**Creating and Starting Java Threads**

* [Java Threads Video Tutorial](http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html#java-threads-video-tutorial)
* [Creating and Starting Threads](http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html#creating-and-starting-a-thread)
* [Thread Subclass](http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html#thread-subclass)
* [Runnable Interface Implementation](http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html#runnable-interface-implementation)
* [Subclass or Runnable?](http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html#subclass-or-runnable)
* [Common Pitfall: Calling run() Instead of start()](http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html#common-pitfall)
* [Thread Names](http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html#thread-names)
* [Thread.currentThread()](http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html#thread-currentthread)
* [Java Thread Example](http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html#java-thread-example)

Java线程是对象，正如Java的其他对象。线程是类java.lang.Thread或者它的子类的实例化。因为是对象，java线程也可以执行代码。

**Java Threads Video Tutorial**

这是Java线程教学的视频版本。

**Creating and Starting Threads**

在Java中创建一个线程操作如下：

Thread thread = new Thread();

为了开始这个Java线程，你将调用它的start()方法，如下：

thread.start();

这个例子不特定任何代码给线程去执行。线程将会立即在它开始后再次停止。

有两种方式指定线程应当执行什么代码。第一种是创建Thread的子类，重写run()方法。第二种方法是传递一个实现Runnable(java.lang.Runnable)的对象给Thread的构造器。两种方法在下面说明。

**Thread Subclass**

第一种方法指定线程跑什么代码，是创建一个线程子类，重写run()方法。run()方法被你调用start()方法之后的线程执行。这是一个例子，创建一个Java Thread子类：

public class MyThread extends Thread {

public void run(){

System.out.println("MyThread running");

}

}

为了创建和开始上述线程你可以这样做：

MyThread myThread = new MyThread();

myTread.start();

start()调用将会在线程被开始后立即返回。不会等到run()方法完成。run()方法将会执行好像被一个不同CPU执行。当run()方法执行时会打印文本“MyThread running”。

你也可以创建一个匿名的Thread的子类如下：

Thread thread = new Thread(){

public void run(){

System.out.println("Thread Running");

}

}

thread.start();

例子将会打印文本“Thread running”一旦run()方法被新线程执行。

**Runnable Interface Implementation**

第二种方法指定一个线程跑什么代码是通过创建一个类实现java.lang.Runnable。Runnable 对象可以被一个Thread执行。

这是一个Java Runnable 例子:

public class MyRunnable implements Runnable {

public void run(){

System.out.println("MyRunnable running");

}

}

为了run() 方法被一个线程执行，传递一个MyRunnable的实例给Thread的构造器。以下是如何实现的：

Thread thread = new Thread(new MyRunnable());

thread.start();

当线程启动时会调用MyRunnable 实例的run()方法而非执行它自己的run()方法。以上的例子将会打印出文本“MyRunnable running”。

你也可以创建一个Runnable的匿名实现，像这样：

Runnable myRunnable = new Runnable(){

public void run(){

System.out.println("Runnable running");

}

}

Thread thread = new Thread(myRunnable);

thread.start();

**Subclass or Runnable?**

没有规则关于这两种方法哪种最好。两种方法都行得通。然而个人，我倾向于实现Runnable，然后传递一个实现类的实例给Thead的实例。当让Runnable被一个线程池([**thread pool**](http://tutorials.jenkov.com/java-concurrency/creating-and-starting-threads.html))执行时，很容易将Runnable实例排队，直到池里的线程是空闲的。这个使用Thread子类有点难以实现。

有时你可能需要实现Runnable接口也要继承Thread类。例如，创建一个Thread的子类执行多个Runnable。这是典型的实现一个线程池的案例。

**Common Pitfall: Calling run() Instead of start()**

当创建并开始一个线程，一个常见的错误是调用Thread类的run()方法而不是start()方法，例如：

Thread newThread = new Thread(MyRunnable());

newThread.run(); //should be start();

首先你可能没注意到任何事因为Runnable的run()方法像你期望那样执行了。然而，不是被你刚刚创建的新线程执行的。而是被创建这个线程的线程执行的，或者说，是执行上述两行代码的线程。为了让MyRunnable实体类的run()方法被新创建的线程(newThread)调用，你必须调用newThread.start()方法。

**Thread Names**

当你创建一个Java线程你可以给它一个名字。名字可以帮你区分不同的线程。例如，如果多线程写入System.out，它可以很方便看到哪个线程写的文本。这是个例子：

Thread thread = new Thread("New Thread") {

public void run(){

System.out.println("run by: " + getName());

}

};

thread.start();

System.out.println(thread.getName());

注意到字符串“New Thread”作为参数被传递给Thread构造器。字符串可以被获取通过Thread类的getName()方法。你可以传递一个名字给一个Thread当使用一个Runnable实现类。这就是它的样子：

MyRunnable runnable = new MyRunnable();

Thread thread = new Thread(runnable, "New Thread");

thread.start();

System.out.println(thread.getName());

但是，请注意，因为MyRunnable类不是Thread的子类，所以它不能访问执行它的线程的getName()方法。

**Thread.currentThread()**

Thread.currentThread()方法返回一个引用，指向执行currentThread()方法的Thread实例。这样，您就可以访问表示执行给定代码块的线程的Java线程对象。这是一个例子，如何使用Thread.currentThread()：

Thread thread = Thread.currentThread();

一旦你获得指向Thread类对象的引用，你可以调用它上面的方法。例如，你可以获得当前执行代码的线程名，像这样：

String threadName = Thread.currentThread().getName();

**Java Thread Example**

这是个小例子。首先打印出执行main()方法的线程名。这个线程由JVM分配。然后它启动10个线程并给它们用数字命名(“” + i)。每个线程然后打印出它的名字，然后停止执行。

public class ThreadExample {

public static void main(String[] args){

System.out.println(Thread.currentThread().getName());

for(int i=0; i<10; i++){

new Thread("" + i){

public void run(){

System.out.println("Thread: " + getName() + " running");

}

}.start();

}

}

}

注意，即使线程按顺序（1、2、3等）启动，它们也可能不会按顺序执行，这意味着线程1可能不是第一个将其名称写入System.out的线程。这是因为线程原则上是并行执行而不是顺序执行的。JVM和/或操作系统确定线程执行的顺序。该顺序不必是它们启动时的相同顺序。