第二次作业 ：多线程程序 产生死锁

小组成员：修玥 石雨宁 王超 周学士

分工如下： 代码编写：修玥

文字编辑：石雨宁

运行：修玥 石雨宁 王超 周学士（由于结果一样 只附一张照片）

死锁：是指两个或两个以上的线程在执行过程中，由于竞争资源或者由于彼此通信而造成的一种阻塞的现象。

思考： （1）当一个线程持有一个锁的时候，其他尝试获取这个锁的线程都会阻塞。

（2）那么如果，线程A持有了锁lock1，并且尝试获取锁lock2，线程B持有了锁 lock2，并且尝试获取锁lock1。

（3）那么这两个线程将会永远阻塞下去，这就是最简单的死锁。

下面给出简单的死锁代码实现：

**import java.util.concurrent.locks.ReentrantLock;**

**class Task1 implements Runnable{**

**private ReentrantLock lock1;**

**private ReentrantLock lock2;**

**public Task1(ReentrantLock lock1,ReentrantLock lock2){**

**this.lock1 = lock1;**

**this.lock2 = lock2;**

**}**

**@Override**

**public void run() {**

**lock1.lock();**

**try{**

**System.out.println("Task1 get lock1,start sleep...");**

**try {**

**Thread.sleep(500);**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**lock2.lock();**

**System.out.println("task1:");**

**}finally {**

**lock2.unlock();**

**lock1.unlock();**

**}**

**}**

**}**

**class Task2 implements Runnable{**

**private ReentrantLock lock1;**

**private ReentrantLock lock2;**

**public Task2(ReentrantLock lock1,ReentrantLock lock2){**

**this.lock1 = lock1;**

**this.lock2 = lock2;**

**}**

**@Override**

**public void run() {**

**lock2.lock();**

**try{**

**System.out.println("Task2 get lock2,start sleep...");**

**try {**

**Thread.sleep(500);**

**} catch (InterruptedException e) {**

**e.printStackTrace();**

**}**

**lock1.lock();**

**System.out.println("task1: ");**

**}finally {**

**lock1.unlock();**

**lock2.unlock();**

**}**

**}**

**}**

**public class DeadLockTest {**

**private static ReentrantLock lock1 = new ReentrantLock();**

**private static ReentrantLock lock2 = new ReentrantLock();**

**public static  void main(String[] args){**

**Thread task1 = new Thread(new Task1(lock1,lock2));**

**Thread task2 = new Thread(new Task2(lock1,lock2));**

**task1.start();**

**task2.start();**

**}**

**}**

上面代码执行后控制台输出：

