**作业：多线程程序，产生死锁**

**小组成员：高钰淼 邹澳航 田苗苗**

**分工如下：代码编写：高钰淼**

**文字编辑：邹澳航**

**运行：高钰淼 邹澳航 田苗苗**

**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，若无外力作用，它们都将无法推进下去。**

**思考：**

**（1）创建两个线程，分别持有两个lock，比如是lock1和lock2**

**（2）线程1和线程2在run方法里分别先获取lock1和lock2的锁，之后分别sleep一段时间（避免一上来一个线程就获得了两个锁）,然后线程1和线程2分别尝试去获取lock2和lock1**

**（3）此时因为锁已经被对方占有，并且对方还在等待自己持有的锁，这个时候死锁就发生了。代码如下：**

**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));**

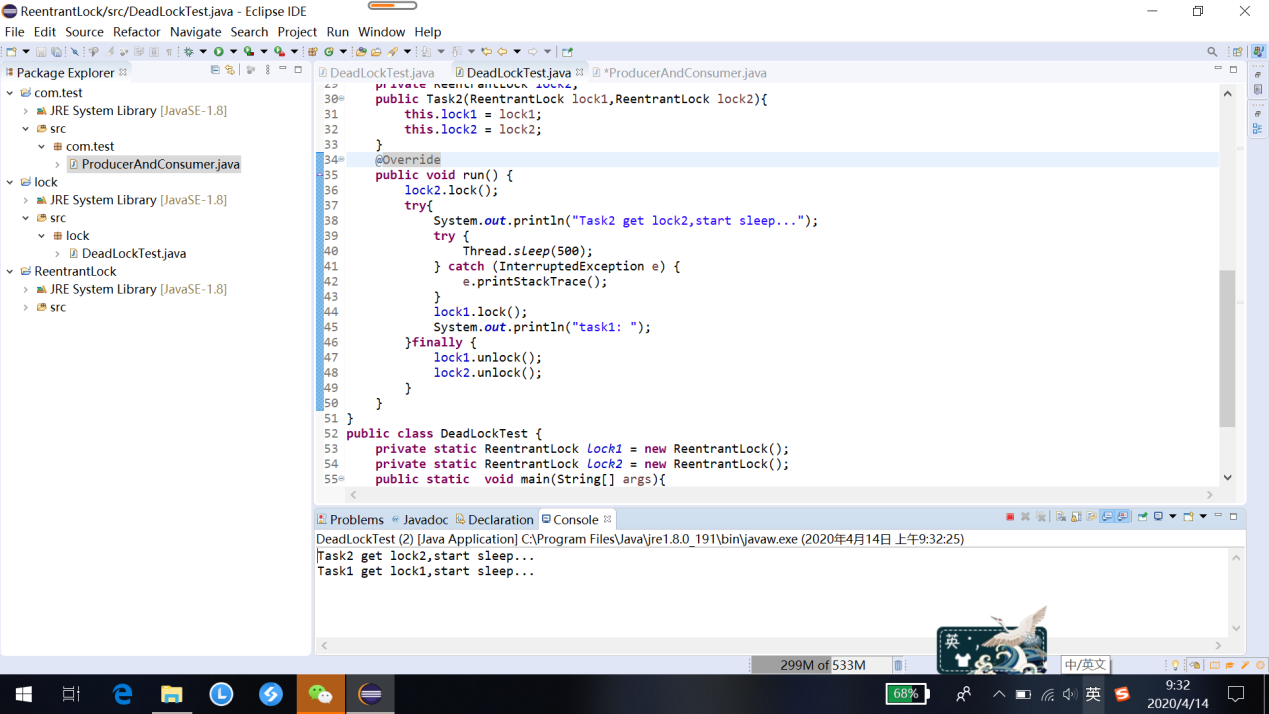
**task1.start();**

**task2.start();**

**}**

**}**

**运行结果如图**

****