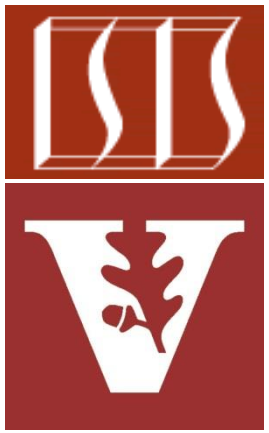
Java ReentrantLock (Part 3)

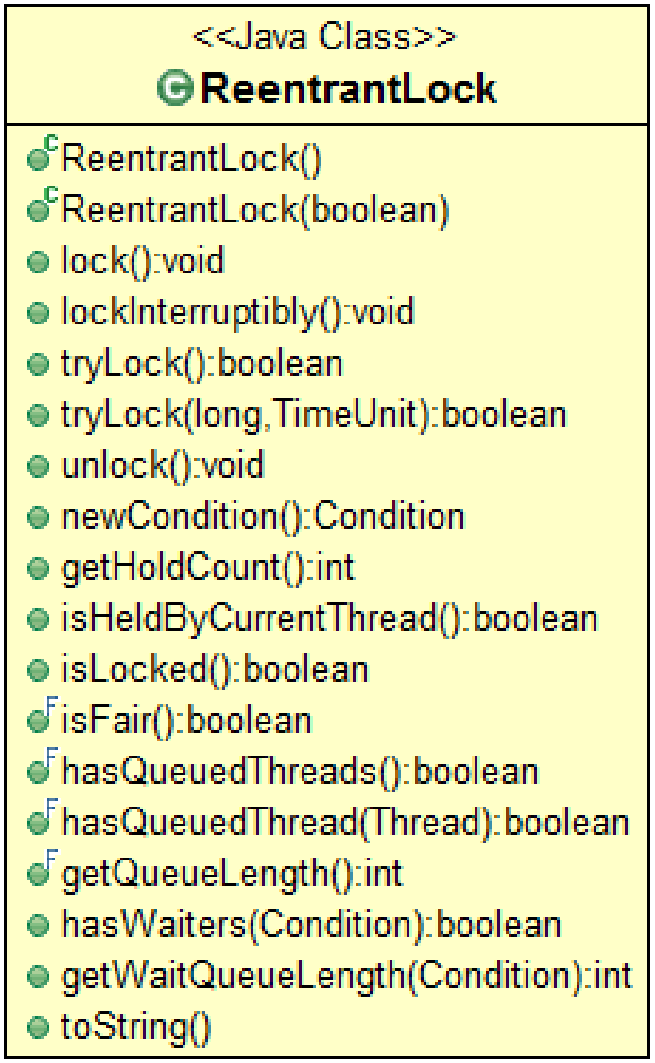


Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Part of the Lesson

- Understand how the concept of mutual exclusion in concurrent programs
- Recognize how Java ReentrantLock provides mutual exclusion to concurrent programs
- Know the key methods defined by the Java ReentrantLock class



Overview of Key ReentrantLock Methods

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public void lock() { sync.lock(); }

  public void lockInterruptibly()
      throws InterruptedException {
    sync.acquireInterruptibly(1);
  }

  public boolean tryLock() {
    return sync.nonfairTryAcquire(1);
  }

  public void unlock() {
    sync.release(1);
  }
  ...
```

See src/share/classes/java/util/concurrent/locks/ReentrantLock.java

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock

These methods are defined in the Java Lock interface

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
...
  public void lock() { sync.lock(); }

  public void lockInterruptibly()
      throws InterruptedException {
    sync.acquireInterruptibly(1);
  }

  public boolean tryLock() {
    return sync.nonfairTryAcquire(1);
  }

  public void unlock() {
    sync.release(1);
  }
  ...
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/locks/Lock.html

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock

These methods simply forward to their implementor methods, which largely inherit from AbstractQueuedSynchronizer

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
...
public void lock() { sync.lock(); }

public void lockInterruptibly()
      throws InterruptedException {
    sync.acquireInterruptibly(1);
}

public boolean tryLock() {
  return sync.nonfairTryAcquire(1);
}

public void unlock() {
  sync.release(1);
}
...
```

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/locks/AbstractQueuedSynchronizer.html

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public void lock() {
    sync.lock();
  }
  ...
```

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available
    - If lock isn't available its implementation depends on the "fairness" policy

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public void lock() {
    sync.lock();
  }
  ...
```

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available
    - If lock isn't available its implementation depends on the "fairness" policy
      - Non-fair implementations are optimized in hardware

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public void lock() {
    sync.lock();
  }
  ...
```

See en.wikipedia.org/wiki/Spinlock

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available
    - If lock isn't available its implementation depends on the "fairness" policy
      - Non-fair implementations are optimized in hardware
      - Fair implementations "park" themselves on a wait queue in FIFO order

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
...
public void lock() {
  sync.lock();
}
...
```

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available
    - If lock isn't available its implementation depends on the "fairness" policy
  - lock() is not interruptible

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public void lock() {
    sync.lock();
  }
  ...
```

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available
  - lockInterruptibly() acquires lock unless interrupted


```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public void lockInterruptibly()
      throws InterruptedException {
    sync.acquireInterruptibly(1);
  }
  ...
```

See lesson on "*Managing the Java Thread Lifecycle*"

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available
  - lockInterruptibly() acquires lock unless interrupted
  - tryLock() acquires lock only if it's not held by another thread at invocation time

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public boolean tryLock() {
    sync.nonfairTryAcquire(1);
  }
  ...
```

Untimed tryLock() doesn't honor fairness setting & can "barge"

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available
  - lockInterruptibly() acquires lock unless interrupted
  - tryLock() acquires lock only if it's not held by another thread at invocation time
  - unlock() attempts to release the lock

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public void unlock() {
    sync.release(1);
  }
  ...
```

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available
  - lockInterruptibly() acquires lock unless interrupted
  - tryLock() acquires lock only if it's not held by another thread at invocation time
  - unlock() attempts to release the lock

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public void unlock() {
    sync.release(1);
  }
  ...
```

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available
  - lockInterruptibly() acquires lock unless interrupted
  - tryLock() acquires lock only if it's not held by another thread at invocation time
  - unlock() attempts to release the lock
    - IllegalMonitorStateException is thrown if calling thread doesn't hold lock

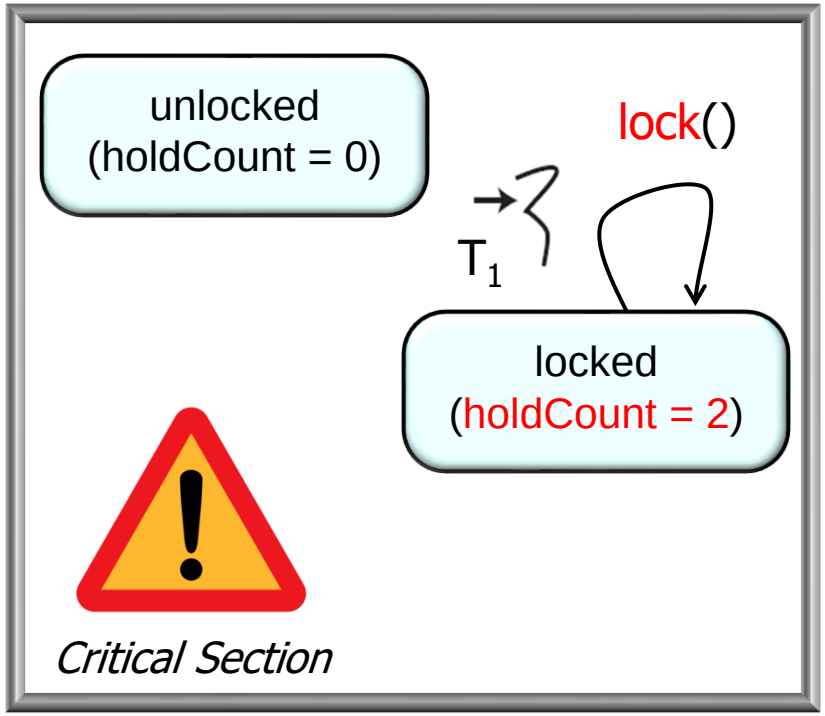
```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public void unlock() {
    sync.release(1);
  }
  ...
```

i.e., a ReentrantLock is "fully bracketed"!

Overview of Key ReentrantLock Methods

- It key methods acquire & release the lock
  - lock() acquires the lock if it's available
  - lockInterruptibly() acquires lock unless interrupted
  - tryLock() acquires lock only if it's not held by another thread at invocation time
- unlock() attempts to release the lock
  - IllegalMonitorStateException is thrown if calling thread doesn't hold lock
  - If hold count > 1 then lock is not released

```
public class ReentrantLock
             implements Lock,
             java.io.Serializable {
  ...
  public void unlock() {
    sync.release(1);
  }
  ...
```



See en.wikipedia.org/wiki/Reentrant_mutex

Overview of Other ReentrantLock Methods

Overview of Other ReentrantLock Methods

- There are many other ReentrantLock methods

boolean	tryLock(long timeout, TimeUnit unit) – Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted
boolean	isFair() – Returns true if this lock has fairness set true
boolean	isLocked() – Queries if this lock is held by any thread
Condition	newCondition() – Returns a Condition instance for use with this Lock instance
…	…

These methods go above & beyond what's available from Java's synchronized statements/methods

Overview of Other ReentrantLock Methods

- There are many other ReentrantLock methods

boolean	tryLock(long timeout, TimeUnit unit) – Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted
boolean	isFair() – Returns true if this lock has fairness set true
boolean	isLocked() – Queries if this lock is held by any thread
Condition	newCondition() – Returns a Condition instance for use with this Lock instance
…	…

Timed tryLock() *does* honor fairness setting & can't "barge"

Overview of Other ReentrantLock Methods

- There are many other ReentrantLock methods

boolean	tryLock(long timeout, TimeUnit unit) – Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted
boolean	isFair() – Returns true if this lock has fairness set true
boolean	isLocked() – Queries if this lock is held by any thread
Condition	newCondition() – Returns a Condition instance for use with this Lock instance
...	...

Overview of Other ReentrantLock Methods

- There are many other ReentrantLock methods

boolean	tryLock(long timeout, TimeUnit unit) – Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted
boolean	isFair() – Returns true if this lock has fairness set true
boolean	isLocked() – Queries if this lock is held by any thread
Condition	newCondition() – Returns a Condition instance for use with this Lock instance
…	…

Not very useful due to non-determinism of concurrency..

Overview of Other ReentrantLock Methods

- There are many other ReentrantLock methods

boolean	tryLock(long timeout, TimeUnit unit) – Acquires the lock if it is not held by another thread within the given waiting time and the current thread has not been interrupted
boolean	isFair() – Returns true if this lock has fairness set true
boolean	isLocked() – Queries if this lock is held by any thread
Condition	newCondition() – Returns a Condition instance for use with this Lock instance
…	…

See upcoming lesson on "*Java ConditionObject*"

End of Java ReentrantLock (Part 3)