```
//使用runnable接口的方式实现多线程的好处：
        //①解决了java程序的单继承的缺点
        //②当实现的对象的数据是一个固定的数据的时候，使用实现runnable接口的方式更加合理
        //创建一个子类的对象
        sub sub = new sub();
        //调用线程的start（），启动此线程，调用相应的run方法
        sub.start();
        int sum = 0;
        for (int i = 0; i <= 100; i++) {
                sum+=i;
                System.out.print(Thread.currentThread().getName()+"::");
                System.out.println(i);
        }

//创建一个继承于thread的子类
class sub extends Thread{
        //重写thread类的run（）方法，方法内实现此子线程要完成的功能
        public void run(){
                int sum =0;
                for (int i = 0; i <= 100; i++) {
                        sum+=i;
                        System.out.print(Thread.currentThread().getName()+"::");
                        System.out.println(i);
                }
        }
}
```

使用多线程的优点：

①提高应用程序的响应，对图形用户界面更有意义，增强用户体验

②提高计算机系统CPU的利用率

③改善程序结构，将既长又复杂的进程分为多个线程，独立运行，利于理解和修改

线程的生命周期：

jdk中用thread.state枚举表示了线程的几种状态。要想实现多线程，必须在主线程中创建新的线程对象，java语言使用thread类及其子类的对象来表示线程，在它的一个生命周期中通常要经历如下的五种状态。

①新建（start）：当一个thread类及其子类的对象被声明并创建时，新生的线程对象处于新建状态

②就绪（yield）：处于新建状态的线程被start（）后，将进入线程队列等待CPU时间片，此时他已经

具备了运行的条件

③运行（run）：当就绪的线程被调度并获得处理器资源时，便进入运行状态，run（）方法定义了线

程的操作和功能

④阻塞（sleep）：在某种特殊情况下，被人为挂起或执行输入输出操作时，让出CPU并临时中止自

己的执行，进入阻塞状态

⑤死亡（stop）：线程完成了它的全部工作或线程被提前强制性中止