多线程死锁案例：

package test;

/\*\*

\* 内容：创建两个静态对象和两个线程，第一个线程给第一个对象加锁，第二个线程给第二个对象加锁，当第一个线程要获取第二个对象的锁时

\* 发现第二个对象的钥匙正在被第二个线程持有，所以第一个线程要的等待第二个线程释放该锁，而第二个线程要释放该锁就有执行该synchronize块，

\* 所以要去拿第一个对象的锁，而第一个对象的锁，正在被第一个线程占有，于是两个线程互相等待，形成死锁。

\* @author Administrator

\*

\*/

public class DeadLock {

private static Object lock1=new Object();

private static Object lock2=new Object();

public static void main(String[] args) {

// TODO Auto-generated method stub

new Thread(){

public void run(){

synchronized (lock1) {

System.out.println("thread1 get lock1;");

try{

Thread.sleep(100);

}catch (Exception e) {

// TODO: handle exception

}

synchronized (lock2) {

System.out.println("thread1 get lock2;");

}

}

System.out.println("thread1 end...");

}

}.start();

new Thread(){

public void run(){

synchronized (lock2) {

System.out.println("thread2 get lock2;");

try{

Thread.sleep(100);

}catch (Exception e) {

// TODO: handle exception

}

synchronized (lock1) {

System.out.println("thread2 get lock1;");

}

}

System.out.println("thread2 end");

}

}.start();

}

}