

18 线程_安全问题与死锁

JAVAAEE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：购票案例实现

1、Thread实现

```
1.  class MyThread extends Thread {
2.      private static int number = 100;
3.      static Object obj = new Object();
4.
5.      public MyThread() {
6.          super();
7.
8.      }
9.      public MyThread(String name) {
10.          super(name);
11.      }
12.
13.      public void run() {
14.          while (true) {
15.              synchronized (obj) {
16.                  if (number <= 0)
17.                      break;
18.                  try {
19.                      Thread.sleep(1000); // 线程1, 2, 3睡觉
20.                  } catch (InterruptedException e) {
21.                      e.printStackTrace();
22.                  }
```

```

23.         System.out.println(getName() + "第" + number-- + "号
    票");
24.     }
25. }
26. }
27. }

```

```

1.  public static void main(String[] args) {
2.      MyThread t1 = new MyThread();
3.      MyThread t2 = new MyThread();
4.      MyThread t3 = new MyThread();
5.      t1.setName("窗口1");
6.      t2.setName("窗口2");
7.      t3.setName("窗口3");
8.      t1.start();
9.      t2.start();
10.     t3.start();
11. }

```

2、Runnable接口

```

1.  class MyThread implements Runnable {
2.      private static int number = 100;
3.      static Object obj = new Object();
4.      public void run() {
5.          while (true) {
6.              synchronized (obj) {
7.                  if (number <= 0)
8.                      break;
9.                  try {
10.                     Thread.sleep(1000); // 线程1, 2, 3睡觉
11.                 } catch (InterruptedException e) {
12.                     e.printStackTrace();
13.                 }
14.                 System.out.println(Thread.currentThread().getName() + "
    第" + number-- + "号票");
15.             }
16.         }
17.     }
18. }

```

```

1.     public static void main(String[] args) {
2.         MyThread t1 = new MyThread();
3.         new Thread(t1).start();
4.         new Thread(t1).start();
5.         new Thread(t1).start();
6.     }

```

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

```

1.     private static String s1 = "筷子左";
2.     private static String s2 = "筷子右";
3.
4.     public static void main(String[] args) {
5.         new Thread() {
6.             public void run() {
7.                 while (true) {
8.                     synchronized (s1) {
9.                         System.out.println(getName() + "获取" + s1 + "等
10. 待" + s2);
11.                         synchronized (s2) {
12.                             System.out.println(getName() + "拿到" + s2 +
13. "开吃");
14.                         }
15.                     }
16.                 }
17.             }.start();
18.
19.             new Thread() {
20.                 public void run() {
21.                     while (true) {
22.                         synchronized (s2) {
23.                             System.out.println(getName() + "获取" + s2 + "等
24. 待" + s1);
25.                             synchronized (s1) {
26.                                 System.out.println(getName() + "拿到" + s1 +

```

```
"开吃");
```

```
25.         }
```

```
26.     }
```

```
27. }
```

```
28. }
```

```
29. }.start();
```

```
30. }
```