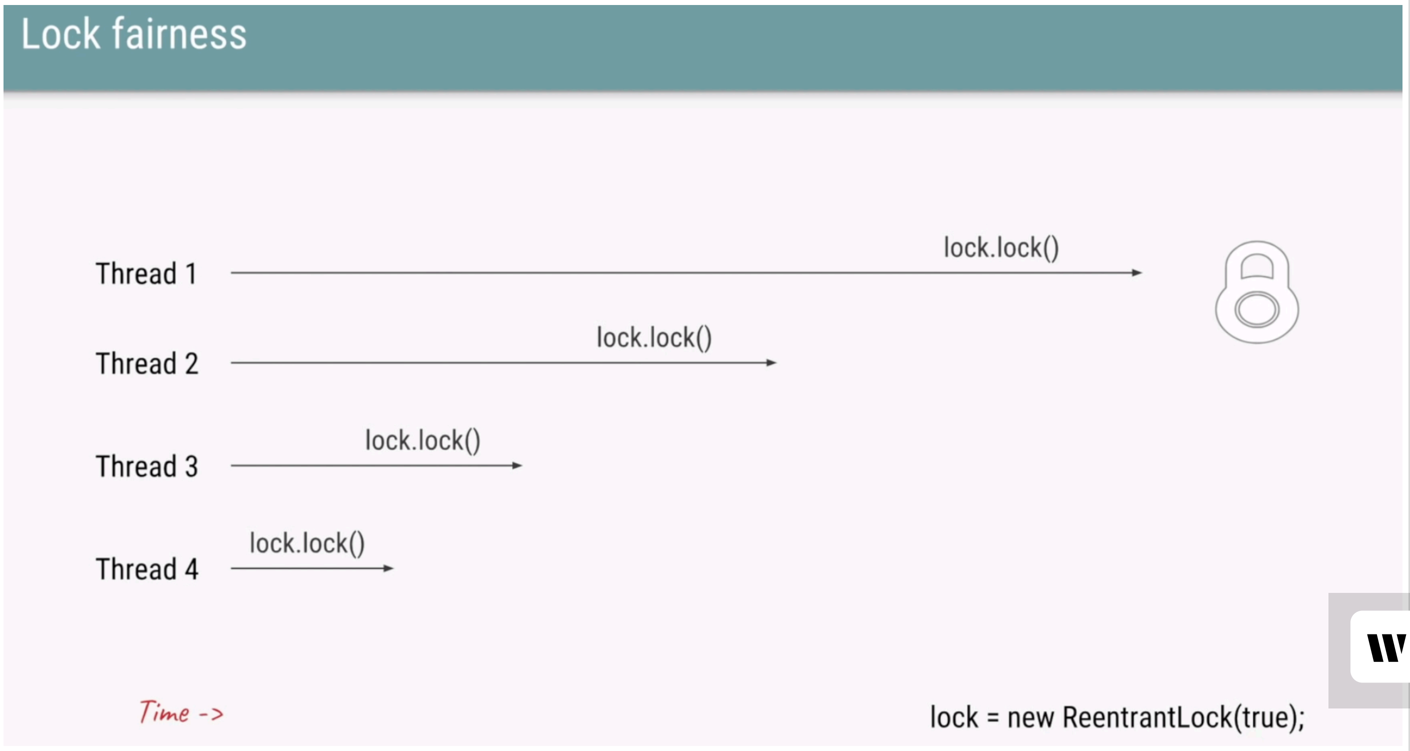
Multithreading 8 : ReentrantLock Part 2:

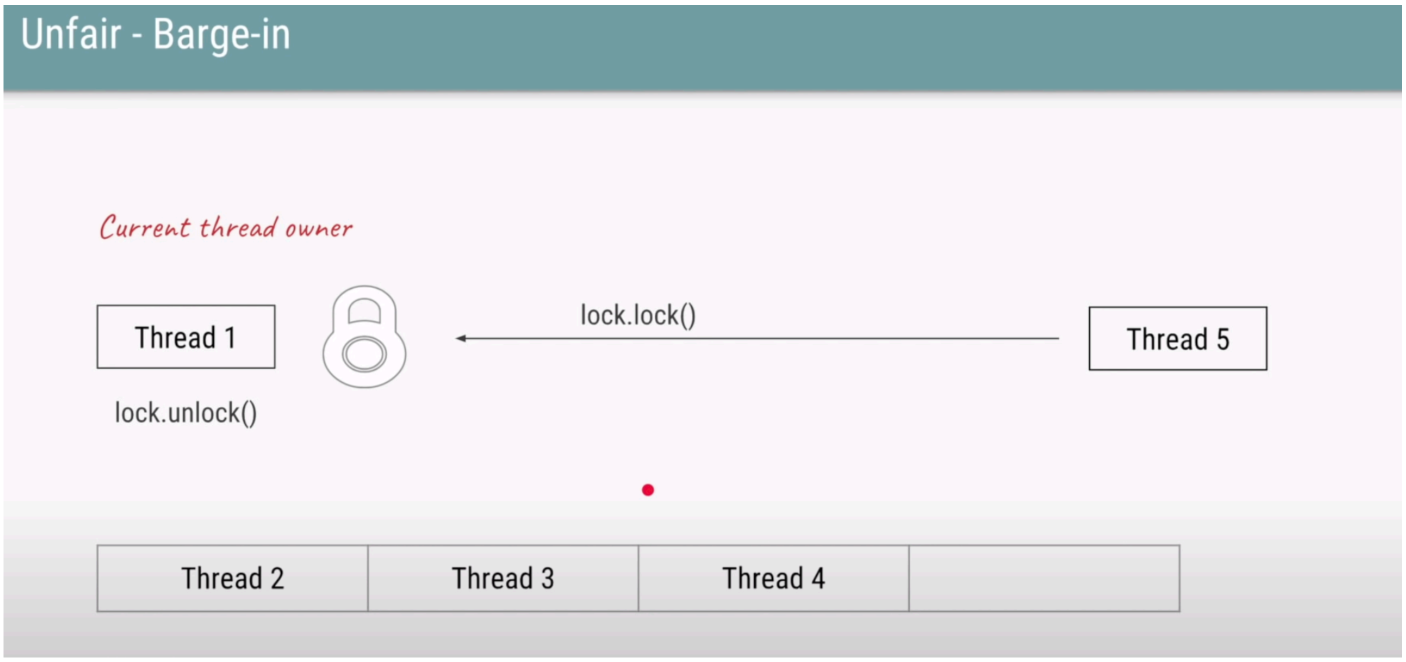


Lock fairness

lock = new ReentrantLock(true);

According to fairness : In the queue longest waiting thread will get a change to take the lock

Unfair logic:

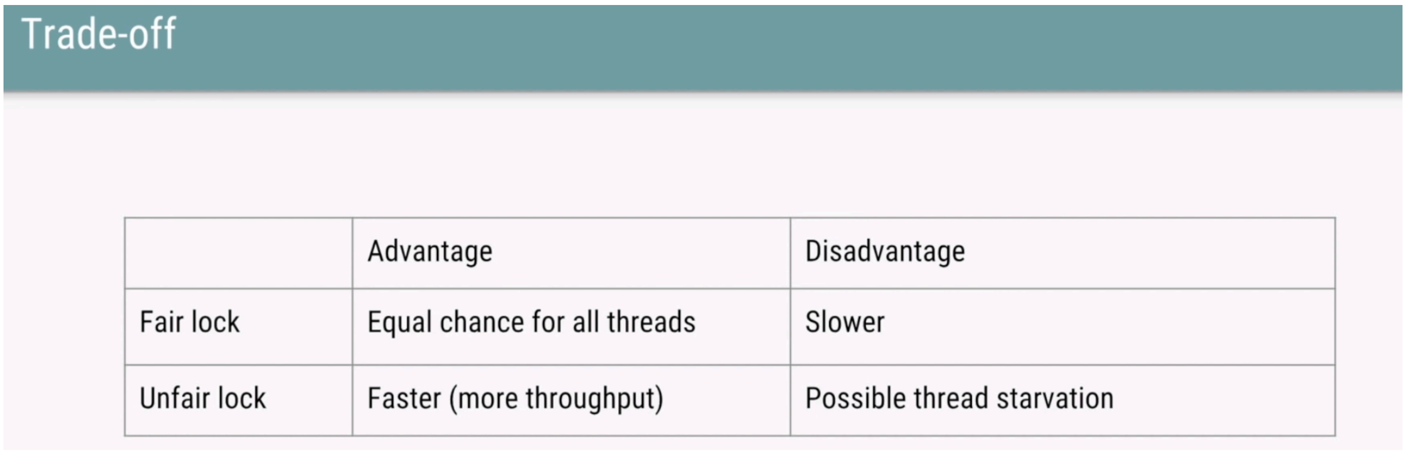


Unfair - Barge-in

Wait Queue

All threads waiting for the lock

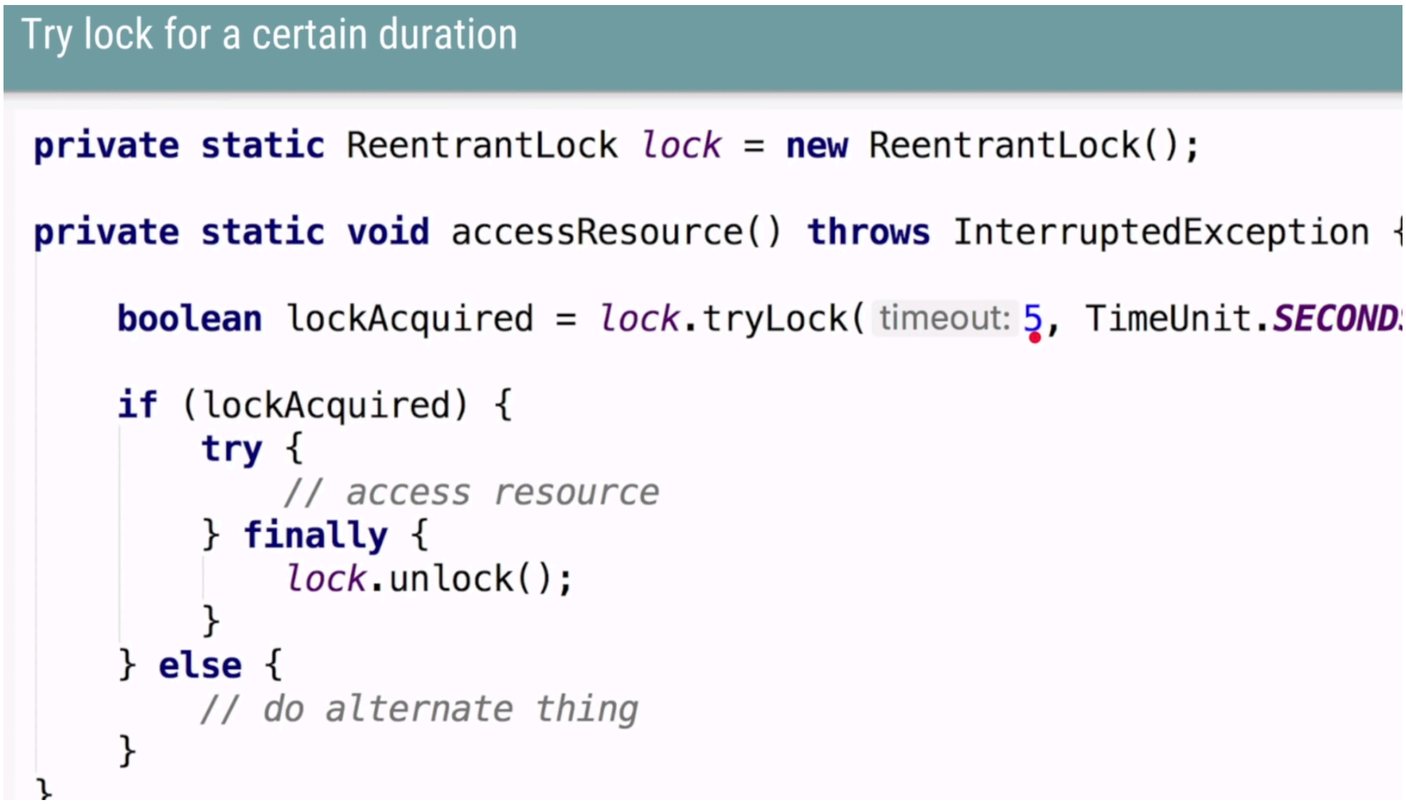
Trade off b/w Fairness or unFair lock pattern:



Trade-off

	Advantage	Disadvantage
Fair lock	Equal chance for all threads	Slower
Unfair lock	Faster (more throughput)	Possible thread starvation

Some other methods like trylock for more functionalities:



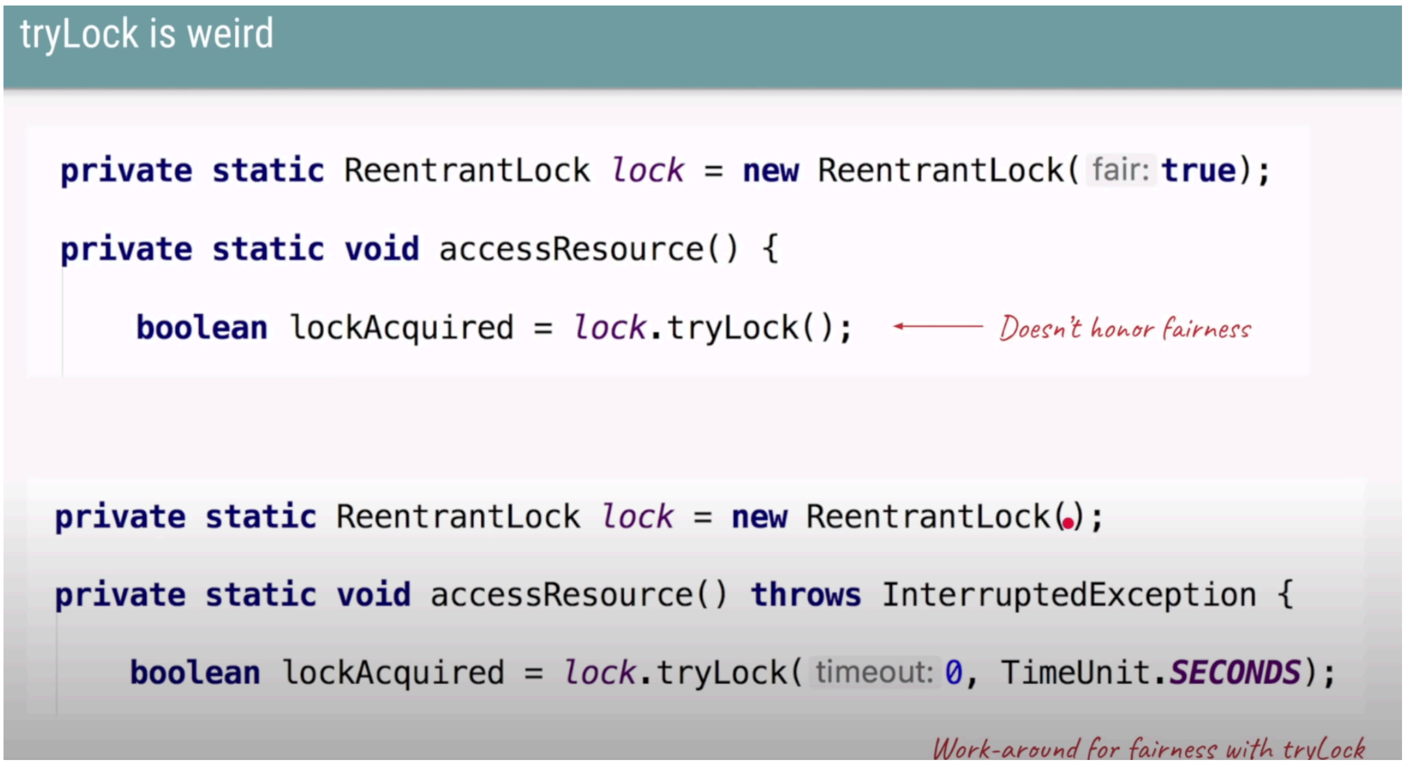
Try lock for a certain duration

```java
private static ReentrantLock lock = new ReentrantLock();

private static void accessResource() throws InterruptedException {

    boolean lockAcquired = lock.tryLock( timeout: 5, TimeUnit.SECOND

    if (lockAcquired) {
        try {
            // access resource
        } finally {
            lock.unlock();
        }
    } else {
        // do alternate thing
    }
}
```



tryLock is weird

```
private static ReentrantLock lock = new ReentrantLock( fair: true);

private static void accessResource() {

    boolean lockAcquired = lock.tryLock();
```

```
private static ReentrantLock lock = new ReentrantLock();

private static void accessResource() throws InterruptedException {

    boolean lockAcquired = lock.tryLock( timeout: 0, TimeUnit.SECONDS);
```

For this timeout is given as 0 so that it won't be unfair.