其它两个线程就已经启动了，其中有一个线程能够停止到这个断点，选择Debug栏的所有断点，选择到我们需要设置的断点， 然后将将suspend从All修改成Thread， 然后点Done，此时就Ok了。



Breakpoints

- Java Line Breakpoints
  - LockDemoReetrantLock.java:26
- Java Exception Breakpoints
  - Any exception
- JavaScript Exception Breakpoints
  - Any exception

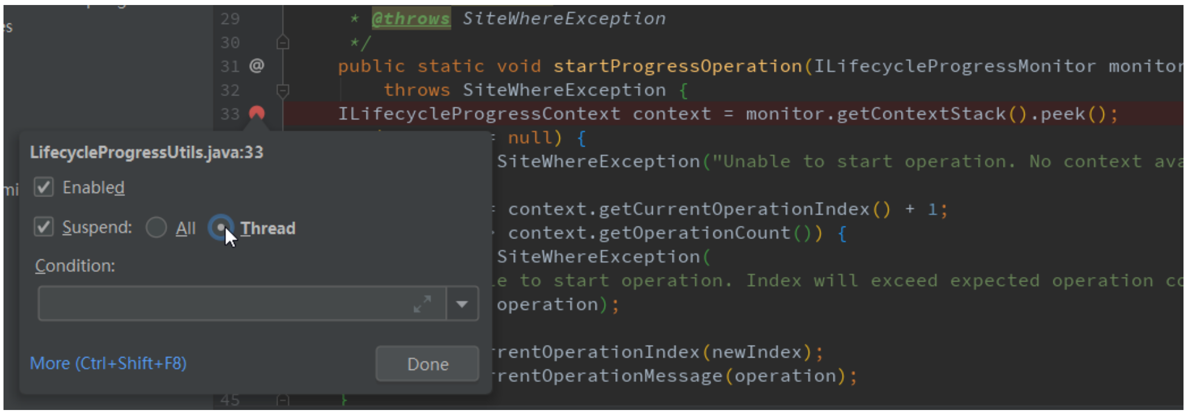
LockDemoReetrantLock.java:26

- [✓] Enabled
- [✓] Suspend: [✓ All] Thread
- [ ] Condition:
- Log: [ ] "Breakpoint hit" message [ ] Stack trace
- [ ] Evaluate and log:
- [ ] Remove once hit
- Disable until breakpoint is hit: <None>
- After hit: Disable again / Leave enabled
- [ ] Instance filters:
- [ ] Class filters:
- [ ] Pass count:
- [ ] Caller filters:

```
public void inCreate(){
    reentrantLock.lock();

    try{
```

File Edit View Navigate Code Analyze Refactor Build Run Tools VCS Window Help
MutiThreadDemo src main java com gaven lock LockDemoReetrantLock
我们打上断点的类
把All改为Thread线程级别的
Done
https://blog.csdn.net/qq_29235677

②.我们在代码侧边栏，设置断点，然后打上断点，右键红色小断点，然后将suspend从All修改成Thread， 然后点Done，可以设置成全局都是Thread， 否则相关的代码断点可能会跳过，因此就需要重新去设置。点击Done后就完成了设置。*推荐使用这种方式，简单明了，一目了然。*



```
     * @throws SiteWhereException
     */
    public static void startProgressOperation(ILifecycleProgressMonitor monitor
        throws SiteWhereException {
    ILifecycleProgressContext context = monitor.getContextStack().peek();
                          null) {
                       SiteWhereException("Unable to start operation. No context ava

                        context.getCurrentOperationIndex() + 1;
                        context.getOperationCount()) {
                       SiteWhereException(
                       e to start operation. Index will exceed expected operation co
                       operation);

                       rentOperationIndex(newIndex);
                       rentOperationMessage(operation);
```

LifecycleProgressUtils.java:33

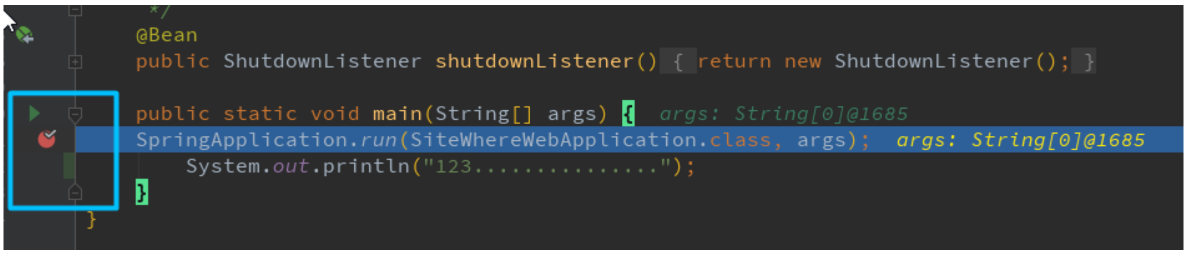
[✓] Enabled

[✓] Suspend: All [✓ Thread]

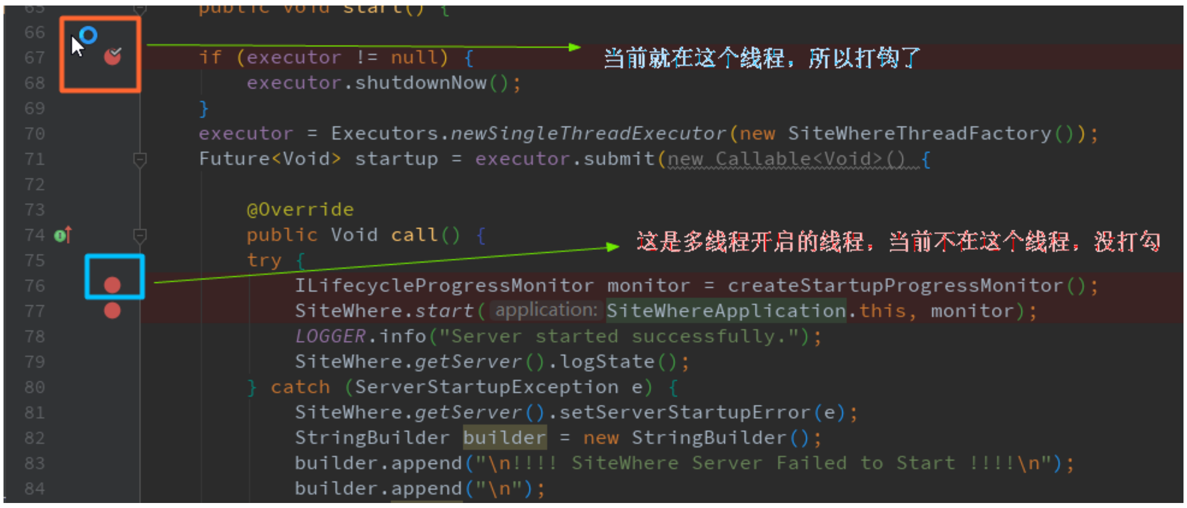
Condition:

More (Ctrl+Shift+F8)　Done

注意：我们开启程序后，进入到相关的线程，就会在断点上打到，如果现在运行的不是这个线程，那么久不会在这个断点上面打钩， 只有运行到上面，才会打钩。



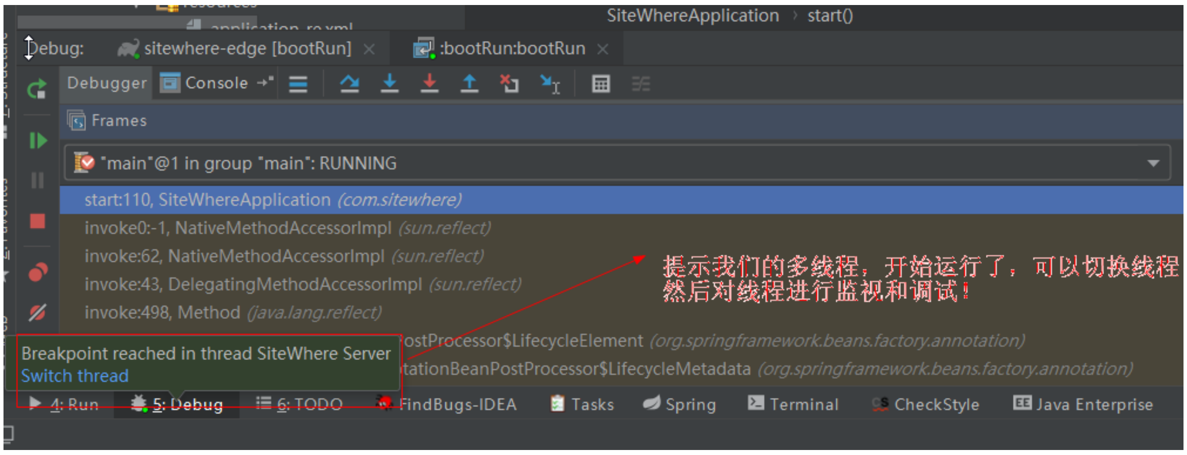
```
    @Bean
    public ShutdownListener shutdownListener() { return new ShutdownListener(); }

    public static void main(String[] args) {  args: String[0]@1685
    SpringApplication.run(SiteWhereWebApplication.class, args);  args: String[0]@1685
        System.out.println("123..............");
    }
}
```



```
        public void start() {
            if (executor != null) {
                executor.shutdownNow();
            }
            executor = Executors.newSingleThreadExecutor(new SiteWhereThreadFactory());
            Future<Void> startup = executor.submit(new Callable<Void>() {

                @Override
                public Void call() {
                try {
                    ILifecycleProgressMonitor monitor = createStartupProgressMonitor();
                    SiteWhere.start( application: SiteWhereApplication.this, monitor);
                    LOGGER.info("Server started successfully.");
                    SiteWhere.getServer().logState();
                } catch (ServerStartupException e) {
                    SiteWhere.getServer().setServerStartupError(e);
                    StringBuilder builder = new StringBuilder();
                    builder.append("\n!!!! SiteWhere Server Failed to Start !!!!\n");
                    builder.append("\n");
```

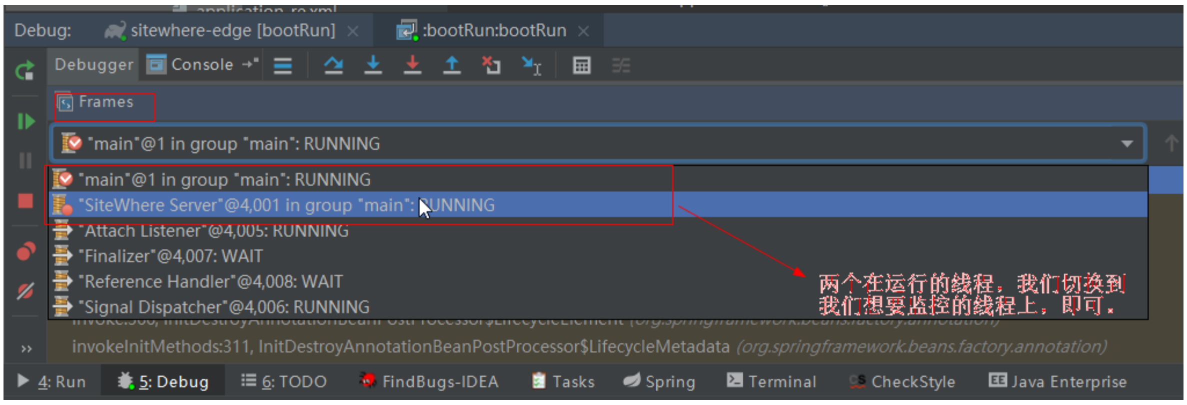
当前就在这个线程，所以打钩了
这是多线程开启的线程，当前不在这个线程，没打勾

开始单步调试执行，会有如下提示查看Frames：



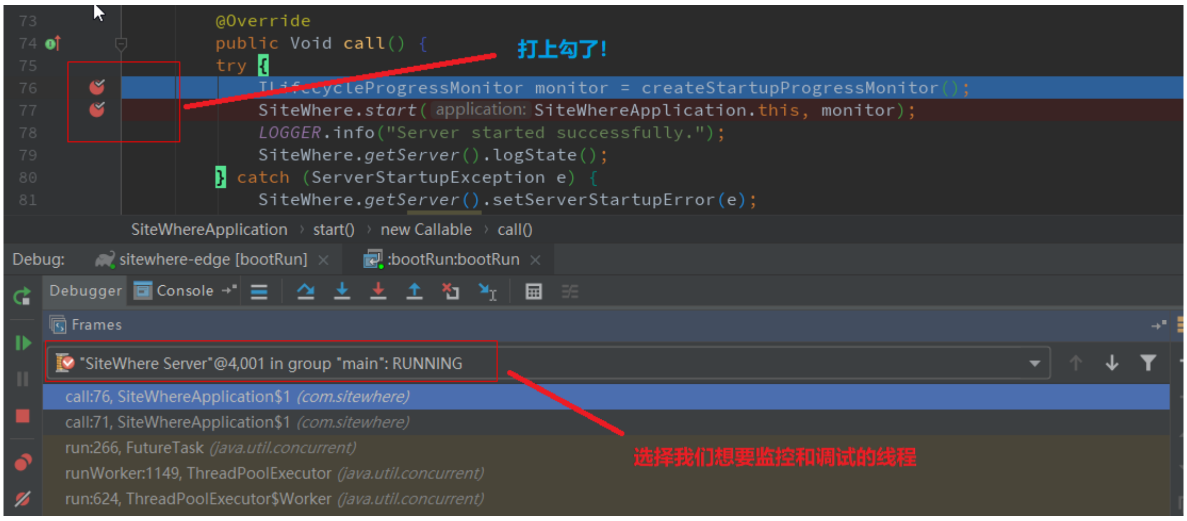
```
SiteWhereApplication › start()
Debug: sitewhere-edge [bootRun]   :bootRun:bootRun
Debugger  Console
Frames
"main"@1 in group "main": RUNNING
start:110, SiteWhereApplication (com.sitewhere)
invoke0:-1, NativeMethodAccessorImpl (sun.reflect)
invoke:62, NativeMethodAccessorImpl (sun.reflect)
invoke:43, DelegatingMethodAccessorImpl (sun.reflect)
invoke:498, Method (java.lang.reflect)
ostProcessor$LifecycleElement (org.springframework.beans.factory.annotation)
tationBeanPostProcessor$LifecycleMetadata (org.springframework.beans.factory.annotation)
Breakpoint reached in thread SiteWhere Server
Switch thread
```

查看Frames, 选择我们想要调试的线程：



```
"main"@1 in group "main": RUNNING
"SiteWhere Server"@4,001 in group "main": RUNNING
"Attach Listener"@4,005: RUNNING
"Finalizer"@4,007: WAIT
"Reference Handler"@4,008: WAIT
"Signal Dispatcher"@4,006: RUNNING
invokeInitMethods:311, InitDestroyAnnotationBeanPostProcessor$LifecycleMetadata (org.springframework.beans.factory.annotation)
```

此时效果如下，我们就可以开始多线程的调试了：

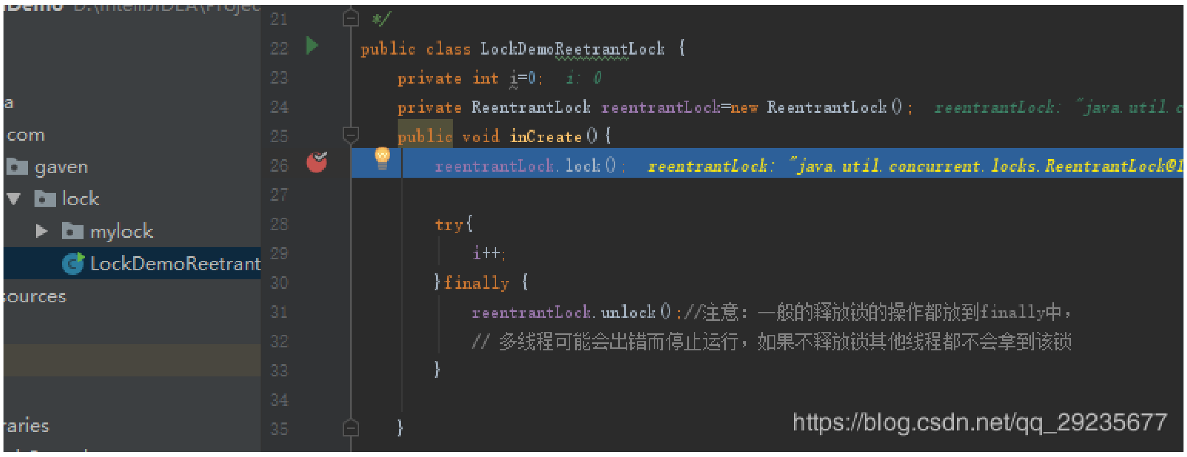


```java
	@Override
	public Void call() {
	    try {
		ILifecycleProgressMonitor monitor = createStartupProgressMonitor();
		SiteWhere.start( application: SiteWhereApplication.this, monitor);
		LOGGER.info("Server started successfully.");
		SiteWhere.getServer().logState();
	    } catch (ServerStartupException e) {
		SiteWhere.getServer().setServerStartupError(e);
```

```
"SiteWhere Server"@4,001 in group "main": RUNNING
call:76, SiteWhereApplication$1 (com.sitewhere)
call:71, SiteWhereApplication$1 (com.sitewhere)
run:266, FutureTask (java.util.concurrent)
runWorker:1149, ThreadPoolExecutor (java.util.concurrent)
run:624, ThreadPoolExecutor$Worker (java.util.concurrent)
```

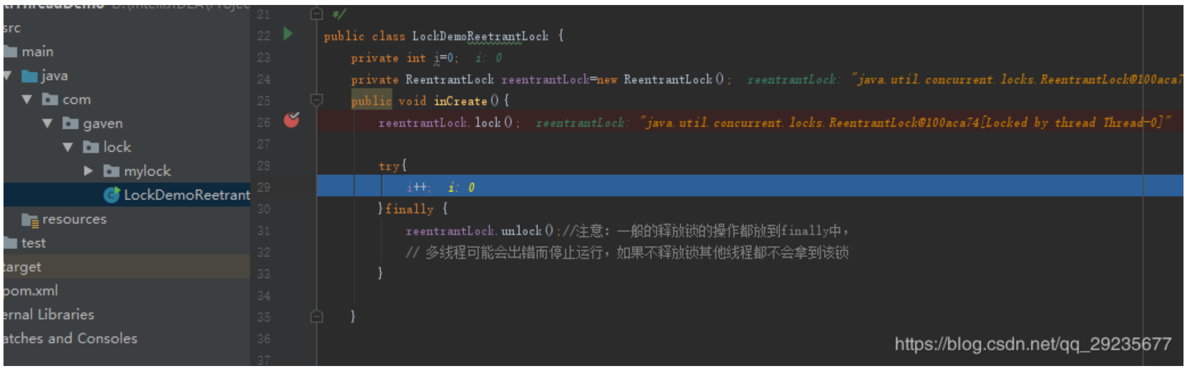
打上勾了!
SiteWhereApplication › start() › new Callable › call()
Debug:
sitewhere-edge [bootRun]
:bootRun:bootRun
Debugger
Console
Frames
选择我们想要监控和调试的线程

多线程调试案例

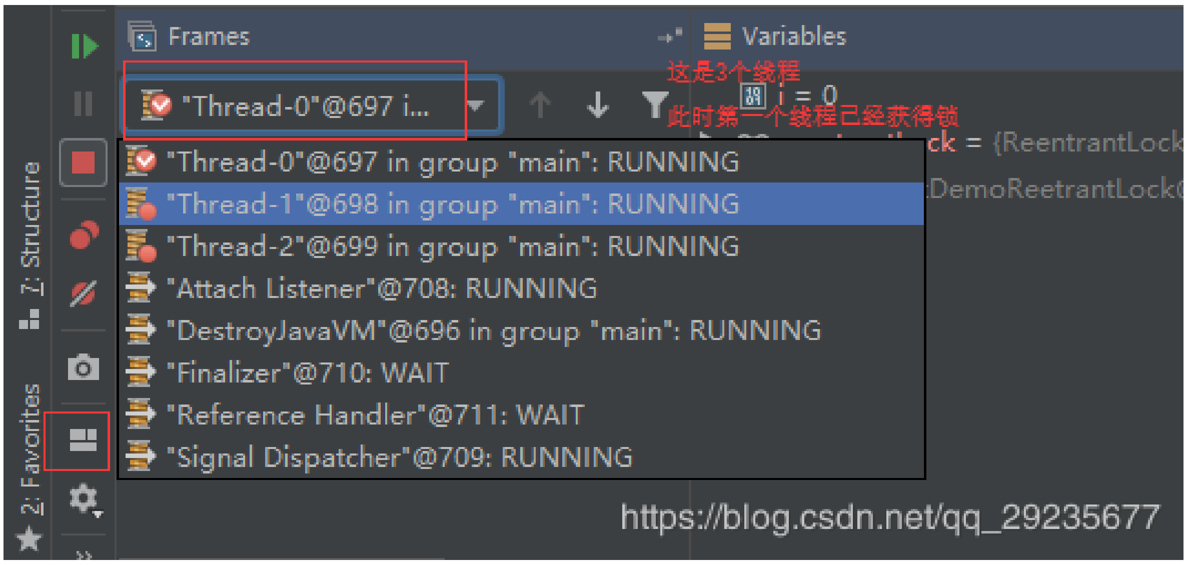


```java
*/
public class LockDemoReetrantLock {
    private int i=0;  i: 0
    private ReentrantLock reentrantLock=new ReentrantLock();  reentrantLock: "java.util.c
    public void inCreate(){
        reentrantLock.lock();  reentrantLock: "java.util.concurrent.locks.ReentrantLock@1

        try{
            i++;
        }finally {
            reentrantLock.unlock();//注意：一般的释放锁的操作都放到finally中，
            // 多线程可能会出错而停止运行，如果不释放锁其他线程都不会拿到该锁
        }

    }
```

开始调试， 单步执行



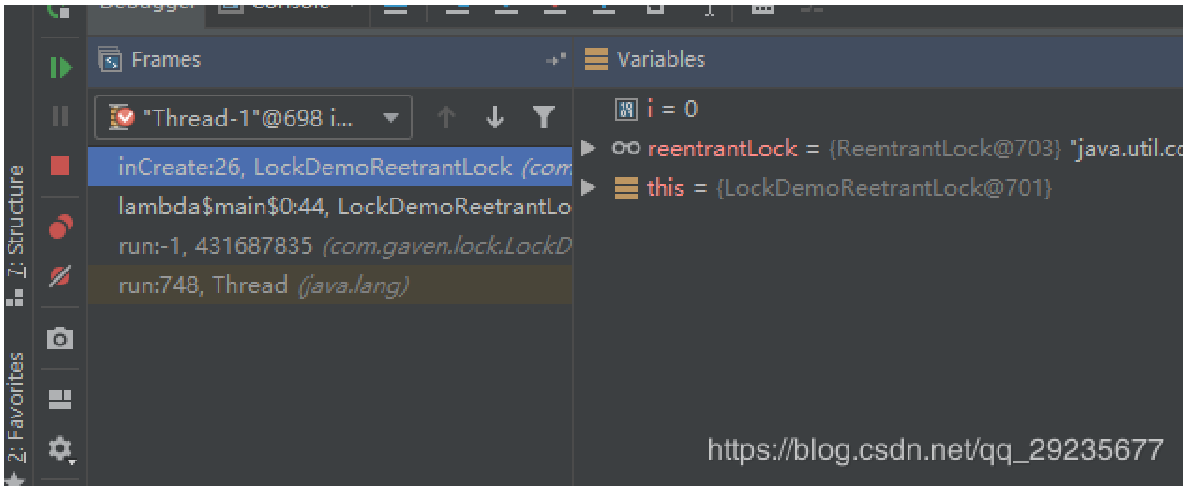
```java
*/
public class LockDemoReetrantLock {
    private int i=0;  i: 0
    private ReentrantLock reentrantLock=new ReentrantLock();  reentrantLock: "java.util.concurrent.locks.ReentrantLock@100aca7
    public void inCreate(){
        reentrantLock.lock();  reentrantLock: "java.util.concurrent.locks.ReentrantLock@100aca74[Locked by thread Thread-0]"

        try{
            i++;  i: 0
        }finally {
            reentrantLock.unlock();//注意：一般的释放锁的操作都放到finally中，
            // 多线程可能会出错而停止运行，如果不释放锁其他线程都不会拿到该锁
        }

    }
```



```
"Thread-0"@697 i...
i = 0
ck = {ReentrantLock
DemoReetrantLock
"Thread-0"@697 in group "main": RUNNING
"Thread-1"@698 in group "main": RUNNING
"Thread-2"@699 in group "main": RUNNING
"Attach Listener"@708: RUNNING
"DestroyJavaVM"@696 in group "main": RUNNING
"Finalizer"@710: WAIT
"Reference Handler"@711: WAIT
"Signal Dispatcher"@709: RUNNING
```

https://blog.csdn.net/qq_29235677

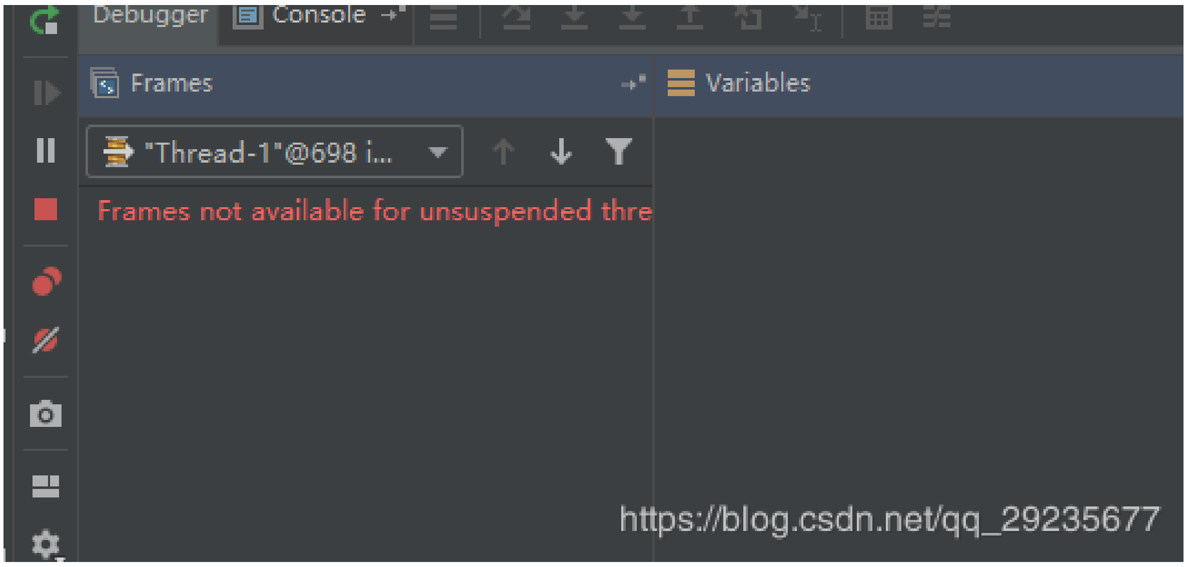
接下来我们看第二个线程是否获得锁，点入该线程(012线程顺序是随机的)



```
"Thread-1"@698 i...
inCreate:26, LockDemoReetrantLock (com
lambda$main$0:44, LockDemoReetrantLo
run:-1, 431687835 (com.gaven.lock.LockD
run:748, Thread (java.lang)

i = 0
reentrantLock = {ReentrantLock@703} "java.util.co
this = {LockDemoReetrantLock@701}
```

https://blog.csdn.net/qq_29235677

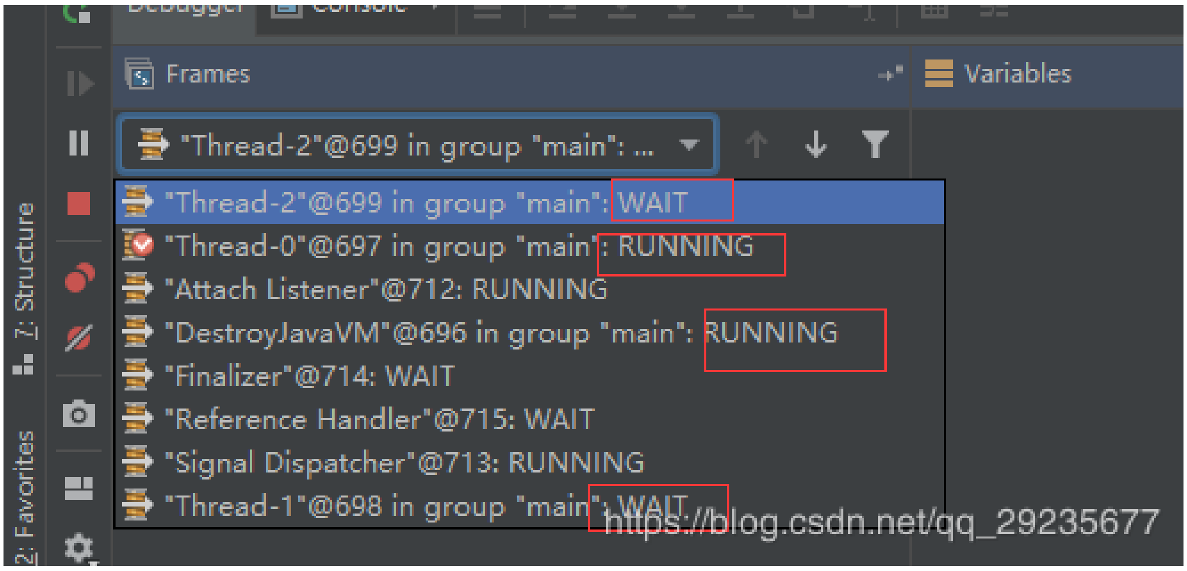
F8显示未挂起的线程不可用 该线程没能获取到该锁（同理Thread2也不能获取该锁）



```
"Thread-1"@698 i...
Frames not available for unsuspended thre
```

https://blog.csdn.net/qq_29235677

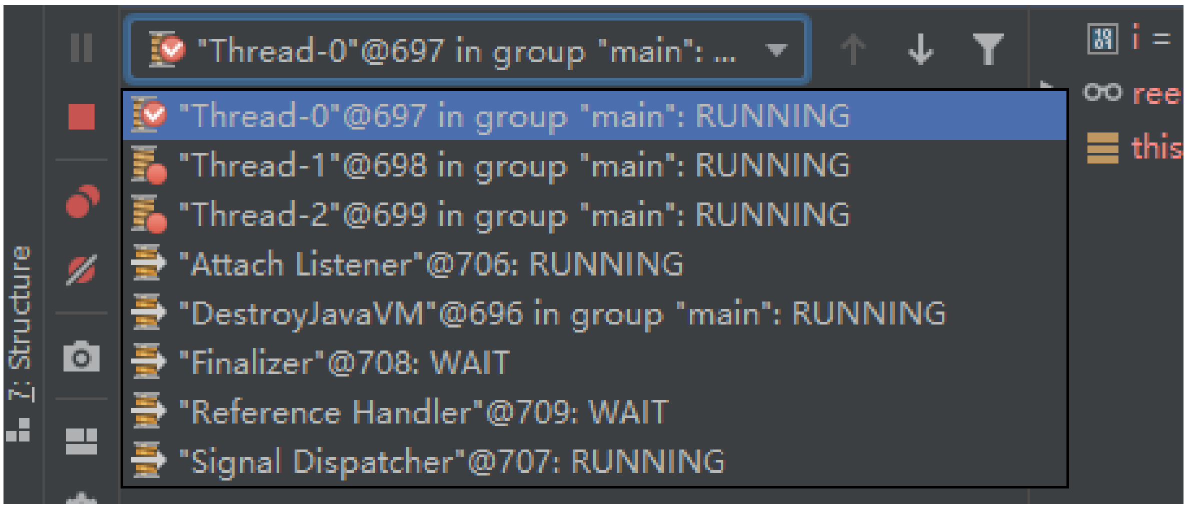
线程1和2 wait 线程0和主线程running，线程1和2都在等待资源



```
"Thread-2"@699 in group "main": WAIT
"Thread-0"@697 in group "main": RUNNING
"Attach Listener"@712: RUNNING
"DestroyJavaVM"@696 in group "main": RUNNING
"Finalizer"@714: WAIT
"Reference Handler"@715: WAIT
"Signal Dispatcher"@713: RUNNING
"Thread-1"@698 in group "main": WAIT
```

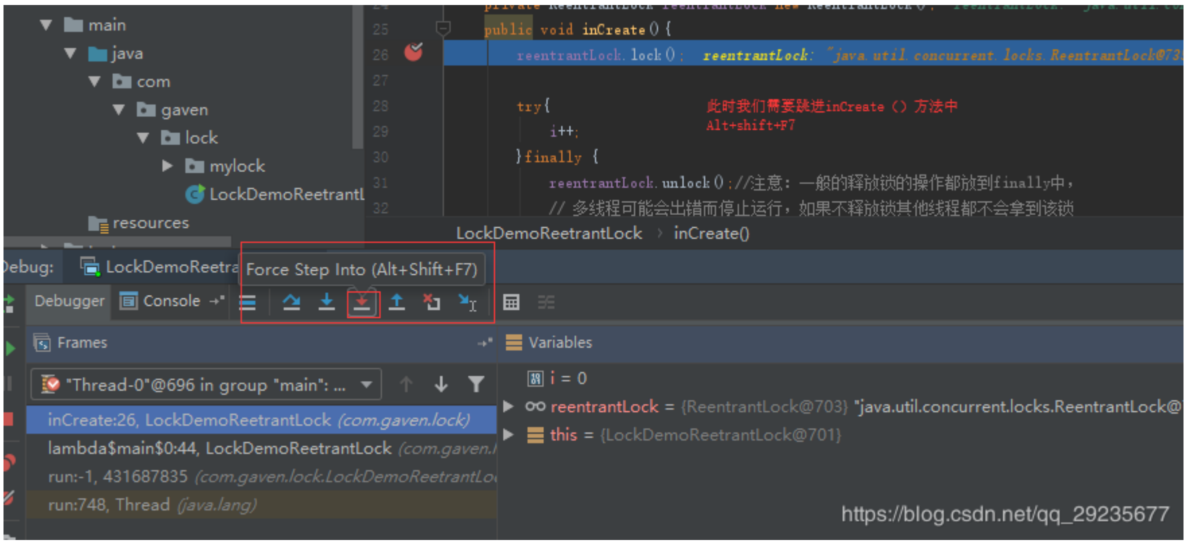
https://blog.csdn.net/qq_29235677

接下来看ReetrantLock 的执行过程，重新启动



```
"Thread-0"@697 in group "main": RUNNING
"Thread-1"@698 in group "main": RUNNING
"Thread-2"@699 in group "main": RUNNING
"Attach Listener"@706: RUNNING
"DestroyJavaVM"@696 in group "main": RUNNING
"Finalizer"@708: WAIT
"Reference Handler"@709: WAIT
"Signal Dispatcher"@707: RUNNING
```

此时3个线程都停留在这



```
public void inCreate(){
    reentrantLock.lock();   reentrantLock: "java.util.concurrent.locks.ReentrantLock@73

    try{                    此时我们需要跳进inCreate（）方法中
        i++;                Alt+shift+F7
    }finally {
        reentrantLock.unlock();//注意：一般的释放锁的操作都放到finally中，
        // 多线程可能会出错而停止运行，如果不释放锁其他线程都不会拿到该锁
```

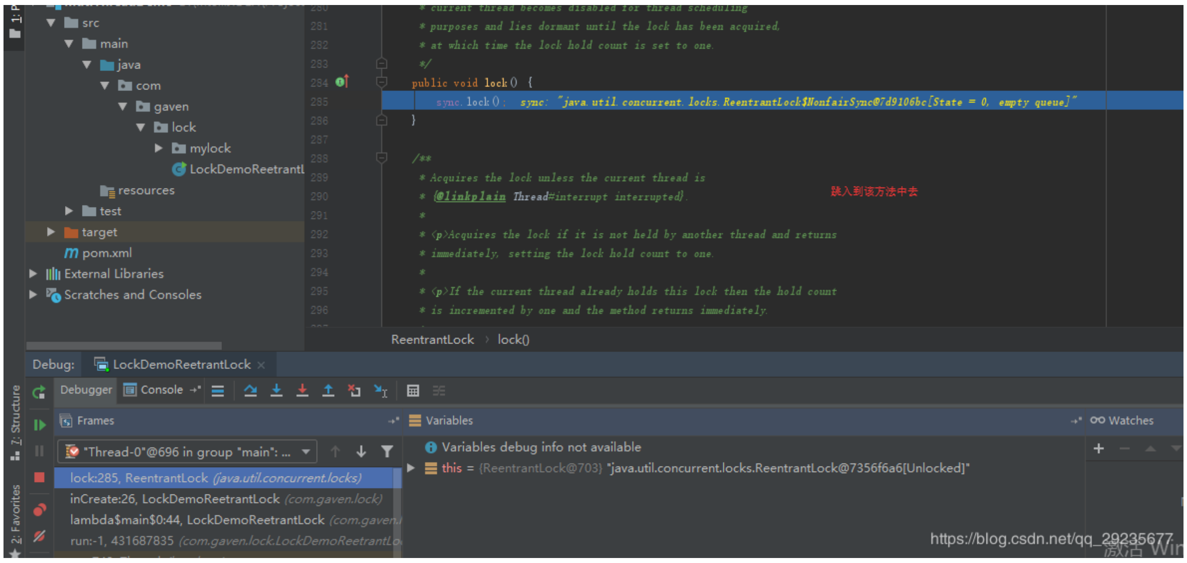
Force Step Into (Alt+Shift+F7)

```
"Thread-0"@696 in group "main": ...
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.l
run:-1, 431687835 (com.gaven.lock.LockDemoReetrantLo
run:748, Thread (java.lang)
```

```
i = 0
reentrantLock = {ReentrantLock@703} "java.util.concurrent.locks.ReentrantLock@
this = {LockDemoReetrantLock@701}
```

https://blog.csdn.net/qq_29235677

此时跳入inCreat方法



```
     * current thread becomes disabled for thread scheduling
     * purposes and lies dormant until the lock has been acquired,
     * at which time the lock hold count is set to one.
     */
    public void lock() {
        sync.lock();   sync: "java.util.concurrent.locks.ReentrantLock$NonfairSync@7d9106bc[State = 0, empty queue]"
    }

    /**
     * Acquires the lock unless the current thread is
     * {@linkplain Thread#interrupt interrupted}.
     *
     * <p>Acquires the lock if it is not held by another thread and returns
     * immediately, setting the lock hold count to one.
     *
     * <p>If the current thread already holds this lock then the hold count
     * is incremented by one and the method returns immediately.
```

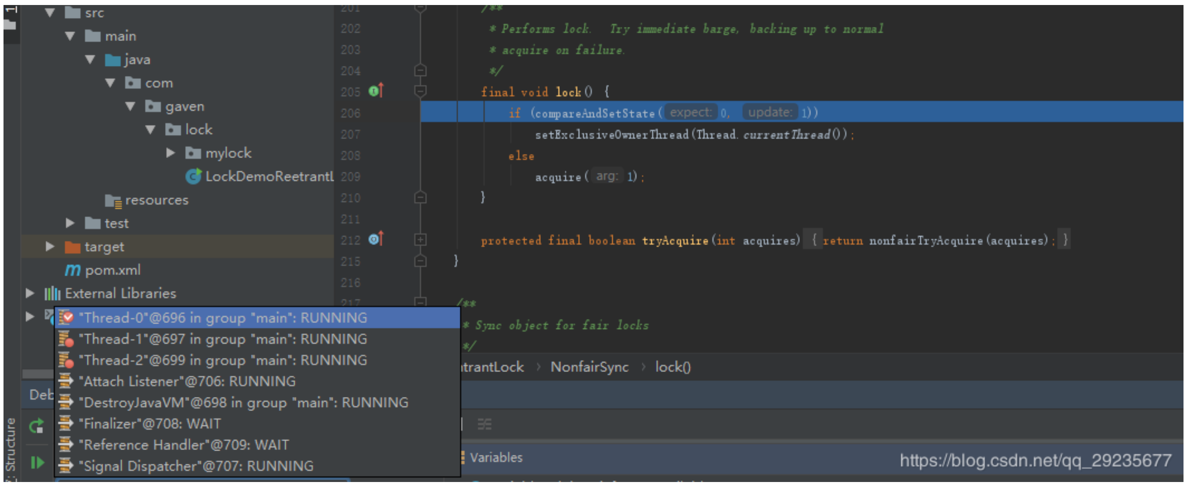
跳入到该方法中去

ReentrantLock › lock()

```
"Thread-0"@696 in group "main": ...
lock:285, ReentrantLock (java.util.concurrent.locks)
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.
run:-1, 431687835 (com.gaven.lock.LockDemoReetrantLo
```

```
Variables debug info not available
this = {ReentrantLock@703} "java.util.concurrent.locks.ReentrantLock@7356f6a6[Unlocked]"
```

再跳进到lock方法中去 进入到非公平锁的实现

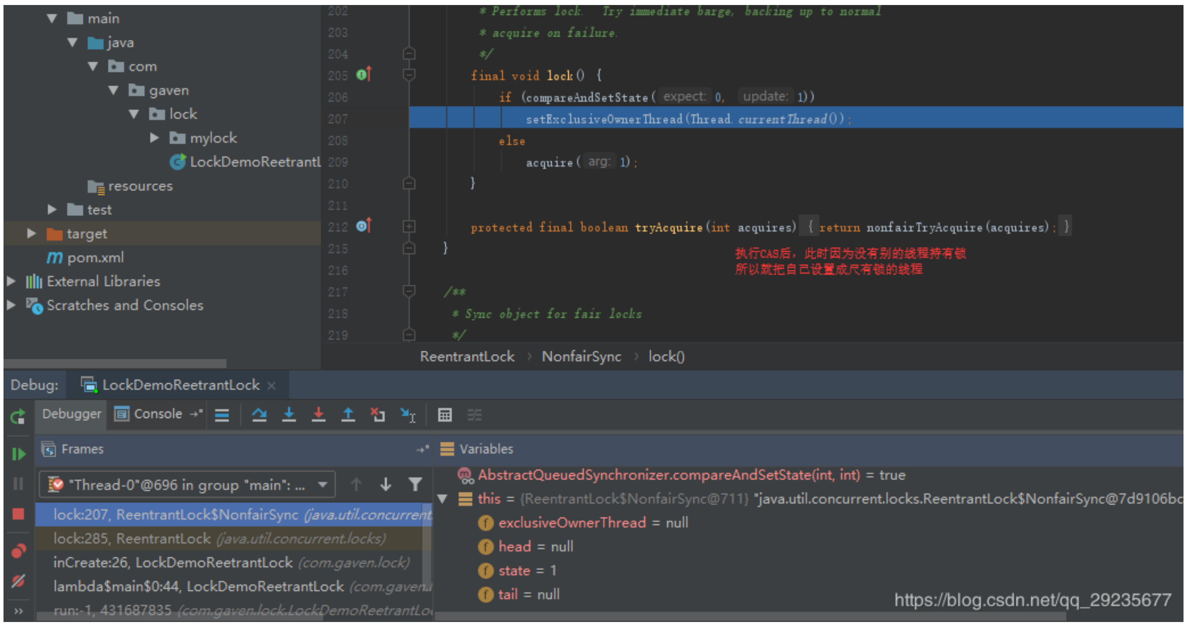


```
    /**
     * Performs lock.  Try immediate barge, backing up to normal
     * acquire on failure.
     */
    final void lock() {
        if (compareAndSetState( expect: 0,  update: 1))
            setExclusiveOwnerThread(Thread.currentThread());
        else
            acquire( arg: 1);
    }

    protected final boolean tryAcquire(int acquires) { return nonfairTryAcquire(acquires); }
}

/**
 * Sync object for fair locks
 */
```

```
"Thread-0"@696 in group "main": RUNNING
"Thread-1"@697 in group "main": RUNNING
"Thread-2"@699 in group "main": RUNNING
"Attach Listener"@706: RUNNING
"DestroyJavaVM"@698 in group "main": RUNNING
"Finalizer"@708: WAIT
"Reference Handler"@709: WAIT
"Signal Dispatcher"@707: RUNNING
```

ReentrantLock › NonfairSync › lock()

F8首先执行CAS



```
     * Performs lock.  Try immediate barge, backing up to normal
     * acquire on failure.
     */
    final void lock() {
        if (compareAndSetState( expect: 0,  update: 1))
            setExclusiveOwnerThread(Thread.currentThread());
        else
            acquire( arg: 1);
    }

    protected final boolean tryAcquire(int acquires) { return nonfairTryAcquire(acquires); }
}

/**
 * Sync object for fair locks
 */
```

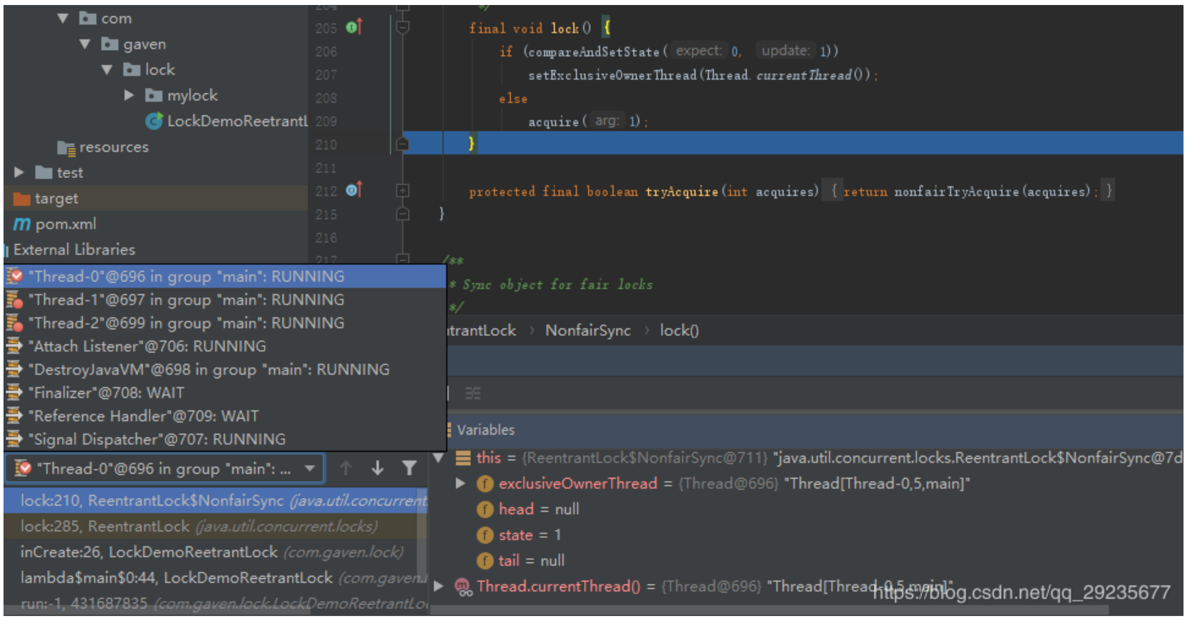
执行CAS后，此时因为没有别的线程持有锁
所以就把自己设置成尺有锁的线程

ReentrantLock › NonfairSync › lock()

```
"Thread-0"@696 in group "main": ...
lock:207, ReentrantLock$NonfairSync (java.util.concurrent
lock:285, ReentrantLock (java.util.concurrent.locks)
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.
run:-1, 431687835 (com.gaven.lock.LockDemoReetrantLo
```

```
AbstractQueuedSynchronizer.compareAndSetState(int, int) = true
this = {ReentrantLock$NonfairSync@711} "java.util.concurrent.locks.ReentrantLock$NonfairSync@7d9106bc
    exclusiveOwnerThread = null
    head = null
    state = 1
    tail = null
```

其他线程就不会执行



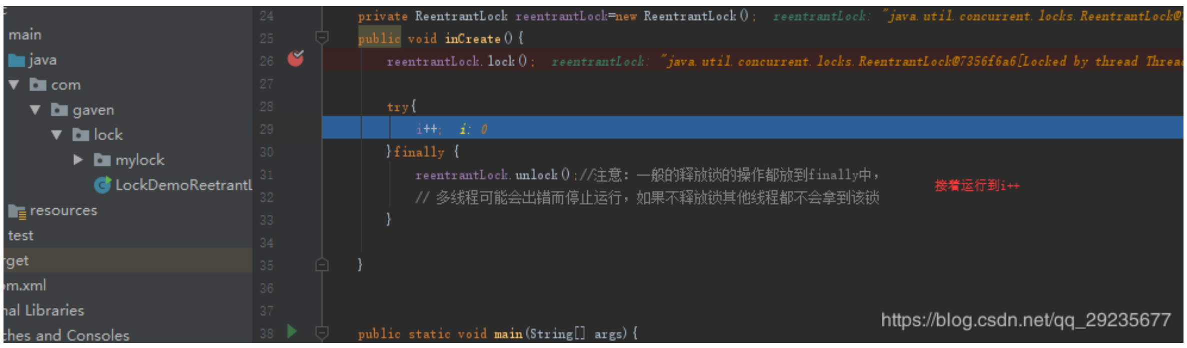
```
final void lock() {
    if (compareAndSetState( expect: 0,  update: 1))
        setExclusiveOwnerThread(Thread.currentThread());
    else
        acquire( arg: 1);
}

protected final boolean tryAcquire(int acquires) { return nonfairTryAcquire(acquires); }
}

/**
 * Sync object for fair locks
 */
```

```
"Thread-0"@696 in group "main": RUNNING
"Thread-1"@697 in group "main": RUNNING
"Thread-2"@699 in group "main": RUNNING
"Attach Listener"@706: RUNNING
"DestroyJavaVM"@698 in group "main": RUNNING
"Finalizer"@708: WAIT
"Reference Handler"@709: WAIT
"Signal Dispatcher"@707: RUNNING
```

```
lock:210, ReentrantLock$NonfairSync (java.util.concurrent
lock:285, ReentrantLock (java.util.concurrent.locks)
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.l
run:-1, 431687835 (com.gaven.lock.LockDemoReetrantLo
```

```
this = {ReentrantLock$NonfairSync@711} "java.util.concurrent.locks.ReentrantLock$NonfairSync@7d
  exclusiveOwnerThread = {Thread@696} "Thread[Thread-0,5,main]"
  head = null
  state = 1
  tail = null
Thread.currentThread() = {Thread@696} "Thread[Thread-0,5,main]"
```

https://blog.csdn.net/qq_29235677

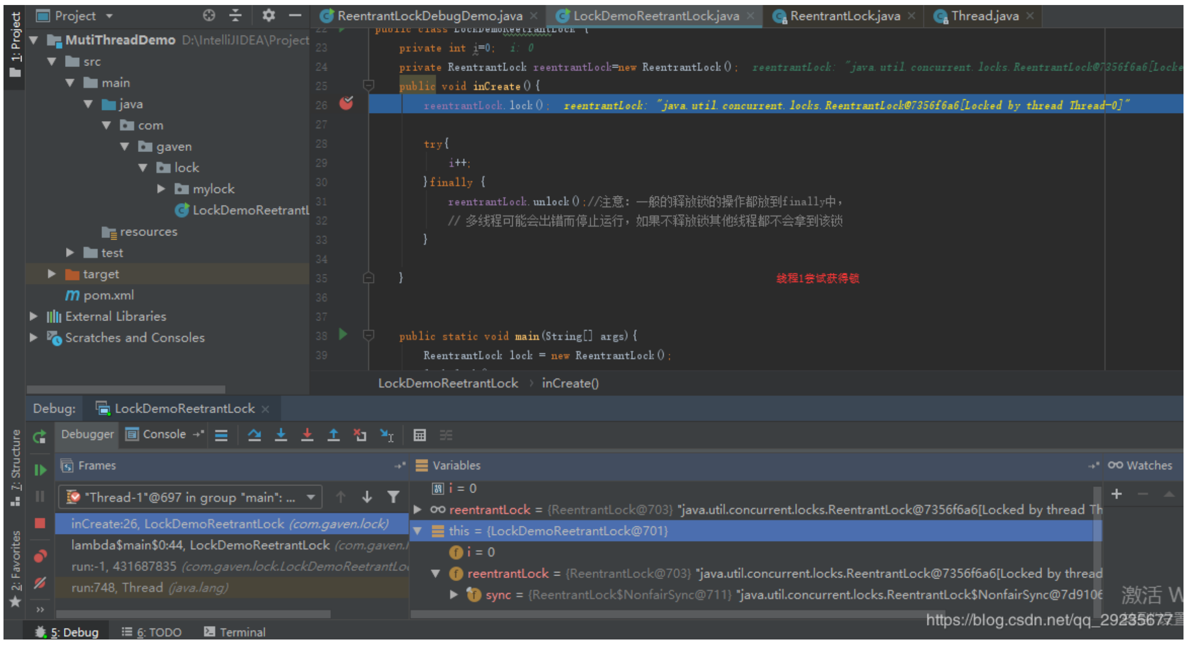


```
private ReentrantLock reentrantLock=new ReentrantLock();  reentrantLock: "java.util.concurrent.locks.ReentrantLock@
public void inCreate(){
    reentrantLock.lock();  reentrantLock: "java.util.concurrent.locks.ReentrantLock@7356f6a6[Locked by thread Threa

    try{
        i++;  i: 0
    }finally {
        reentrantLock.unlock();//注意：一般的释放锁的操作都放到finally中，
        // 多线程可能会出错而停止运行，如果不释放锁其他线程都不会拿到该锁
    }

}


public static void main(String[] args){
```

接着运行到i++

https://blog.csdn.net/qq_29235677

此时由于是线程0先执行的，我们再开一下线程1（012执行顺序是随机的这里假定0先执行），接下来看线程1



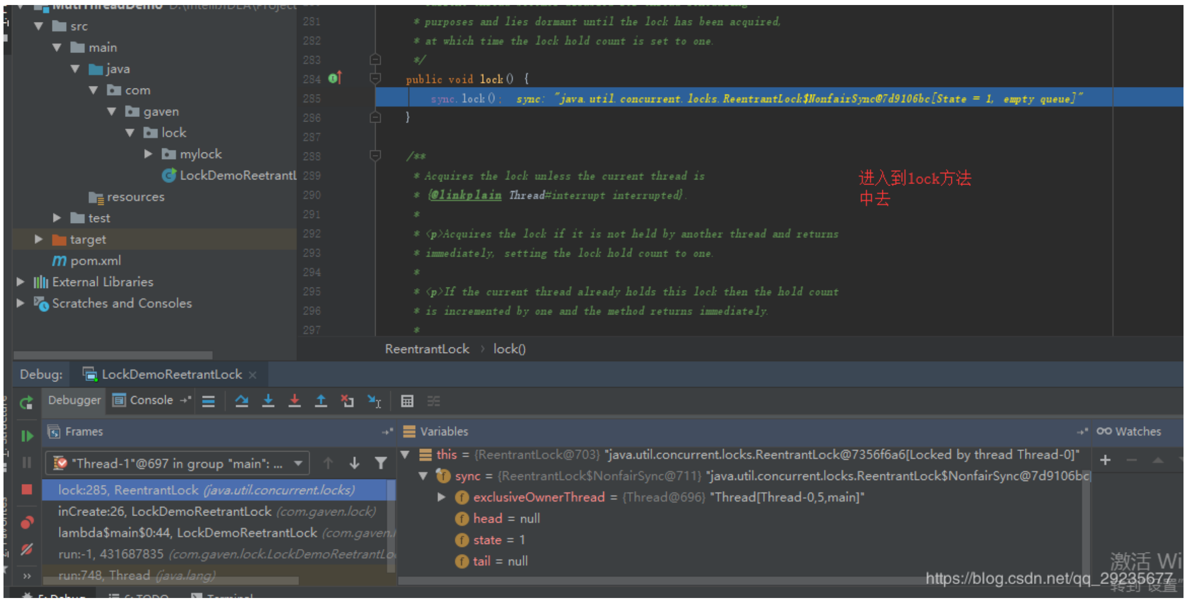
```
private int i=0;  i: 0
private ReentrantLock reentrantLock=new ReentrantLock();  reentrantLock: "java.util.concurrent.locks.ReentrantLock@7356f6a6[Locke
public void inCreate(){
    reentrantLock.lock();  reentrantLock: "java.util.concurrent.locks.ReentrantLock@7356f6a6[Locked by thread Thread-0]"

    try{
        i++;
    }finally {
        reentrantLock.unlock();//注意：一般的释放锁的操作都放到finally中，
        // 多线程可能会出错而停止运行，如果不释放锁其他线程都不会拿到该锁
    }

}


public static void main(String[] args){
    ReentrantLock lock = new ReentrantLock();
```

线程1尝试获得锁

```
"Thread-1"@697 in group "main": ...
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.l
run:-1, 431687835 (com.gaven.lock.LockDemoReetrantLo
run:748, Thread (java.lang)
```

```
i = 0
reentrantLock = {ReentrantLock@703} "java.util.concurrent.locks.ReentrantLock@7356f6a6[Locked by thread Th
this = {LockDemoReetrantLock@701}
  i = 0
  reentrantLock = {ReentrantLock@703} "java.util.concurrent.locks.ReentrantLock@7356f6a6[Locked by thread
    sync = {ReentrantLock$NonfairSync@711} "java.util.concurrent.locks.ReentrantLock$NonfairSync@7d910
```

https://blog.csdn.net/qq_29235677

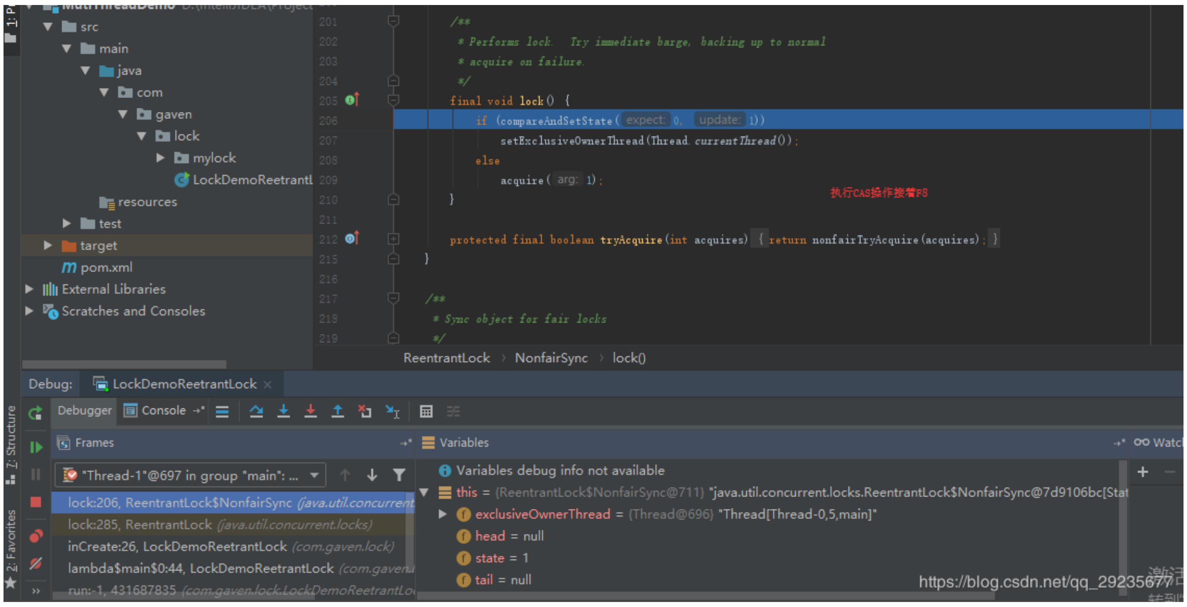


```
     * purposes and lies dormant until the lock has been acquired,
     * at which time the lock hold count is set to one.
     */
    public void lock() {
        sync.lock();   sync: "java.util.concurrent.locks.ReentrantLock$NonfairSync@7d9106bc[State = 1, empty queue]"
    }

    /**
     * Acquires the lock unless the current thread is
     * {@linkplain Thread#interrupt interrupted}.
     *
     * <p>Acquires the lock if it is not held by another thread and returns
     * immediately, setting the lock hold count to one.
     *
     * <p>If the current thread already holds this lock then the hold count
     * is incremented by one and the method returns immediately.
     *
```

```
Frames
"Thread-1"@697 in group "main": ...
lock:285, ReentrantLock (java.util.concurrent.locks)
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.
run:-1, 431687835 (com.gaven.lock.LockDemoReetrantLo
run:748, Thread (java.lang)
```

```
Variables
this = {ReentrantLock@703} "java.util.concurrent.locks.ReentrantLock@7356f6a6[Locked by thread Thread-0]"
  sync = {ReentrantLock$NonfairSync@711} "java.util.concurrent.locks.ReentrantLock$NonfairSync@7d9106bc
    exclusiveOwnerThread = {Thread@696} "Thread[Thread-0,5,main]"
    head = null
    state = 1
    tail = null
```

https://blog.csdn.net/qq_29235677

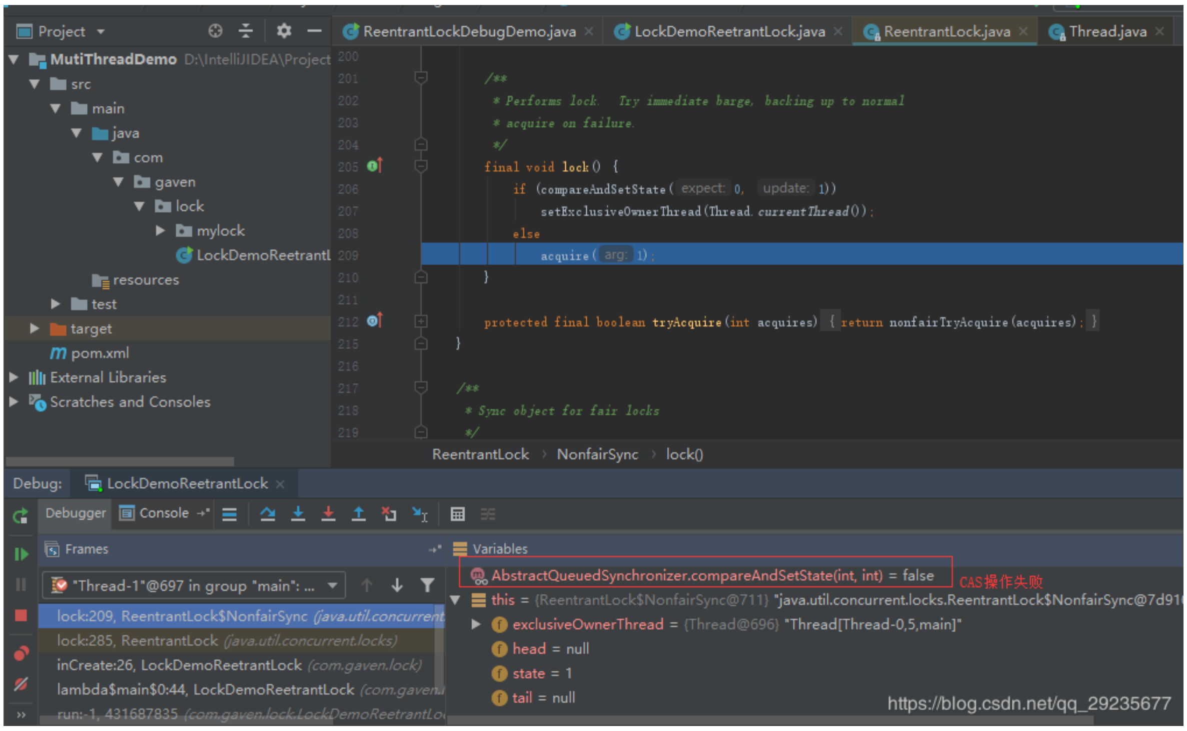


```
    /**
     * Performs lock.  Try immediate barge, backing up to normal
     * acquire on failure.
     */
    final void lock() {
        if (compareAndSetState( expect: 0,  update: 1))
            setExclusiveOwnerThread(Thread.currentThread());
        else
            acquire( arg: 1);
    }

    protected final boolean tryAcquire(int acquires) { return nonfairTryAcquire(acquires); }
}

/**
 * Sync object for fair locks
 */
```

```
Frames
"Thread-1"@697 in group "main": ...
lock:206, ReentrantLock$NonfairSync (java.util.concurrent
lock:285, ReentrantLock (java.util.concurrent.locks)
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.
run:-1, 431687835 (com.gaven.lock.LockDemoReetrantLo
```

```
Variables
Variables debug info not available
this = {ReentrantLock$NonfairSync@711} "java.util.concurrent.locks.ReentrantLock$NonfairSync@7d9106bc[Stat
  exclusiveOwnerThread = {Thread@696} "Thread[Thread-0,5,main]"
  head = null
  state = 1
  tail = null
```

https://blog.csdn.net/qq_29235677



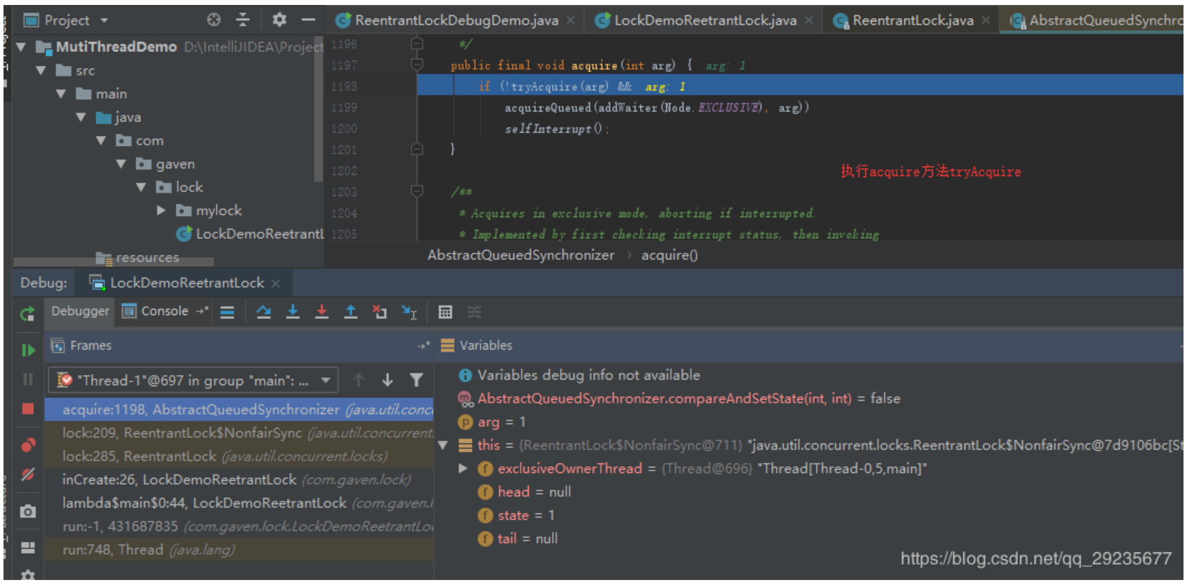
```
    /**
     * Performs lock.  Try immediate barge, backing up to normal
     * acquire on failure.
     */
    final void lock() {
        if (compareAndSetState( expect: 0,  update: 1))
            setExclusiveOwnerThread(Thread.currentThread());
        else
            acquire( arg: 1);
    }

    protected final boolean tryAcquire(int acquires) { return nonfairTryAcquire(acquires); }
}

/**
 * Sync object for fair locks
 */
```

```
Frames
"Thread-1"@697 in group "main": ...
lock:209, ReentrantLock$NonfairSync (java.util.concurrent
lock:285, ReentrantLock (java.util.concurrent.locks)
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.
run:-1, 431687835 (com.gaven.lock.LockDemoReetrantLo
```

```
Variables
AbstractQueuedSynchronizer.compareAndSetState(int, int) = false
this = {ReentrantLock$NonfairSync@711} "java.util.concurrent.locks.ReentrantLock$NonfairSync@7d91
  exclusiveOwnerThread = {Thread@696} "Thread[Thread-0,5,main]"
  head = null
  state = 1
  tail = null
```

https://blog.csdn.net/qq_29235677

之后执行acquire方法

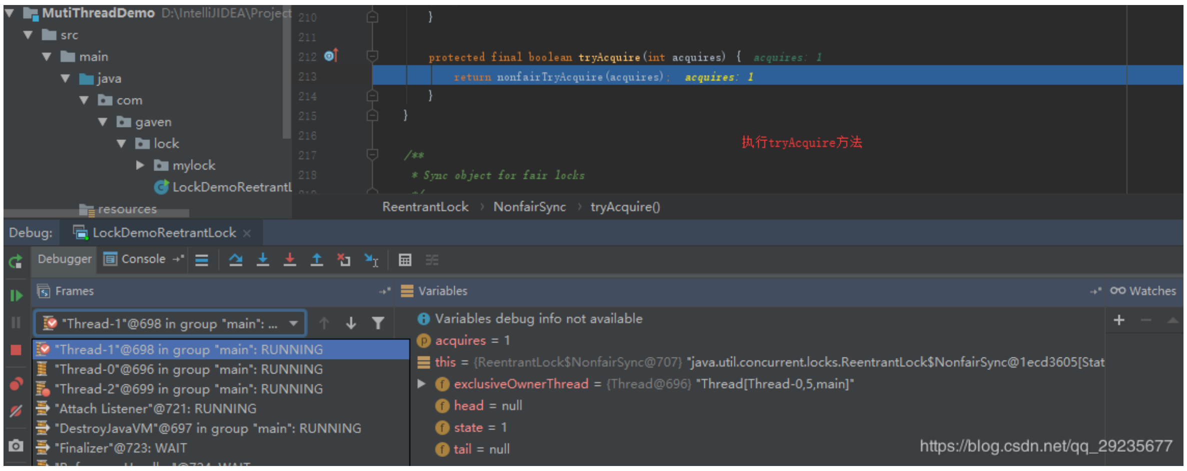


```java
     */
    public final void acquire(int arg) {   arg: 1
        if (!tryAcquire(arg) &&   arg: 1
            acquireQueued(addWaiter(Node.EXCLUSIVE), arg))
            selfInterrupt();
    }

    /**
     * Acquires in exclusive mode, aborting if interrupted.
     * Implemented by first checking interrupt status, then invoking
```

执行acquire方法tryAcquire

```
Frames
"Thread-1"@697 in group "main": ...
acquire:1198, AbstractQueuedSynchronizer (java.util.conc
lock:209, ReentrantLock$NonfairSync (java.util.concurrent
lock:285, ReentrantLock (java.util.concurrent.locks)
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.l
run:-1, 431687835 (com.gaven.lock.LockDemoReetrantLo
run:748, Thread (java.lang)

Variables
Variables debug info not available
AbstractQueuedSynchronizer.compareAndSetState(int, int) = false
arg = 1
this = {ReentrantLock$NonfairSync@711} "java.util.concurrent.locks.ReentrantLock$NonfairSync@7d9106bc[St
    exclusiveOwnerThread = {Thread@696} "Thread[Thread-0,5,main]"
    head = null
    state = 1
    tail = null
```

https://blog.csdn.net/qq_29235677



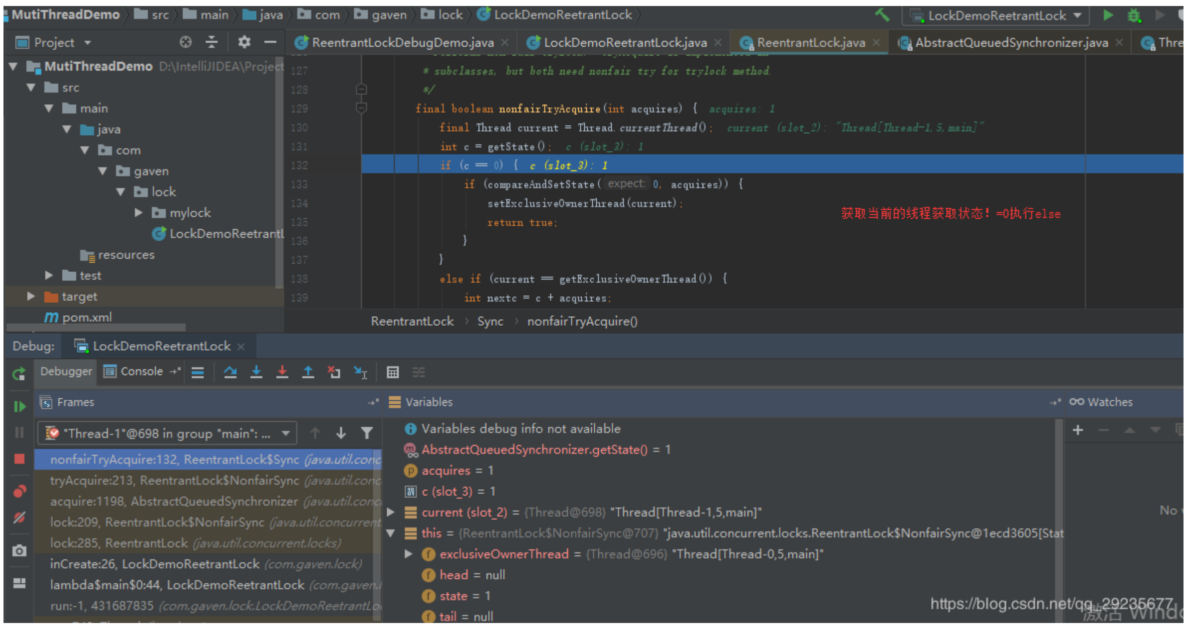
```java
        }

        protected final boolean tryAcquire(int acquires) {   acquires: 1
            return nonfairTryAcquire(acquires);   acquires: 1
        }
    }

    /**
     * Sync object for fair locks
```

执行tryAcquire方法

```
Frames
"Thread-1"@698 in group "main": ...
"Thread-1"@698 in group "main": RUNNING
"Thread-0"@696 in group "main": RUNNING
"Thread-2"@699 in group "main": RUNNING
"Attach Listener"@721: RUNNING
"DestroyJavaVM"@697 in group "main": RUNNING
"Finalizer"@723: WAIT

Variables
Variables debug info not available
acquires = 1
this = {ReentrantLock$NonfairSync@707} "java.util.concurrent.locks.ReentrantLock$NonfairSync@1ecd3605[Stat
    exclusiveOwnerThread = {Thread@696} "Thread[Thread-0,5,main]"
    head = null
    state = 1
    tail = null
```

https://blog.csdn.net/qq_29235677

再跳

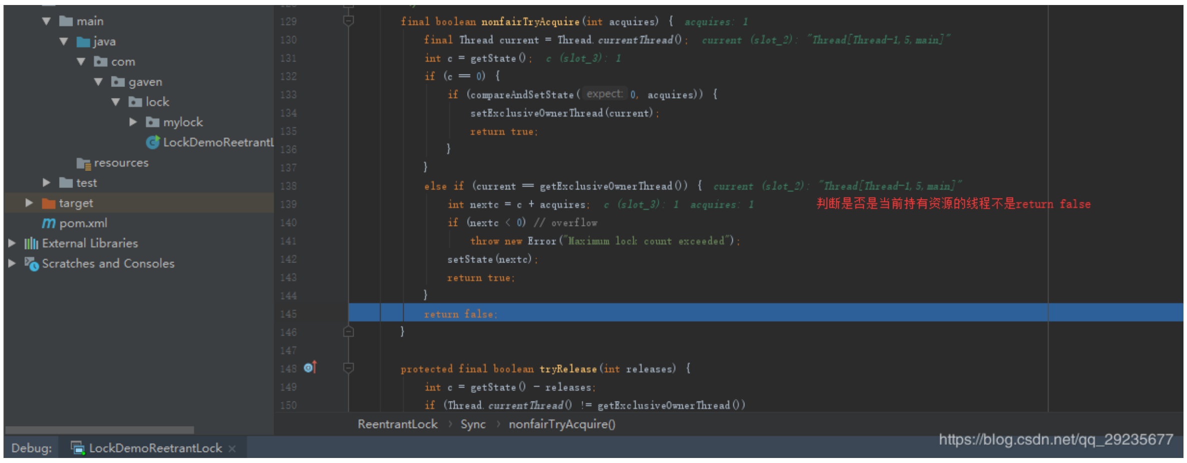


```java
         * subclasses, but both need nonfair try for trylock method.
         */
        final boolean nonfairTryAcquire(int acquires) {   acquires: 1
            final Thread current = Thread.currentThread();   current (slot_2): "Thread[Thread-1,5,main]"
            int c = getState();   c (slot_3): 1
            if (c == 0) {   c (slot_3): 1
                if (compareAndSetState( expect: 0, acquires)) {
                    setExclusiveOwnerThread(current);
                    return true;
                }
            }
            else if (current == getExclusiveOwnerThread()) {
                int nextc = c + acquires;
```

获取当前的线程获取状态！=0执行else

```
Frames
"Thread-1"@698 in group "main": ...
nonfairTryAcquire:132, ReentrantLock$Sync (java.util.conc
tryAcquire:213, ReentrantLock$NonfairSync (java.util.conc
acquire:1198, AbstractQueuedSynchronizer (java.util.conc
lock:209, ReentrantLock$NonfairSync (java.util.concurrent
lock:285, ReentrantLock (java.util.concurrent.locks)
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.l
run:-1, 431687835 (com.gaven.lock.LockDemoReetrantLo

Variables
Variables debug info not available
AbstractQueuedSynchronizer.getState() = 1
acquires = 1
c (slot_3) = 1
current (slot_2) = {Thread@698} "Thread[Thread-1,5,main]"
this = {ReentrantLock$NonfairSync@707} "java.util.concurrent.locks.ReentrantLock$NonfairSync@1ecd3605[Stat
    exclusiveOwnerThread = {Thread@696} "Thread[Thread-0,5,main]"
    head = null
    state = 1
    tail = null
```

https://blog.csdn.net/qq_29235677

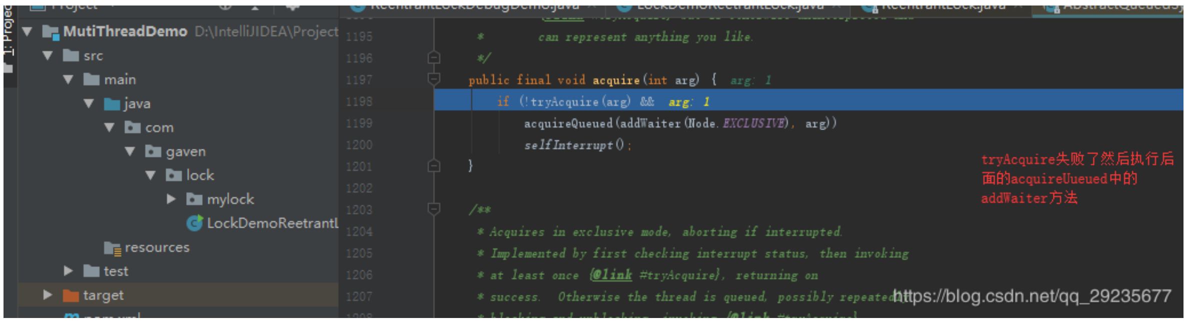


```java
final boolean nonfairTryAcquire(int acquires) {  acquires: 1
    final Thread current = Thread.currentThread();  current (slot_2): "Thread[Thread-1,5,main]"
    int c = getState();  c (slot_3): 1
    if (c == 0) {
        if (compareAndSetState( expect: 0, acquires)) {
            setExclusiveOwnerThread(current);
            return true;
        }
    }
    else if (current == getExclusiveOwnerThread()) {  current (slot_2): "Thread[Thread-1,5,main]"
        int nextc = c + acquires;  c (slot_3): 1  acquires: 1
        if (nextc < 0) // overflow
            throw new Error("Maximum lock count exceeded");
        setState(nextc);
        return true;
    }
    return false;
}

protected final boolean tryRelease(int releases) {
    int c = getState() - releases;
    if (Thread.currentThread() != getExclusiveOwnerThread())
```

判断是否是当前持有资源的线程不是return false

ReentrantLock › Sync › nonfairTryAcquire()

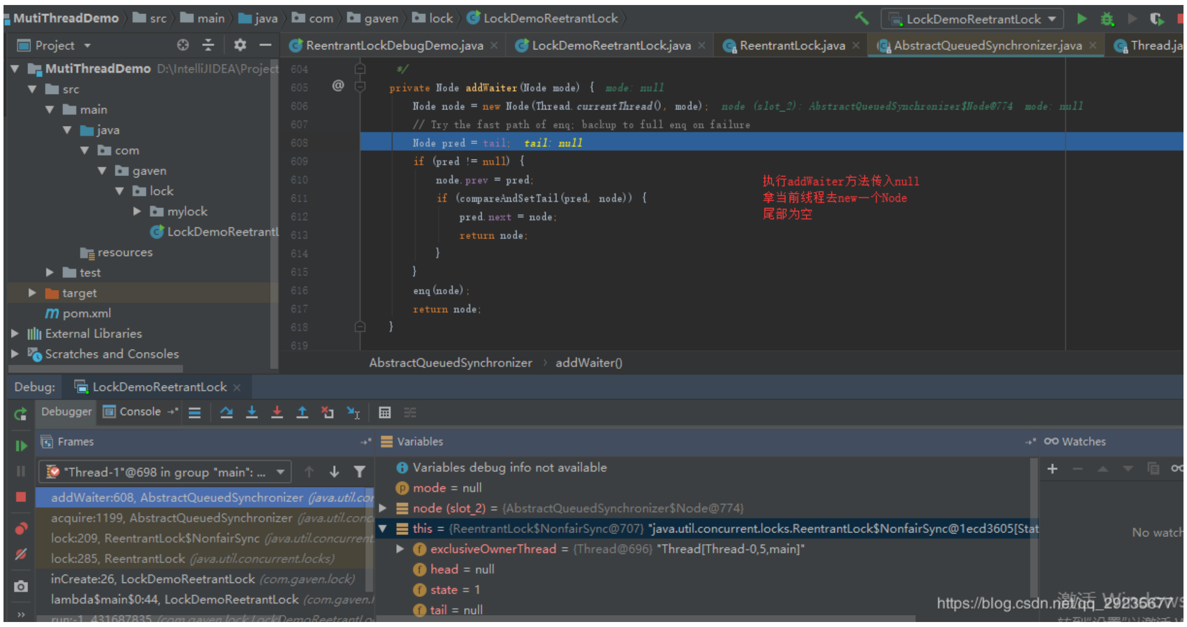
https://blog.csdn.net/qq_29235677



```java
 *        can represent anything you like.
 */
public final void acquire(int arg) {  arg: 1
    if (!tryAcquire(arg) &&  arg: 1
        acquireQueued(addWaiter(Node.EXCLUSIVE), arg))
        selfInterrupt();
}

/**
 * Acquires in exclusive mode, aborting if interrupted.
 * Implemented by first checking interrupt status, then invoking
 * at least once {@link #tryAcquire}, returning on
 * success.  Otherwise the thread is queued, possibly repeatedly
```

tryAcquire失败了然后执行后面的acquireUeued中的addWaiter方法

https://blog.csdn.net/qq_29235677



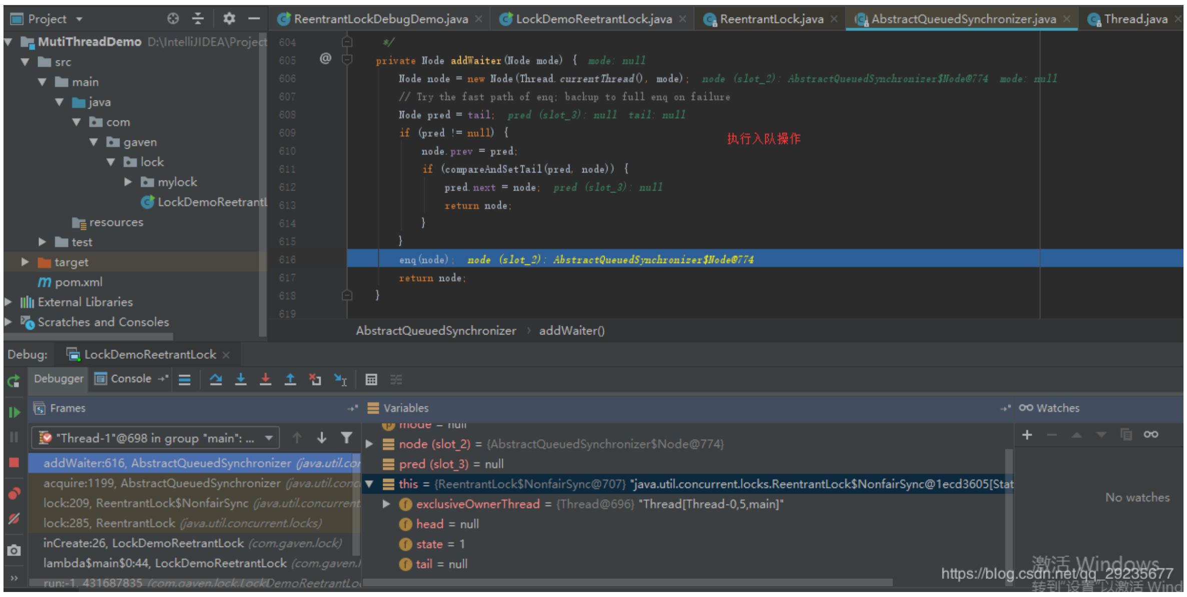
```java
 */
private Node addWaiter(Node mode) {  mode: null
    Node node = new Node(Thread.currentThread(), mode);  node (slot_2): AbstractQueuedSynchronizer$Node@774  mode: null
    // Try the fast path of enq; backup to full enq on failure
    Node pred = tail;  tail: null
    if (pred != null) {
        node.prev = pred;
        if (compareAndSetTail(pred, node)) {
            pred.next = node;
            return node;
        }
    }
    enq(node);
    return node;
}
```

执行addWaiter方法传入null
拿当前线程去new一个Node
尾部为空

AbstractQueuedSynchronizer › addWaiter()

```
Frames
"Thread-1"@698 in group "main": ...
addWaiter:608, AbstractQueuedSynchronizer (java.util.con
acquire:1199, AbstractQueuedSynchronizer (java.util.conc
lock:209, ReentrantLock$NonfairSync (java.util.concurrent
lock:285, ReentrantLock (java.util.concurrent.locks)
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.l

Variables
Variables debug info not available
mode = null
node (slot_2) = {AbstractQueuedSynchronizer$Node@774}
this = {ReentrantLock$NonfairSync@707} "java.util.concurrent.locks.ReentrantLock$NonfairSync@1ecd3605[Stat
    exclusiveOwnerThread = {Thread@696} "Thread[Thread-0,5,main]"
    head = null
    state = 1
    tail = null
```

https://blog.csdn.net/qq_29235677



```java
 */
private Node addWaiter(Node mode) {  mode: null
    Node node = new Node(Thread.currentThread(), mode);  node (slot_2): AbstractQueuedSynchronizer$Node@774  mode: null
    // Try the fast path of enq; backup to full enq on failure
    Node pred = tail;  pred (slot_3): null  tail: null
    if (pred != null) {
        node.prev = pred;
        if (compareAndSetTail(pred, node)) {
            pred.next = node;  pred (slot_3): null
            return node;
        }
    }
    enq(node);  node (slot_2): AbstractQueuedSynchronizer$Node@774
    return node;
}
```

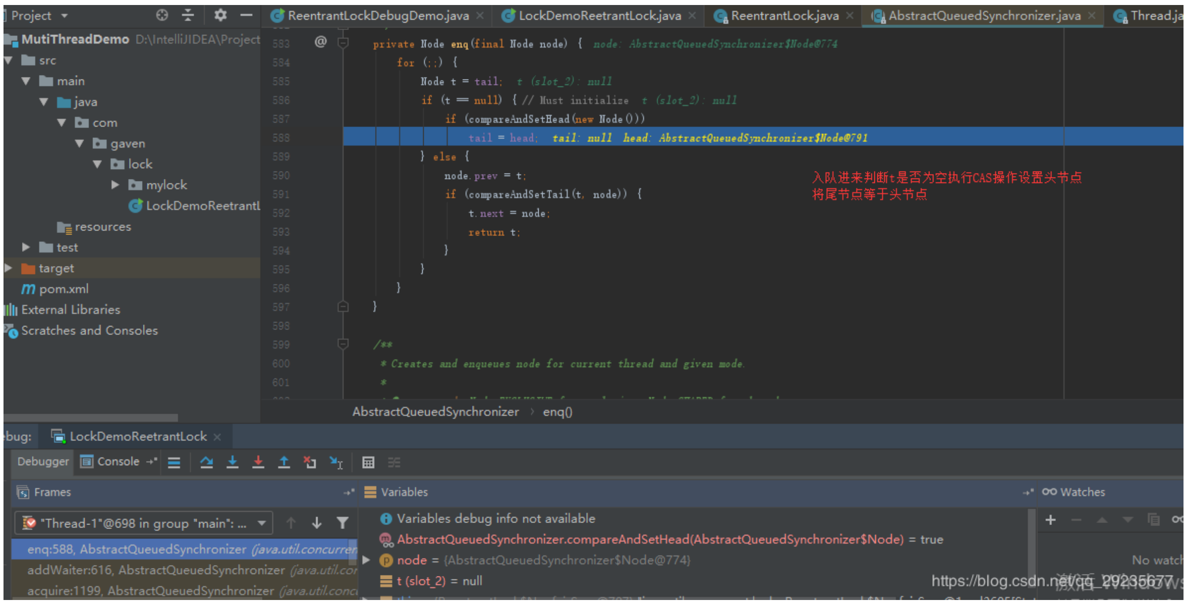
执行入队操作

AbstractQueuedSynchronizer › addWaiter()

```
Frames
"Thread-1"@698 in group "main": ...
addWaiter:616, AbstractQueuedSynchronizer (java.util.co
acquire:1199, AbstractQueuedSynchronizer (java.util.conc
lock:209, ReentrantLock$NonfairSync (java.util.concurrent
lock:285, ReentrantLock (java.util.concurrent.locks)
inCreate:26, LockDemoReetrantLock (com.gaven.lock)
lambda$main$0:44, LockDemoReetrantLock (com.gaven.l

Variables
node (slot_2) = {AbstractQueuedSynchronizer$Node@774}
pred (slot_3) = null
this = {ReentrantLock$NonfairSync@707} "java.util.concurrent.locks.ReentrantLock$NonfairSync@1ecd3605[Stat
    exclusiveOwnerThread = {Thread@696} "Thread[Thread-0,5,main]"
    head = null
    state = 1
    tail = null
```

https://blog.csdn.net/qq_29235677

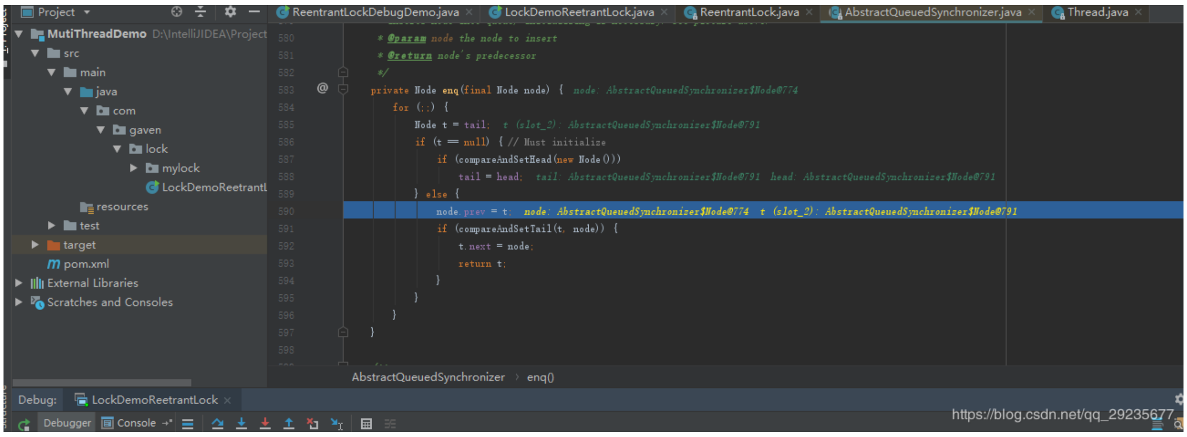
跳进去



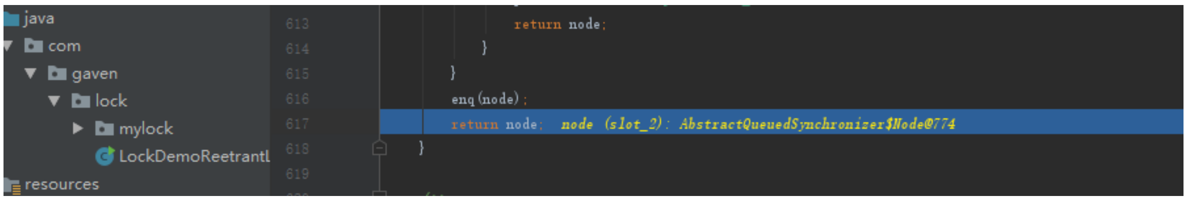
```
    private Node enq(final Node node) {  node: AbstractQueuedSynchronizer$Node@774
        for (;;) {
            Node t = tail;  t (slot_2): null
            if (t == null) { // Must initialize  t (slot_2): null
                if (compareAndSetHead(new Node()))
                    tail = head;  tail: null  head: AbstractQueuedSynchronizer$Node@791
            } else {
                node.prev = t;
                if (compareAndSetTail(t, node)) {
                    t.next = node;
                    return t;
                }
            }
        }
    }

    /**
     * Creates and enqueues node for current thread and given mode.
     *
```

入队进来判断t是否为空执行CAS操作设置头节点
将尾节点等于头节点

```
"Thread-1"@698 in group "main": ...
enq:588, AbstractQueuedSynchronizer (java.util.concurrent...)
addWaiter:616, AbstractQueuedSynchronizer (java.util.con...)
acquire:1199, AbstractQueuedSynchronizer (java.util.conc...)

Variables debug info not available
AbstractQueuedSynchronizer.compareAndSetHead(AbstractQueuedSynchronizer$Node) = true
node = {AbstractQueuedSynchronizer$Node@774}
t (slot_2) = null
```

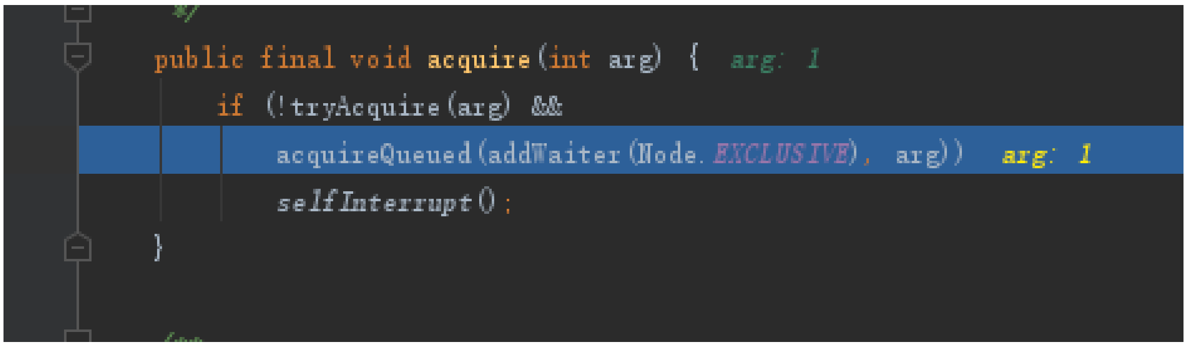


```
     * @param node the node to insert
     * @return node's predecessor
     */
    private Node enq(final Node node) {  node: AbstractQueuedSynchronizer$Node@774
        for (;;) {
            Node t = tail;  t (slot_2): AbstractQueuedSynchronizer$Node@791
            if (t == null) { // Must initialize
                if (compareAndSetHead(new Node()))
                    tail = head;  tail: AbstractQueuedSynchronizer$Node@791  head: AbstractQueuedSynchronizer$Node@791
            } else {
                node.prev = t;  node: AbstractQueuedSynchronizer$Node@774  t (slot_2): AbstractQueuedSynchronizer$Node@791
                if (compareAndSetTail(t, node)) {
                    t.next = node;
                    return t;
                }
            }
        }
    }
```



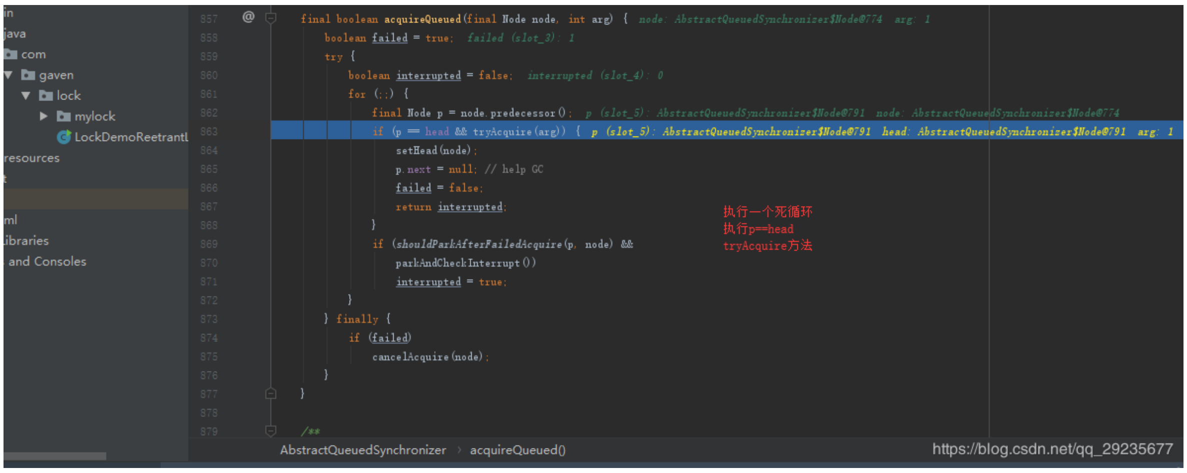
```
                return node;
            }
        }
        enq(node);
        return node;  node (slot_2): AbstractQueuedSynchronizer$Node@774
    }
```

此时addWaiter执行完毕



```
    public final void acquire(int arg) {  arg: 1
        if (!tryAcquire(arg) &&
            acquireQueued(addWaiter(Node.EXCLUSIVE), arg))  arg: 1
            selfInterrupt();
    }
```

接着执行acquireQueued方法

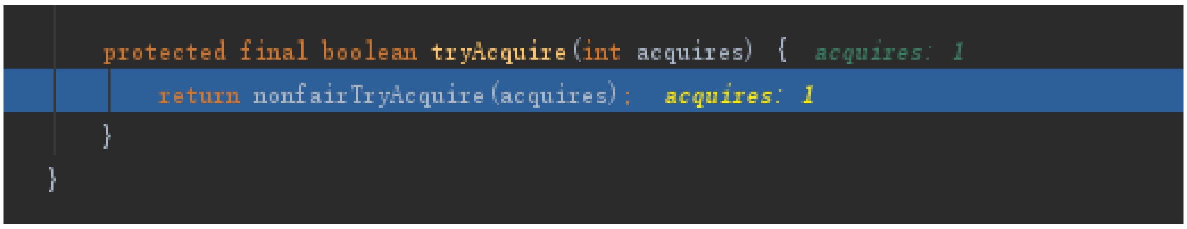


```java
final boolean acquireQueued(final Node node, int arg) {  node: AbstractQueuedSynchronizer$Node@774  arg: 1
    boolean failed = true;  failed (slot_3): 1
    try {
        boolean interrupted = false;  interrupted (slot_4): 0
        for (;;) {
            final Node p = node.predecessor();  p (slot_5): AbstractQueuedSynchronizer$Node@791  node: AbstractQueuedSynchronizer$Node@774
            if (p == head && tryAcquire(arg)) {  p (slot_5): AbstractQueuedSynchronizer$Node@791  head: AbstractQueuedSynchronizer$Node@791  arg: 1
                setHead(node);
                p.next = null; // help GC
                failed = false;
                return interrupted;
            }
            if (shouldParkAfterFailedAcquire(p, node) &&
                parkAndCheckInterrupt())
                interrupted = true;
        }
    } finally {
        if (failed)
            cancelAcquire(node);
    }
}

/**
```

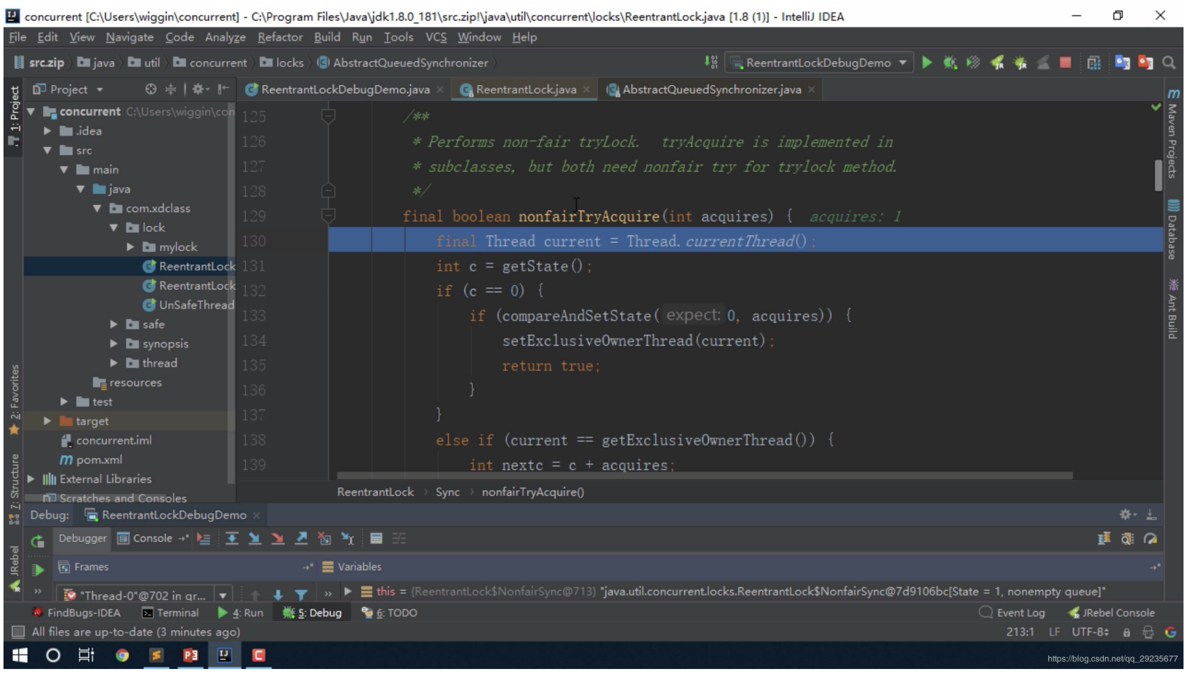
执行一个死循环
执行p==head
tryAcquire方法

AbstractQueuedSynchronizer › acquireQueued()

https://blog.csdn.net/qq_29235677



```java
protected final boolean tryAcquire(int acquires) {  acquires: 1
    return nonfairTryAcquire(acquires);  acquires: 1
}
}
```



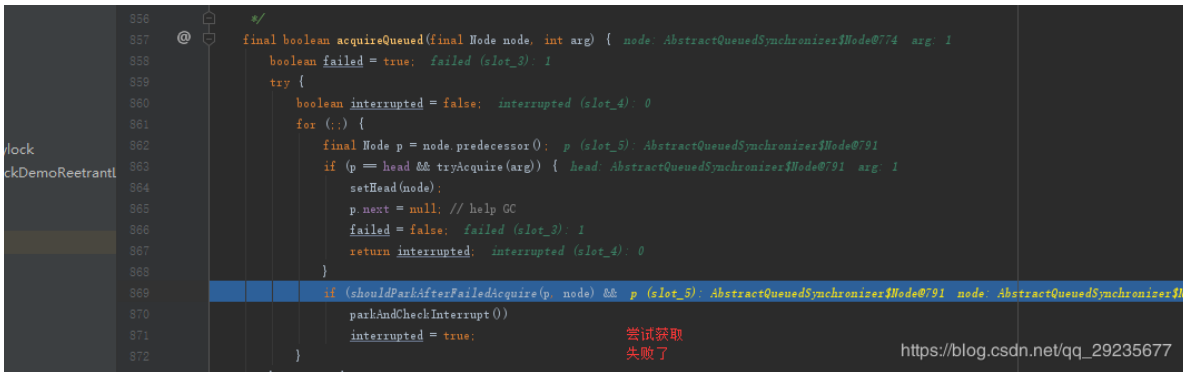
concurrent [C:\Users\wiggin\concurrent] - C:\Program Files\Java\jdk1.8.0_181\src.zip!\java\util\concurrent\locks\ReentrantLock.java [1.8 (1)] - IntelliJ IDEA

```java
/**
 * Performs non-fair tryLock.  tryAcquire is implemented in
 * subclasses, but both need nonfair try for trylock method.
 */
final boolean nonfairTryAcquire(int acquires) {  acquires: 1
    final Thread current = Thread.currentThread();
    int c = getState();
    if (c == 0) {
        if (compareAndSetState( expect: 0, acquires)) {
            setExclusiveOwnerThread(current);
            return true;
        }
    }
    else if (current == getExclusiveOwnerThread()) {
        int nextc = c + acquires;
```

ReentrantLock › Sync › nonfairTryAcquire()

```
this = {ReentrantLock$NonfairSync@713} "java.util.concurrent.locks.ReentrantLock$NonfairSync@7d9106bc[State = 1, nonempty queue]"
```

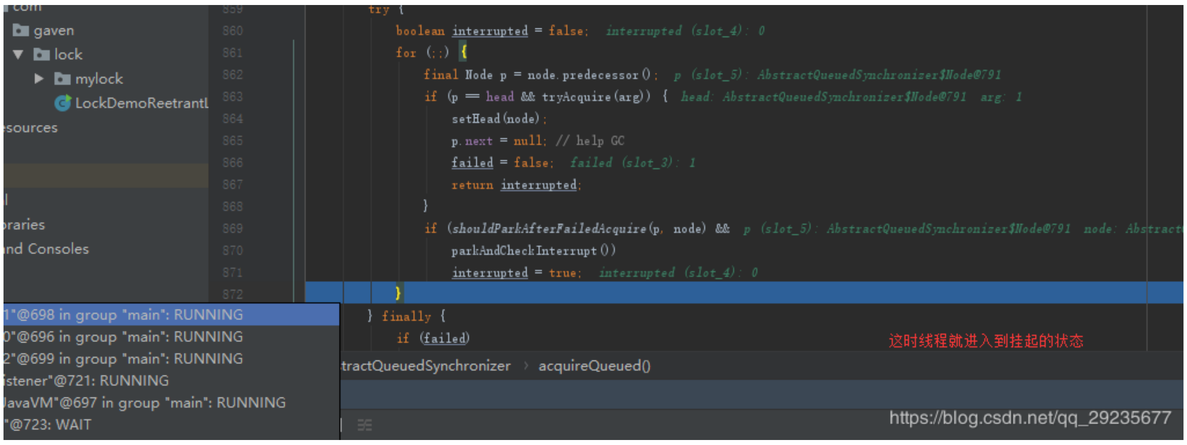
https://blog.csdn.net/qq_29235677



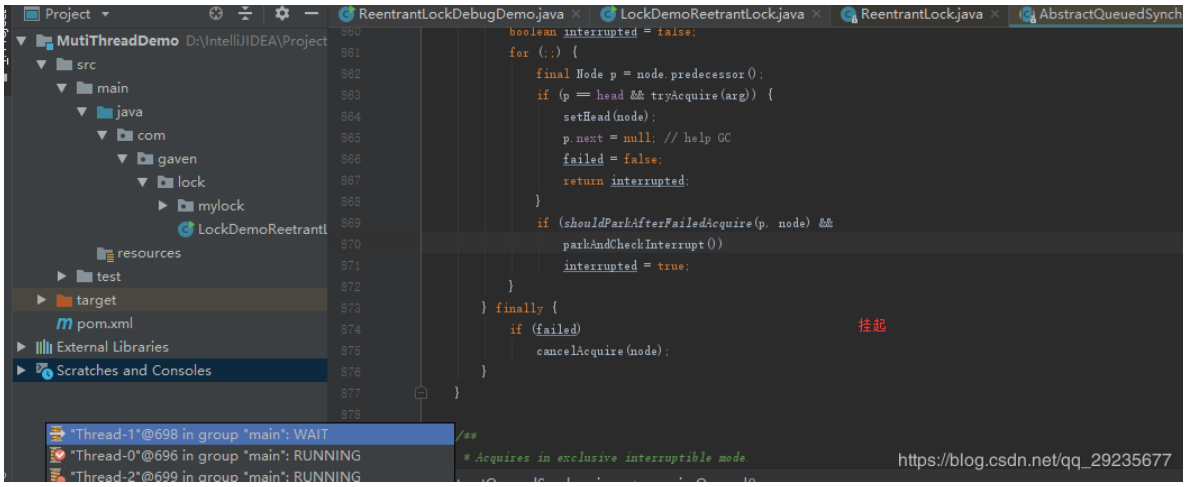
```java
 */
final boolean acquireQueued(final Node node, int arg) {  node: AbstractQueuedSynchronizer$Node@774  arg: 1
    boolean failed = true;  failed (slot_3): 1
    try {
        boolean interrupted = false;  interrupted (slot_4): 0
        for (;;) {
            final Node p = node.predecessor();  p (slot_5): AbstractQueuedSynchronizer$Node@791
            if (p == head && tryAcquire(arg)) {  head: AbstractQueuedSynchronizer$Node@791  arg: 1
                setHead(node);
                p.next = null; // help GC
                failed = false;  failed (slot_3): 1
                return interrupted;  interrupted (slot_4): 0
            }
            if (shouldParkAfterFailedAcquire(p, node) &&  p (slot_5): AbstractQueuedSynchronizer$Node@791  node: AbstractQueuedSynchronizer$N
                parkAndCheckInterrupt())
                interrupted = true;
        }
```

尝试获取
失败了

https://blog.csdn.net/qq_29235677

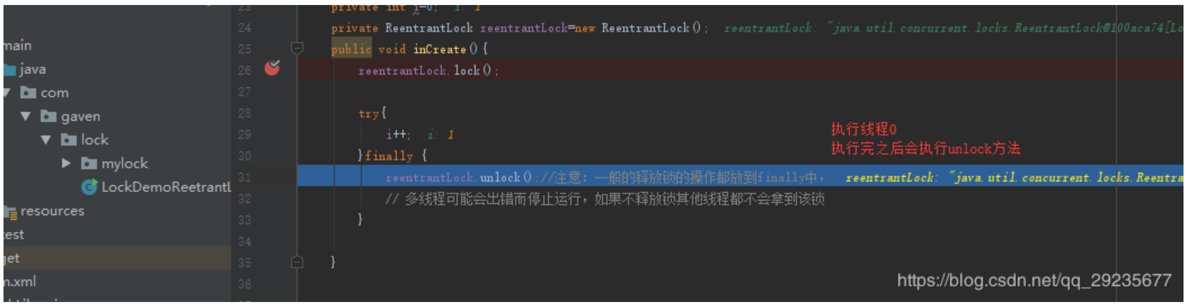

```
        try {
            boolean interrupted = false;  interrupted (slot_4): 0
            for (;;) {
                final Node p = node.predecessor();  p (slot_5): AbstractQueuedSynchronizer$Node@791
                if (p == head && tryAcquire(arg)) {  head: AbstractQueuedSynchronizer$Node@791  arg: 1
                    setHead(node);
                    p.next = null; // help GC
                    failed = false;  failed (slot_3): 1
                    return interrupted;
                }
                if (shouldParkAfterFailedAcquire(p, node) &&  p (slot_5): AbstractQueuedSynchronizer$Node@791  node: Abstract
                    parkAndCheckInterrupt())
                    interrupted = true;  interrupted (slot_4): 0
            }
        } finally {
            if (failed)
```
这时线程就进入到挂起的状态

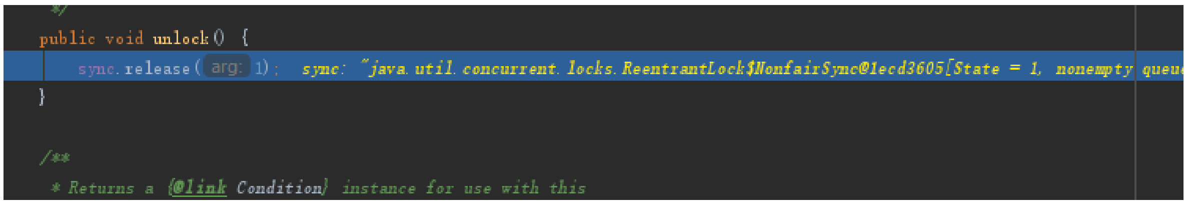
```
1"@698 in group "main": RUNNING
0"@696 in group "main": RUNNING
2"@699 in group "main": RUNNING
stener"@721: RUNNING
JavaVM"@697 in group "main": RUNNING
"@723: WAIT
```


```
        boolean interrupted = false;
        for (;;) {
            final Node p = node.predecessor();
            if (p == head && tryAcquire(arg)) {
                setHead(node);
                p.next = null; // help GC
                failed = false;
                return interrupted;
            }
            if (shouldParkAfterFailedAcquire(p, node) &&
                parkAndCheckInterrupt())
                interrupted = true;
        }
    } finally {
        if (failed)
            cancelAcquire(node);
    }
}

/**
 * Acquires in exclusive interruptible mode.
```
挂起

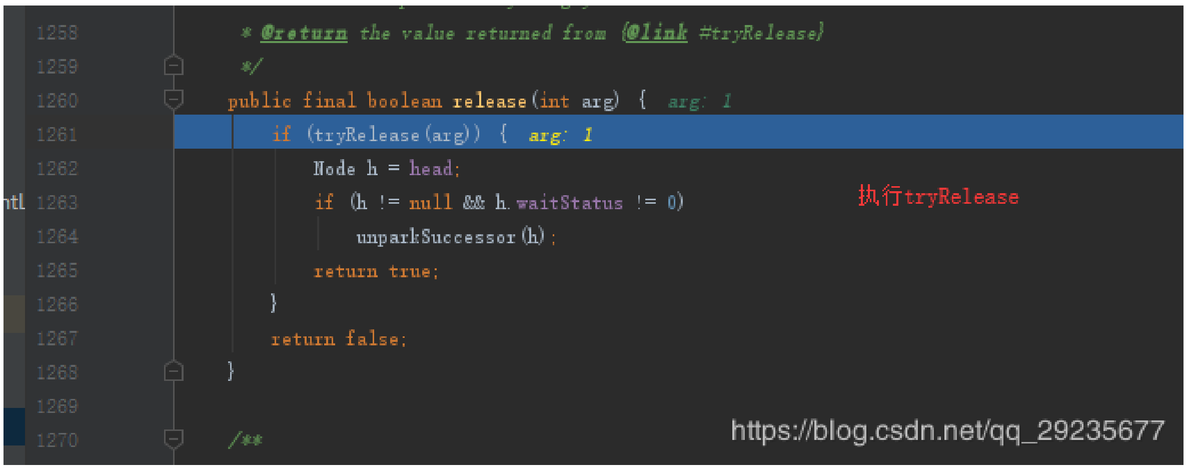
```
"Thread-1"@698 in group "main": WAIT
"Thread-0"@696 in group "main": RUNNING
"Thread-2"@699 in group "main": RUNNING
```

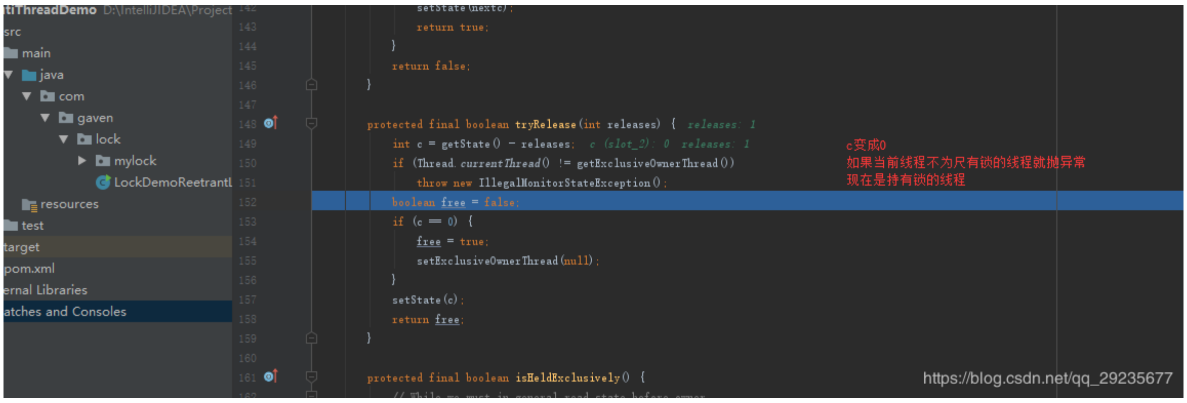
同理线程2也这样


```
    private ReentrantLock reentrantLock=new ReentrantLock();  reentrantLock: "java.util.concurrent.locks.ReentrantLock@100aca74[Lo
    public void inCreate(){
        reentrantLock.lock();

        try{
            i++;  i: 1
        }finally {
            reentrantLock.unlock();//注意：一般的释放锁的操作都放到finally中，  reentrantLock: "java.util.concurrent.locks.Reentra
            // 多线程可能会出错而停止运行，如果不释放锁其他线程都不会拿到该锁
        }

    }
```
执行线程0
执行完之后会执行unlock方法


```
    public void unlock() {
        sync.release( arg: 1);  sync: "java.util.concurrent.locks.ReentrantLock$NonfairSync@1ecd3605[State = 1, nonempty queu
    }

    /**
     * Returns a {@link Condition} instance for use with this
```

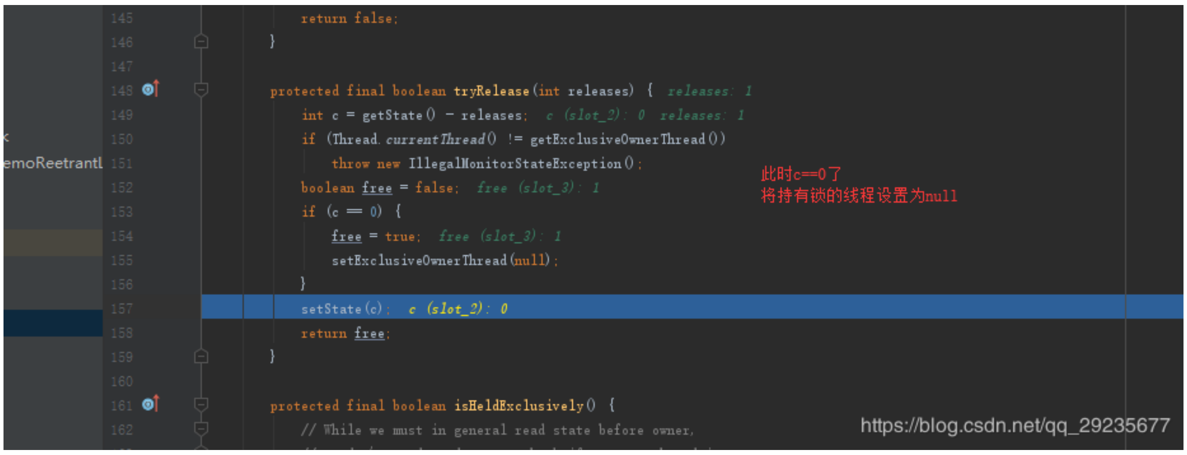
执行tryRelease方法


```
     * @return the value returned from {@link #tryRelease}
     */
    public final boolean release(int arg) {  arg: 1
        if (tryRelease(arg)) {  arg: 1
            Node h = head;
            if (h != null && h.waitStatus != 0)
                unparkSuccessor(h);
            return true;
        }
        return false;
    }

    /**
```
执行tryRelease



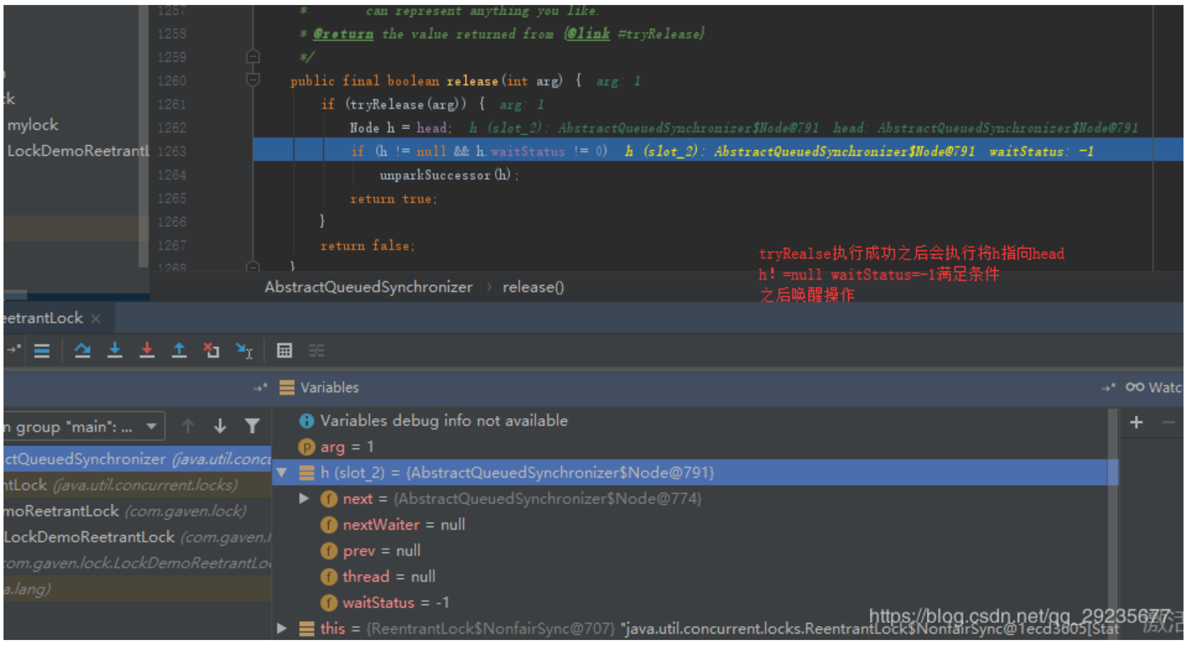
```
            setState(nextc);
            return true;
        }
        return false;
    }

    protected final boolean tryRelease(int releases) {  releases: 1
        int c = getState() - releases;  c (slot_2): 0  releases: 1
        if (Thread.currentThread() != getExclusiveOwnerThread())
            throw new IllegalMonitorStateException();
        boolean free = false;
        if (c == 0) {
            free = true;
            setExclusiveOwnerThread(null);
        }
        setState(c);
        return free;
    }

    protected final boolean isHeldExclusively() {
```

c变成0
如果当前线程不为尺有锁的线程就抛异常
现在是持有锁的线程
https://blog.csdn.net/qq_29235677



```
        return false;
    }

    protected final boolean tryRelease(int releases) {  releases: 1
        int c = getState() - releases;  c (slot_2): 0  releases: 1
        if (Thread.currentThread() != getExclusiveOwnerThread())
            throw new IllegalMonitorStateException();
        boolean free = false;  free (slot_3): 1
        if (c == 0) {
            free = true;  free (slot_3): 1
            setExclusiveOwnerThread(null);
        }
        setState(c);  c (slot_2): 0
        return free;
    }

    protected final boolean isHeldExclusively() {
        // While we must in general read state before owner,
```

tryRelease方法执行成功

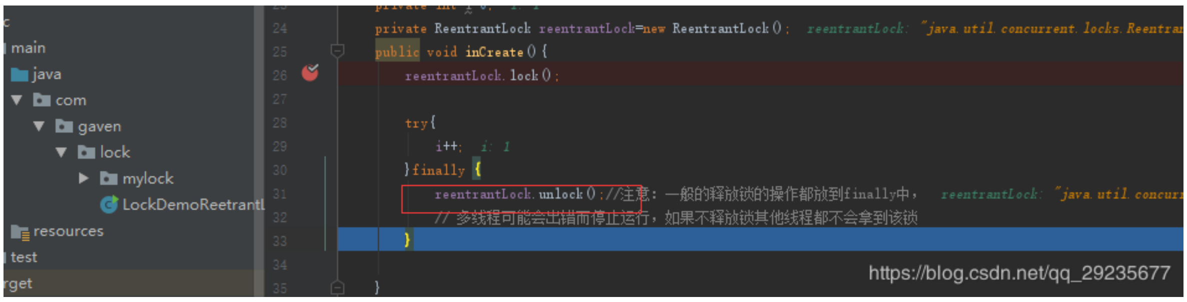


```
     *        can represent anything you like.
     * @return the value returned from {@link #tryRelease}
     */
    public final boolean release(int arg) {  arg: 1
        if (tryRelease(arg)) {  arg: 1
            Node h = head;  h (slot_2): AbstractQueuedSynchronizer$Node@791  head: AbstractQueuedSynchronizer$Node@791
            if (h != null && h.waitStatus != 0)  h (slot_2): AbstractQueuedSynchronizer$Node@791  waitStatus: -1
                unparkSuccessor(h);
            return true;
        }
        return false;
    }
```

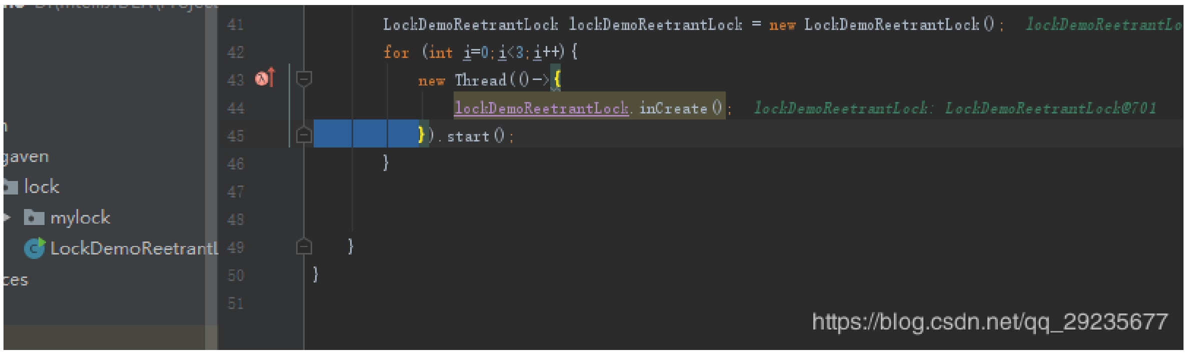
```
AbstractQueuedSynchronizer › release()

Variables
Variables debug info not available
arg = 1
h (slot_2) = {AbstractQueuedSynchronizer$Node@791}
    next = {AbstractQueuedSynchronizer$Node@774}
    nextWaiter = null
    prev = null
    thread = null
    waitStatus = -1
this = {ReentrantLock$NonfairSync@707} "java.util.concurrent.locks.ReentrantLock$NonfairSync@1ecd3605[Stat
```

tryRealse执行成功之后会执行将h指向head
h! =null waitStatus=-1满足条件
之后唤醒操作
etrantLock
n group "main": ...
ctQueuedSynchronizer (java.util.conc
ntLock (java.util.concurrent.locks)
moReentrantLock (com.gaven.lock)
LockDemoReentrantLock (com.gaven.l
om.gaven.lock.LockDemoReentrantLo
a.lang)
Watc
https://blog.csdn.net/qq_29235677

执行完成之后就会唤醒其他线程

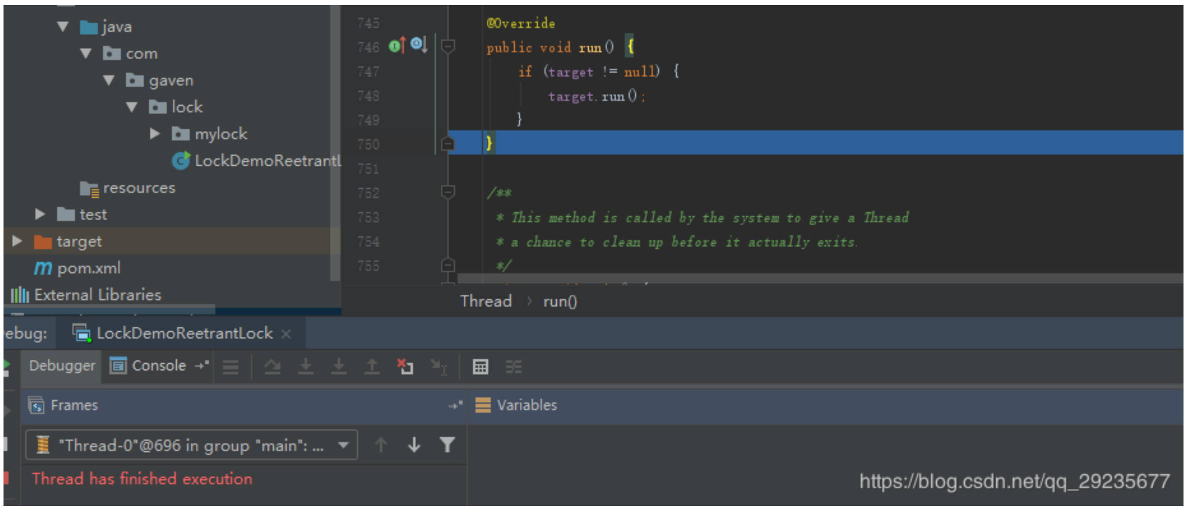


```
    private ReentrantLock reentrantLock=new ReentrantLock();  reentrantLock: "java.util.concurrent.locks.Reentra
    public void inCreate(){
        reentrantLock.lock();

        try{
            i++;  i: 1
        }finally {
            reentrantLock.unlock();//注意：一般的释放锁的操作都放到finally中，  reentrantLock: "java.util.concurr
            // 多线程可能会出错而停止运行，如果不释放锁其他线程都不会拿到该锁
        }

    }
```



```
        LockDemoReetrantLock lockDemoReetrantLock = new LockDemoReetrantLock();   lockDemoReetrantLo
        for (int i=0;i<3;i++){
            new Thread(()->{
                lockDemoReetrantLock.inCreate();   lockDemoReetrantLock: LockDemoReetrantLock@701
            }).start();
        }

    }
}
```

https://blog.csdn.net/qq_29235677



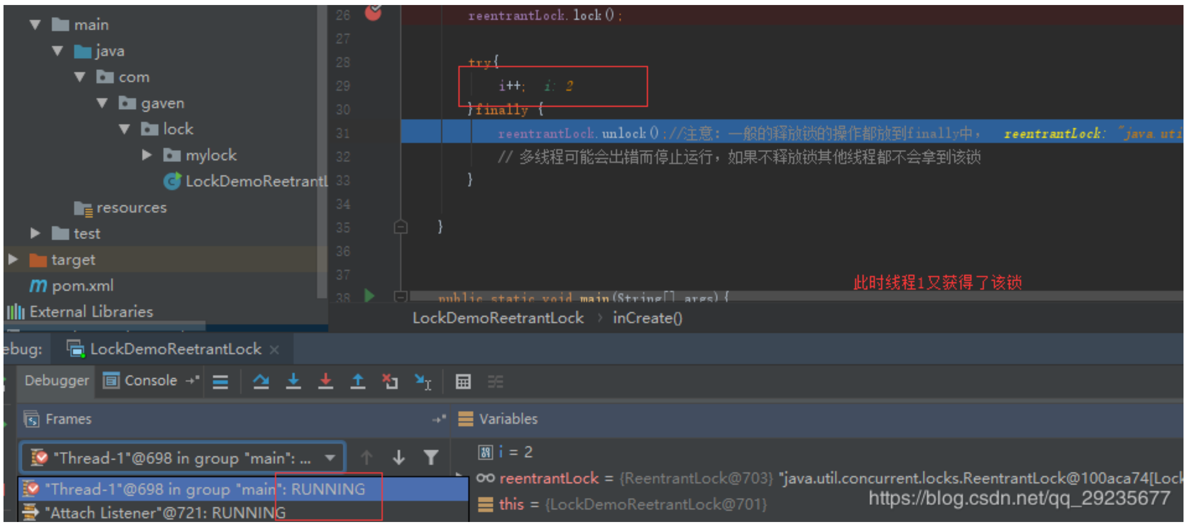
```
    @Override
    public void run() {
        if (target != null) {
            target.run();
        }
    }

    /**
     * This method is called by the system to give a Thread
     * a chance to clean up before it actually exits.
     */
```

Thread > run()

```
"Thread-0"@696 in group "main": ...
Thread has finished execution
```

https://blog.csdn.net/qq_29235677

该线程执行完毕。接着查看其他线程（1，2）



```
            reentrantLock.lock();

            try{
                i++;   i: 2
            }finally {
                reentrantLock.unlock();//注意：一般的释放锁的操作都放到finally中，   reentrantLock: "java.uti
                // 多线程可能会出错而停止运行，如果不释放锁其他线程都不会拿到该锁
            }

        }

    public static void main(String[] args) {
```

此时线程1又获得了该锁

LockDemoReetrantLock > inCreate()

```
"Thread-1"@698 in group "main": ...
"Thread-1"@698 in group "main": RUNNING
"Attach Listener"@721: RUNNING
```

```
i = 2
reentrantLock = {ReentrantLock@703} "java.util.concurrent.locks.ReentrantLock@100aca74[Lock
this = {LockDemoReetrantLock@701}
```

https://blog.csdn.net/qq_29235677

---

ref: 1.多线程——多线程debug调试