17 线程_同步 && 生命周期

JAVAEE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

```
class Printer {
    public void print1() {
            System.out.print("print1:进");
            System.out.print("行");
            System.out.print("同");
            System.out.print("步");
            System.out.print("\r\n");

    }
    public void print2() {
            System.out.print("print2:进");
            System.out.print("行");
            System.out.print("同");
            System.out.print("步");
            System.out.print("\r\n");
    }
}
```

```
//没有同步:会出现乱序的现象
public static void main(String[] args) {
         Printer p = new Printer();
        new Thread() {
            public void run() {
                while (true) {
                    p.print1();
                }
            }
        }.start();

        new Thread() {
            public void run() {
                while (true) {
                    p.print2();
                }
            }
        }.start();
    }
```

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
```

2、同步之后没有任何的乱序现象

```
class Printer {
    public void print1() {
        synchronized (this) {
            System.out.print("print1:进");
            System.out.print("行");
            System.out.print("同");
            System.out.print("步");
            System.out.print("\r\n");
    }

    }
    public void print2() {
        // 锁对象不能用匿名对象,因为匿名对象不是同一个对象  new .....
        synchronized (this) {
            System.out.print("print2:进");
            System.out.print("行");
            System.out.print("同");
            System.out.print("步");
            System.out.print("\r\n");
    }
    }
}
```

```
    public static void main(String[] args) {
         Printer p = new Printer();
        new Thread() {
            public void run() {
                while (true) {
                    p.print1();
                }
```

```
            }
        }.start();

        new Thread() {
            public void run() {
                while (true) {
                    p.print2();
                }
            }
        }.start();
    }
```

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

```
class Printer {
    public static  synchronized void print1() {
            System.out.print("print1:进");
            System.out.print("行");
            System.out.print("同");
            System.out.print("步");
            System.out.print("\r\n");

    }
    public   static  synchronized void print2() {
        // 锁对象不能用匿名对象,因为匿名对象不是同一个对象  new .....
            System.out.print("print2:进");
            System.out.print("行");
            System.out.print("同");
            System.out.print("步");
            System.out.print("\r\n");
    }
}
```

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，在它的一个完整的生命周期中

> 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：

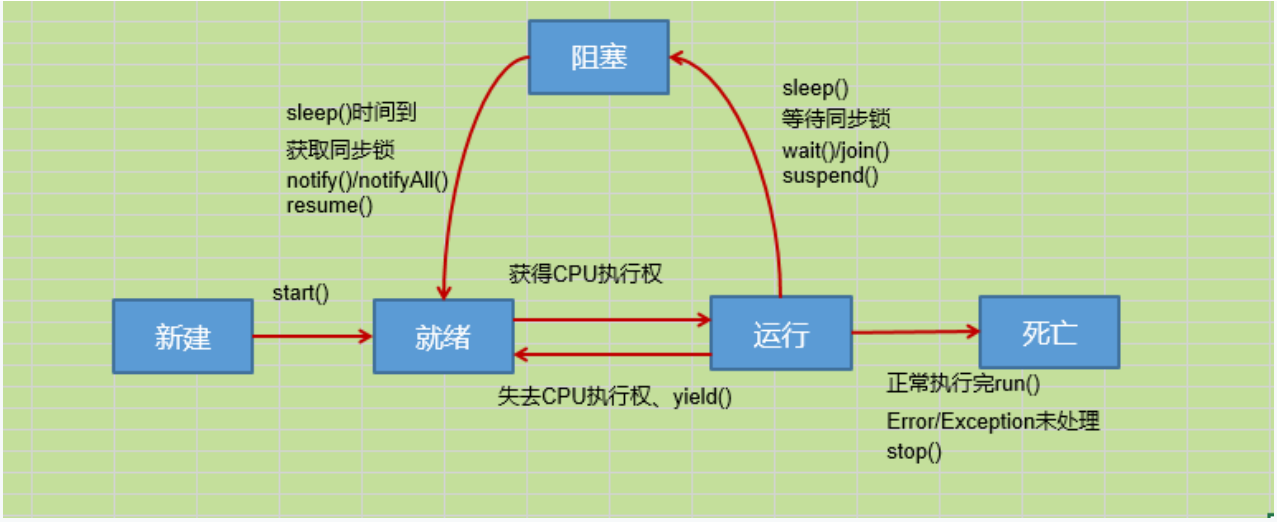
> 1、当就绪的线程被调度并获得处理器资源时,便进入运行状态， run()方法定义了线程的操作和功能

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


阻塞
sleep()时间到
获取同步锁
notify()/notifyAll()
resume()
sleep()
等待同步锁
wait()/join()
suspend()
获得CPU执行权
start()
新建
就绪
运行
死亡
失去CPU执行权、yield()
正常执行完run()
Error/Exception未处理
stop()