软件架构：

1、 Telnet 协议

1.1、简述

Telnet 协议是通过网络来调用服务器，并使自己的机器作为服务器的终端的协议。它为用户提供了在本地计算机上完成远程主机工作的能力。

1.2、使用端口

Telnet 协议使用的端口是 23，所以我们再使用该协议时需要将端口取值为 23。

1.3、协商

为了能成为虚拟终端，服务器和客户端双方必须确认客户端要采用的界面控制功能和服务器端所采用的终端控制功能，这个操作过程称为协商。

下表尾部分 telnet 命令及相应的代码和描述，也是程序中所以用到 telnet 指令。

名称	代码（10 进制）	描述
IAC	255	标志符,代表是一个 TELNET 指令,接到该指令之后的数据都是命令
DONT	254	表示一方要求另一方停止使用，或者确认你不再希望另一方使用指定的选项。
DO	253	表示一方要求另一方使用，或者确认你希望另一方使用指定的选项。
WONT	252	表示拒绝使用或者继续使用指定的选项。
WILL	251	表示希望开始使用或者确认所使用的是指定的选项。

下表为协商选项，用于设置协商的具体选项。通常 telnet 指令为<IAC,telnet 指令,协商选项>，如<IAC,DO,24>。在代码实现中，由于只使用 NVT 功能进行通讯，所以需要拒绝服务器端的所有有关的执行选项的请求，所以对于服务器端的发送 DO 请求，回复 WONT。

选项标识（10 进制）	名称
1	回显
3	抑制继续进行
5	状态
6	定时标记
24	终端类型
31	窗口大小
32	终端速率
33	远程流量控制
34	行方式
36	环境变量

2、 代码结构

2.1、代码中实现中，利用 23 号端口进行 telnet 协议的通讯。为了使代码结构更加清晰，这里使用了两个主要的类。

TelnetClass：包含 telnet 的通讯的所有主要部分	
变量及方法	描述
host	String 类型。指定连接的域名或者 ip 地址
port	Int 类型。指定服务器端口
telnetSocket	Socket 类型。利用 host 和 port 创建
serverOutput	OutputStream 类型。用于对服务器发送数据。
ServerInput	InputStream 类型。用于获取服务器返回数据。
TelnetClass (String host,int port)	构造函数
openConnection()	利用 host 和 port 创建 telnetSocket，并获得 serverOutput 和 serverInput
interactWithTelnetServer()	创建一个 InputToServer 类对象，开启一个子线程，用于将指令发送到服务器。 创建一个 OutputToScreen 对象，用于将服务器返回的数据显示到显示屏中。
negotiate()	进行协调，对于服务器端的 DO 命令回复 WONT 命令。
close()	关闭 telnetSocket

InputToServer：用于将指令发送到服务器，继承 Runnable 接口	
变量和方法	**描述**
input	BufferedInputStream 类型。用 System.in 创建。将数据从键盘输入。
Output	OutputStream 类型。用 serverOutput 创建。将数据输出到服务器。
run()	在一个 while(true)循环中，保持两个输入输出端的交互。

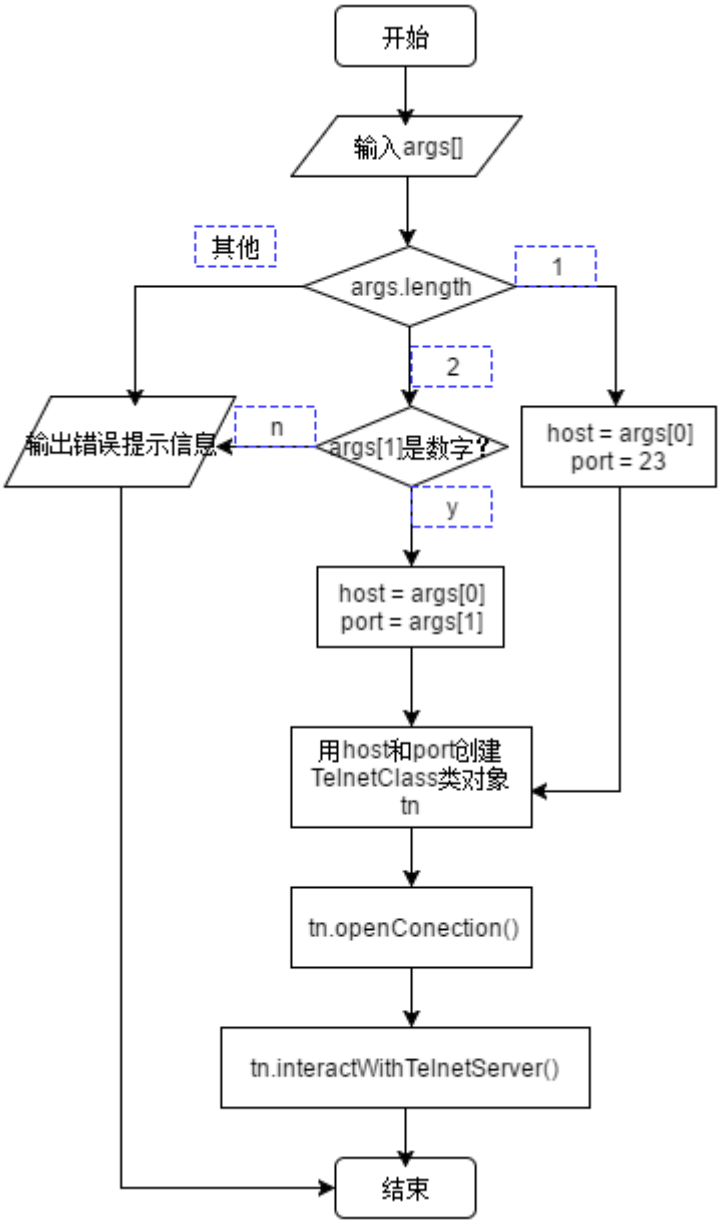
OutputToScreen：用于将指令发送到服务器，继承 Runnable 接口	
变量和方法	**描述**
reader	BufferedInputStream 类型。用 serverInput 创建。将数据从服务器返回的数据流的读取出来并进行“GBK”编码，从而允许终端进行颜色的渲染。
printer	用 System.out 创建。将数据输出到显示屏。如果运行系统为 Linux 则保留颜色标签用于渲染字符，否则通过正则表达式消除所有的颜色标签。
colorCodeRegex	颜色标签匹配的正则表达式。
run()	在一个 while(true)循环中，保持两个输入输出端的交互。

特别说明：

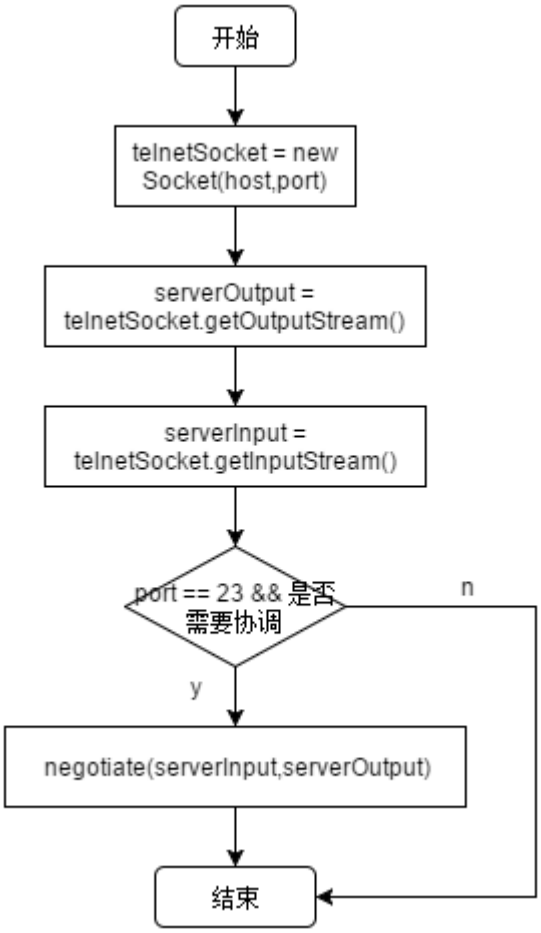
1、OutputToScreen 会将获取到的数据流用“GBK”进行编码，然后输出到控制台或者终端，如果是在 Linux 下的终端中运行的话，颜色标签就会渲染成相应的颜色，在 windows 下运行则只能看到颜色标签的字符表示方法。

2、程序会根据运行该程序的操作系统决定是否需要使用 colorCodeRegex 去消除得到的服务器返回的数据中的颜色标签如：_[1;33m 或者_[11;16H 或者_[m 等多种形式（这里的下滑线表示为 ASCII 码为 27 的不可视字符）。如果是在 Linux 下不会消除，这些颜色标签将用于渲染字符颜色，否则将用 colorCodeRegex 正则表达式消除所有的颜色标签，使得显示的结果尽管没有颜色，但是界面会比较清晰。

2.2、代码主体框架



2.3、tn.openConection()的实现



2.4、其他函数都比较简单，可以从上面的表格读懂。这里不做多余的描述。这里需要说明一点。按照我查找到的资料，如果在进行 telnet 协议通讯时，没有进行协调，那么返回来的应该是用于协调的符号序列 y#yy'y$。也就是说如果客户端没有向服务器发送 telnet 协调命令，能够读取到的数据只有那么一个序列而已。所以应该进行的操作是，对得到的序列进行分析，判断指令的具体内容，让后再进行发送相应的信息，这里设置的是对于服务器的 DO 指令发送 WONT 信息。然而，在实际的通讯过程中，在没有发送任何协调命令的情况下，我仍然获得了全部的消息数据。而且可以进行正常的通讯。相反，如果我进行协调的话，因为需要获取数据开始的 3 个 byte，进行命令分析，所以在输出中开始的 3 个 byte 的数据没有了。

通过实验的现象，我决定设置一个全局的变量 needNegotiate，并设置为 false，使得不进行协调以获得全部的消息。如果在之后的学习，找到原因的所在方便进行修改。

除此之外，在运行的过程中，由于后台的两个子线程会不断的运行，即使程序已经结束了，或者遇到异常而结束，子线程都会一直运行，这样产生的效果非常不好，所以我设置了一个全部的变量 stop 并初始化为 false，在每次读写循环中，判断 stop 的值是否为 true，如果在异常发生时或者其他情况下改变为 true 了，run 函数结束，线程结束。