**Java Lock Example – ReentrantLock**

Most of the times, synchronized keyword is the way to go but it has some shortcomings that lead the way to inclusion of Lock API in Java Concurrency package. Java 1.5 Concurrency API came up with java.util.concurrent.locks package with Lock interface and some implementation classes to improve the Object locking mechanism.

Some important interfaces and classes in Java Lock API are:

1. **Lock**: This is the base interface for Lock API. It provides all the features of synchronized keyword with additional ways to create different Conditions for locking, providing timeout for thread to wait for lock. Some of the important methods are lock() to acquire the lock, unlock() to release the lock, tryLock() to wait for lock for a certain period of time, newCondition() to create the Condition etc.
2. **Condition**: Condition objects are similar to [Object wait-notify](http://www.journaldev.com/1037/java-thread-wait-notify-and-notifyall-example) model with additional feature to create different sets of wait. A Condition object is always created by Lock object. Some of the important methods are await() that is similar to wait() and signal(), signalAll() that is similar to notify() and notifyAll() methods.
3. **ReadWriteLock**: It contains a pair of associated locks, one for read-only operations and another one for writing. The read lock may be held simultaneously by multiple reader threads as long as there are no writer threads. The write lock is exclusive.
4. **ReentrantLock**: This is the most widely used implementation class of Lock interface. This class implements the Lock interface in similar way as synchronized keyword. Apart from Lock interface implementation, **ReentrantLock** contains some utility methods to get the thread holding the lock, threads waiting to acquire the lock etc.

synchronized block are reentrant in nature i.e if a thread has lock on the monitor object and if another synchronized block requires to have the lock on the same monitor object then thread can enter that code block. I think this is the reason for the class name to be ReentrantLock. Let’s understand this feature with a simple example.

public class Test{

public synchronized foo(){

//do something

bar();

}

public synchronized bar(){

//do some more

}

}

If a thread enters foo(), it has the lock on Test object, so when it tries to execute bar() method, the thread is allowed to execute bar() method since it’s already holding the lock on the Test object i.e same as synchronized(this).

**Java Lock Example – ReentrantLock in Java**

Now let’s see a simple example where we will replace synchronized keyword with Java Lock API.

Let’s say we have a Resource class with some operation where we want it to be thread-safe and some methods where thread safety is not required.

package com.journaldev.threads.lock;

public class Resource {

public void doSomething(){

//do some operation, DB read, write etc

}

public void doLogging(){

//logging, no need for thread safety

}

}

Now let’s say we have a Runnable class where we will use Resource methods.

package com.journaldev.threads.lock;

public class SynchronizedLockExample implements Runnable{

private Resource resource;

public SynchronizedLockExample(Resource r){

this.resource = r;

}

@Override

public void run() {

synchronized (resource) {

resource.doSomething();

}

resource.doLogging();

}

}

Notice that I am using synchronized block to acquire the lock on Resource object. We could have created a dummy object in the class and used that for locking purpose.

Now let’s see how we can use java Lock API and rewrite above program without using synchronized keyword. We will use ReentrantLock in java.

package com.journaldev.threads.lock;

import java.util.concurrent.TimeUnit;

import java.util.concurrent.locks.Lock;

import java.util.concurrent.locks.ReentrantLock;

public class ConcurrencyLockExample implements Runnable{

private Resource resource;

private Lock lock;

public ConcurrencyLockExample(Resource r){

this.resource = r;

this.lock = new ReentrantLock();

}

@Override

public void run() {

try {

if(lock.tryLock(10, TimeUnit.SECONDS)){

resource.doSomething();

}

} catch (InterruptedException e) {

e.printStackTrace();

}finally{

//release lock

lock.unlock();

}

resource.doLogging();

}

}

As you can see that, I am using tryLock() method to make sure my thread waits only for definite time and if it’s not getting the lock on object, it’s just logging and exiting. Another important point to note is the use of try-finally block to make sure lock is released even if doSomething() method call throws any exception.

**Java Lock vs synchronized**

Based on above details and program, we can easily conclude following differences between Java Lock and synchronization.

1. Java Lock API provides more visibility and options for locking, unlike synchronized where a thread might end up waiting indefinitely for the lock, we can use tryLock() to make sure thread waits for specific time only.
2. Synchronization code is much cleaner and easy to maintain whereas with Lock we are forced to have try-finally block to make sure Lock is released even if some exception is thrown between lock() and unlock() method calls.
3. synchronization blocks or methods can cover only one method whereas we can acquire the lock in one method and release it in another method with Lock API.
4. synchronized keyword doesn’t provide fairness whereas we can set fairness to true while creating ReentrantLock object so that longest waiting thread gets the lock first.
5. We can create different conditions for Lock and different thread can await() for different conditions.

That’s all for Java Lock example, ReentrantLock in java and a comparative analysis with synchronized keyword.

FILED UNDER: [JAVA](http://www.journaldev.com/dev/java)

**About Pankaj**

If you have come this far, it means that you liked what you are reading. Why not reach little more and connect with me directly on [**Google Plus**](https://plus.google.com/118104420597648001532/posts?rel=author), **[Facebook](https://www.facebook.com/journaldev)** or [**Twitter**](https://twitter.com/JournalDev). I would love to hear your thoughts and opinions on my articles directly.

Recently I started creating video tutorials too, so do check out my videos on **[Youtube](https://www.youtube.com/user/JournalDev)**.

[« Core Java Interview Questions and Answers](http://www.journaldev.com/2366/core-java-interview-questions-and-answers)

[Java 8 Features with Examples »](http://www.journaldev.com/2389/java-8-features-with-examples)

**Comments**

1. **Riteeka says**

[JUNE 10, 2017 AT 9:12 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-38358)

Hi Pankaj,

Very good explanation. I tried this example working fine. But I would like to high light one case which is not handle by this example.

In finally block before releasing lock, we need to make sure Mock is held by current thread otherwise it will throw  
IllegalMonitorStateException. Attaching tested snippet code.

finally{  
System.out.println(Thread.currentThread().getName() + ” tries to unlock”);

if(Thread.holdsLock(resource)){  
lock.unlock();  
}  
}

Best Regards.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-38358)

* + **Riteeka says**

[JUNE 10, 2017 AT 10:12 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-38359)

Edited : find snippet below

if (((ReentrantLock) lock).isHeldByCurrentThread()) {  
System.out.println(Thread.currentThread().getName() + ” unlock resource obj”);  
lock.unlock();  
}

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-38359)

1. **Ravi says**

[NOVEMBER 22, 2016 AT 1:48 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-36893)

Hi Pankaj,

I like your posts a lot, awesome posts, very clear and precise!!!

I have a doubt on reentrant lock example above, what are you trying to explain here? wanna thread safe doSomthing() method, isn’t it? If so, synchronisation/lock logic shouldn’t be inside class of doSomething method instead inside run method of Thread? With above example how can you ensure the thread safety of doSomething() method?

Thanks  
Ravi.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-36893)

1. **Gunatron says**

[SEPTEMBER 8, 2016 AT 6:37 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-36264)

and where was the ReEntrantLock example? i see only synchronized vs Lock, not ReentrantLock

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-36264)

* + [**Pankaj**](http://www.journaldev.com/)**says**

[SEPTEMBER 8, 2016 AT 9:59 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-36265)

There is explanation of ReentrantLock as well as used in example program, what else do you want?

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-36265)

* + **Sachin says**

[JUNE 28, 2017 AT 12:18 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-38498)

Lock is just an interface and ReentrantLock is an implementation class of Lock interface. If you looked at the code with focus so you will find the code as mentioned below:-

public ConcurrencyLockExample(Resource r){  
this.resource = r;  
this.lock = new ReentrantLock();  
}

so the above code defines the ReentrantLock implementation.  
Thanks

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-38498)

1. **Riley says**

[JANUARY 11, 2016 AT 9:53 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33866)

please see below, also the the Runnable classes:

public class SynchronizedExample implements Runnable{

private Resource resource;

public SynchronizedExample(Resource r){  
this.resource = r;  
}

@Override  
public void run() {  
synchronized (resource) {  
resource.doSomething();  
resource.doLogging(“synchronized hold released”);  
}

}  
}

public class LockExample implements Runnable{

private Resource resource;  
private Lock lock;

public LockExample(Resource r){  
this.resource = r;  
this.lock = new ReentrantLock();  
}

@Override  
public void run() {  
try {  
if(lock.tryLock(10, TimeUnit.SECONDS)){  
resource.doSomething();  
}  
} catch (InterruptedException e) {  
e.printStackTrace();  
}finally{  
//release lock  
resource.doLogging(“lock released”);  
lock.unlock();  
}

}

}

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33866)

1. **Riley says**

[JANUARY 11, 2016 AT 7:51 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33864)

here’s the code i used based upon yours:

public class Resource {

public void doSomething(){  
//do some operation, DB read, write etc  
try {  
System.out.println(“resource locked by thread: ” + Thread.currentThread().getName()) ;  
Thread.sleep(4000);  
} catch (InterruptedException e) {  
e.printStackTrace();  
}  
}

public void doLogging(String str){  
//logging, no need for thread safety  
System.out.println(str) ;  
}  
}

public class LockTest {

public static void main(String[] args) throws InterruptedException {

Resource res = new Resource() ;  
LockExample ex = new LockExample(res);  
Thread t1 = new Thread(ex, “t1”);  
t1.start();  
Thread t2 = new Thread(ex, “t2”);  
t2.start();  
t1.join();  
t2.join();  
System.out.println(“completed lock example”);  
}  
}

public class SynchronizedTest {

public static void main(String[] args) throws InterruptedException {

Resource res = new Resource() ;  
SynchronizedExample ex = new SynchronizedExample(res);  
Thread t1 = new Thread(ex, “t1”);  
t1.start();  
Thread t2 = new Thread(ex, “t2”);  
t2.start();  
t1.join();  
t2.join();  
System.out.println(“completed synchronized example”);  
}  
}

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33864)

1. **jitendra says**

[DECEMBER 29, 2015 AT 10:31 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33775)

when using lock.tryLock() one has to remember that lock.unlock() needs to be done in finally only when lock.tryLock() was true i.e the local was acquired . hence something like below should be done .

public void run() {  
boolean b = false;  
if (Thread.currentThread().getName().equalsIgnoreCase(“second”)) {  
try {  
Thread.sleep(1200);  
} catch (InterruptedException e) {  
e.printStackTrace();  
}  
System.out.println(“in second thread”);  
}  
try {  
b = lock.tryLock();  
System.out.println(“b in ” + Thread.currentThread().getName() + ” thread is : ” + b);  
if(b){  
}  
} catch (Exception e) {  
e.printStackTrace();  
}finally{  
//release lock  
if (b) {  
lock.unlock();  
}  
}  
}

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33775)

1. **tELLtHEtRUTH says**

[NOVEMBER 22, 2015 AT 8:46 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33560)

Please, leave out all these fake comments.  
I think you spend more time commenting your own post that in actually writing.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33560)

* + **Melwin Dmello says**

[DECEMBER 4, 2015 AT 7:29 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33622)

Mr. “tELLtHEtRUTH” please move on to a different site. This one is obviously not for you.  
Pankaj is doing an awesome job with this site and your “TRUTH” has no relevance here!!!  
Talking about “FAKE”, you obviously are very shy to reveal your name ROFL!

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33622)

1. **himJEL says**

[SEPTEMBER 17, 2015 AT 9:00 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33180)

I fall in love in love with whatever topic I read from your website.  
Its like if I need to learn something, then first I ll search on your website and if found then I feel so much relief as if by just getting that topic on your website I learnt everything in that.   
What I want to explain is the confidence I got in your website that if any topic is available on our website then I ll master it.  
Keep up the good work and helping others.  
In my opinion you are doing a million dollar social service in this way.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33180)

* + [**Pankaj**](http://www.journaldev.com/)**says**

[SEPTEMBER 17, 2015 AT 11:02 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33183)

Thanks Himani, your appreciation means a lot.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33183)

1. **Gopinath says**

[AUGUST 23, 2015 AT 9:10 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33029)

Thanks for the wonderful tutorial on Threads:) Simple and precise, upto point tutorial so far:) Thanks a ton:)

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-33029)

1. **Wells Lee says**

[APRIL 29, 2015 AT 6:48 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-32340)

Hi Pankaj,  
this demo i get a correct result ,i was expecting the second thread should wait till the first thread execution completed but both the thread were running simultaneously.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-32340)

1. **Veranga says**

[APRIL 21, 2015 AT 3:38 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-32263)

Good Article.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-32263)

1. **farrell2k says**

[APRIL 11, 2015 AT 9:00 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-32180)

Do the locking outside of the try. You want to avoid throwing an IllegalMonitorStateException.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-32180)

1. **Lalita Kamde says**

[JANUARY 22, 2015 AT 12:24 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-31509)

Thanks! Finally understated locks

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-31509)

1. **Prakash says**

[NOVEMBER 18, 2014 AT 4:36 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-30772)

Suppose there are 10 threads name like T1, T2,… T10. T1 is in running stage. All other threads T2,…T10 are in waiting stage. When T1 finish its operation, then any one of them(T2,..T10) will get chance to start working. If all the threads have same priority then which how to select the thread to start its operation?

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-30772)

* + **praveen kumar says**

[MAY 18, 2015 AT 12:13 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-32452)

If multiple threads having same priority then Thread Schedular will decide which thread to start ,It is Varient on JVM.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-32452)

1. **shiva says**

[NOVEMBER 17, 2014 AT 11:06 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-30770)

What happens if a thread waits for certain period, say 10 secs and not able to get lock. Will it not try to lock again? I think this may inconstant behavior

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-30770)

1. **Bhagath Sagar says**

[NOVEMBER 3, 2014 AT 9:48 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-30659)

Hi ,

How this ConcurrencyLockExample work as you are not sharing the lock across the threads ?.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-30659)

1. **shailendra says**

[OCTOBER 9, 2014 AT 5:41 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-30422)

Resource resource = new Resource();  
ConcurrencyLockExample c = new ConcurrencyLockExample(resource);

Thread t = new Thread(c, “My Thread”);

Thread t1 = new Thread(c, “My Thread”);  
t.start();  
t1.start();

I used above code to Test Lock function I found lock.tryLock(10, TimeUnit.MILLISECONDS) never return false;

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-30422)

1. **J Saxton says**

[AUGUST 26, 2014 AT 4:08 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-29906)

Not checked with code, but I think your finally block actually needs to be  
  
finally {  
if( lock.isHeldByCurrentThread() {  
lock.unlock();  
}  
  
otherwise there is a risk of an IllegalMonitorStateException if an exception was thrown due to tryLock being interrupted (the InterruptedException e you’re catching)

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-29906)

1. **Suhita says**

[JUNE 17, 2014 AT 1:13 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-29263)

Thanks a lot! Finally understood Re-entrant lock!

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-29263)

1. **Bhaskar says**

[MAY 6, 2014 AT 7:29 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28789)

Thanks Pankaj for this blog. I could get a brief information about the Lock API. It’s simple and good.  
It may become more useful if there is an example for each of those cases: Condition, ReadWriteLock and ReentrantLock.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28789)

1. **Hrishi says**

[APRIL 18, 2014 AT 3:07 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28620)

Hey just wanted to know why do we use ReadLock..? when multiple threads are allowed so what is need of ReadLock…? access it without any lock.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28620)

* + **Bhaskar says**

[MAY 6, 2014 AT 7:33 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28790)

If I understand correctly, ReadWriteLock will allow multiple thread to read the Resource as long as there is no Writer Thread modifying the same Resource. Basically, it seems like, if there is some thread modifying the Resource, it may not be good idea to read either. I believe, this’s not achievable using Synchronized the moment we Synchronize the blocks of Read and Write on the same object it would not even allow to read by multiple thread at the same time.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28790)

1. **HIMANSU NAYAK says**

[APRIL 10, 2014 AT 3:31 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28514)

Hi Pankaj,  
I created a test class to run this code.

public class CocurrencyLockMain {  
public static void main(String… args) {  
Resource resource = new Resource();  
new Thread(new ConcurrencyLock(resource)).start();  
new Thread(new ConcurrencyLock(resource)).start();  
}  
}

i was expecting the second thread should wait till the first thread execution completed but both the thread were running simultaneously.

Can you please explain what is wrong in my understanding.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28514)

* + **HIMANSU NAYAK says**

[APRIL 10, 2014 AT 10:06 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28518)

Sorry i forget move the lock from instance to class reference.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28518)

1. **HIMANSU NAYAK says**

[APRIL 10, 2014 AT 12:01 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28513)

Hi Pankaj,  
Can you please also explain the different Condition of lock & Lock Fairness with an example.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28513)

1. **3biga says**

[APRIL 8, 2014 AT 9:32 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28492)

Hi, I guess you did a mistake here:  
As you can see that, I am using tryLock() method to make sure my thread waits only for definite time and if it’s not getting the lock on object, it’s just logging and exiting.

Since tryLock returns false if the lock hasn’t been acquired and doesn’t raise an exception, you should place resource.doSomething() in IF block.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28492)

* + [**Pankaj**](http://www.journaldev.com/)**says**

[APRIL 8, 2014 AT 4:00 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28496)

Thanks for the observation, you are right. I have updated the program.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28496)

1. **Vijay Yadav says**

[FEBRUARY 27, 2014 AT 2:29 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28209)

Good

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28209)

1. **amit nandode says**

[FEBRUARY 26, 2014 AT 11:19 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28201)

Hi, I like to thank you for providing such a good explaination with simple example.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28201)

1. **Dharmendra says**

[FEBRUARY 24, 2014 AT 5:37 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28182)

Hello Friend,

Just let me know if I am wrong-

I thing lock.tryLock(10, TimeUnit.SECONDS); in run method i trying to access lock on ConcurrencyLockExample object instead of Resource object.

If you want to get lock on Resource object than Lock lock= new Reentrant();  
should be added in Resource Class;

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28182)

* + [**Pankaj**](http://www.journaldev.com/)**says**

[FEBRUARY 24, 2014 AT 8:09 AM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28187)

Hi Dharmendra,

We don’t want lock on Resource object, we want lock on the code that is executing resource method for thread safety. So lock.tryLock() will get the lock on the object and then execute the doSomething() method. While executing if some other thread is also asking for the lock, that thread will wait until the first thread is finished executing the doSomething() method and has released the lock.

In real life, we don’t always has control on the Resource objects and it’s the responsibility of the caller to look after thread safety. If the threads are not involved, then having locking mechanism in the Resource classes will be an extra overhead to the system.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-28187)

* + - **Ken says**

[SEPTEMBER 20, 2014 AT 12:02 PM](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-30211)

Thanks very much for this great article. However, I am not clear on your comment.

“So lock.tryLock() will get the lock on the object and then execute the doSomething() method.”

I think tryLock will acquire lock on runnable instance itself rather than resource. I somehow don’t see any point where an object of runnable would need to acquire lock on a part (code block) of itself.

[Reply](http://www.journaldev.com/2377/java-lock-example-reentrantlock#comment-30211)