Java线程

1. 线程的创建

线程的创建可以分为两种方式:A）继承Tread类；B）实现Runnable接口

两种创建线程的方式区别和联系主要有哪些?

1）、Java单继承机制，限制了Thread类的使用；然后可以通过实现Runnable接口实现多线程，同时也可继承其他类来实现其他功能；

2）、Runable可以使用多线程去处理同一资源，同时也增加了程序的健壮性，相同的代码可以被多个线程共享，这是Thread类所不具备的;

如买票的经典例子:

|  |
| --- |
| **public** **class** Test {  **public** **static** **void** main(String[] args) {  ThreadTest thread1 = **new** Test.ThreadTest();  ThreadTest thread2 = **new** Test.ThreadTest();  thread1.start();;  thread2.start();  RunableTest runable = **new** Test.RunableTest();  **new** Thread(runable).start();;  **new** Thread(runable).start();;  }  **static** **class** ThreadTest **extends** Thread{  **private** **int** ticket = 10;  @Override  **public** **void** run() {  **for** (**int** i = 0; i < 20; i++) {  **if**(**this**.ticket > 0){  System.***out***.println("[" + Thread.*currentThread*().getName() + "] -- " + **this**.ticket--);  }  }  }  }  **static** **class** RunableTest **implements** Runnable {  **private** **int** ticket = 10;  **public** **void** run() {  **for** (**int** i = 0; i < 20; i++) {  **if**(**this**.ticket > 0){  System.***out***.println("[" + Thread.*currentThread*().getName() + "] -- " + **this**.ticket--);  }  }  }  }  } |

PS:

1、JDK中Thread类也是Runnable接口的子类

2、Thread的run()和start()的关系(JDK源码可知)：

start()方法使用native关键字修饰，该关键字表示调用操作系统的底层函数(JNI);

Thread的start()一旦被调用，JVM则会去调用run()方法;

1. 线程的同步

线程同步的两个特性：可见性和有序性

简单来说，多线程的交互方式往往是通过共享变量的方式来实现。如多线程共享内存中的一个对象，如果某一个线程修改了该对象的某个属性值，对于其他线程而言是可以看到被修改的属性值，即:可见性；再如多线程共享内存中的一个对象，如果某一个线程想修改该对象的某个属性值，而同时另外一个线程也想修改这个属性值，为了避免修改冲突，一定要保证修改的先后顺序，即：有序性。

Java线程同步的实现：volatile、synchronized、Lock

1. volatile

1）、确切的说该关键字并不是真正意义上的线程同步；它本质是告诉jvm当前变量在寄存器中的值是不确定的,需要从主存中读取；

2）、只用作用于变量级别；

1. synchronized

1）、synchronized则是锁定当前变量,只有当前线程可以访问该变量,其他线程被阻塞住；

2）、可作用于变量、方法、块；

1. Lock（JDK1.5以上的锁接口）

1）、ReentrantLock 类实现了 Lock ，它拥有与 synchronized 相同的并发性和内存语义，但是添加了类似锁投票、定时锁等候和可中断锁等候的一些特性。此外，它还提供了在激烈争用情况下更佳的性能。（即：当许多线程都想访问共享资源时，JVM 可以花更少的时候来调度线程，把更多时间用在执行线程上。）

2）、Lock 和 synchronized 有一点明显的区别：Lock必须在finally块中释放。否则，如果受保护的代码将抛出异常，锁就有可能永远得不到释放，This is very terrible!

1. 描述一下锁的概念：

每个锁都有两个队列，一个是就绪队列，一个是阻塞队列，就绪队列存储了将要获得锁的线程，阻塞队列存储了被阻塞的线程，当一个线程被唤醒(notify)后，才会进入到就绪队列，等待CPU的调度，反之，当一个线程被wait后，就会进入阻塞队列，等待下一次被唤醒，一个线程执行互斥代码过程如下：

1）. 获得同步锁；

2）. 清空工作内存；

3）. 从主内存拷贝对象副本到工作内存；

4）. 执行代码(计算或者输出等)；

5）. 刷新主内存数据；

6）. 释放同步锁。

Java线程同步的实现

1. volatile(略)
2. synchronized
3. Lock