可重入锁指的是在一个线程中可以多次获取同一把锁

比如：一个线程在执行一个带锁的方法，该方法中又调用了另一个需要相同锁的方法，则该线程可以直接执行调用的方法，而无需重新获得锁；

在java中ReentrantLock和synchronized都是可重入锁

代码示例

```
/**
 * 可重入锁
 */
public class Reentrant {
    Lock lock = new ReentrantLock();

    public synchronized void method01() {
        System.out.println(Thread.currentThread().getName() + "执行了method01");
        method02();
    }

    public synchronized void method02() {
        System.out.println(Thread.currentThread().getName() + "执行了method02");
    }

    public void method03() {
        try {
            lock.lock();
            System.out.println(Thread.currentThread().getName() + "执行了method03");
            method04();
        } finally {
            lock.unlock();
        }
    }

    public void method04() {
        try {
            lock.lock();
            System.out.println(Thread.currentThread().getName() + "执行了method04");
        } finally {
            lock.unlock();
        }
    }

    public static void main(String[] args) {
        Reentrant reentrant = new Reentrant();
        new Thread(new Runnable() {
            @Override
```

```
            public void run() {
                reentrant.method01();
            }
        }, "t1").start();

        new Thread(new Runnable() {
            @Override
            public void run() {
                reentrant.method03();
            }
        }, "t2").start();
    }
}
```

运行结果：

```
t1执行了method01
t1执行了method02
t2执行了method03
t2执行了method04
```

从结果可以分析得出,t1 线程可以先进入 method01，它拥有了锁，他线程可以继续进入 它已经拥有锁的同步方法 method02,同理，线程 t2 也是一样