Java多线程实现的四种方式

（1）继承Thread类（无返回值）

（2）实现Runnable接口（无返回值）

（3）实现Callable接口通过FutureTask包装器来创建Thread线程（带返回值）

（4）使用ExecutorService、Callable、Future实现有返回结果的多线程。（带返回值）

1.继承Thread类创建线程

Thread类本质上是实现了Runnable接口的一个实例。

启动线程的唯一方法是通过Thread类的start()实例方法，它将启动一个新线程，并执行run()方法。

方式1：匿名内部类方法：

new Thread(){

public void run(){

//重写run方法

}

}.start();

方式2：public class MyThread extends Thread{

public void run(){

//重写run方法

}

}

MyThread myThread1 = new MyThread();

MyThread myThread2 = new MyThread();

myThread1.start();

myThread2.start();

2.实现Runnable接口

方式1：匿名内部类方法

new Thread(new Runnable{

public void run(){

//重写run方法

}

}).start();

或 Thread myThread = new Thread(new Runnable{

public void run(){

//重写run()方法

}});

myThread.start();

或 public class MyThread extends otherClass implements Runnable{

public void run(){

//重写run方法

}

}

new Thread(new MyThread()).start();

MyThread myThread = new Thread();

Thread thread1 = new Thread(myThread);

Thread thread2 = new Thread(myThread);

thread1.start();

thread2.start();

//实现Runnable的好处是可共享变量，对同一变量进行操作，为了防止冲突，可对方法上synchronized，并且必要时候在内部.wait()释放锁。注意.sleep是无法释放锁的

实现Runnable接口相对于继承Thread类的优点

1.适合多个相同程序代码的线程去**处理同一个资源**

2.避免了继承Thread类的Java单继承的局限性

3.增强程序的健壮性，代码能够被多个线程共享，代码与数据是独立的。

4.实现Runnable接口的方式，更加的符合面向对象，线程分为两个部分，一部分线程对象，一部分线程任务。大大降低了线程对象和线程任务的耦合性。所以创建线程的方式都使用Runnable接口。

原因如下：

继承Thread类：

run方法在Thread里。线程对象（Thread）和线程任务（run方法）耦合在一起。一旦创建Thread类的子类对象，既是线程对象，又有线程任务。

实现Runnable接口：

run方法在Runnable接口里。将线程任务单独分离出来封装成对象（实现Runnable接口），用Runnable来标明线程任务，用Thread来明确线程对象。Runnable接口对县城对象和线程任务进行解耦。