Thread 线程类

Java 的线程是通过 java.lang.Thread 类来实现的。VM 启动时会有一个由 main 方法所定义的线程。可以通过创建 Thread 的实例来创建新的线程。每个线程都是通过某个特定 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。通过调用 Thread 类的 **start()**方法来启动一个线程。

Start 方法：

start（）方法来启动线程，真正实现了多线程运行。

start 方法的作用就是将线程由 NEW 状态，变为 RUNABLE 状态。当线程创建成功时，线程处于 NEW（新建）状态，如果你不调用 start( )方法，那么线程永远处于 NEW 状态。调用 start( )后，才会变为 RUNABLE 状态，线程才可以运行。

线程不是马上执行的；准确来说，调用 start( )方法后，线程的状态是“READY（就绪）”状态，而不是“RUNNING（运行中）”状态（关于线程的状态详细。线程要等待 CPU 调度，不同的 JVM 有不同的调度算法，线程何时被调度是未知的。因此，start（）方法的被调用顺序不能决定线程的执行顺序。

由于在线程的生命周期中，线程的状态由 NEW ----> RUNABLE 只会发生一次，因此，一个线程只能调用 start（）方法一次，多次启动一个线程是非法的。特别是当线程已经结束执行后，不能再重新启动。

Run 方法：

run（）方法当作普通方法的方式调用

run( )其实是一个**普通方法**，只不过当线程调用了 start( )方法后，一旦线程被 CPU 调度，处于运行状态，那么线程才会去调用这个 run（）方法；

上面说了，run（）方法是一个普通的对象方法，因此，不需要线程调用 start（）后才可以调用的。可以线程对象可以随时随地调用 run 方法。

下面是 start()和 run()的源码

```
    public synchronized void start() {
        // 如果线程不是"就绪状态"，则抛出异常！
        if (threadStatus != 0)
            throw new IllegalThreadStateException();
        // 将线程添加到 ThreadGroup 中
        group.add(this);
        boolean started = false;
        try {
            // 通过 start0()启动线程,新线程会调用 run()方法
            start0();
            // 设置 started 标记=true
            started = true;
        } finally {
            try {
                if (!started) {
```

```
                    group.threadStartFailed(this);
                }
            } catch (Throwable ignore) {
            }
        }
    }
public void run() {
    if (target != null) {
        target.run();
    }
}
```

Start()的调用会导致两个线程被启动：

导致此线程开始执行; Java 虚拟机调用此线程的 run 方法。结果是两个线程同时运行:当前线程**（从调用返回到 start 方法）** 和**另一个线程（执行其 run 方法）**。

多次 start()线程是不合法的（会检查线程的状态的）