18 线程_安全问题与死锁

JAVAEE高级

一:线程安全问题

> 1、多线程并发操作同一数据时, 就有可能出现线程安全问题
>
> 2、使用同步技术可以解决这种问题, 把操作数据的代码进行同步, 不要多个线程一起操作

二：购买票案例实现

> 1、Thread实现

```
class MyThread extends Thread {
    private static int number = 100;
    static Object obj = new Object();

    public MyThread() {
        super();

    }
    public MyThread(String name) {
        super(name);
    }

    public void run() {
        while (true) {
            synchronized (obj) {
                if (number <= 0)
                    break;
                try {
                    Thread.sleep(1000);// 线程1, 2, 3睡觉
                    } catch (InterruptedException e) {
                    e.printStackTrace();
                }
```

```
                System.out.println(getName() + "第" + number-- + "号票");
            }
        }
    }
}
```

```
public static void main(String[] args) {
        MyThread t1 = new MyThread();
        MyThread t2 = new MyThread();
        MyThread t3 = new MyThread();
        t1.setName("窗口1");
        t2.setName("窗口2");
        t3.setName("窗口3");
        t1.start();
        t2.start();
        t3.start();
    }
```

2、Runnable接口

```
class MyThread implements Runnable {
    private static int number = 100;
    static Object obj = new Object();
    public void run() {
        while (true) {
            synchronized (obj) {
                if (number <= 0)
                    break;
                try {
                    Thread.sleep(1000);// 线程1, 2, 3睡觉
                    } catch (InterruptedException e) {
                    e.printStackTrace();
                }
                System.out.println(Thread.currentThread().getName() + "第" + number-- + "号票");
                }
        }
    }
}
```

```
    public static void main(String[] args) {
        MyThread t1 = new MyThread();
        new Thread(t1).start();
        new Thread(t1).start();
        new Thread(t1).start();
    }
```

三：死锁

> 1、多线程同步的时候, 如果同步代码嵌套, 使用相同锁, 就有可能出现死锁
>
> 2、不同的线程分别占用对方需要的同步资源不放弃，都在等待对方放弃自己需要的同步资源，就形成了线程的死锁

```
private static String s1 = "筷子左";
    private static String s2 = "筷子右";

    public static void main(String[] args) {
        new Thread() {
            public void run() {
                while (true) {
                    synchronized (s1) {
                        System.out.println(getName() + "获取" + s1 + "等待" + s2);
                        synchronized (s2) {
                            System.out.println(getName() + "拿到" + s2 + "开吃");
                        }
                    }
                }
            }.start();

        new Thread() {
            public void run() {
                while (true) {
                    synchronized (s2) {
                        System.out.println(getName() + "获取" + s2 + "等待" + s1);
                        synchronized (s1) {
                            System.out.println(getName() + "拿到" + s1 +
```

```
"开吃");
                            }
                        }
                    }
                }
            }.start();
        }
```