

17 线程_同步 && 生命周期

JAVAAEE高级

一：线程_同步描述

- 1、使用synchronized关键字加上一个锁对象来定义一段代码, 这就叫同步代码块
- 2、多个同步代码块如果使用相同的锁对象, 那么他们就是同步的

1、什么环境需要同步

- 1、当多线程并发, 有多段代码同时执行时, 我们希望某一段代码执行的过程中CPU不要切换到其他线程工作. 这时就需要同步
- 2、如果两段代码是同步的, 那么同一时间只能执行一段, 在一段代码没执行结束之前, 不会执行另外一段代码

2、使用多线程的优势

- 1、提高应用程序的响应
- 2、对图形化界面更有意义, 可增强用户体验。
- 3、提高计算机系统CPU的利用率
- 4、改善程序结构
- 5、将既长又复杂的进程分为多个线程, 独立运行, 利于理解和修改

二：同步代码块

1、没有进行同步的时候

```
1.  class Printer {
2.      public void print1() {
3.          System.out.print("print1:进");
4.          System.out.print("行");
5.          System.out.print("同");
6.          System.out.print("步");
7.          System.out.print("\r\n");
8.
9.      }
10.     public void print2() {
11.         System.out.print("print2:进");
12.         System.out.print("行");
13.         System.out.print("同");
14.         System.out.print("步");
15.         System.out.print("\r\n");
16.     }
17. }
```

```
1.  //没有同步:会出现乱序的现象
2.  public static void main(String[] args) {
3.      Printer p = new Printer();
4.      new Thread() {
5.          public void run() {
6.              while (true) {
7.                  p.print1();
8.              }
9.          }
10.     }.start();
11.
12.     new Thread() {
13.         public void run() {
14.             while (true) {
15.                 p.print2();
16.             }
17.         }
18.     }.start();
19. }
```

乱序

```
print2:进行同步
print2:进行同步
print2:进行同步
print2:进行同步
同步
print1:进行同步
print1:进行同步
print1:进行同步
print1:进行同步
```

2、同步之后没有任何的乱序现象

```

1. class Printer {
2.     public void print1() {
3.         synchronized (this) {
4.             System.out.print("print1:进");
5.             System.out.print("行");
6.             System.out.print("同");
7.             System.out.print("步");
8.             System.out.print("\r\n");
9.         }
10.
11.     }
12.     public void print2() {
13.         // 锁对象不能用匿名对象, 因为匿名对象不是同一个对象    new .....
14.         synchronized (this) {
15.             System.out.print("print2:进");
16.             System.out.print("行");
17.             System.out.print("同");
18.             System.out.print("步");
19.             System.out.print("\r\n");
20.         }
21.     }
22. }

```

```
1. public static void main(String[] args) {
2.     Printer p = new Printer();
3.     new Thread() {
4.         public void run() {
5.             while (true) {
6.                 p.print1();
7.             }

```

```

8.         }
9.     }.start();
10.
11.     new Thread() {
12.         public void run() {
13.             while (true) {
14.                 p.print2();
15.             }
16.         }
17.     }.start();
18. }

```

三：同步方法

1、使用synchronized关键字修饰一个方法, 该方法中所有的代码都是同步的

```

1.  class Printer {
2.      public static synchronized void print1() {
3.          System.out.print("print1:进");
4.          System.out.print("行");
5.          System.out.print("同");
6.          System.out.print("步");
7.          System.out.print("\r\n");
8.      }
9.
10.     public static synchronized void print2() {
11.         // 锁对象不能用匿名对象, 因为匿名对象不是同一个对象 new .....
12.         System.out.print("print2:进");
13.         System.out.print("行");
14.         System.out.print("同");
15.         System.out.print("步");
16.         System.out.print("\r\n");
17.     }
18. }

```

四：生命周期

1、Java语言使用Thread类及其子类的对象来表示线程，在它的一个完整生命周期中

通常要经历如下的五种状态：

1、新建：

- 1、当一个Thread类或其子类的对象被声明并创建时，新生的线程对象处于新建状态

2、就绪：

- 1、处于新建状态的线程被start()后，将进入线程队列等待CPU时间片，此时它已具备了运行的条件

3、运行：

- 1、当就绪的线程被调度并获得处理器资源时,便进入运行状态，run()方法定义了线程的操作和功能

4、阻塞：

- 1、在某种特殊情况下，被人为挂起或执行输入输出操作时，让出 CPU 并临时中止自己的执行，进入阻塞状态

5、死亡：

- 1、线程完成了它的全部工作或线程被提前强制性地中止

