

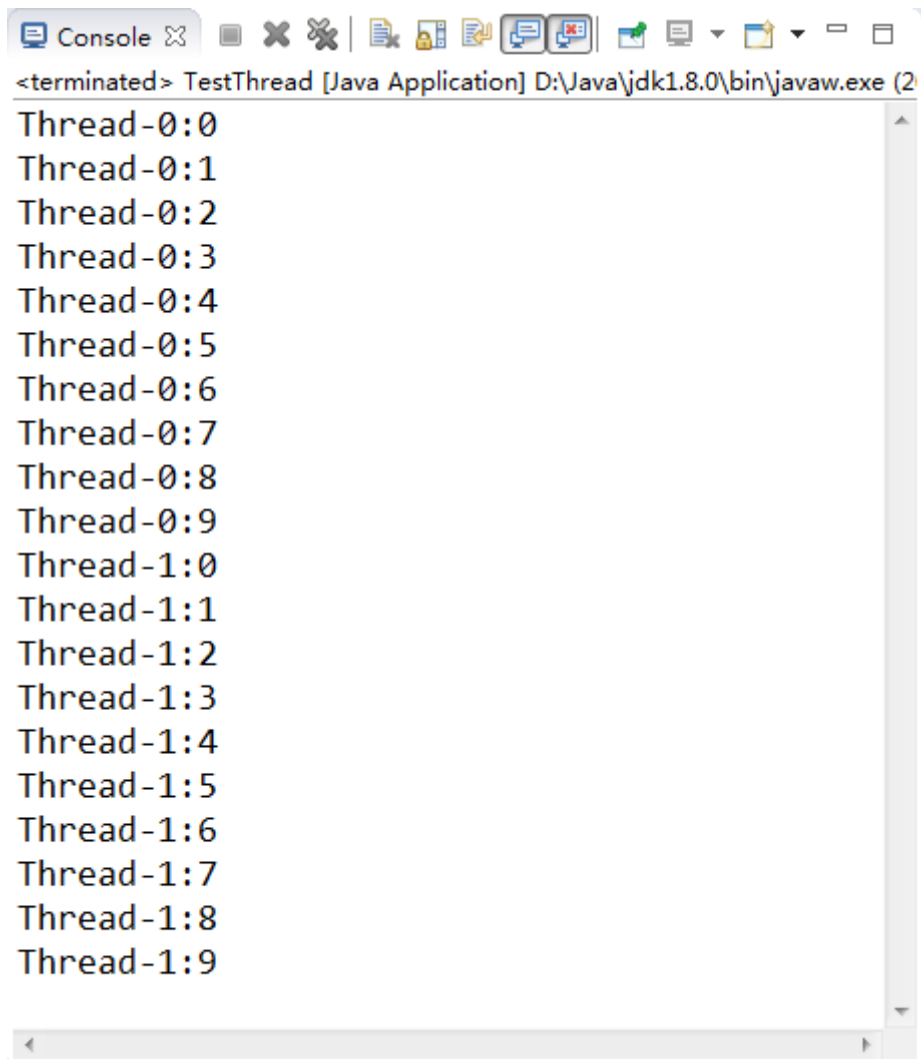
实际上所有的多线程代码都是通过运行Thread的start()方法来运行的。因此，不管是扩展Thread类还是实现Runnable接口来实现多线程，最终还是通过Thread的对象的API来控制线程的。

1. 在Java中负责实现线程功能的类是java.lang.Thread 类。
2. 可以通过创建 Thread的实例来创建新的线程。
3. 每个线程都是通过某个特定的Thread对象所对应的方法run()来完成其操作的，方法run()称为线程体。
4. 通过调用Thread类的start()方法来启动一个线程。

缺点：如果我们的类已经继承了一个类(如小程序必须继承自 Applet 类)，则无法再继承 Thread 类。

通过继承Thread类实现多线程

```
1 public class TestThread extends Thread { //自定义类继承Thread类
2     //需要实现 run() 方法，因为 Thread 类也实现了 Runnable 接口。
3     public void run() {
4         for (int i = 0; i < 10; i++) {
5             System.out.println(this.getName() + ":" + i);
6             //getName() 方法是返回线程名称
7         }
8     }
9
10    public static void main(String[] args) {
11        TestThread thread1 = new TestThread(); //创建线程对象
12        thread1.start(); //启动线程
13        TestThread thread2 = new TestThread();
14        thread2.start();
15    }
}
```



The image shows a screenshot of a Java IDE's console window. The title bar at the top reads "Console" followed by standard window control buttons (minimize, maximize, close). Below the title bar, the console output shows the status of a Java application: "<terminated> TestThread [Java Application] D:\Java\jdk1.8.0\bin\javaw.exe (2)". The main area of the console lists 20 thread names, organized into two groups. The first group, labeled "Thread-0:", contains threads 0 through 9. The second group, labeled "Thread-1:", contains threads 0 through 9. The text is displayed in a monospaced font, and a vertical scrollbar is visible on the right side of the console area.

```
<terminated> TestThread [Java Application] D:\Java\jdk1.8.0\bin\javaw.exe (2)
Thread-0:0
Thread-0:1
Thread-0:2
Thread-0:3
Thread-0:4
Thread-0:5
Thread-0:6
Thread-0:7
Thread-0:8
Thread-0:9
Thread-1:0
Thread-1:1
Thread-1:2
Thread-1:3
Thread-1:4
Thread-1:5
Thread-1:6
Thread-1:7
Thread-1:8
Thread-1:9
```