实验报告

题目:Socket网络编程

组员分工：

学号：

指导老师：

1. 实验题目：

Socket网络编程

二、实验内容及要求：

要求：基于Java Socket API搭建简单的HTTP客户端和服务器端程序

说明：

1. 不允许基于netty等框架，完全基于Java Socket API进行编写

2. 不分区使用的IO模型，BIO、NIO和AIO都可以

3. 实现基础的HTTP请求、响应功能，具体要求如下：

​ 3.1 HTTP客户端可以发送请求报文、呈现响应报文（命令行和GUI都可以）

​ 3.2 HTTP客户端对301、302、304的状态码做相应的处理

​ 3.3 HTTP服务器端支持GET和POST请求

​ 3.4 HTTP服务器端支持200、301、302、304、404、405、500的状态码

​ 3.5 HTTP服务器端实现长连接

​ 3.6 MIME至少支持三种类型，包含一种非文本类型

三、实验方法：

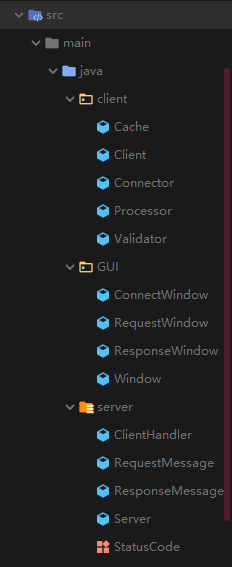
1、本次实验使用GUI与用户交互。

2、实验通过客户端发送请求报文，接收并呈现响应报文。

四、实验过程

1、确定编程语言与架构方式。

1.1：编程语言：Java

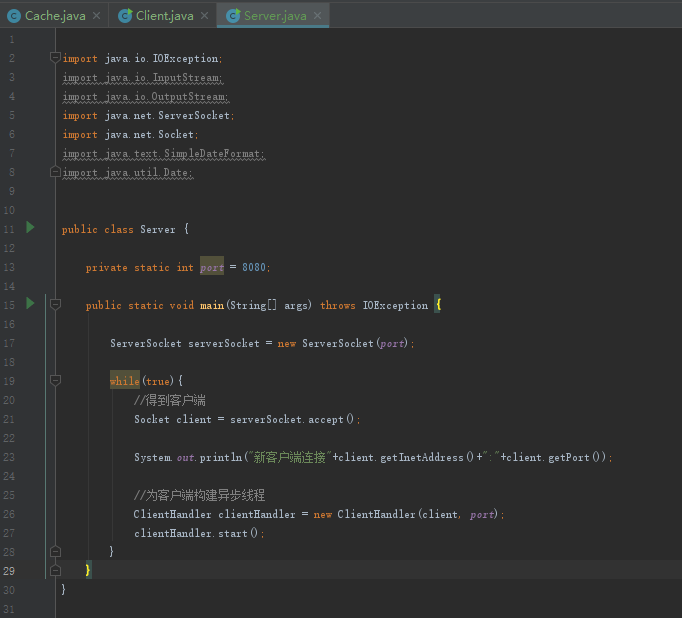
1.2架构方式：

client为客户端，GUI为交互窗口，server为服务端，StatusCode为需要测试的状态码

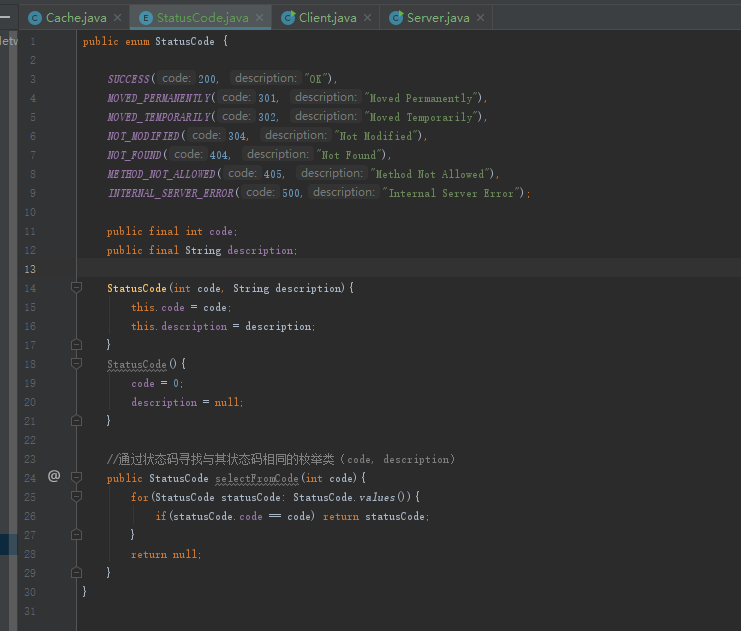
1.2.1:客户端实现的核心代码



1.2.2：服务端实现的核心代码



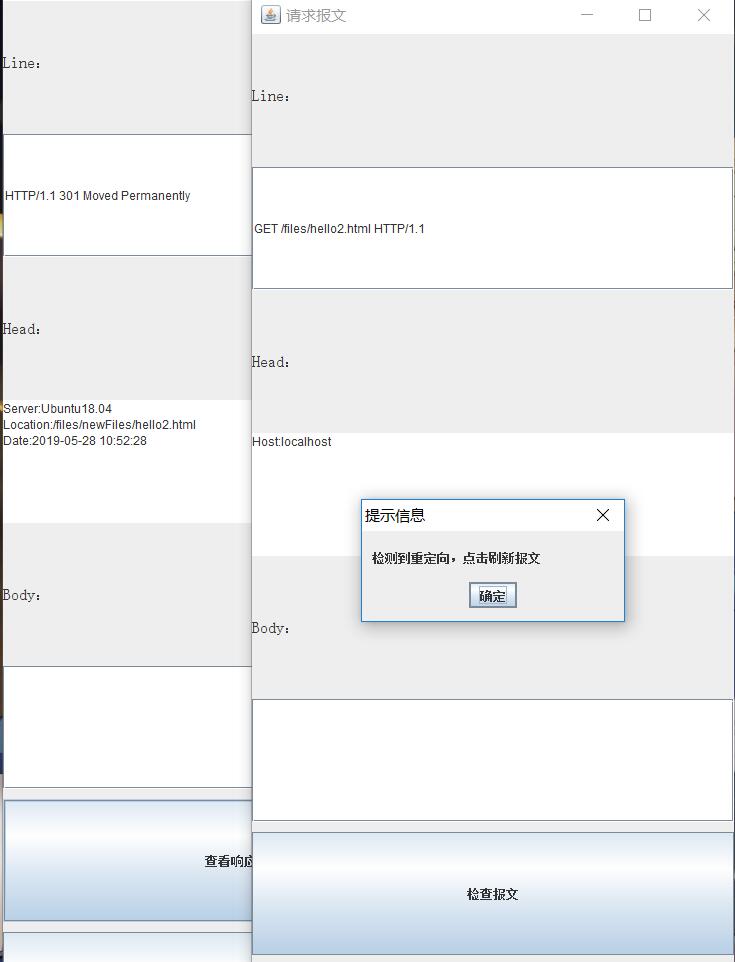
1.2.3：需要完成测试的状态码



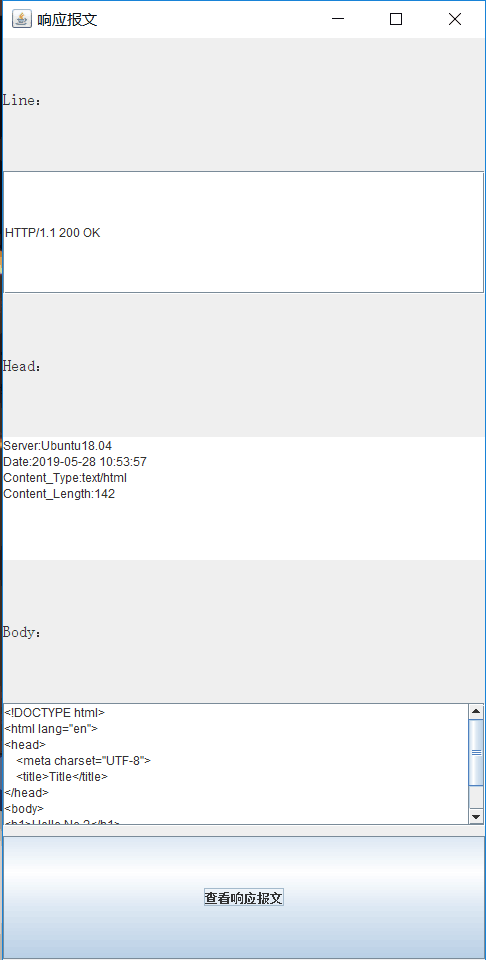
2、客户端对301状态码的处理：

2.1:先输入远程主机地址和端口号，此处默认设为localhost：8080

2.2：对301 状态码做出处理

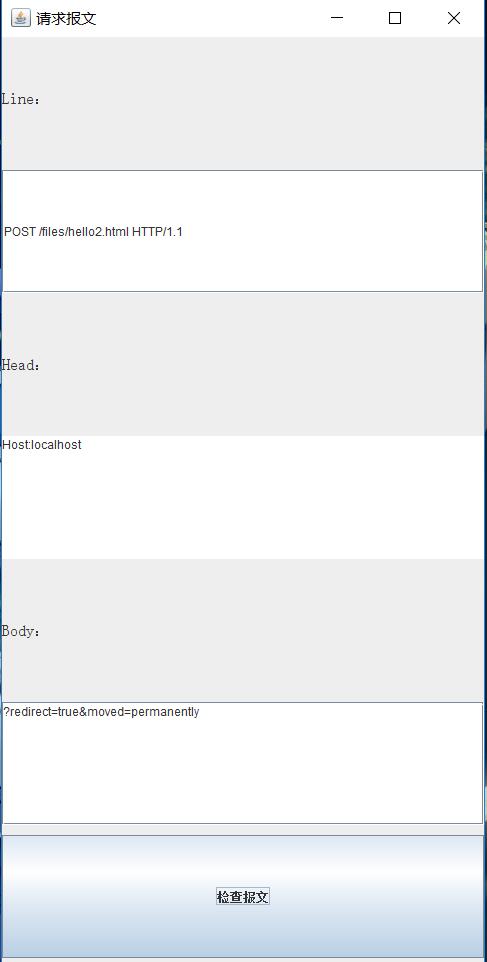


2.3: 点击确认后，浏览器自动定向到新URI，并发送响应报文。因为此处GET请求成功，所以返回状态码为200

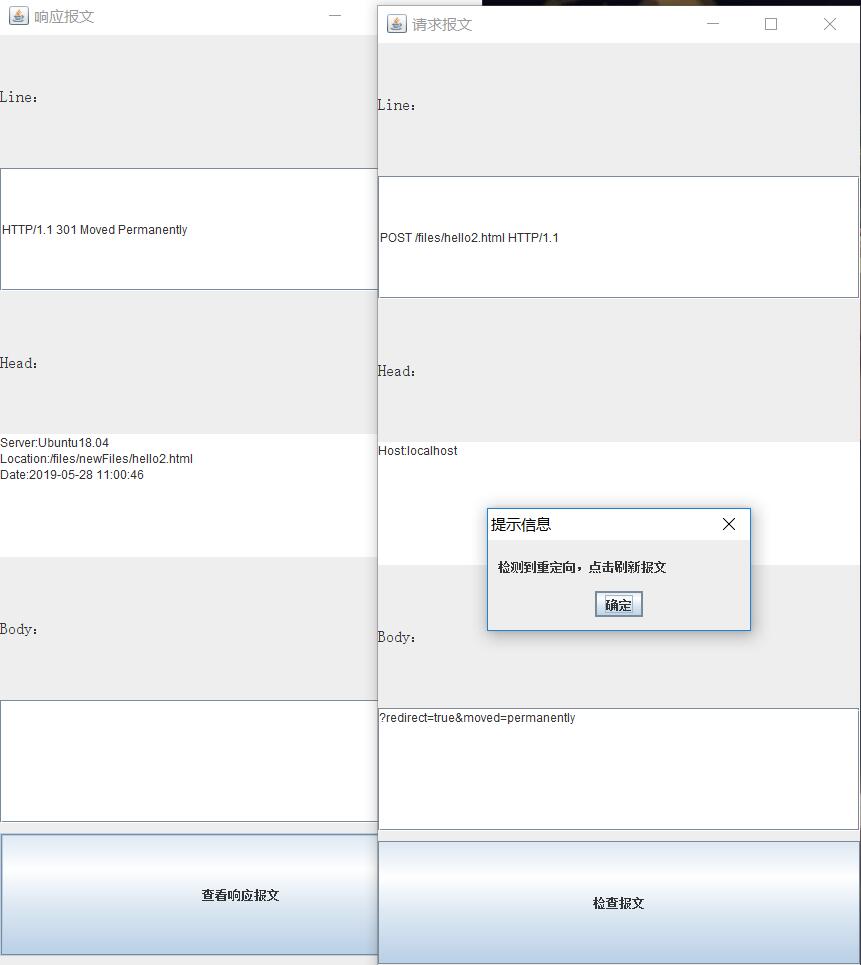


3、POST请求

3.1：发送POST请求

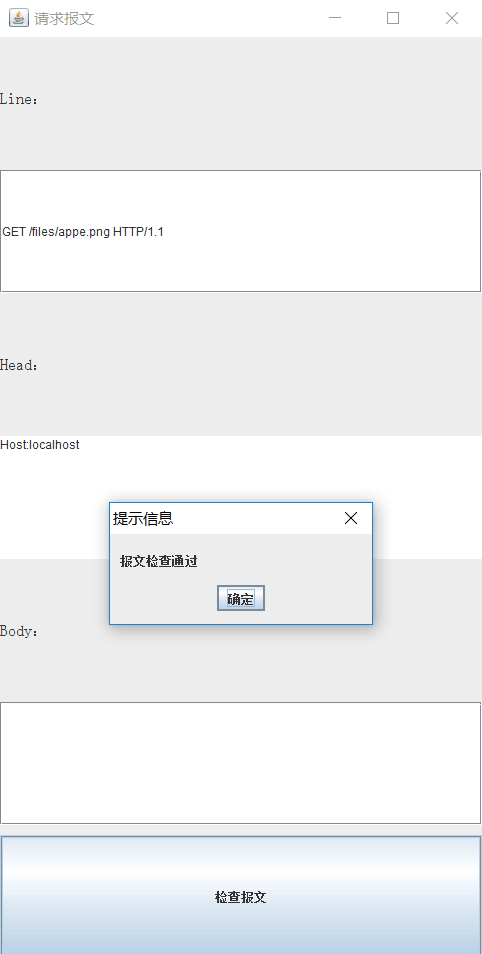


3.2：对于POST请求做出正确的响应

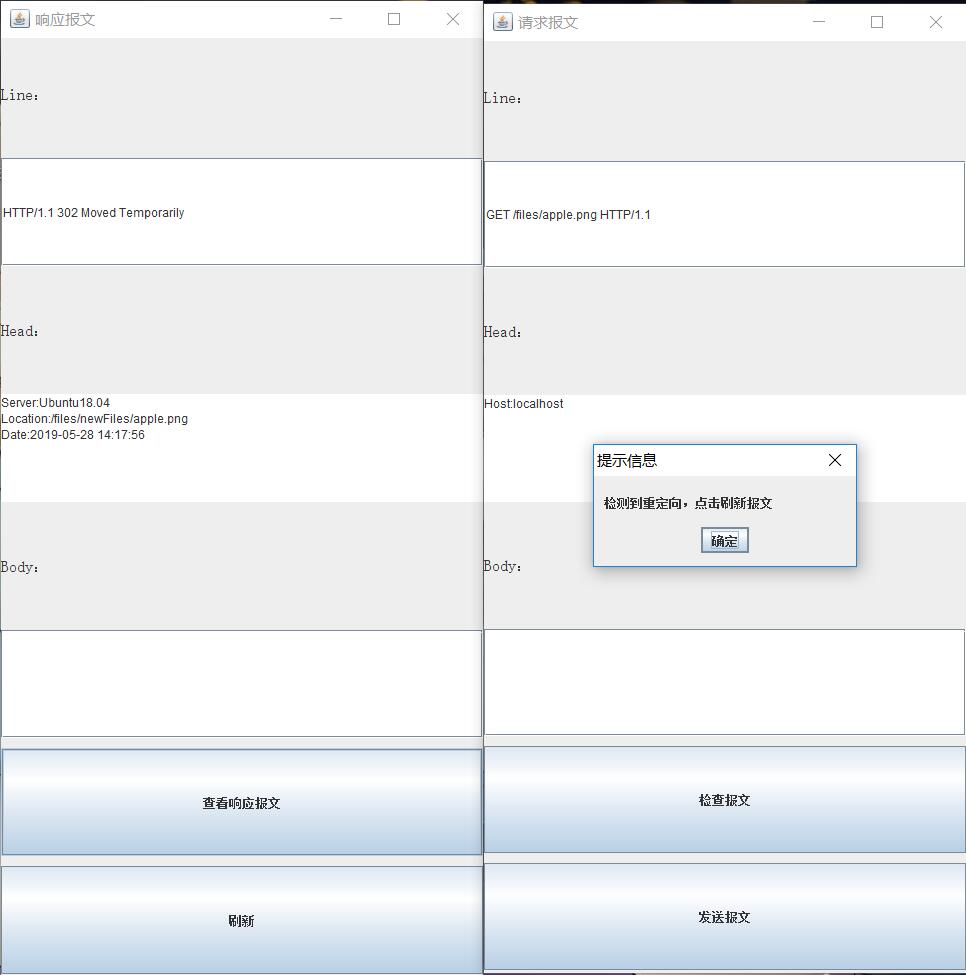


4、处理302状态码

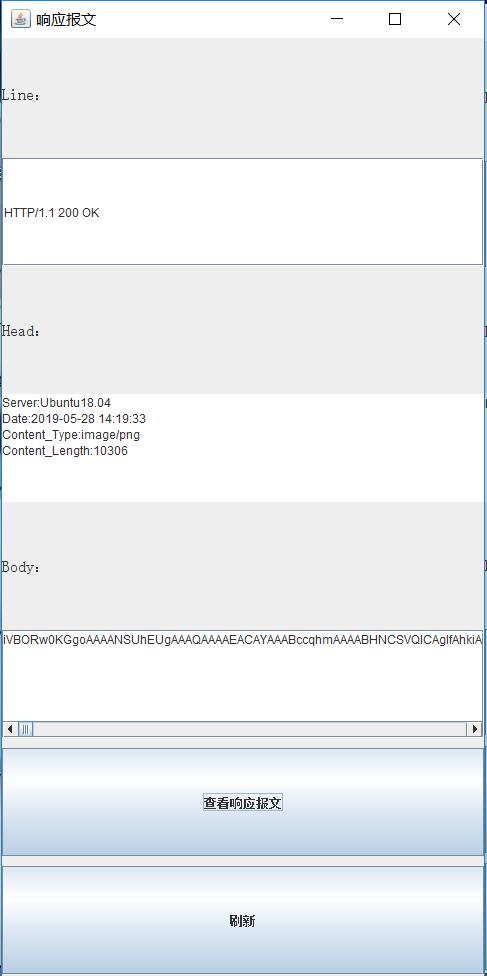
4.1：发送下一条请求报文，请求打开的文件类型为image/png



4.2：服务器端支持302状态码，并对302状态码做出相应处理

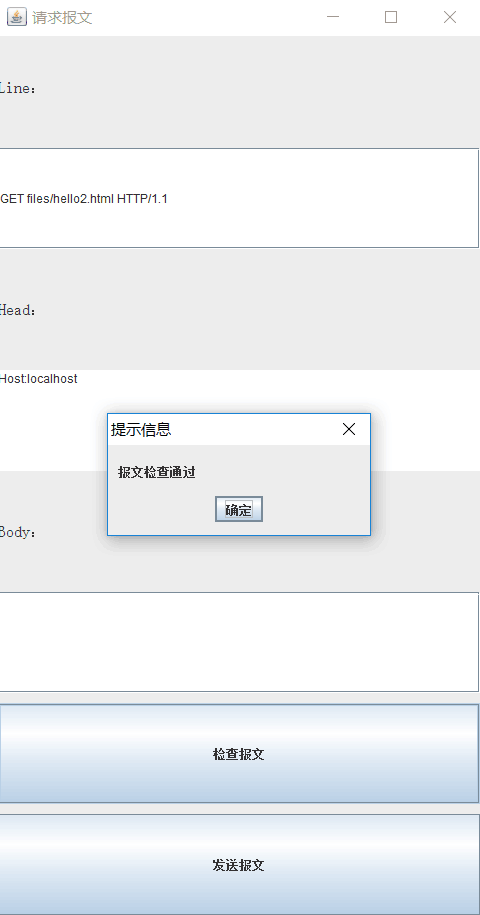


4.3：呈现了响应报文，此处表示请求成功

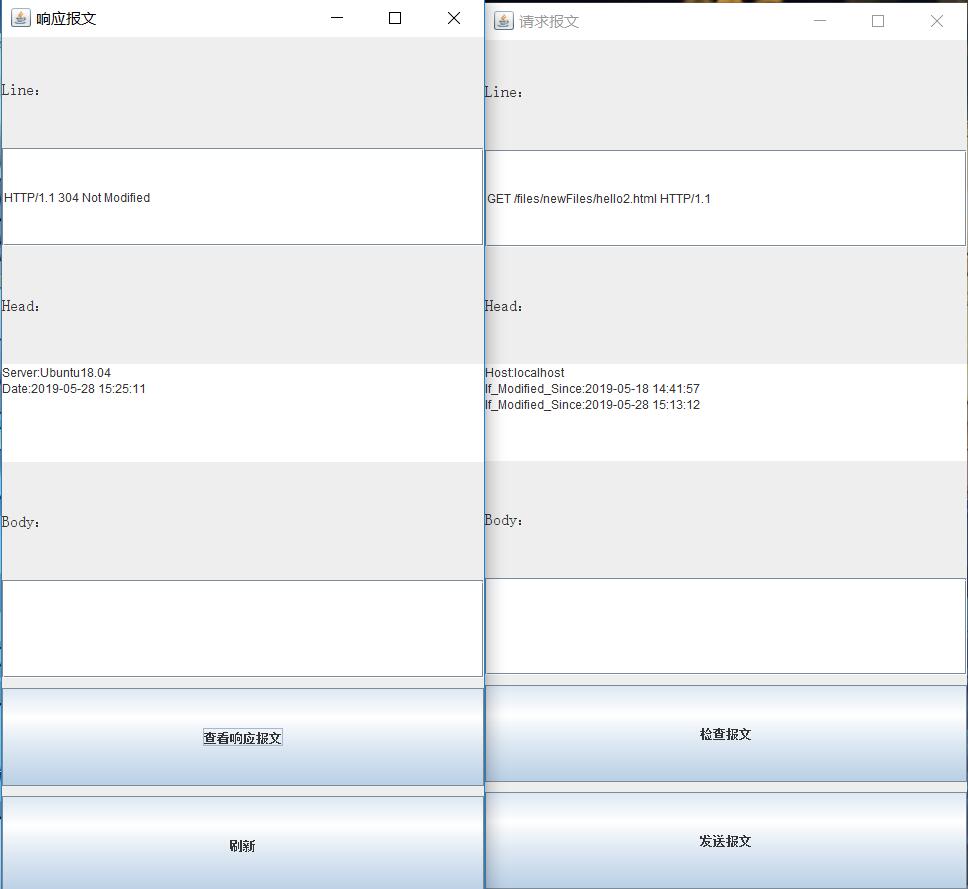


5、处理304状态码

5.1：发送请求报文

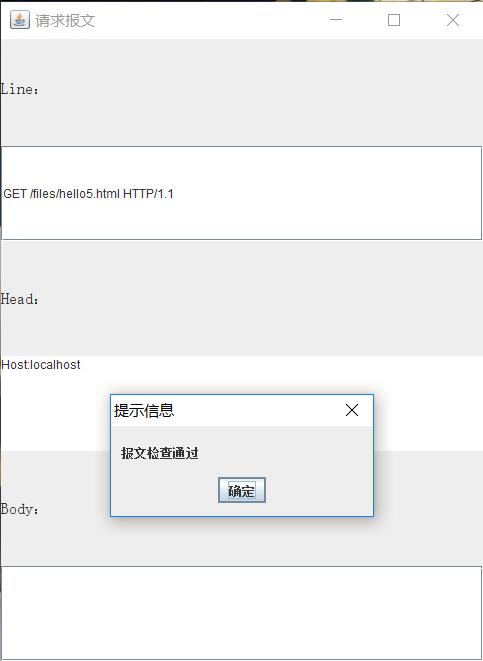


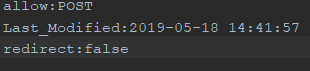
5.2：304状态码的响应报文



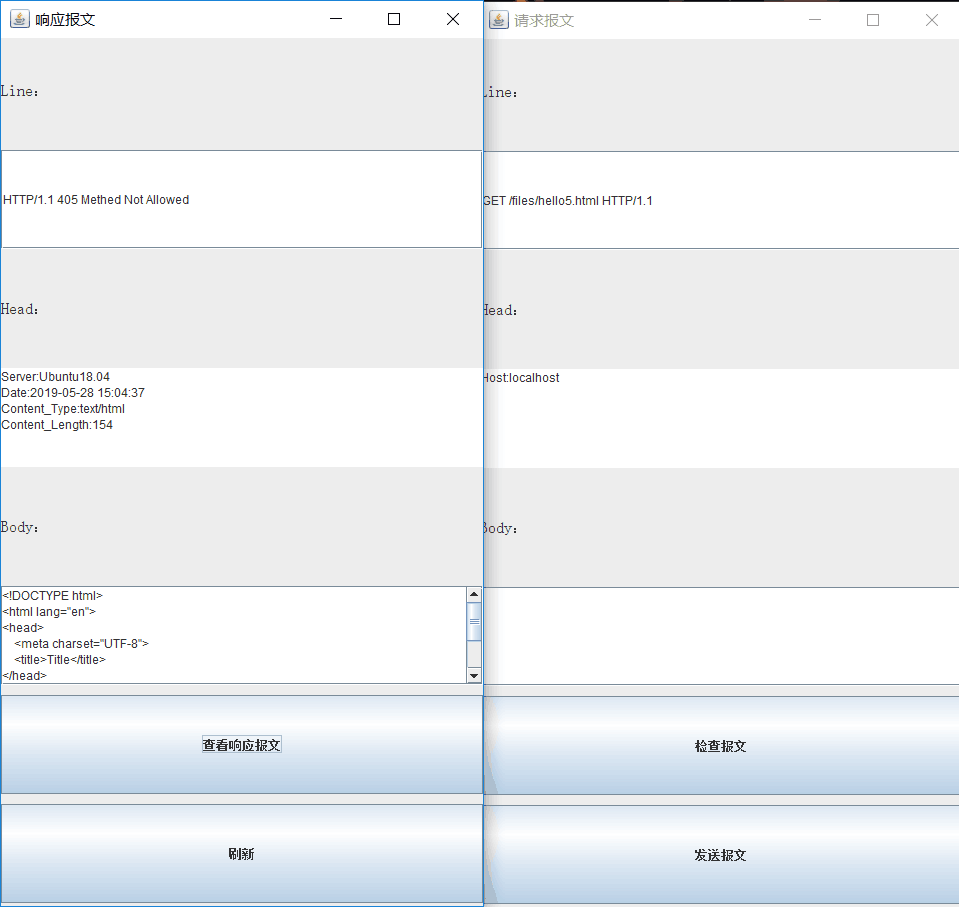
6、处理状态码 405

6.1：发送下一条请求，该请求的参数被设置为只支持POST请求



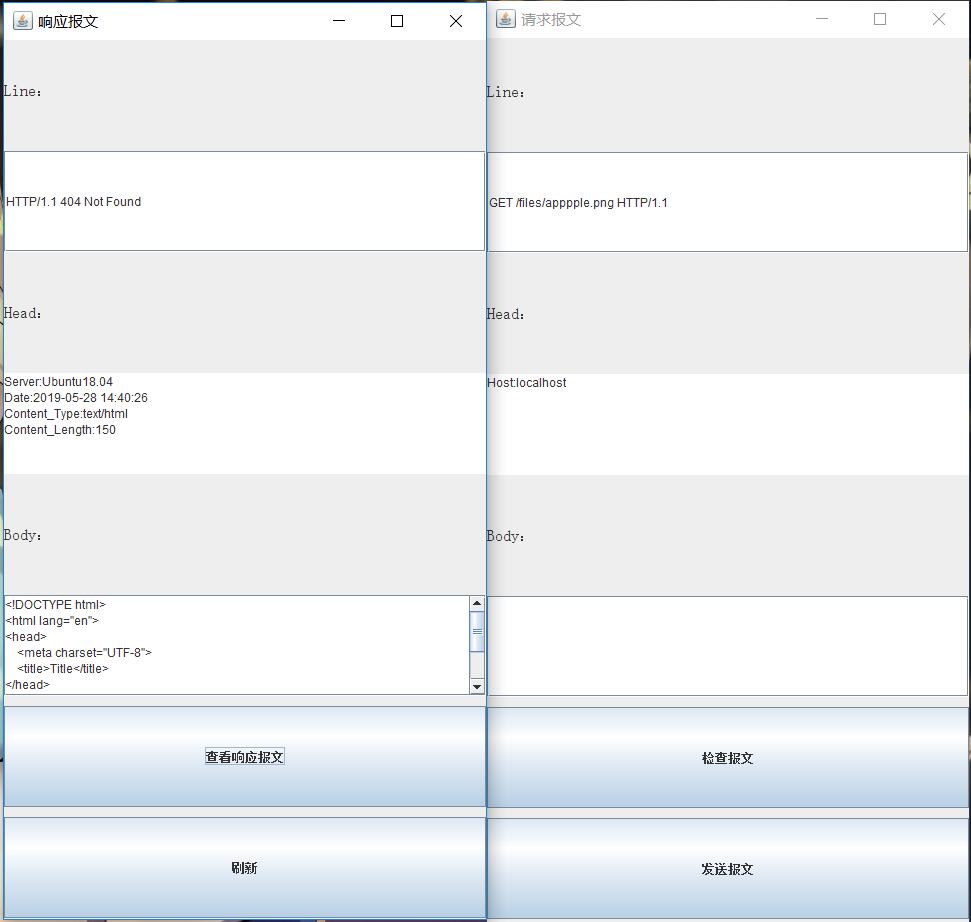


6.2：由于客户端请求中使用的是GET方法，该方法不被允许，于是返回状态码405



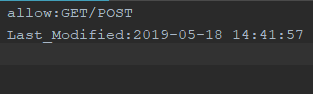
7、404状态码

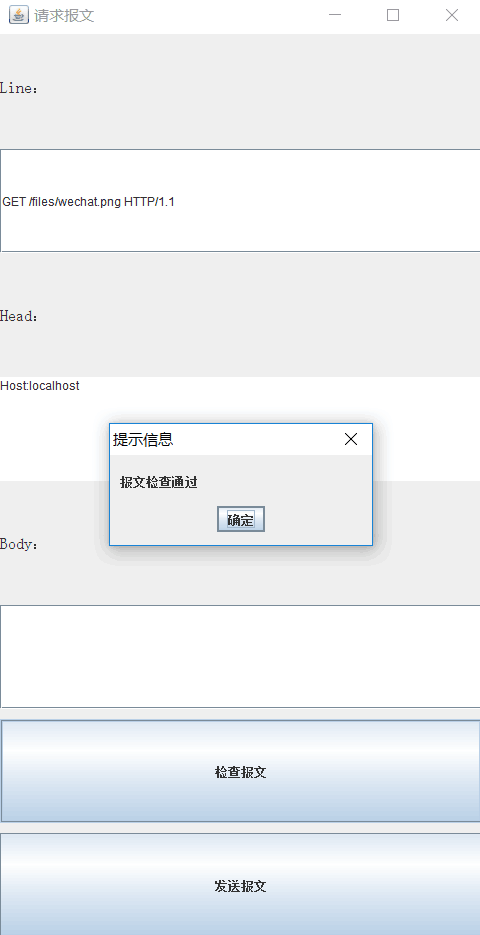
7.1：输入错误的文件名，系统找不到指定文件，于是返回了状态码404：



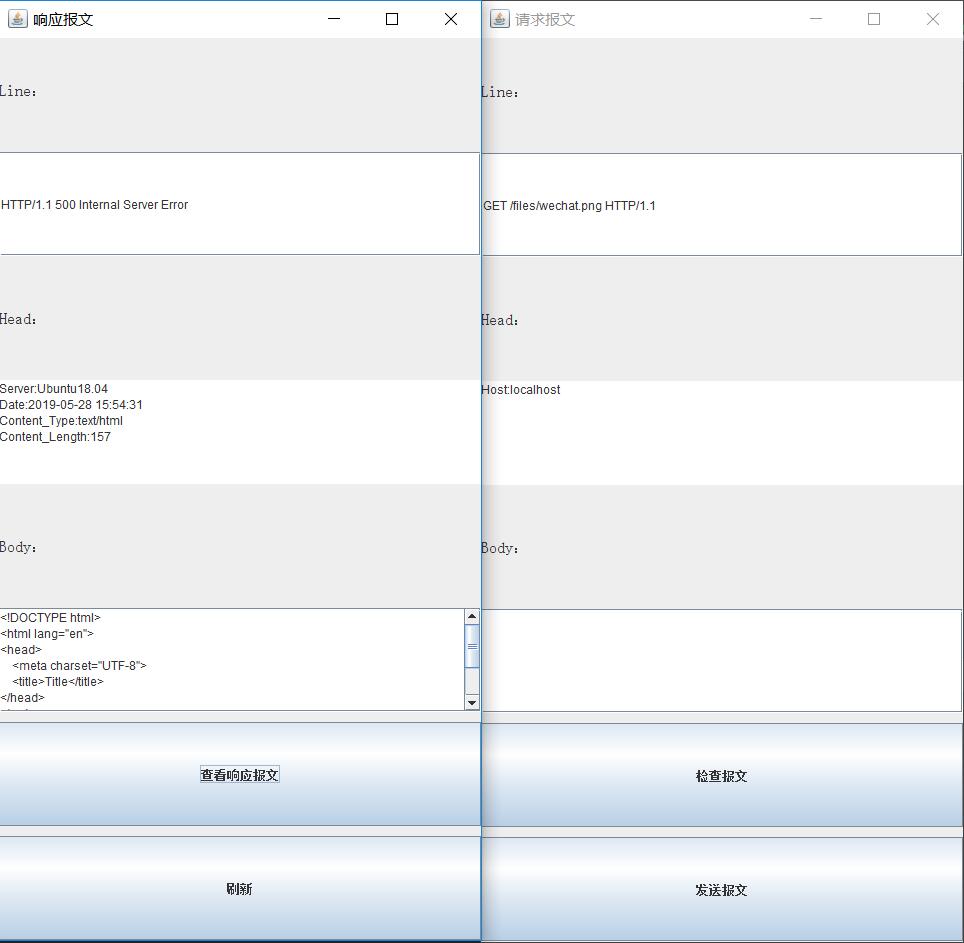
8、处理状态码500

8.1：此报文没有设置重定向这一参数，来测试状态码500

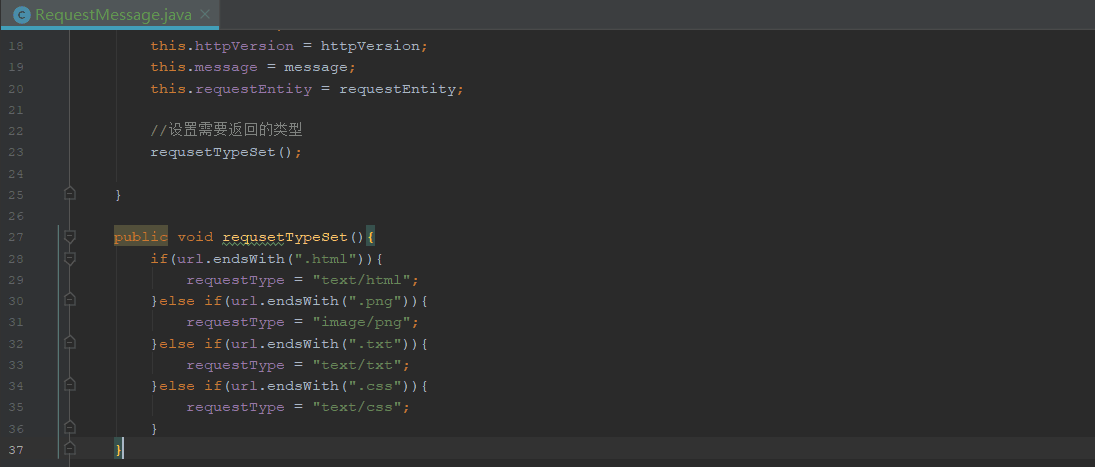




8.2：因重定向参数没有设置值，导致服务器内部出现空指针错误，于是返回状态码500



9、至少支持三种类型的文件，且包括了非文本文件(如.png格式的文件)



五、实验总结

通过完成实验，学习了简单的socket编程，了解到一些基础的客户端与服务端网络传输用到的协议，更重要的是对于平时我们都会遇到的部分状态码有了更深刻地理解，不在是表面上含义的理解，而是在原理层次上的理解。当然，这次实验也遇到了一些困难，其中最麻烦的就是传输MINE类型为图片处理。

实验要求实现三种MINE类型的报文传输，我们选择的时html/txt/png，其中html和txt格式的文件最容易传输，在实验中两者区别并不大，如果时相应报文的话，直接将String类型的文本转换为byte,然后塞到输出流中即可，客户端读取可以直接从输入流之中得到；当然，png格式也是这样，但是直接将png格式转为byte,然后通过输出流输出，最后客服端读入，其实读入的是一堆乱码，没法转换为正常png格式的图片。通过学习，了解到一种类似公钥和私钥的的处理办法；在服务端首先将图片读入到Byte类型的data中，然后Base64.encode(data)进行加密，就可以直接传输了；到达客户端以后，客服端再使用Base64.decode(data)进行解密，之后输出为图片就可以了，这样就确保传输过后图片可以正常得到。