```
package com.sxt.syn;
/**
 * 死锁：过多的同步可能造成相互不释放资源
 * 从而相互等待，一般发生于同步中持有多个对象的锁
 *
 * 避免：不要在同一个代码块中，同时持有多个对象的锁
 * @author DELL
 *
 */

public class DeadLock13 {
   public static void main(String[] args) {
           Markup g1=new Markup(1, "张柏芝");
           Markup g2=new Markup(2, "王菲");
           g1.start();
           g2.start();

}
}

//口红
class Lipstick{

}

//镜子
class Mirror{



}

//化妆
class Markup extends Thread{
        static Lipstick lipstick=new Lipstick();
        static Mirror mirror=new Mirror();
```

```
//选择
int choice;
//名字
String girl;
public Markup(int choice,String girl) {
        this.choice=choice;
        this.girl=girl;
}
@Override
public void run() {
        //化妆
        markup();
}

//相互持有对方的对象锁-->可能造成死锁
public void markup() {
        if(choice==0) {
                synchronized(lipstick) {//获得口红的锁
                    System.err.println(this.girl+"涂口红");
                    //1秒后想获得镜子的锁
                    try {
                            Thread.sleep(1000);
                    } catch (InterruptedException e) {
                            e.printStackTrace();
                    }
                synchronized(mirror) {
                    System.err.println(this.girl+"照镜子");
                    }
                }
        }else {
                synchronized(mirror) {//获得镜子的锁
                        System.err.println(this.girl+"照镜子");
                //2秒后想拥有口红的锁
                try {
                        sleep(2000);
```

```
                } catch (InterruptedException e) {
                        e.printStackTrace();
                }
                synchronized(lipstick) {
                        System.err.println(this.girl+"涂口红");
                }
                }
        }

    }



}
```