# Android基本功三Java多线程

搜索关键字：

线程

多线程

难点：

线程的同步

## 1. 章节简介

在学习前先介绍一下什么是线程。

线程是进程中的一个实体，被系统独立调度和分派的基本单位。一开始的程序都是用单线程的形式来完成程序的运行，即一个程序从头到尾只有一条执行线索，不能同时执行多个任务。这样会使得系统的使用效率大打折扣。因此，多线程的使用将会大大提高系统资源的利用率。

多线程就是使得程序可以同时运行多个任务，共享同一进程中相同的资源和内存。简单的说多线程，类似我们平时边看电视同时还可以跟朋友聊天。

Java提供了对多线程的支持，其中线程的创建有两种方法，通过使用Thread类和实现runnable接口都可轻松实现多线程，但同时多线程的使用要非常严谨，在程序中非法使用多线程不但会使程序变慢，而且还会使得程序崩溃。同时每个线程有它自己的生命周期、优先级、基本控制。

在Android 4.0及以上版本，主线程只负责处理UI的更新，而禁止在主线程中访问网络，如果在访问网络的需求下，必需要新建一条线程去访问网络，得到需要的结果后再在主线程中更新UI。所以，学好Java多线程对以后Android的学习时很有好处。

## 2案例的设计和实现

### 2.1 两种创建线程的方法

1、简述

方式一：通过继承Thread 类，可以从java.land.Thread类派生一个新的线程类，通过重载它的run()方法，run()方法决定了该线程所做的工作。然后创建子类的对象，并通过start方法启动线程。

方式二：用类来实现Runnable接口，而这个类实例将用一个线程来调用启动。Runnable接口只有一个run()方法，其作用和Thread类的run()方法相同。

2、实例与解析

实例1，通过方式一的继承Thread类派生出一个新的线程类PongPing1。

实现方法如图4C-1所示。

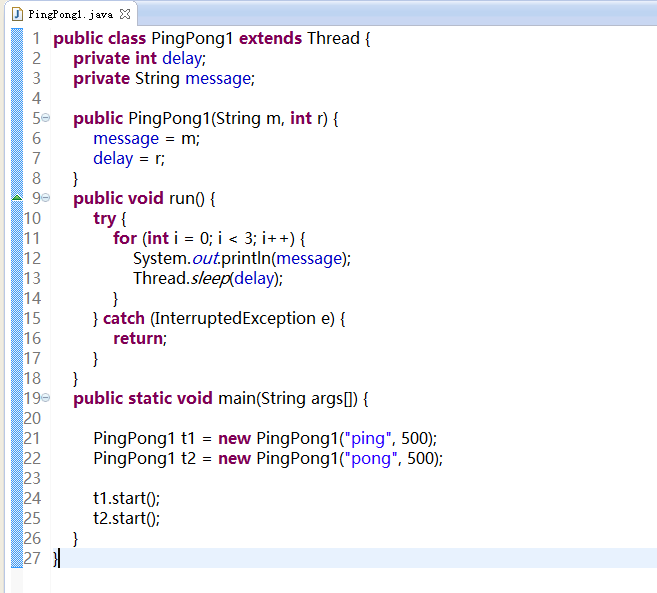


图4C-1 继承Thread类创建线程

该实例中先创建一个PingPong1类继承父类Thread类，然后重构了一个构造函数，通过构造函数初始化delay和message变量，然后重写了run()方法定义线程体，那么当PingPong1类对象t1，t2调用start方法的时候就会找到run()方法执行。

执行结果，如图4C-2所示。

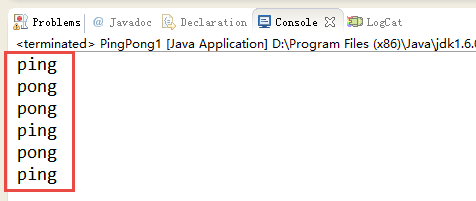


图4C-2 实例1执行结果

【说明】在本例中创建了两个线程，所以在执行线程的时候，cpu会交替执行程序。

实例2，通过方式二中使用Pingpong1类来实现Runnable接口来创建线程。实现过程如图4C-3所示。

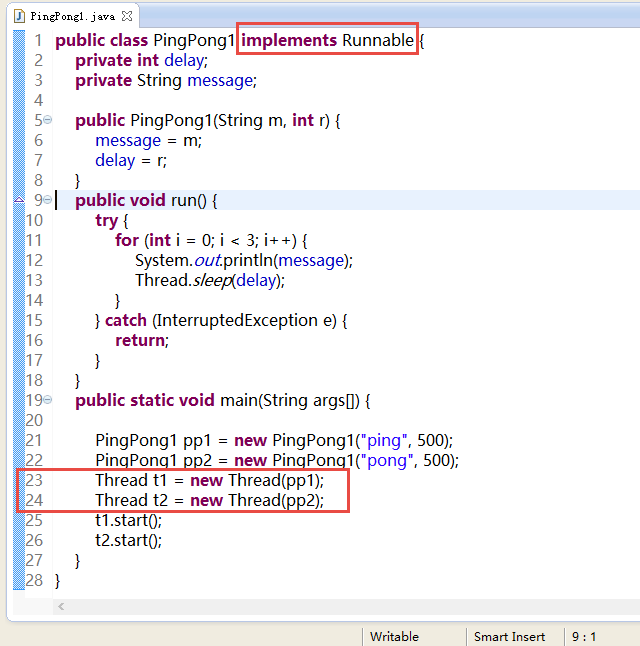


图4C-3 通过继承Runnable创建线程

执行结果，如图4C-4所示。

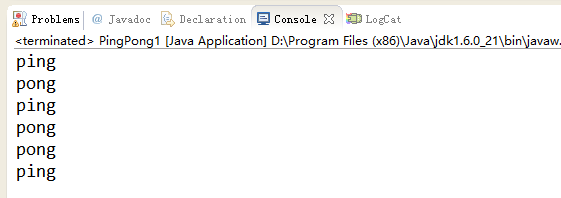


图4C-4 案例二执行结果

【说明】本例用PingPong2类来实现Runnable接口， 并重写了run()方法，与Thread类的创建线程不同的是这里不能直接创建目标类对象并运行它，而是将目标类作为参数创建Thread类的对象。

例：

PingPong1 pp1 = new PingPong1(“ping”,500);

Thread t1 = new Thread(pp1);

然后使用Thread类的对象调用start()方法启动线程，如上：t1.start()。

3、两种创建线程方法的比较

（1）继承Thread 类创建的线程的方法简单方便，可以直接使用线程，但不能再继承其他类。

（2）使用Runnable 借口方法可以将cpu、代码和数据分开，形成清晰的模 型，并且还可以继承其他类。

### 2.2. 线程的生命周期

线程的生命周期，如图4C-5所示。

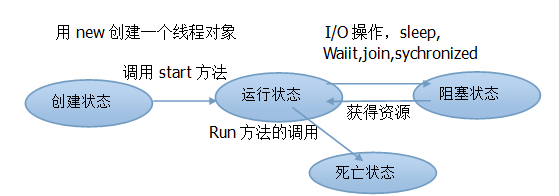


图4C-5 线程的生命周期

#### 简述

从图4C-5可以知道，线程一共有4个状态，分别是——创建状态、运行状态、阻塞状态、死亡状态，下面将会把在这些状态中的线程一一介绍。

（1）创建状态

1 ） 当用new关键字创建一个线程的时候，该线程就处于新建状态，并已初始化。

2 ）当调用start( )方法后，线程就进入了就绪状态。但不能立即进入运行状态，要等待JVM里线程调度器的调度。

（2）运行和阻塞状态

1 ) 当就绪状态的线程得到了cpu，进入运行状态，就开始执行run( )方法。

2 ) 当遇到下面几种情况，线程就会进入阻塞状态。

a . 线程调用 sleep() 方法主动放弃所占用的处理器资源。

b. 线程调用了阻塞式IO方法，在该方法返回之时，线程被阻塞。

c. 线程试图获得一个同步监控器，但该同步监控器被其他线程所有。

d. 线程等待某个通知（notify）。

e. 线程调用了线程的suspend ( )方法将线程挂起。

3 ) 要把线程恢复有下面几种方法

a. 等待其休眠制定时间之后，自动脱离阻塞状态。

b. 系统自己调用特定的指令使该线程恢复可运行状态。

c. 线程成功获取了试图取得的同步监视器。

d. 线程正在等待的某个通知时，其他线程发出了一个通知（signal）。

e. 挂起的线程调用了resume（）恢复方法。

（3）线程死亡状态

1 ) run ( ) 或者call ( )方法执行完毕，线程正常结束。

2 ) 线程抛出一个未捕获的Exception 或 Error;

3 ) 直接调用线程的stop（）方法结束线程，不推荐使用。

【说明】检查线程是否死亡可以使用isAlive ( ) 方法测试，该方法当线程处于就绪、运行、阻塞状态时，就返回true;当线程处于新建、死亡状态时，返回false。

#### 示例与解析

线程生命周期实例，如图4C-6所示。

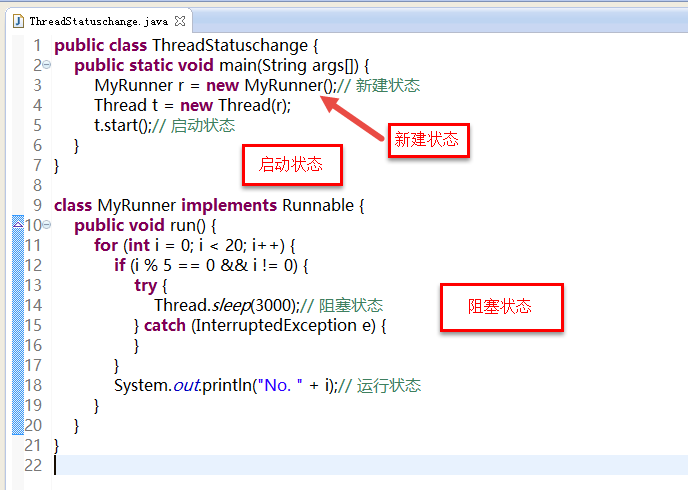


图4C-6 线程生命周期实例

执行结果，如图4C-7所示。

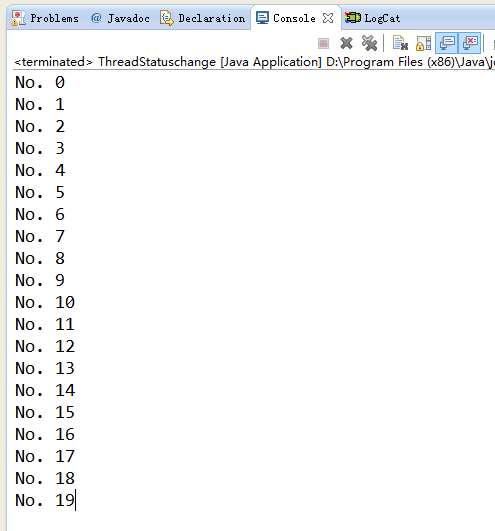


图4C-7 线程生命周期实例输出结果

【说明】在本实例中新建状态是通过new MyRunner（）来新建线程，使用对象调用start()方法后将启动线程，使线程进入运行状态，而遇到Thread.sleep（）睡眠方法时，线程将进入阻塞状态，当run()方法运行结束时线程将进入死亡状态。

### 2.3. 线程的类型

Java中线程分为两种类型：用户线程和守护线程。通过Thread.setDaemon(false)设置为用户线程；通过Thread.setDaemon(true)设置为守护线程。如果不设置该属性，默认使用用户线程。下面就来讲解一下用户线程和守护线程的区别。

#### 1、用户类型

（1） 由用户创建，在前台执行

a.当用户线程运行时，JVM不会退出，JVM就是Java 的虚拟机

b.所有用户线程都执行完毕时，JVM将终止所有守护线程，然后退出。

#### 2、守护类型

（1） 为其他线程提供服务的线程，一般是一个独立的线程，它的run()方法是一个无限循环。

（2） 该类型通常在后台运行，例如：定时器、垃圾回收器等。

（3） 某些用户的线程通过调用setDaemon() 方法在后台运行。

3、 存活状态

当主线程结束后，用户线程还会继续运行，JVM依然存活；而如果没有用户线程，即全部都是守护线程，那么当主线程结束后，所有守护线程以及JVM都会结束。

### 2.4. 线程的优先级

#### 1. 简述

Java线程的优先级拥有优先级这个概念，并且可以用数字来表示优先级级数（1~10）,数字越大表示优先级越高，默认值为5。且Java有三种静态常量：

MAX\_PRIORITY, 最大优先级（值为10）

MIN\_PRIORITY, 最小优先级（值为1）

NORM\_PRIORITY, 默认优先级（值为5）

常用方法：

a. Int getPriority() :获得线程的优先级

b. setPriority() :设置线程的优先级

#### 2、实例与解析

用ThreadPriorityDemo类继承Thread类，并创建两个线程,两个类对象 为t1、t2,并用方法getPriority()设定他们的优先级,并使用MIN\_PRIORITY 和MAX\_PRIORITY来设定他们的优先级。优先级设定为MAX\_PRIORITY线 输出，优先级为MIN\_PRIORITY为后输出。

如图4C-8所示。

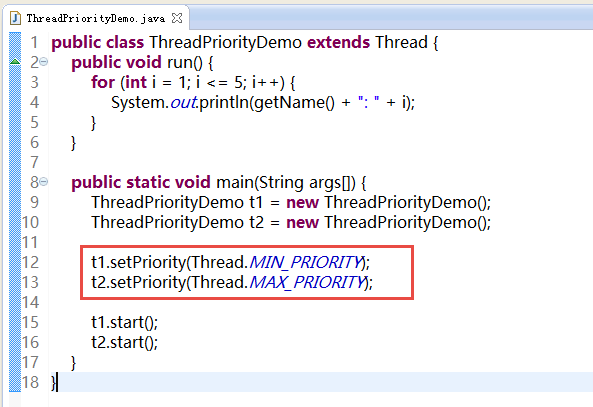


图4C-8 线程优先级示例实现

执行结果如图4C-9所示。

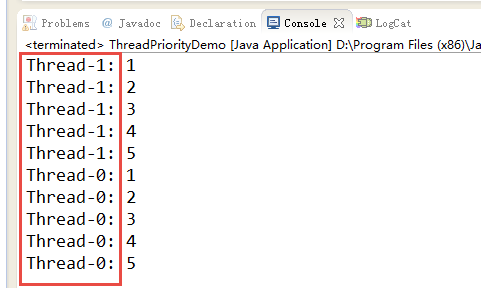


图4C-9 线程优先级示例执行结果

从图4C-9中可以知道，使用了优先级的最大的值和最小值得两个线程，首先是优先级大的先输出，优先级小的后输出。若程序改为图4C-10所示。



图4C-10 对调线程优先级级数

结果如下，图4C-11。

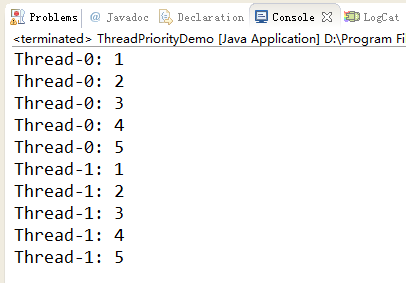


图4C-11 对调线程优先级级数输出结果

结合图4C-10和4C-11可知，优先级调换之后，结果发生了变化，可见线程优先级发挥的作用。

### 2.5. 线程的基本控制

#### 1、简述

在使用线程的时候，有很多方法可以对县城进行操作、如让当前线程休眠、将当前线程占用的cpu让予其他线程等。下面讲解几个比较常用的线程操作方法。

Sleep():使一个线程在特定时间内休眠

a. public static void sleep (long millisec);(毫秒)

b. Public static void sleeo(long millisec , int nanosec);（纳秒）

yield():将cpu让予其他线程，将导致正在运行的线程暂时停止，让行予其他线程。

正在休眠或等待的线程通过interruopt（）方法中断，导致抛出一个异常interruptedException 异常。通过isInterrupted()方法，可以了解是否被中断。

#### 2、实例与解析

设计一个案例使得当前线程休眠1小时，代码如图4C-12所示：

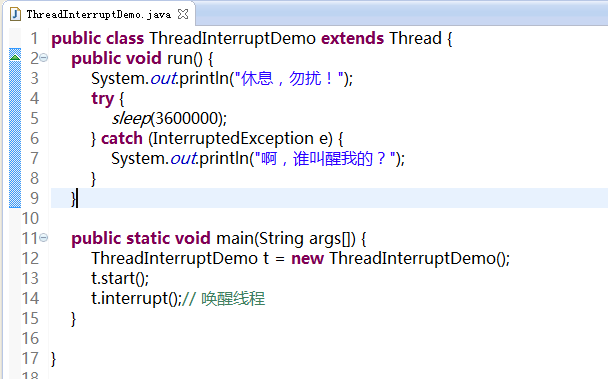


图4C-12 使当前线程休眠1小时案例实现

案例输出结果如图4C-13所示。

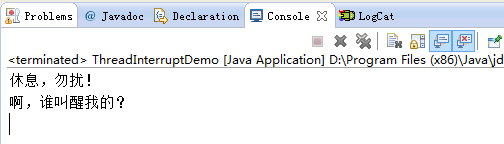


图4C-13 使当前线程休眠1小时案例输出结果

【注意】如果程序中不使用 interrupt() 方法将线程唤醒，线程将等到由sleep()方法设定的时间结束后线程才被唤醒，若使用 interrupt() 方法线程将被提前唤醒。

### 2.6. 线程同步

#### 1、简述

要处理线程同步，可以修改数据的方法，用关键字synchronized 修饰，当一个方法使用synchronized 修饰时，如果A线程要使用该方法，其他线程想使用该方法的线程必须等待线程A使用完才能使用。

而所谓的线程同步就是多个线程都需要使用同一个synchronized 修饰的方法。使用synchronized 修饰的方法需要注意几点：

1. 当两个并发线程访问同一个对象object 中的synchronized( this )同步代码时，同一时间内只有一个线程得到执行，另一个必须等待当前的额线程使用完后才能执行。

2. 当一个线程访问object 的一个synchronized（this）同步代码块时，另一个线程仍然可以访问该object中的非synchronized（this）同步代码块。

3. 当一个线程访问object 的一个synchronized（this）同步代码块时，其他线程对object 中所有其他synchronized（this）同步代码块的访问将被阻塞。

4. 当一个线程访问object 的一个synchronized（this）同步代码块时，它就获得了这个object的对象锁，结果，其他线程对该object对象所有同步代码部分的访问都被暂时阻塞。

#### 2、同步方法

1. wait():使调用线程放弃它正在占用的监视程序并进入睡眠状态，直到某个其他线程进入同一个监视程序并用notify();

2. 唤醒调用对象上的wait() 方法的第一线程。

3. notifyAll():用于唤醒已调用对象上的wait()方法的所有线程。

## 3. 项目心得

通过对多线程的学习，创建线程的两个方法和其区别，由继承Thread类创建线程的方法简单方便，可以直接操作线程，无需使用Thread.currentThread()。但不能再继承其他类。使用Runnable接口方法可以将CPU，代码和数据分开，形成清晰的模型，还可以从其他类继承。保持程序风格的一致性。还有线程的五个状态，新建、就绪、运行、阻塞和死亡，并通过实例知道了它们的使用方法。在java线程中使用优先级将有利于人们去调度多个线程的协调，避免多个线程争用有限的资源而导致系统死机或者崩溃。而线程同步实现了对共享资源的一致性维护。

## 4.常见问题

分不清什么时候使用Run()方法，什么时候直接使用start()方法？

调用start()方法就是一个线程的启动，使线程所代表的虚拟处理机处于可运行状态，这意味着它可以由JVM（java虚拟机）调度并执行。run()方法可以产生必须退出的标志来停止一个线程，即进入死亡状态。在创建线程的时候都要重写run()方法，而要启动线程时就用类的对象来调用start()方法。