

**NANCHANG UNIVERSITY**

**网络编程实训大作业报告**



**题 目:** 远程屏幕监控与控制软件

**学 院：** 软件学院

**专 业：** 信息安全

**班 级：** 193班

**完成人数：** 1人

**人 员：** 8003119100 丁俊

**起讫日期：** 2021.9.20-2021.10.7

**任课教师：** 邹春华 职称： 副教授

**完成时间：** 2021.6.18

**填表日期：** 2021 年 10 月 7 日

目录

[一、选题和需求分析 3](#_Toc84355658)

[1.1选题 3](#_Toc84355659)

[1.2需求介绍 3](#_Toc84355660)

[1.2.1 系统功能需求 3](#_Toc84355661)

[1.2.2系统环境 3](#_Toc84355662)

[二、软件结构和分析设计 4](#_Toc84355663)

[2.1 程序文件结构 4](#_Toc84355664)

[2.2 整体分析设计 5](#_Toc84355665)

[2.2.1类图分析 5](#_Toc84355666)

[2.2.2流程介绍 7](#_Toc84355667)

[三、核心技术和功能实现 8](#_Toc84355668)

[3.1 Client实现 8](#_Toc84355669)

[3.1.1ClientUI界面 8](#_Toc84355670)

[3.1.2 按键信息KeyInfo 8](#_Toc84355671)

[3.1.3 Client.java 9](#_Toc84355672)

[3.1.4 图片发送线程ImageThread 10](#_Toc84355673)

[3.1.5 接收文件线程receiveFile 11](#_Toc84355674)

[3.1.6 处理控制消息getAction 11](#_Toc84355675)

[3.1.7 监听事件 13](#_Toc84355676)

[3.2 Server实现 14](#_Toc84355677)

[3.2.1 按键信息类KeyInfo 14](#_Toc84355678)

[3.2.2 服务器界面Server 14](#_Toc84355679)

[3.2.3 接收截图线程receivePicThread 14](#_Toc84355680)

[3.2.4 监控界面showUI 15](#_Toc84355681)

[四、软件运行结果 19](#_Toc84355682)

[五、体会和总结 22](#_Toc84355683)

# 一、选题和需求分析

## 1.1选题

本次软件项目可以实现对同一局域网中的任一台(或多台)主机进行屏幕监控和远程控制功能，比如鼠标和键盘等操作。

## 1.2需求介绍

### 1.2.1 系统功能需求

通过网络编程socket连接服务机和客户机，客户机需要将当前的屏幕显示的内容抓取为图片；将捕捉的图片发送到远程控制主机；远程控制主机接收并在本地显示，以上可以实现服务机对客户机的屏幕监视，将客户机桌面上的内容同步显示在服务机上。服务器端连接客户机后可以向其传送文件，在服务器的监控窗口进行鼠标和键盘操作时，客户机收到之后也会做出相应的操作，并在其界面显示出来。所以这些功能都将采用多线程实现。

### 1.2.2系统环境

开发语言: java语言，面向对象编程

操作系统: Microsoft Windows 10

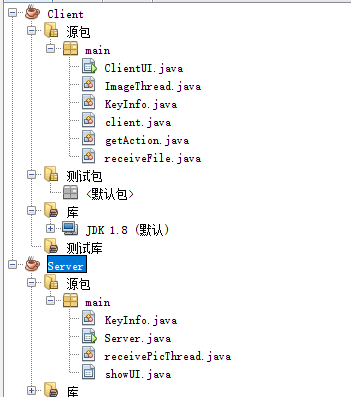
JAVA开发包: JDK1.8

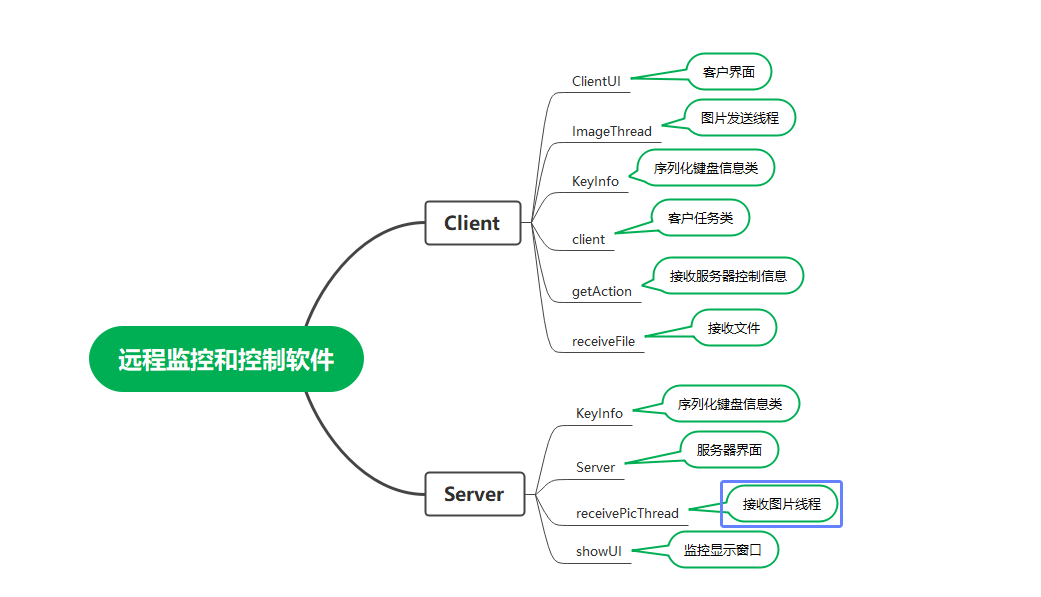
开发平台: NetBeansIDE 8.2

核心技术：网络编程Socket和传输流的应用

# 二、软件结构和分析设计

## 2.1 程序文件结构





## 2.2 整体分析设计

### 2.2.1类图分析

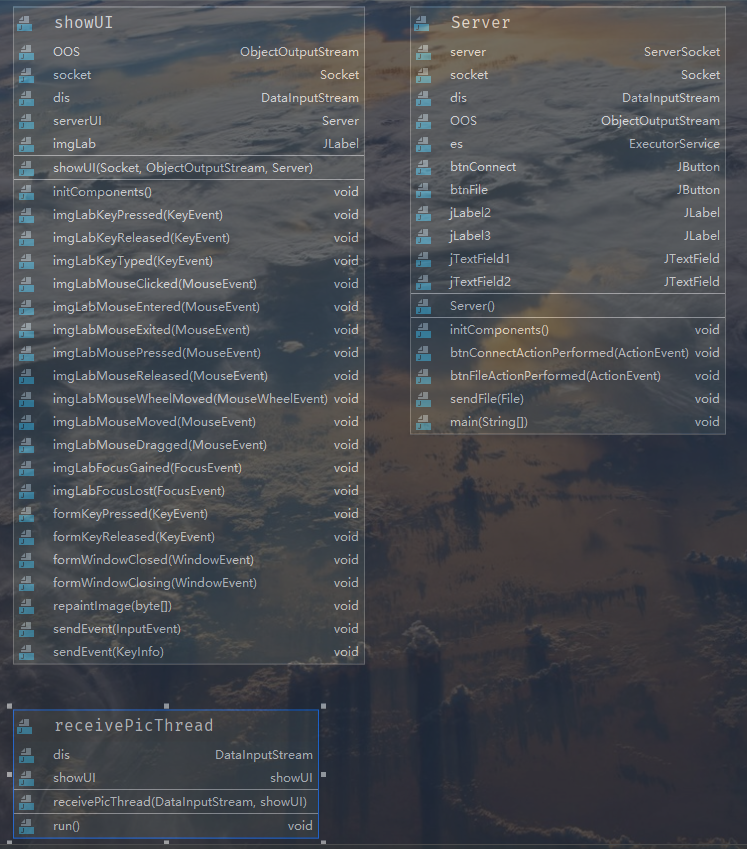
**Client部分**



ClientUI.java定义相关的组件和按钮响应事件；getAction函数中定义了Robot对象用来控制本机屏幕，还有处理相关事件的响应函数；KeyInfo封装了键盘的按键信息并实现了Serializable接口，便于用对象流进行传输；

receiveFile.java是接收文件的线程，用于接收从服务端发过来的文件封装信息；ImageThread是一直循环发送当前屏幕截图给服务器的线程，通过将截图信息转换成字节byte传输给服务器，还可设置每秒发送的图片数量(帧率)。

client是任务对象，内置启动任务线程的方法。



Server类是服务端界面，设置端口和地址，连接成功后创建对应连接的Socket和传输流；showUI是显示监控画面，并设置了很多监听并将操作发送给客户机，其中鼠标监听是被添加在showUI界面的imgLab(Jlabel类型)上的，键盘监听是被添加在showUI(Jframe类型)上的。

receivePicThread类用于接收客户机传送过来的图片并显示在showUI界面上。

### 2.2.2流程介绍

**服务端 客户端**

发送文件，建立连接 接收文件，发起连接

接收图像，发送控制信息 发送图像，接收控制信息

接收文件

发送文件

文件事件

文件发送线程

按钮：启动服务器

按钮：上传文件

事件类型

鼠标滚轮

鼠标事件

键盘事件

接收文件事件

做出事件响应

判断事件

发送窗口的鼠标和键盘事件

监控屏幕

截图发送

传输图像线程

接收控制信息线程

循环接收

创建任务对象

按钮：连接服务器

结束按钮功能

启动图像接收线程

创建监听窗口

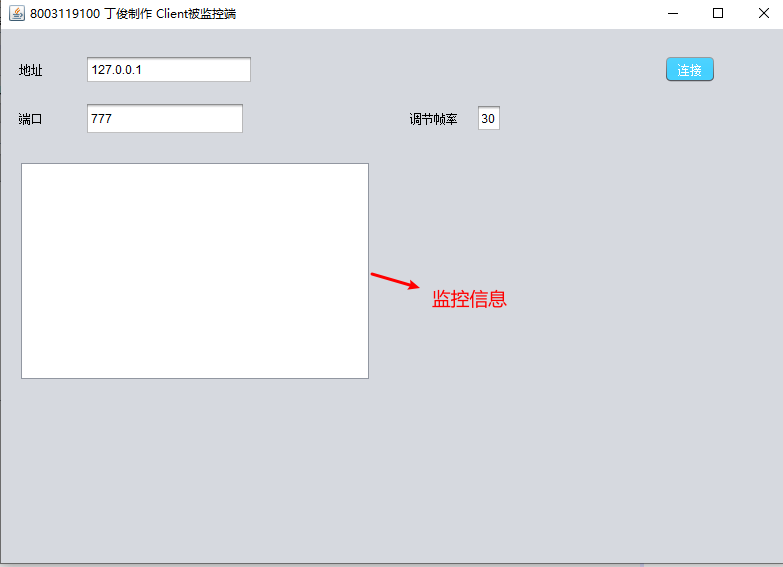
创建监听线程

启动TCP连接

# 三、核心技术和功能实现

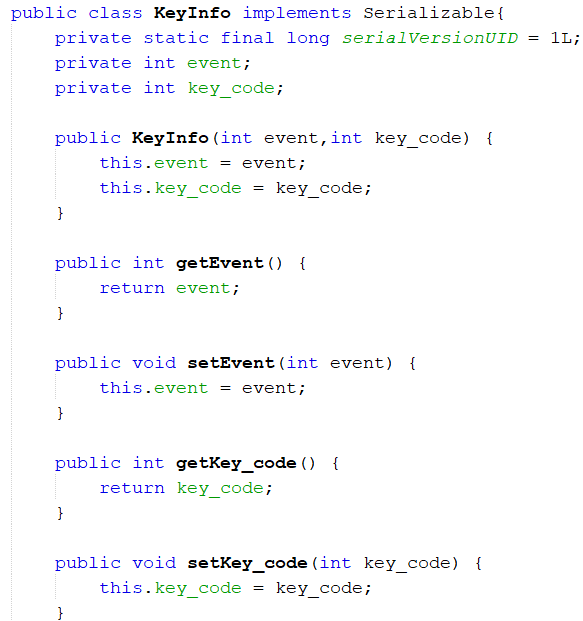
## 3.1 Client实现

### 3.1.1ClientUI界面

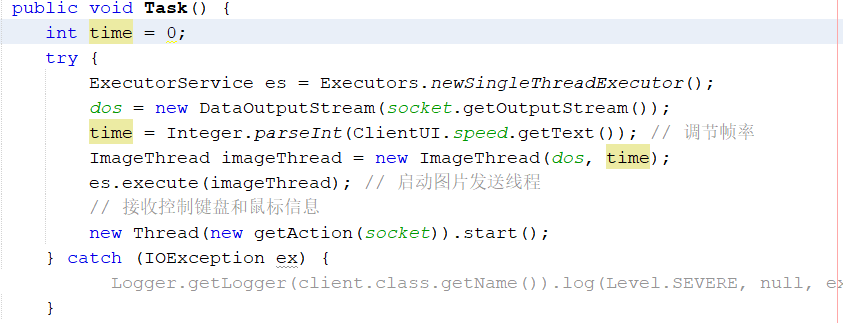


### 3.1.2 按键信息KeyInfo

KeyInfo实现了Serialzable接口，定义按键类型和按键号码，因为在javaswing和socket中不能单纯的通过socket流发送按键事件，所以这里要求实现序列化。



### 3.1.3 Client.java



这里新建一个client类在其中定义Task()方法执行imageThread图片发送线程、getAction处理控制信息线程，传入对象输出流和线程刷新值给ImageThread图片接收线程处理图像数据

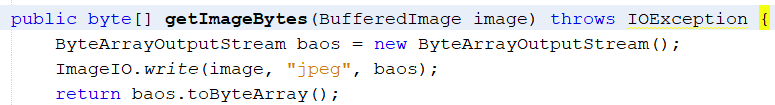
### 3.1.4 图片发送线程ImageThread

1、使用Robot类获取当前屏幕截图

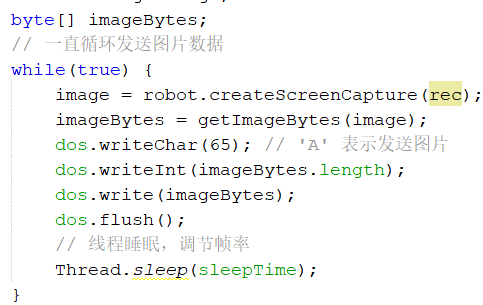




2、将image图片对象转换成字节byte，ImageIO是专门用来读写图片的图像流，可以很好地处理图片



3、使用socket构建的DataOutputStream数据流的write方法写入经过图片转换的字节数据，并且可以通过Thread调节图片数据写入的速度，调节帧率



### 3.1.5 接收文件线程receiveFile

用来接收从服务器传来的文件，因为Server服务端那边是通过输出对象流ObjectOutputStream写入Map键值对的方式传输文件的，所以client这边使用Map进行接收。

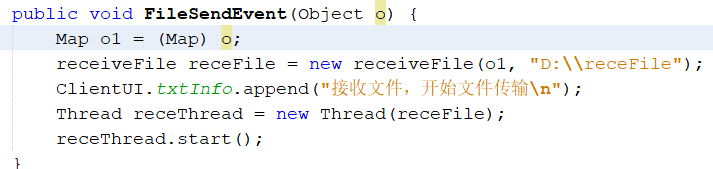


### 3.1.6 处理控制消息getAction

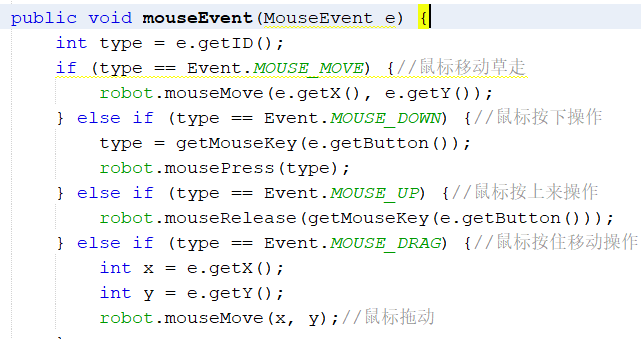
定义了一些响应不同类事件的处理方法，在run方法中一直循环接收来自服务器的对象讯息，通过对象输入流读取对象进而判断对象类型，执行相关的操作，分别有鼠标滚轮事件MouseWheelEvent、键盘事件KeyEvent、鼠标事件MouseEvent、发送文件事件FileSendEvent。

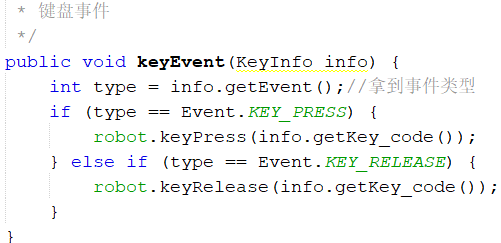


当触发文件发送事件时，立即启动文件接收线程进行文件接收。



当接收到鼠标和键盘等事件时，通过Robot对象和javaswing内置方法获取鼠标键盘信息，然后直接进行相关操作。





### 3.1.7 监听事件

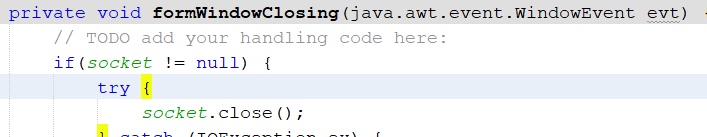
**连接按钮监听**

获取指定服务器地址和端口构建socket，启动任务对象类



**关闭按钮监听**

关闭socket套接字资源

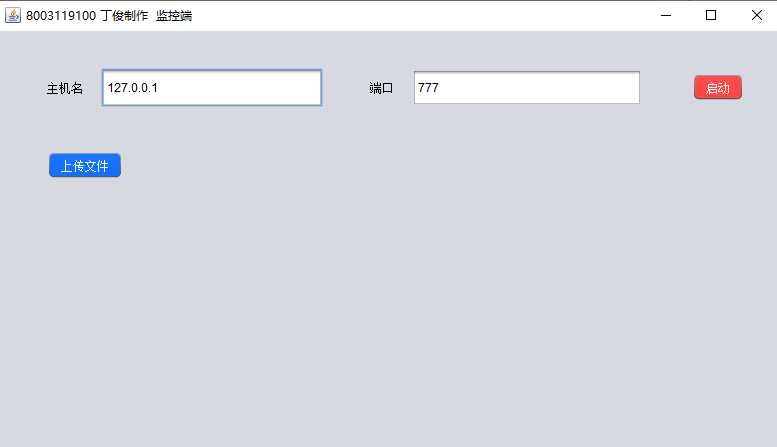


## 3.2 Server实现

### 3.2.1 按键信息类KeyInfo

同Client中的keyInfo类似

### 3.2.2 服务器界面Server



### 3.2.3 接收截图线程receivePicThread

使用数据流DataInputSteam读取流中的字节数据，调用showUI中的repainImage(Byte)方法显示在showUI监控页面中。

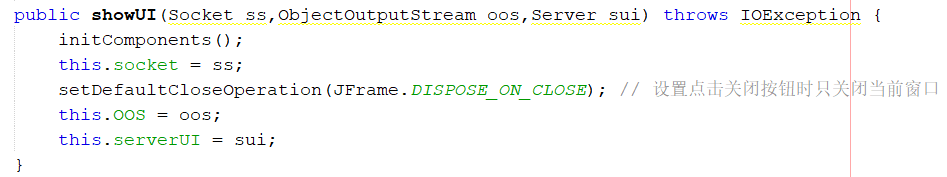


### 3.2.4 监控界面showUI

在showUI界面中添加了一个Jlabel标签ImgLab用于实时显示接收到的图片数据。

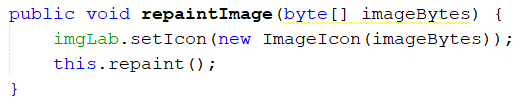
**构造方法**

初始化socket和对象输出流以及传入ServerUI。



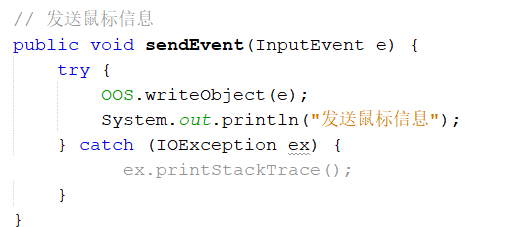
**显示函数**

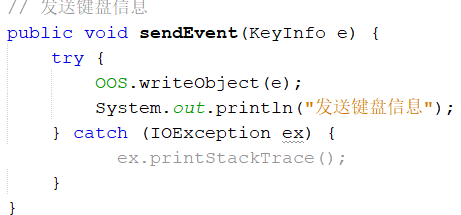
为imgLab设置图标图像setIcon(ImageIcon)，直接使用图片字节数据初始化构造ImageIcon，repaint()方法用来刷新和清空画面。



**发送控制信息**

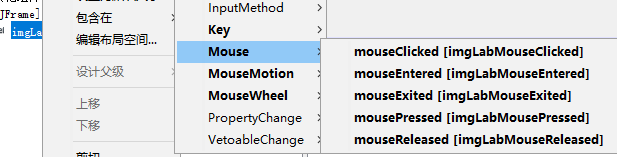
通过对象输出流写入鼠标事件和序列化的按键信息对象





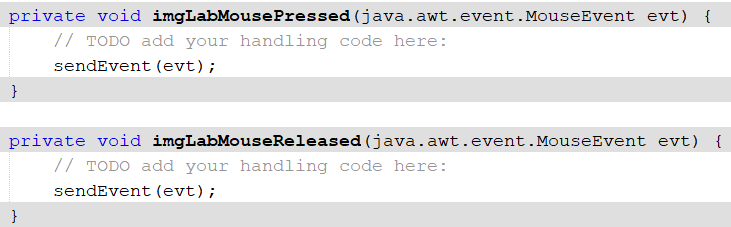
**监听事件**

在imgLab中添加了鼠标滚轮、鼠标事件(点击、移动、拖拽)。



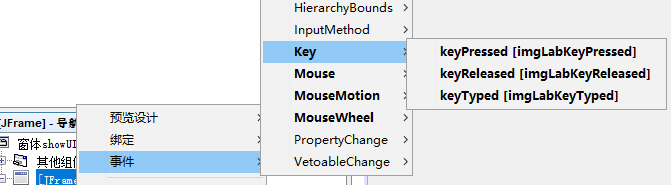




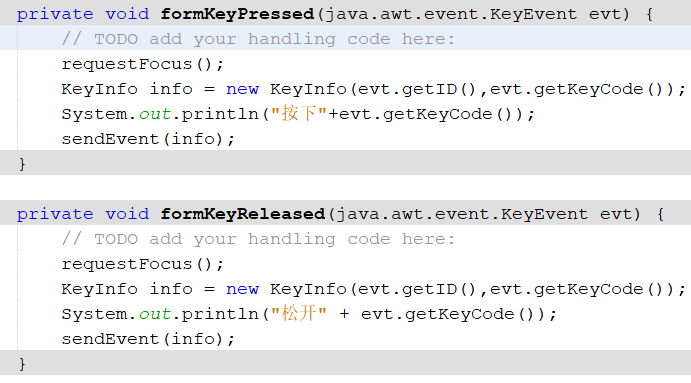


当触发这些事件时，直接在监听函数中调用sendEvent()方法将控制信息发送给客户端。

另外在showUI(JFrame)中单独添加Key键盘监听事件



同样也是当按下或松开键盘某一按键就调用重载的sendEvent函数发送键盘按键信息。



**启动按钮监听**

1、创建serverSocket套接字

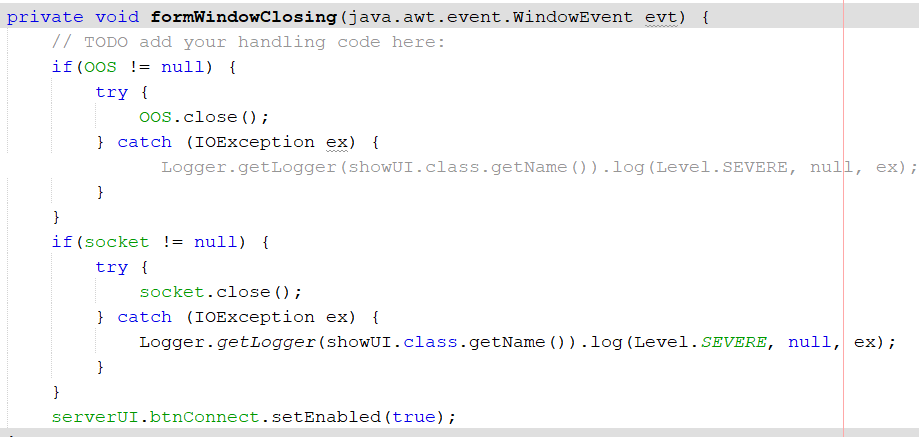


2、新建线程等待客户机连接



将socket和相应构建好的对象输入流dis传入showUI和receiveThread图像接收线程，并使用线程池启动线程。并且这里一直循环等待客户机连接，实现了**监控和控制多台客户机**的操作。

**关闭退出按钮监听**



**发送文件按钮监听**

点击“上传文件”按钮，开启文件选择器框组件，从而进行文件选择，获取当前选择到的文件对象，调用文件发送时间函数。

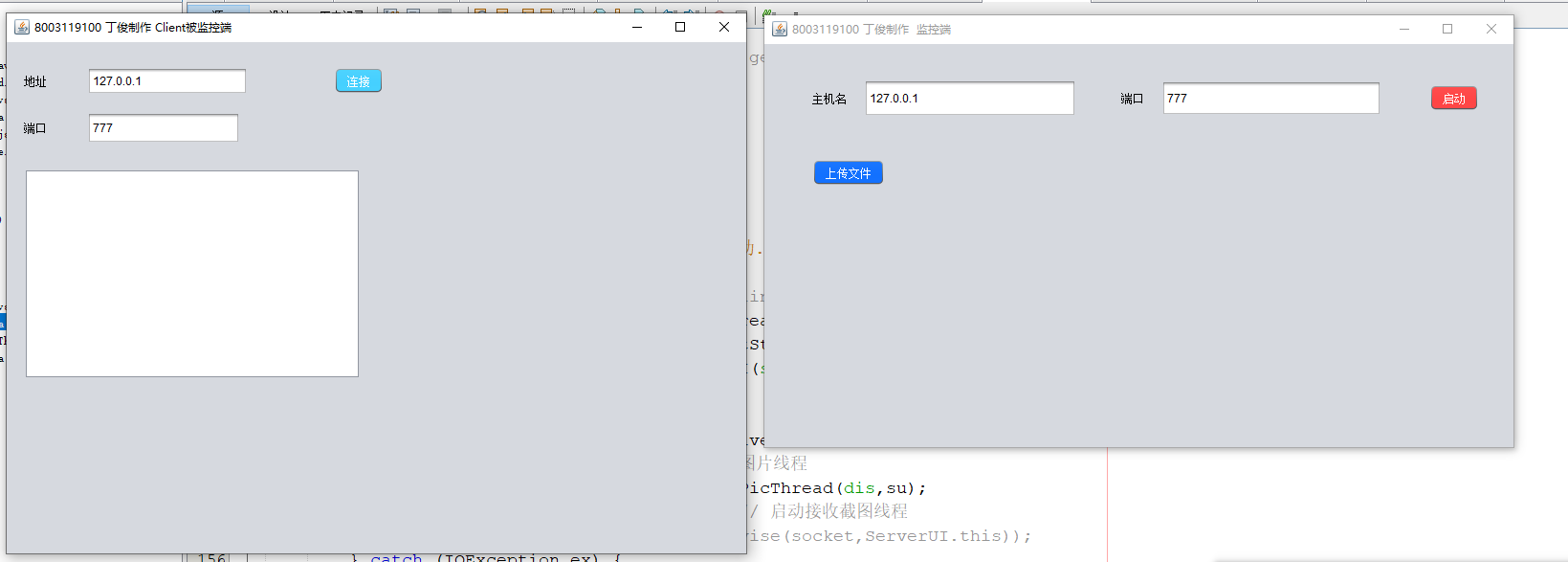


先使用文件输入流对象fis指向该文件对象file，然后利用字节数组输出流bos将从文件中读出的字节写到字节数组输出流bos中，最后dos.toByteArray()调用将流中的数据转换为字节byte类型，构建Map对象，通过OOS对象输出流写入Map数据。



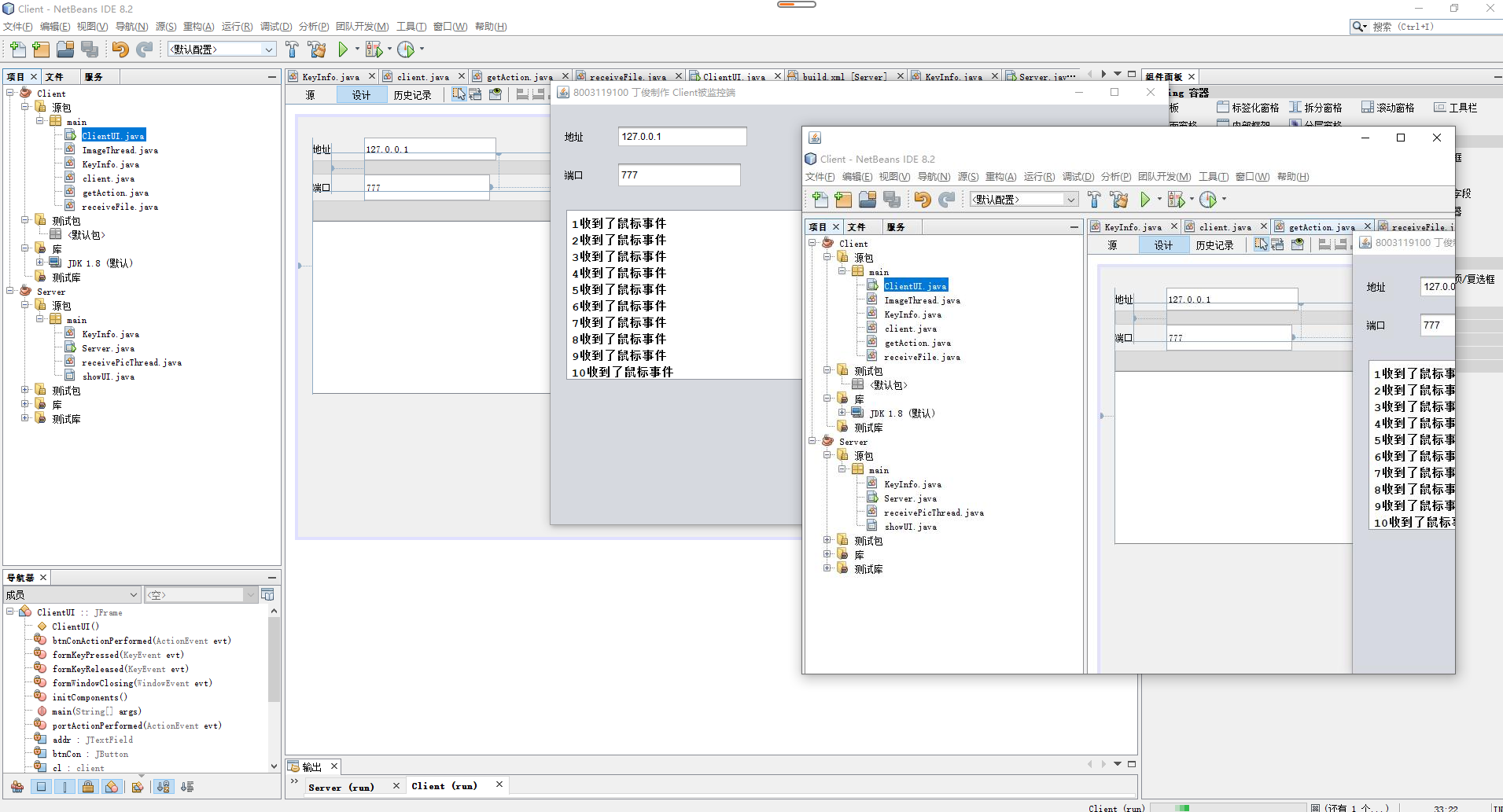
# 四、软件运行结果

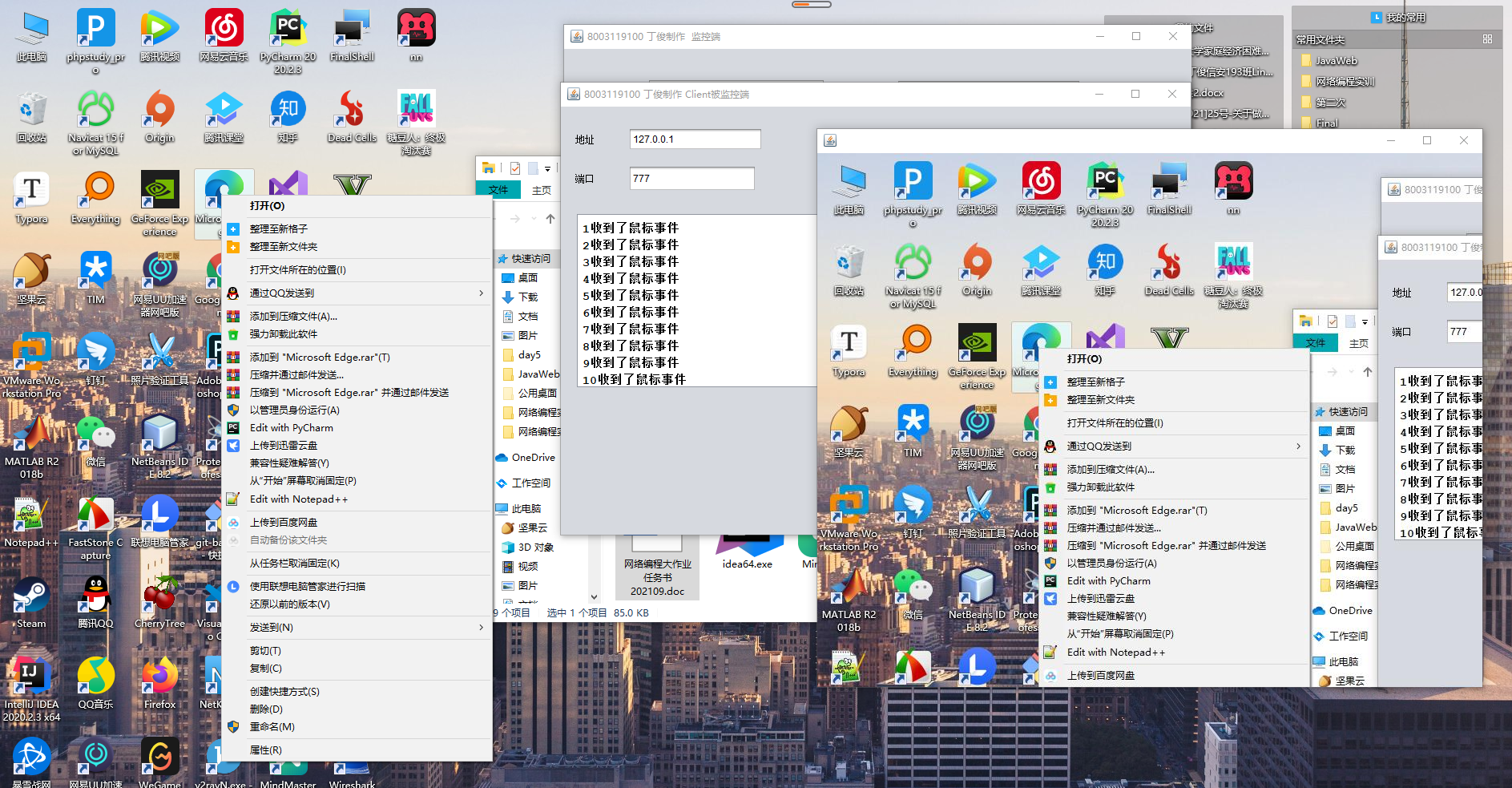
1、首先运行客户端和服务端

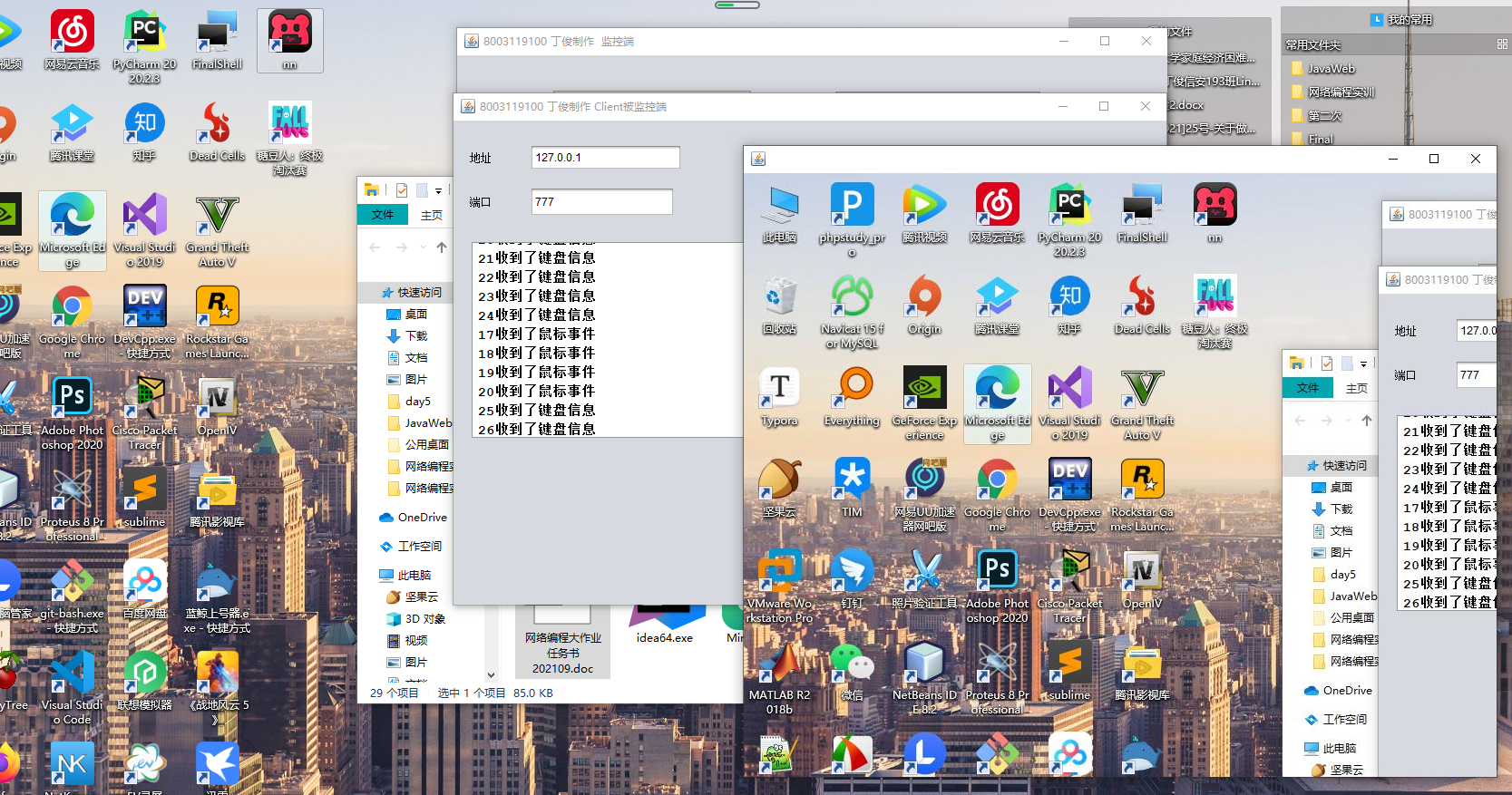


2、先启动服务器，再让客户机进行连接

连接成功之后会弹出showUI监控页面，并且鼠标和键盘一些操作也会显示在客户端的监控文本区域。

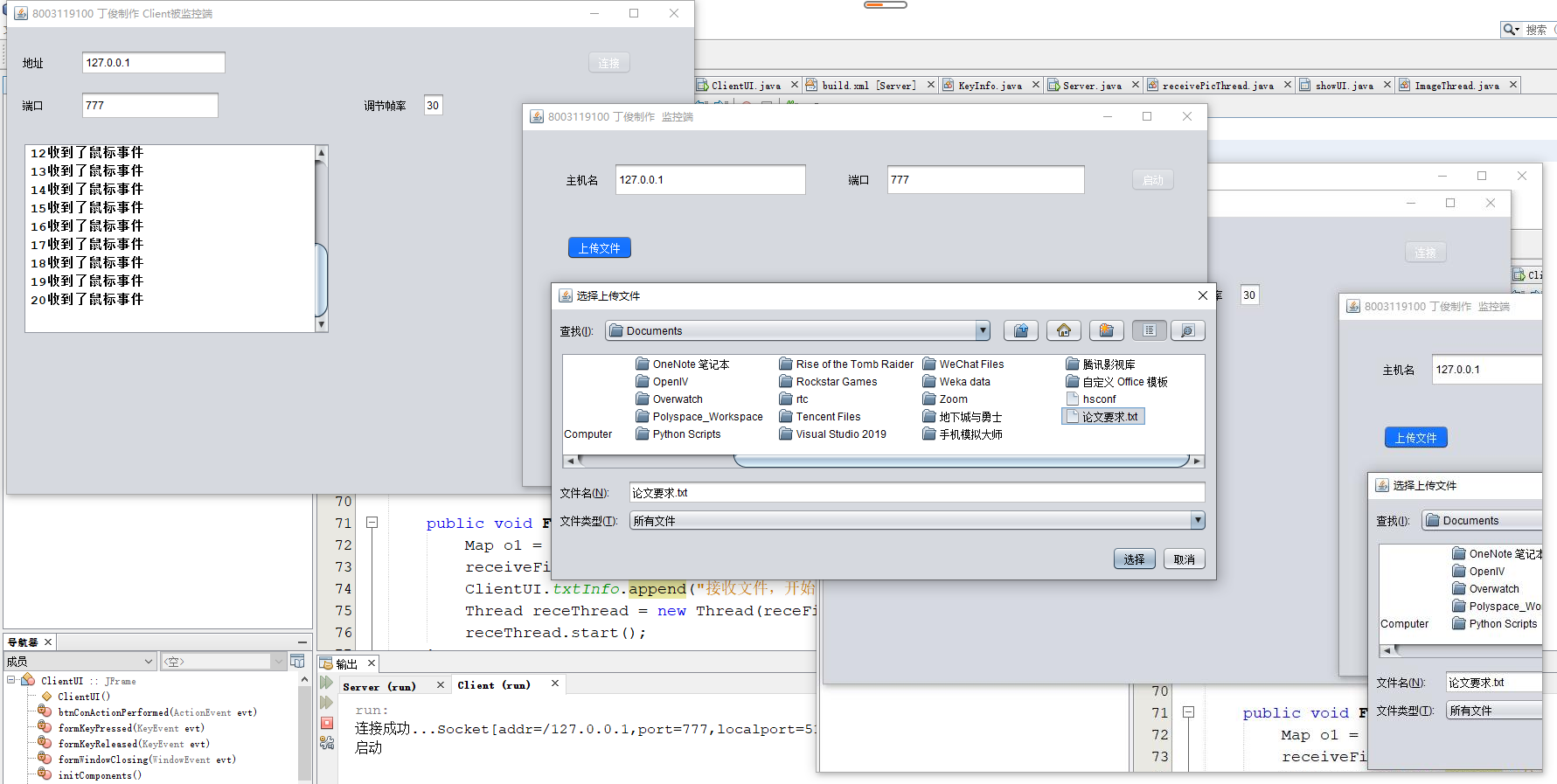




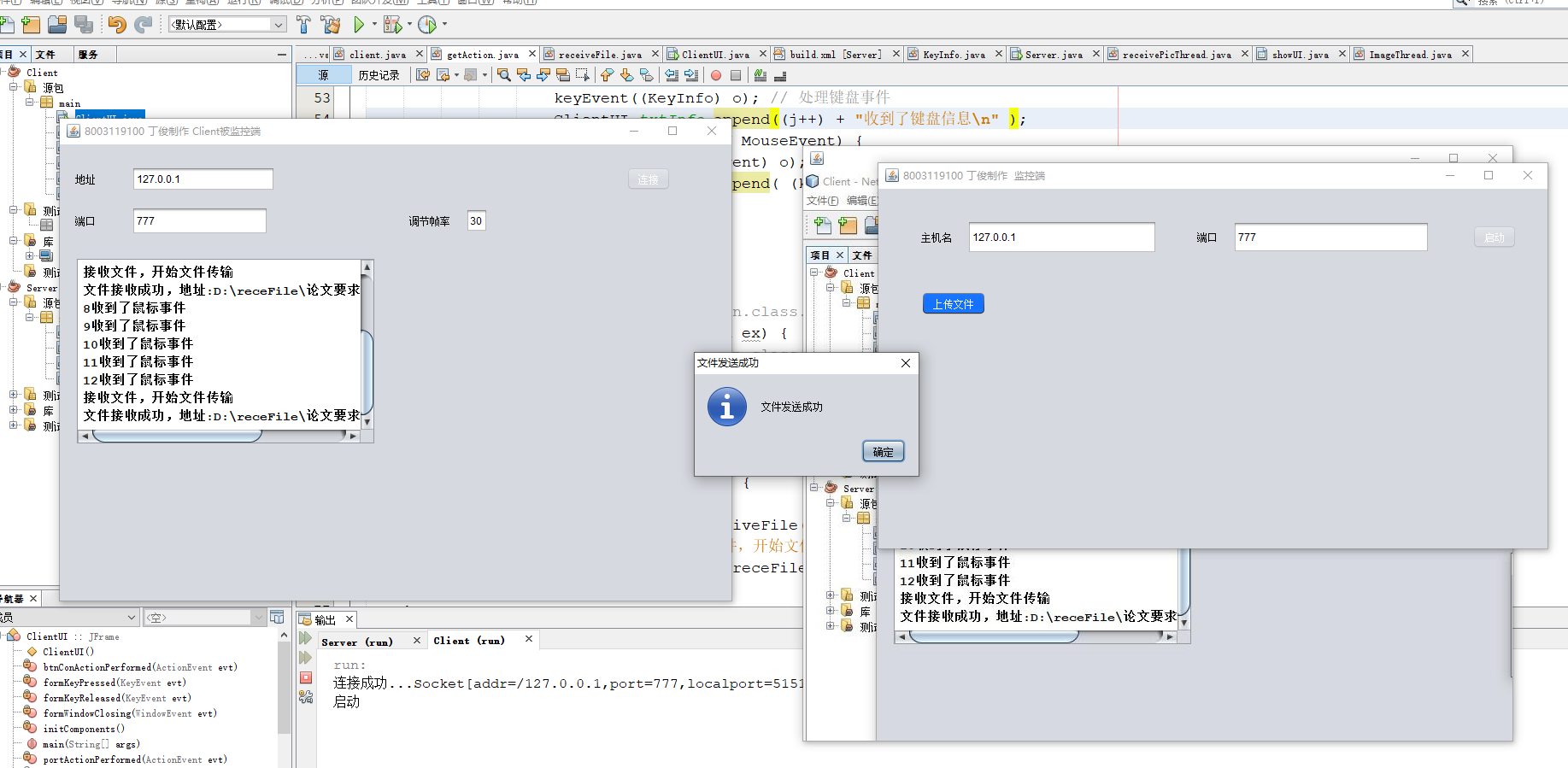


一台电脑对于控制功能展示不了，所以我特地录制了一个视频存放在文件夹中。

**传输文件**



**显示接收文件成功**



# 五、体会和总结

初学java之时，虽然接触过socket编程和一些流的使用，但当初对流的使用和socket“套接字”的理解并不是很深入，其实它是一种规定好的约定和协议，没有很神秘。通过这次网络实训中不断的学习和解决问题，也使得我更加的熟练掌握和运用网络编程知识，并能根据特定需求设计出满足要求的软件或程序。

本次大作业也遇到不少问题，比如线程卡死之类的现象，这是因为设计程序之初没有考虑到线程的协调问题，由于某些堵塞函数或操作使得程序无响应，这时应该开启或协调好新的线程来解决问题；键盘无法控制现象，由于socket不能直接传输javaswing中的键盘信息，所以我这里采用序列化接口将按键信息封装序列化，然后再通过对象流进行传输，再根据具体的对象数据来处理按键信息；监控画面太卡问题，起初我是把屏幕的截图图像信息通过对象序列化加上对象流传输的，在接收方的那头要先用对象输出流写入到文件中，然后再从文件中取出，这样会降低程序的效率，进而使得图片显示过慢造成卡顿现象，另外这样监控控制也不灵敏。后来我直接采用字节流的方式传输，用数据流操作处理字节数据再转成图像。

尽管这个软件还不够系统化，没来得及整合成一个完整的软件包，只能说很不完美，但对我来说也不乏是一种尝试，曾经以为很难的东西也终于在自己的手中实现了，很快乐也很充实。