## What is ReentrantLock in Java

`ReentrantLock` is mutual exclusive lock, similar to implicit locking provided by synchronized keyword in Java, with **extended feature like fairness, which can be used to provide lock to longest waiting thread**. Lock is acquired by `lock()` method and held by Thread until a call to `unlock()` method. Fairness parameter is provided while creating instance of `ReentrantLock` in constructor. ReentrantLock provides same visibility and ordering guarantee, provided by implicitly locking, which means, `unlock()` happens before another thread get `lock()`.

## Difference between ReentrantLock and synchronized keyword in Java

Though `ReentrantLock` provides same visibility and orderings guaranteed as implicit lock, acquired by synchronized keyword in Java, it provides more functionality and differ in certain aspect. **main difference between synchronized and ReentrantLock** is ability to trying for lock with or without timeout. Thread doesn't need to block infinitely, which was the case with synchronized.

Let’s see few more differences between synchronized and Lock in Java.

1) Another significant difference between `ReentrantLock` and synchronized keyword is **fairness**. `synchronized` keyword doesn't support fairness. Any thread can acquire lock once released, no preference can be specified, on the other hand you can make `ReentrantLock` fair by specifying `fairness` property, while creating instance of `ReentrantLock`. Fairness property provides lock to longest waiting thread, in case of conflict.

2) Second difference between synchronized and Reentrant lock is **`tryLock()`** method. ReentrantLock provides convenient `tryLock()` method, which acquires lock only if its available or not held by any other thread. This reduce blocking of thread waiting for lock in Java application.

Similarly `tryLock()` with timeout can be used to timeout if lock is not available in certain time period.

3) ReentrantLock also provides convenient method to get List of all threads waiting for lock.

In short, Lock interface adds lot of power and flexibility and allows some control over lock acquisition process, which can be influenced to write highly scalable systems in Java.

## Disadvantages of ReentrantLock in Java

Major drawback of using ReentrantLock in Java is , wrapping method body inside try-finally block, which makes code unreadable and error-prone. Another disadvantage is that, now programmer is responsible for acquiring and releasing lock, which is a power but also opens gate for new subtle bugs, when programmer forget to release the lock in finally block.