

**Java语言与系统设计课程实验报告**

|  |  |
| --- | --- |
| 学生姓名 | 王云鹏 |
|  |  |
| 学生学号 | 8213180228 |
|  |  |
| 指导教师 | 郭克华 |
|  |  |
| 专业班级 | 物联网工程1802 |
|  |  |
| 完成日期 | 2020.6.1 |

**计算机学院**

**目 录**

[实验一 2](#_Toc500596551)

[一、目的与要求 2](#_Toc500596552)

[二、操作环境 2](#_Toc500596553)

[三、实验内容 2](#_Toc500596554)

[四、实验数据 2](#_Toc500596555)

[实验总结 2](#_Toc500596556)

[参考资料](#_Toc383552634) 3

# 实验一

## 目的与要求

学习Java界面开发基础知识。利用所学到的Java编程知识和编程技巧，设计一个实际的应用软件，初步认识软件设计的基本方法，提高进行工程设计的基本技能及分析、解决实际问题的能力，为后续课程和以后的工程实践打下良好的基础。

**用JavaSocket编程开发云笔记本（每位同学完成）。**

基本功能：该软件支持客户端从服务器下载用户之前保存的笔记。笔记以文本文档的形式保存在服务器中，笔记在客户端的界面上显示。该软件支持用户在客户端修改笔记，并通过保存按钮更新服务器的笔记文件，同一个用户可以在服务器中保存多个笔记。

1、首先，运行服务器。服务器运行之后，可以连接1个客户端。

2、运行客户端。用户可以在客户端的注册和登录界面输入用户昵称和密码，点击“注册”按钮，服务器将客户端输入的用户昵称和密码进行保存。用户在客户端输入用户昵称和密码后，点击“登录”按钮，服务器查询用户昵称和密码是否匹配，如果用户昵称和密码匹配，客户端即可出现功能界面。

客户端功能界面如下：

1、客户端功能界面设有“新建笔记”、“保存笔记”、“打开笔记”、“删除笔记”、“查询笔记、“笔记下载到本地”功能。

2、新建笔记与保存。客户端点击“新建笔记”按钮，在客户端界面，用户可以输入文本笔记。当用户点击“保存笔记”按钮，出现“笔记名称”输入框。当用户输入“笔记名称”后，笔记的文本内容在服务器中以文件保存，例如：服务器中的笔记文件命名为“用户昵称\_笔记名称.txt”。

3、打开笔记与修改保存。客户端从服务器获取该用户的所有笔记名称并显示。在用户选择笔记名称后，客户端从服务器中获取该笔记的内容展示，用户修改后可以进行保存，服务器端笔记内容更新。

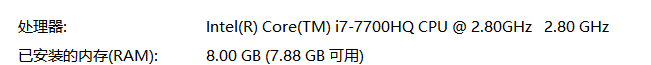
4、笔记删除。客户端可以选择某个笔记并删除，服务器将存储中的笔记文件进行删除。客户端随即更新。

5、笔记查询。客户端输入关键字，服务器全文查询笔记内容，将含有关键字的笔记进行展示，用户可以选择后，打开该笔记。

6、笔记下载到本地。用户选择某个笔记，在客户端点击“笔记下载到本地”按钮后，客户端允许用户选择存储笔记的文件夹路径以及笔记文件名称，将笔记下载到本地。

## 二、操作环境

硬件：



软件：

系统：Windows10

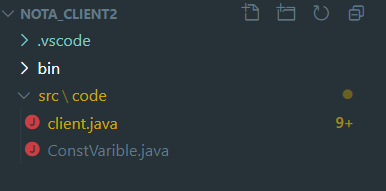
编辑器：Visual Studio Code

## 三、实验内容

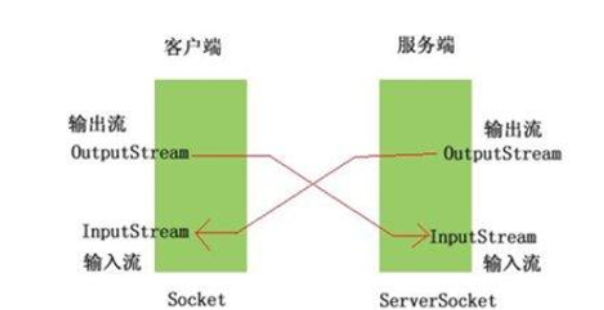
### Server代码文件结构



### Client代码文件结构如下

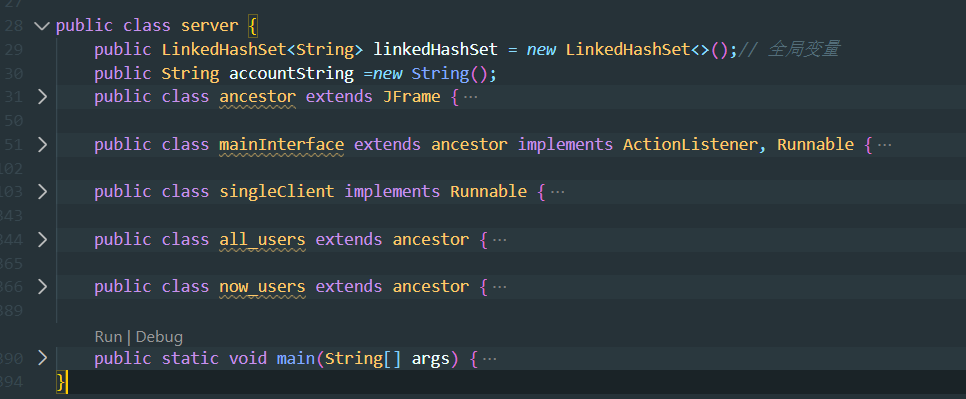


### socket遍程思想流图





### server.java



类：

ancestor：所有类的祖先类，在其中完成GUI布局设置

mainInterface：服务器的主界面，包含两个按钮“所有用户”和“在线用户”

singleClient：一个线程类，用来不断接受用户发来的连接要求，处理用户的需求，为主要的功能实现类

all\_users：完成主界面上的“所有用户”功能

now\_users：完成主界面上的“当前用户”功能

值得一提的函数：



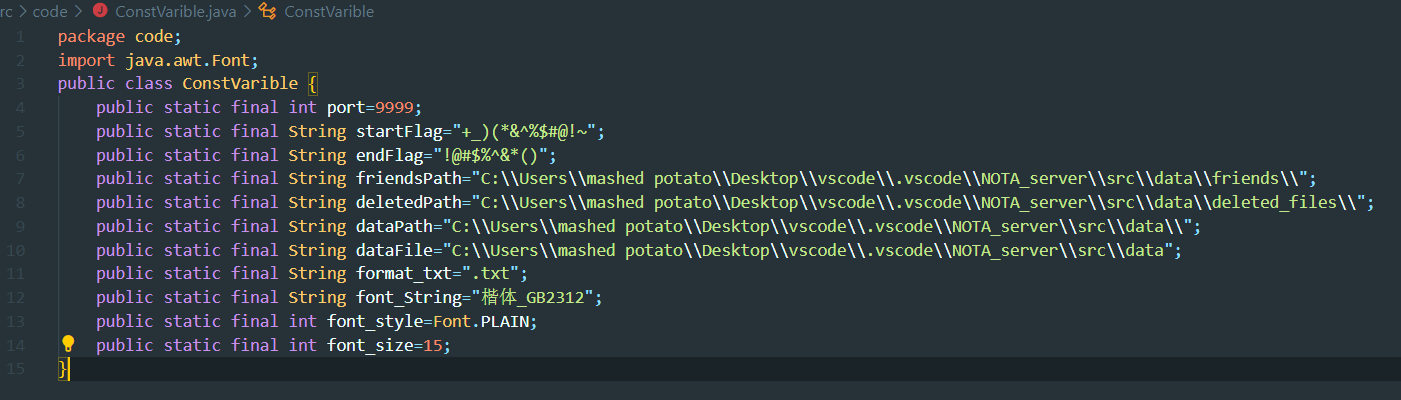
这是singleClient类中的run函数，用于不断接受用户发来的消息，根据首句消息判断用户需要的功能，执行相应代码

### fileOperation.java



fileOperation为一个工具类，封装了各种服务器端的文件操作函数，这些函数为了在服务器上存取和查看用户信息。

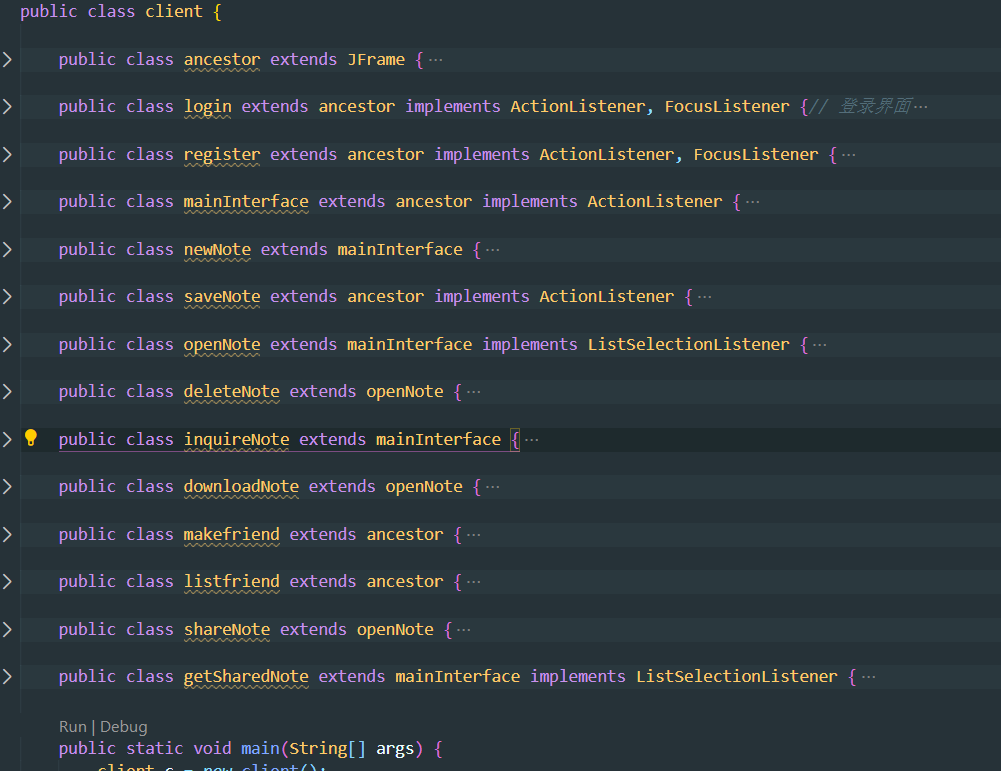
### ConstVarible.java（服务器端）



ConstVarible是一个常量类，封装了各种使用到的常量

包括文件存取地址，断句标识符，字体格式，文件格式等

### client.java

客户端的主要类，包括类：

ancestor：所有类的祖先类，进行了基本的GUI设置，完成了与服务器交流的socket处理

login：完成登录功能

register：完成注册功能

mainInterface：所有之后功能界面的父类，完成了进一步的GUI设置

newNote：新建笔记

saveNote：保存笔记

openNote：打开笔记

deleteNote：删除笔记

inquireNote：查询笔记（中的内容）

downloadNote：下载笔记

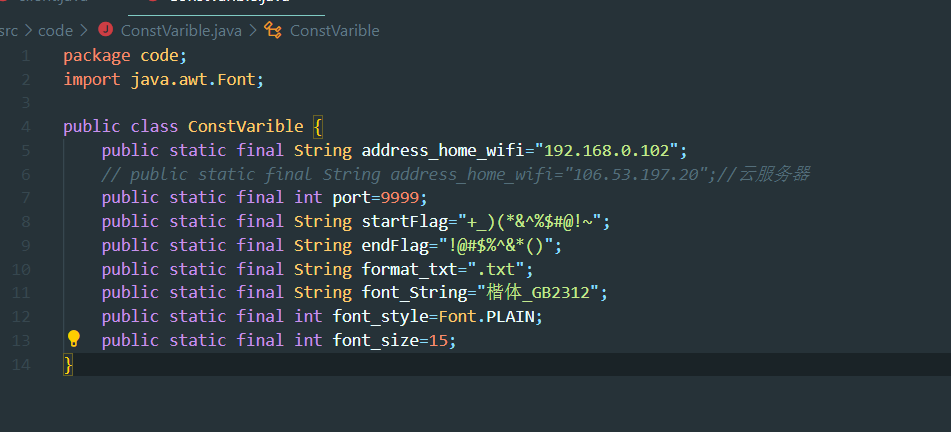
makefriend：加好友

listfriend：显示好友列表

sharefriend：向好友分享自己的笔记

getSharedNote：得到好友分享给自己的笔记

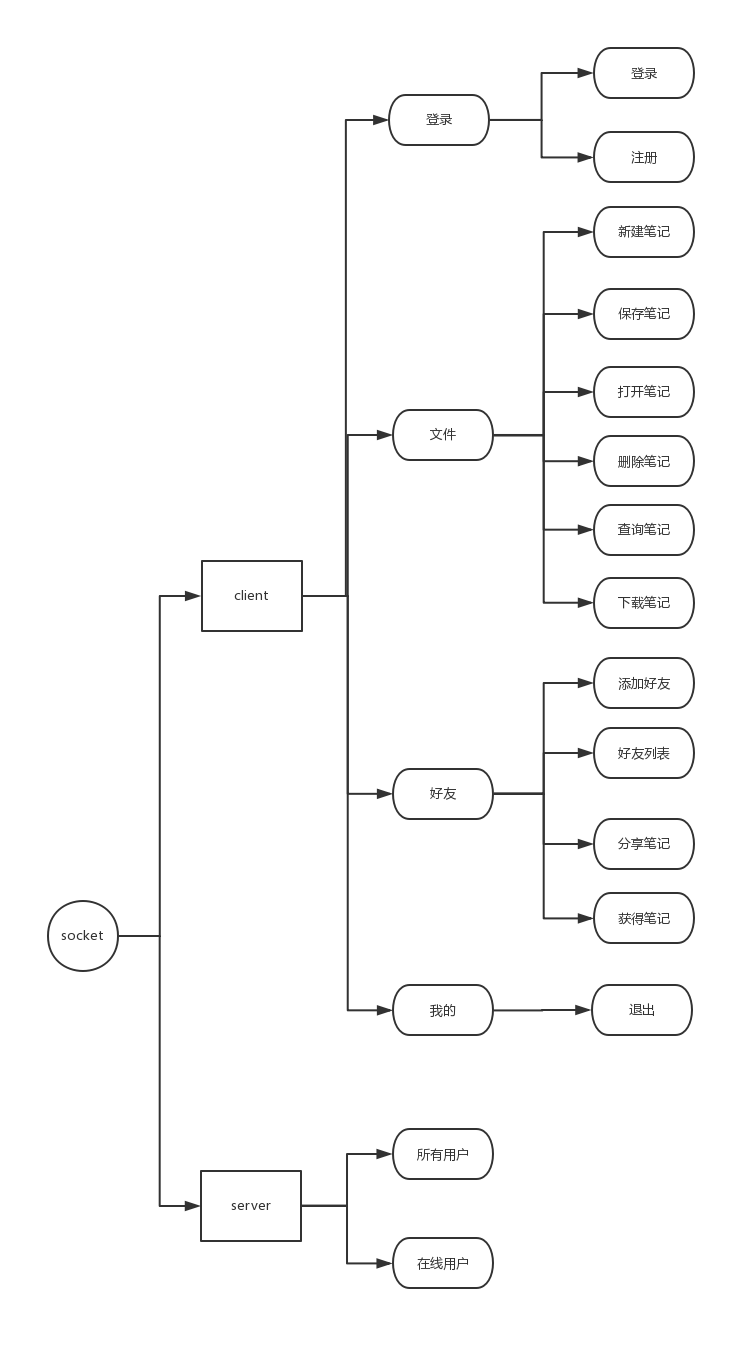
### ConstVarible.java（客户端）



ConstVarible为常量类，封装了各种client中用到的常量，包括服务器的IP地址，短句符，文本格式，字体格式等。

## 实验结果

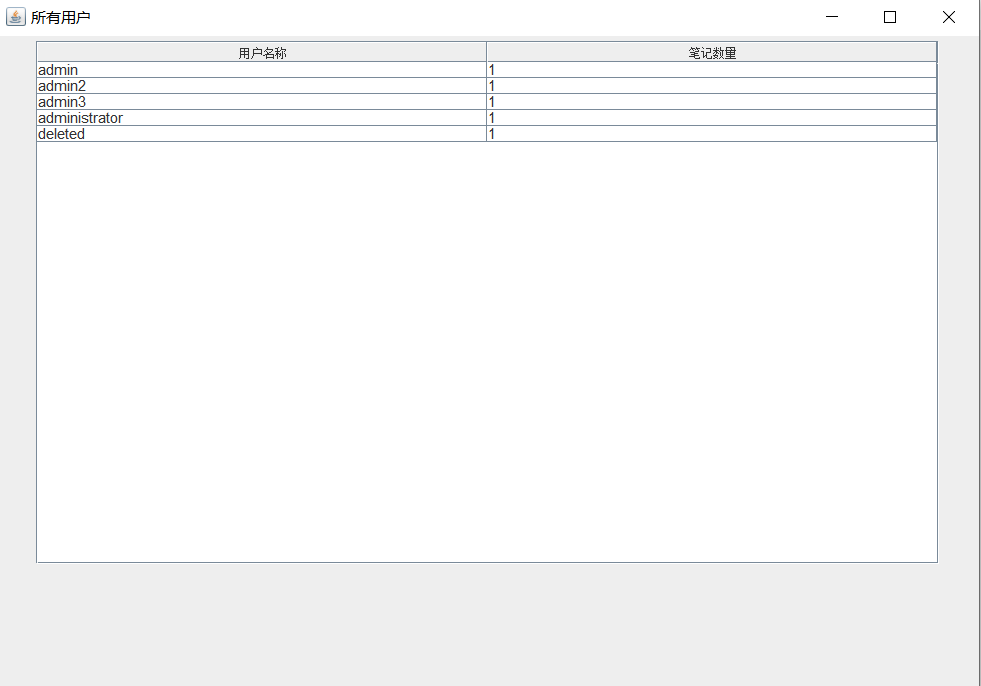
软件功能结构图



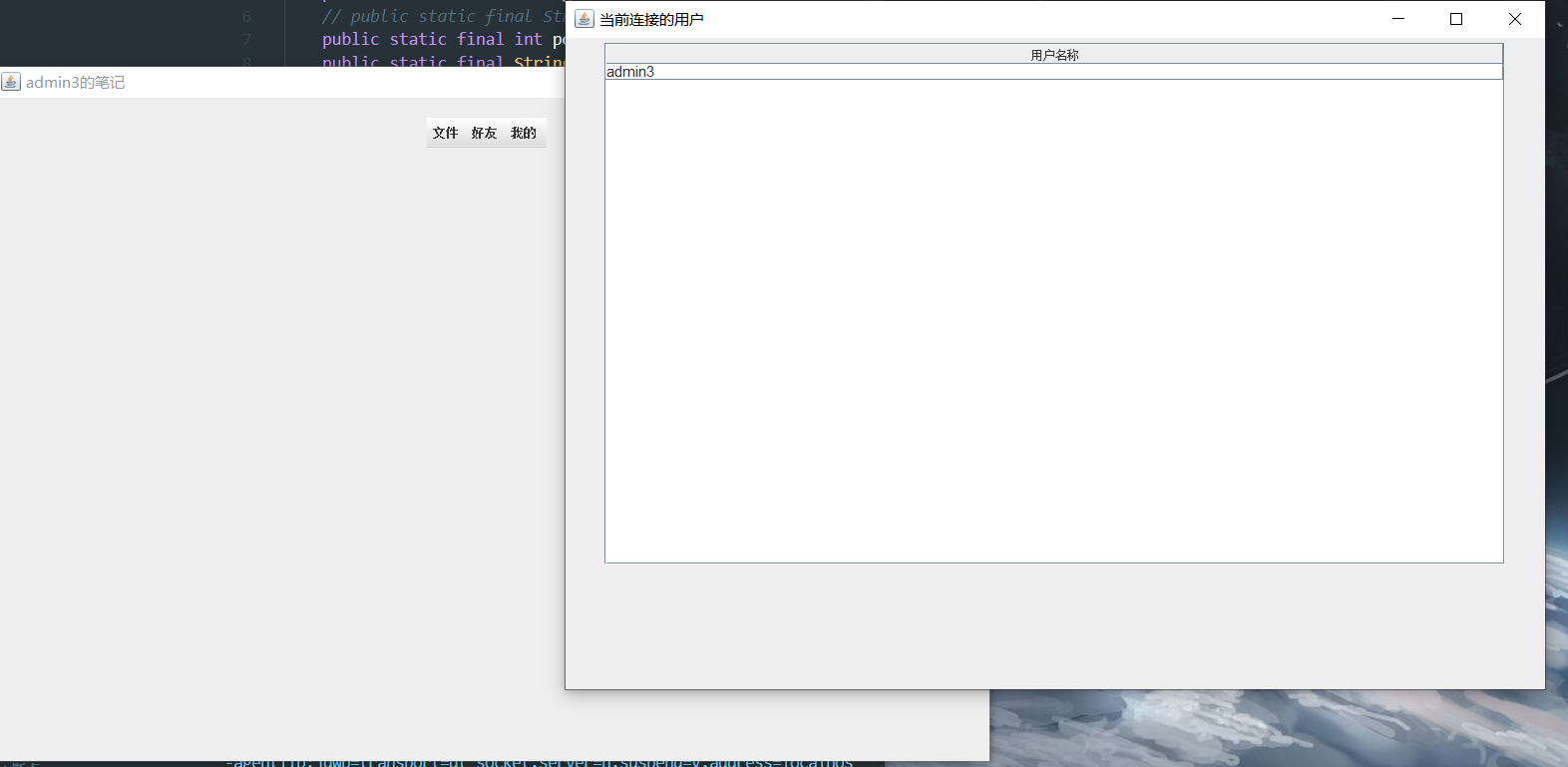
### Server



### Server.所有用户



### Server.在线用户

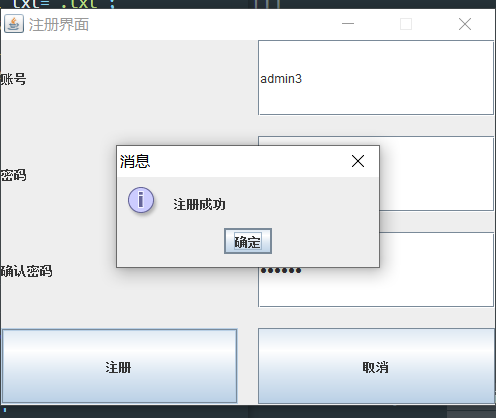


### Client.登录.登录



### Client.登录.注册

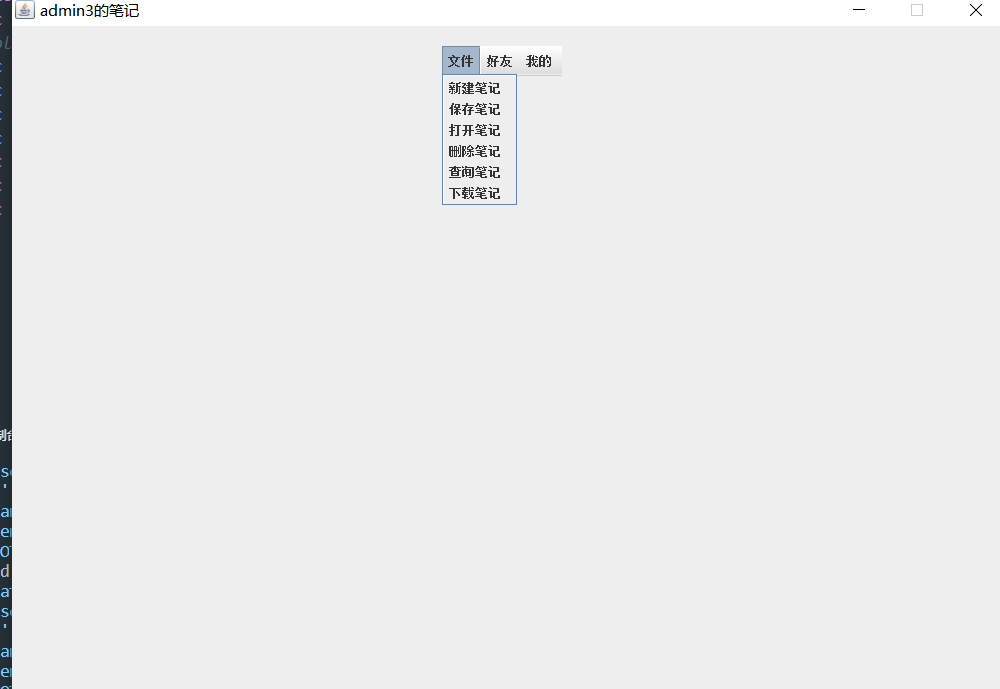




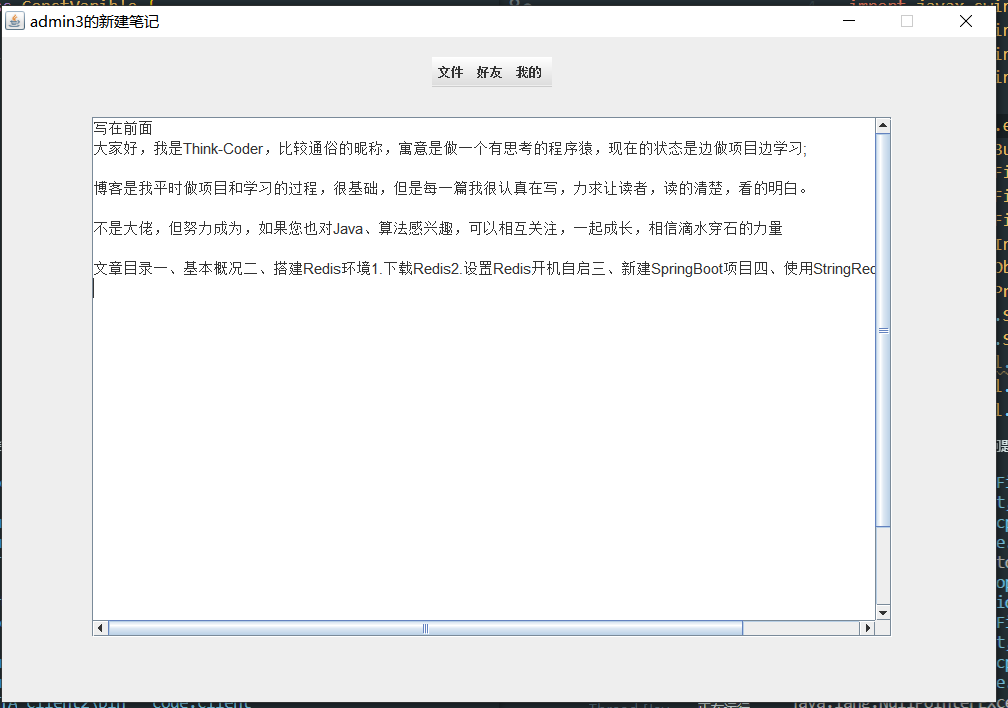
### Client.文件/好友/我的



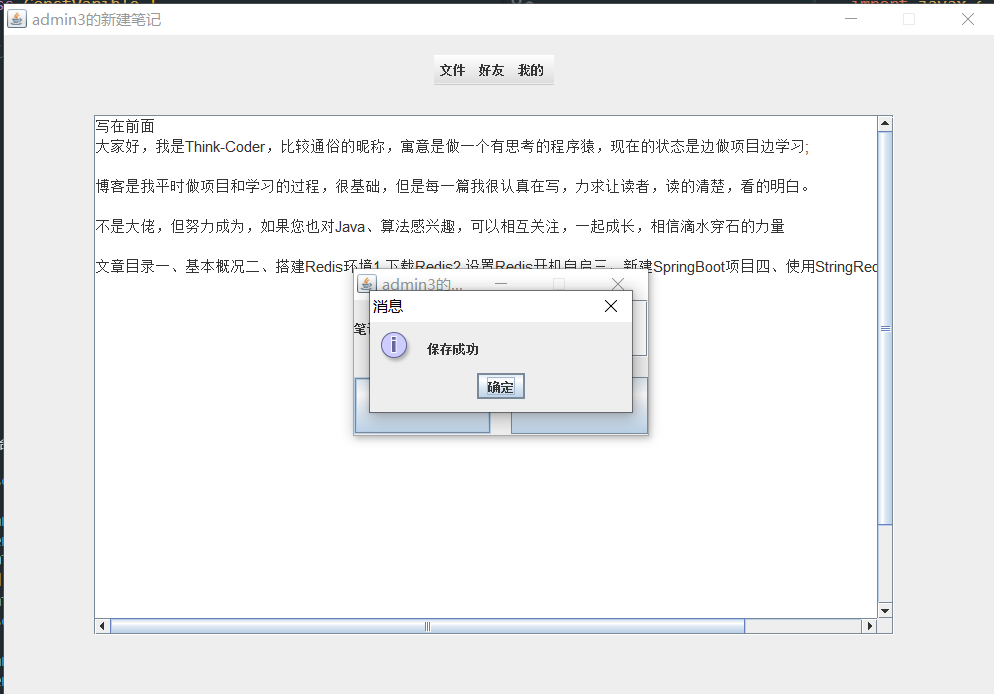
### Client.文件



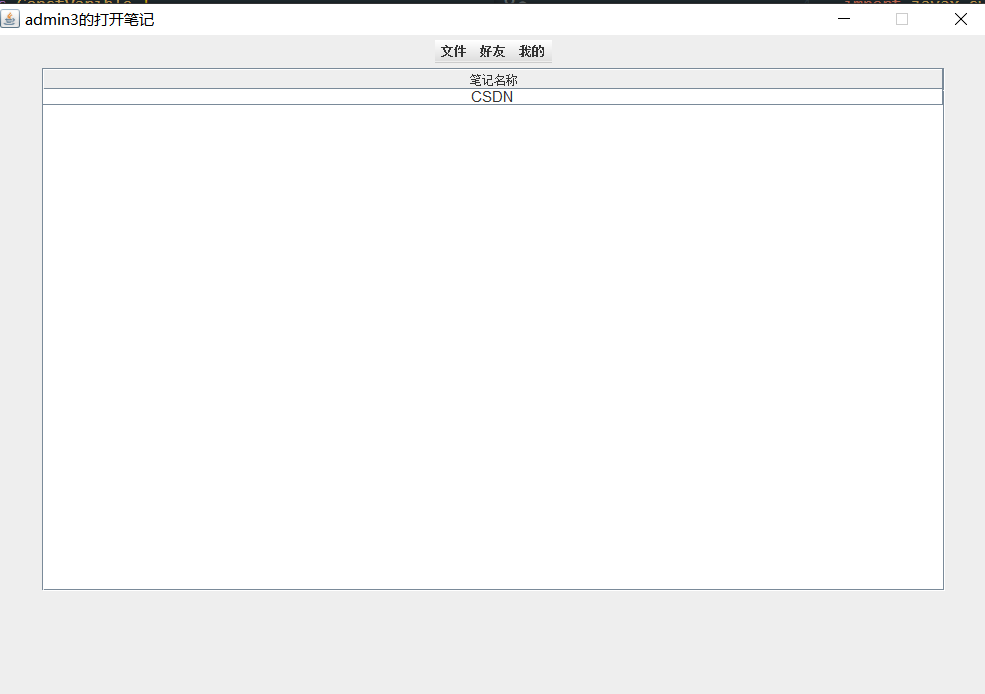
### Client.文件.新建笔记



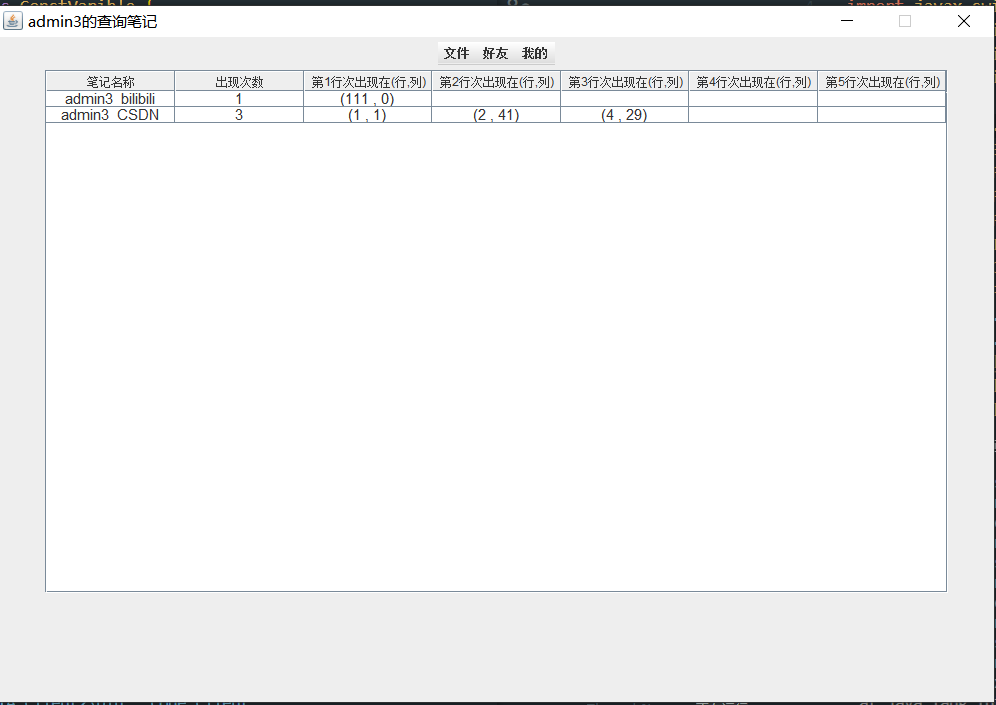
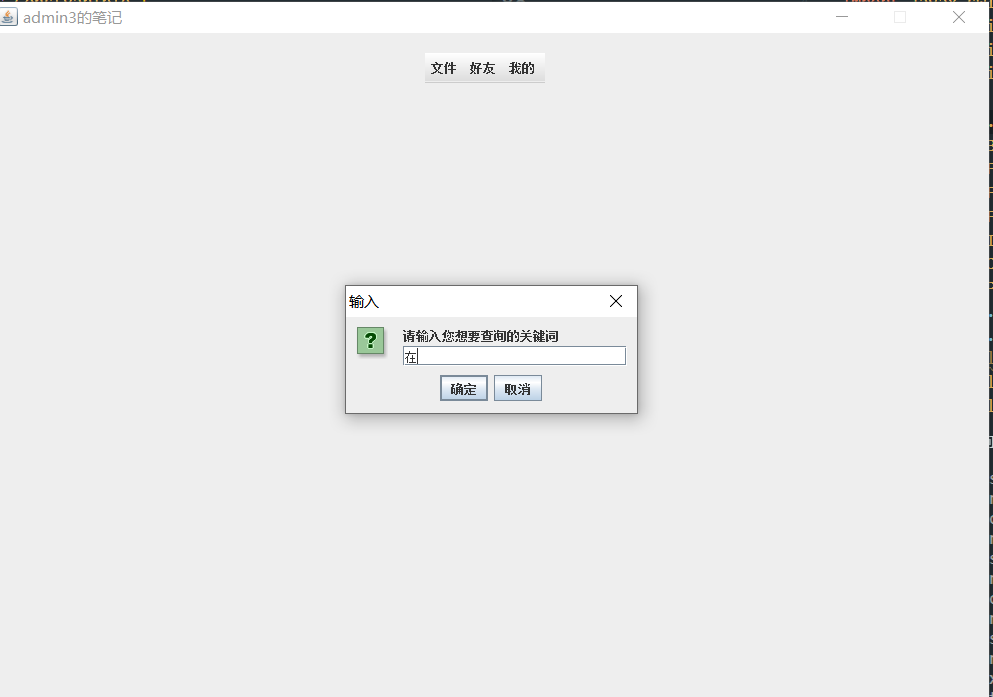
### Client.文件.保存笔记



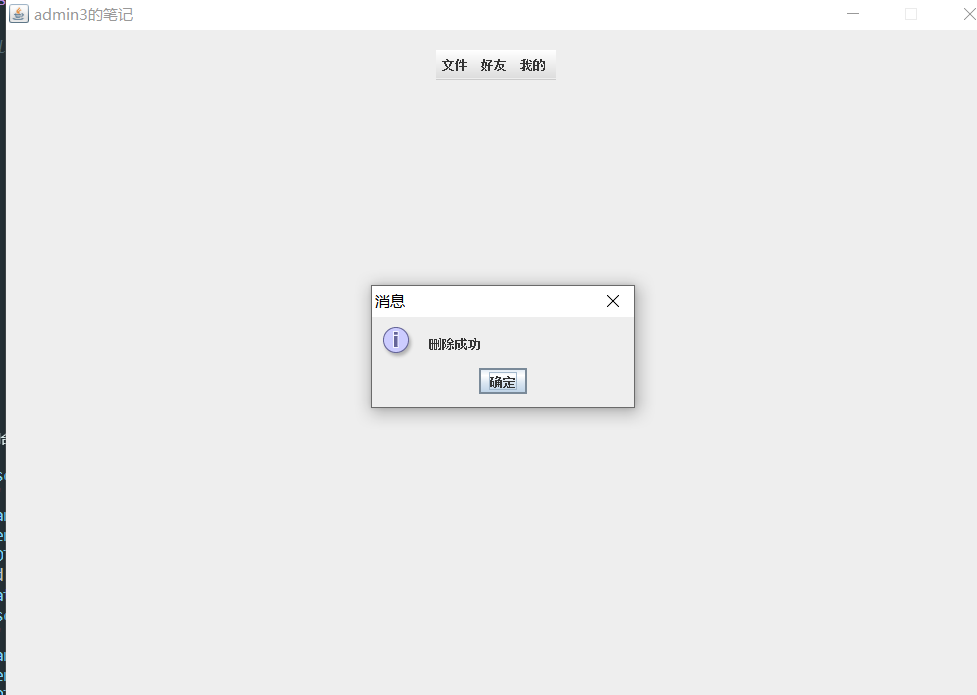
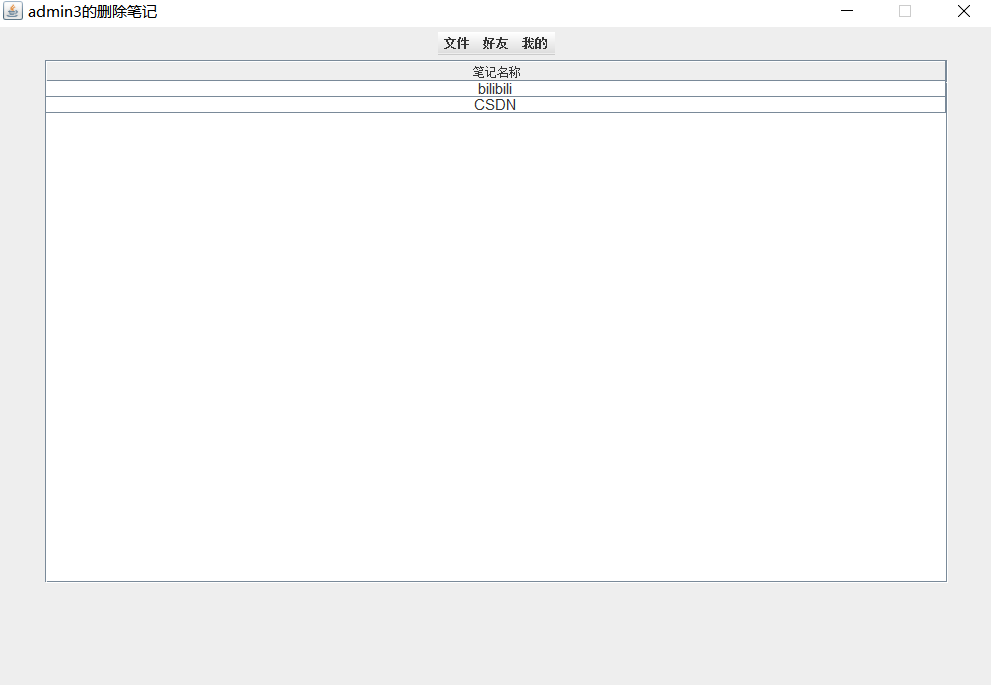
### Client.文件.打开笔记



### Client.文件.查询笔记

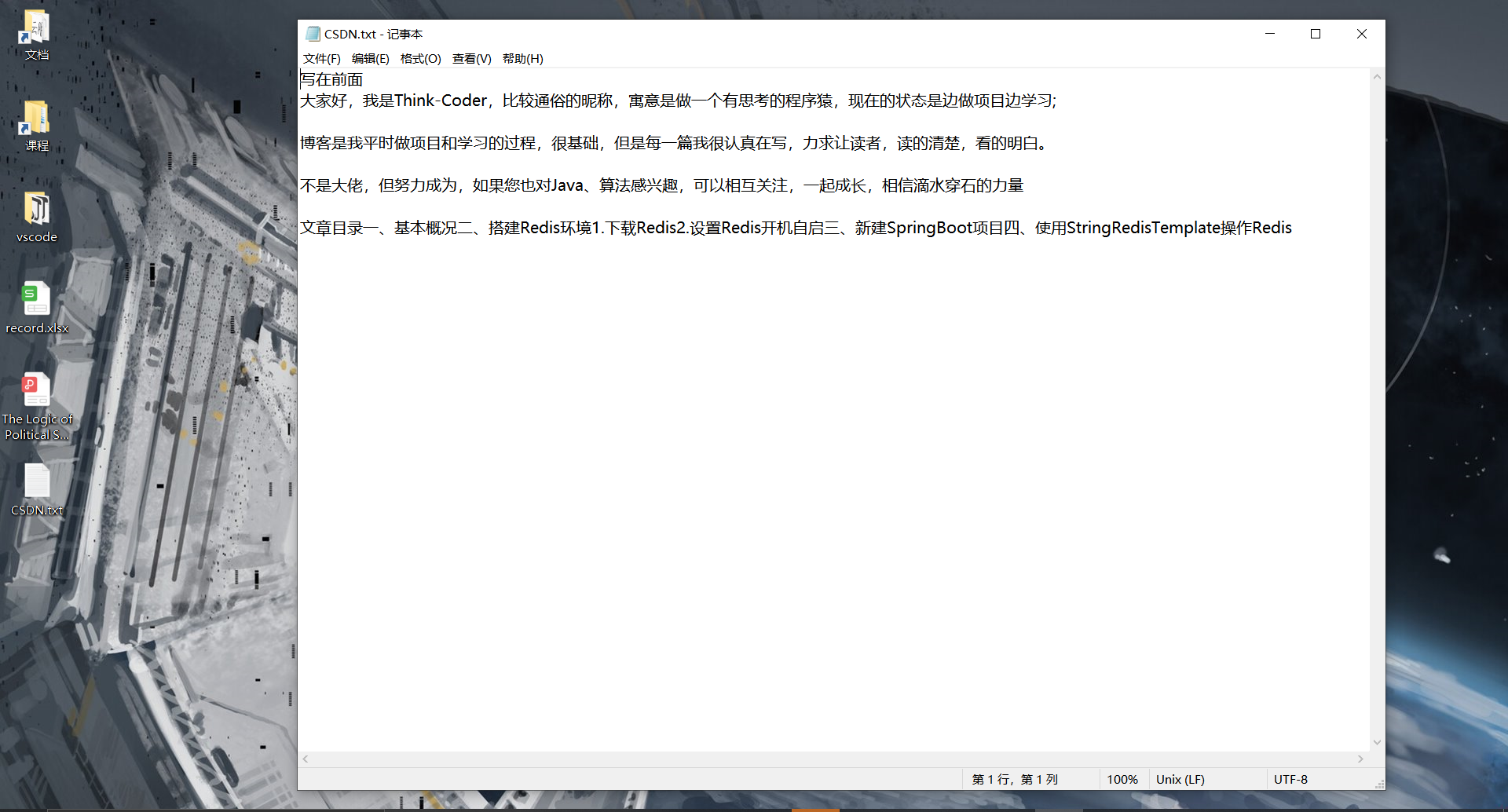
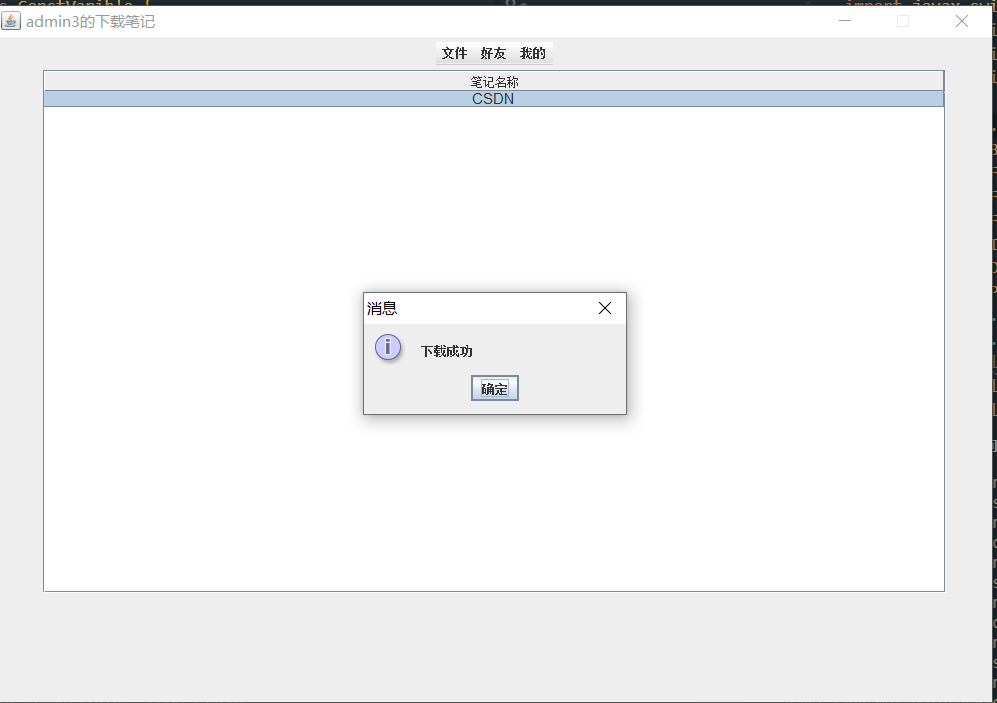
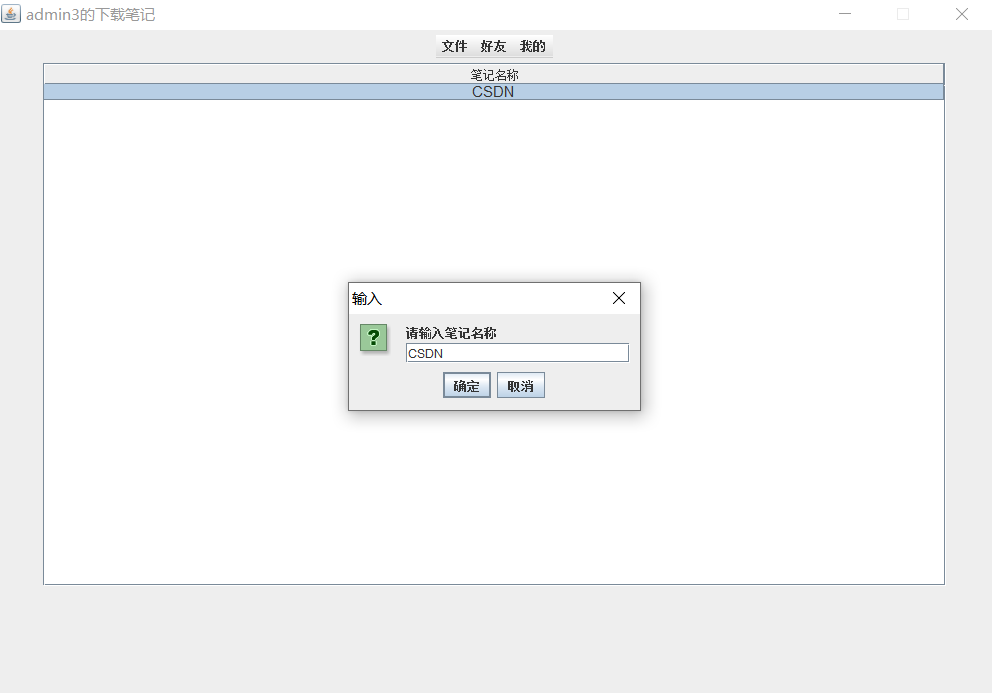
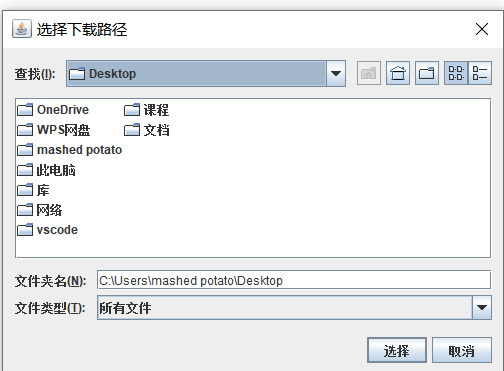
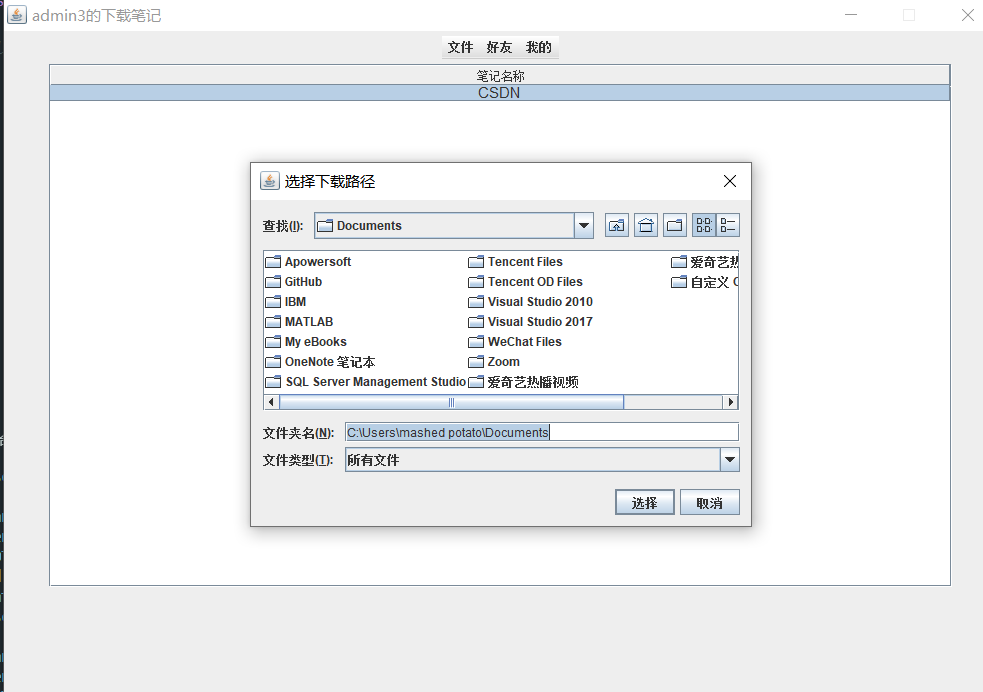
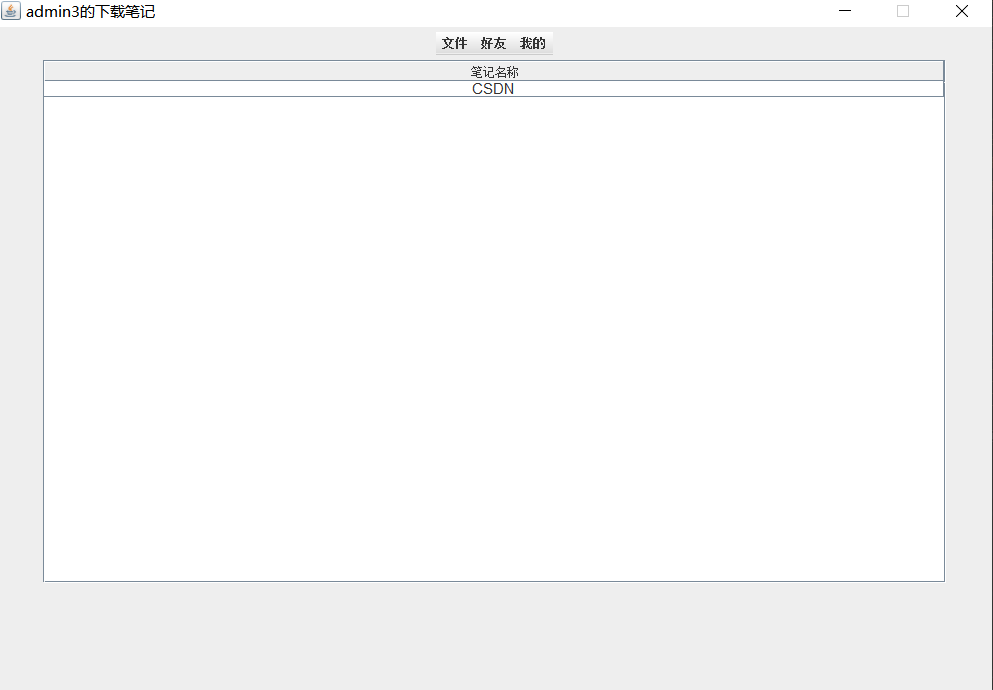


### Client.文件.删除笔记



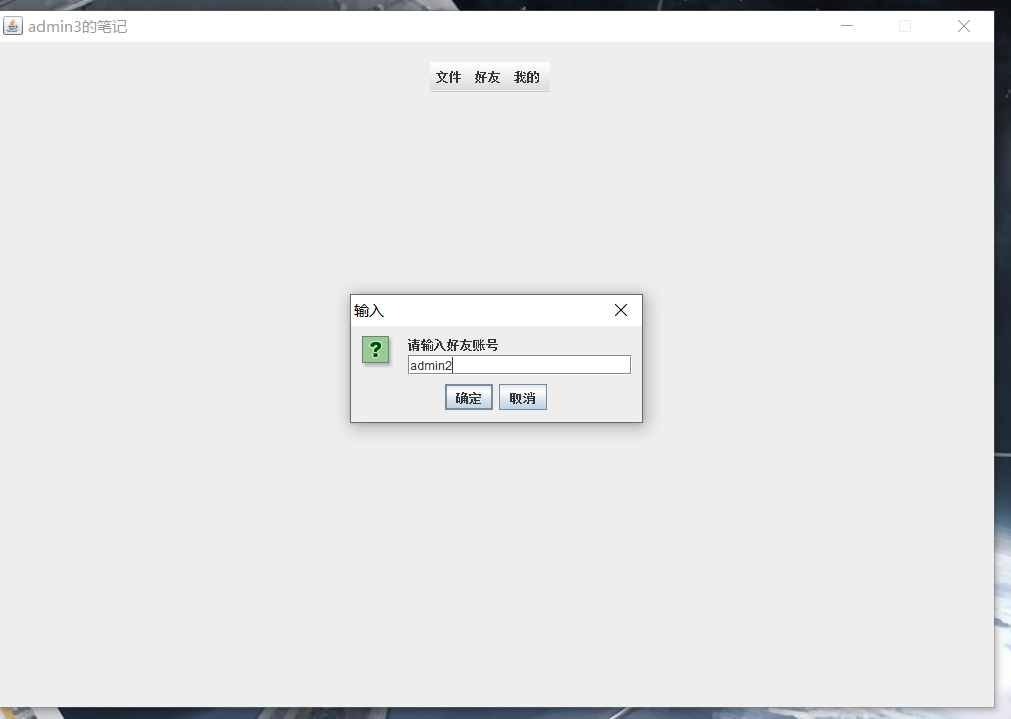


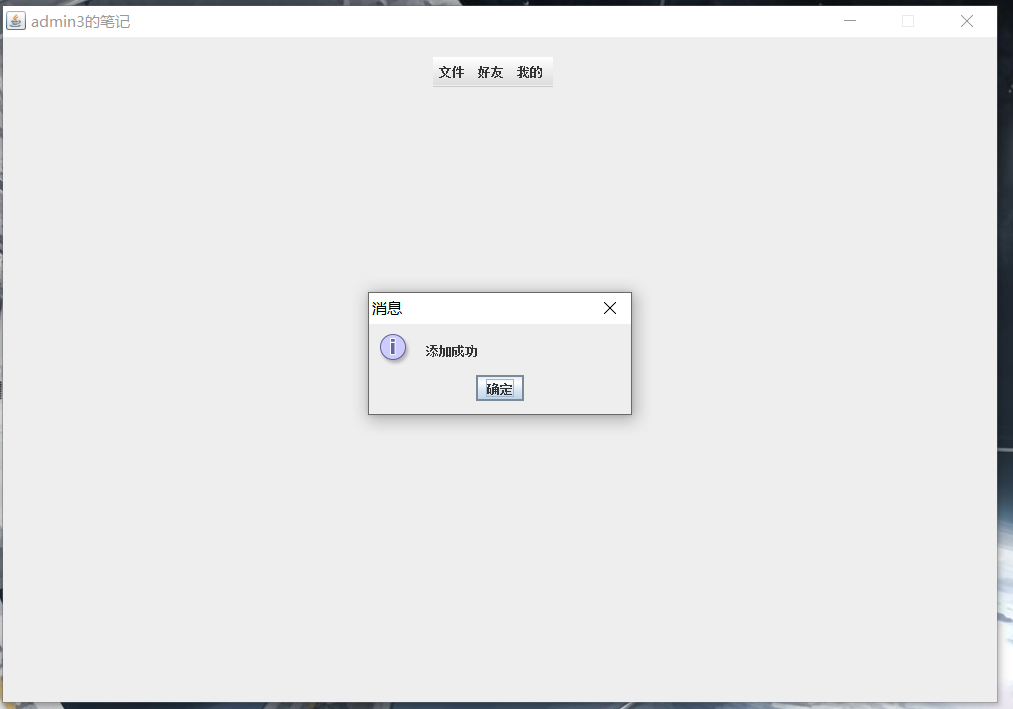
### Client.文件.下载笔记



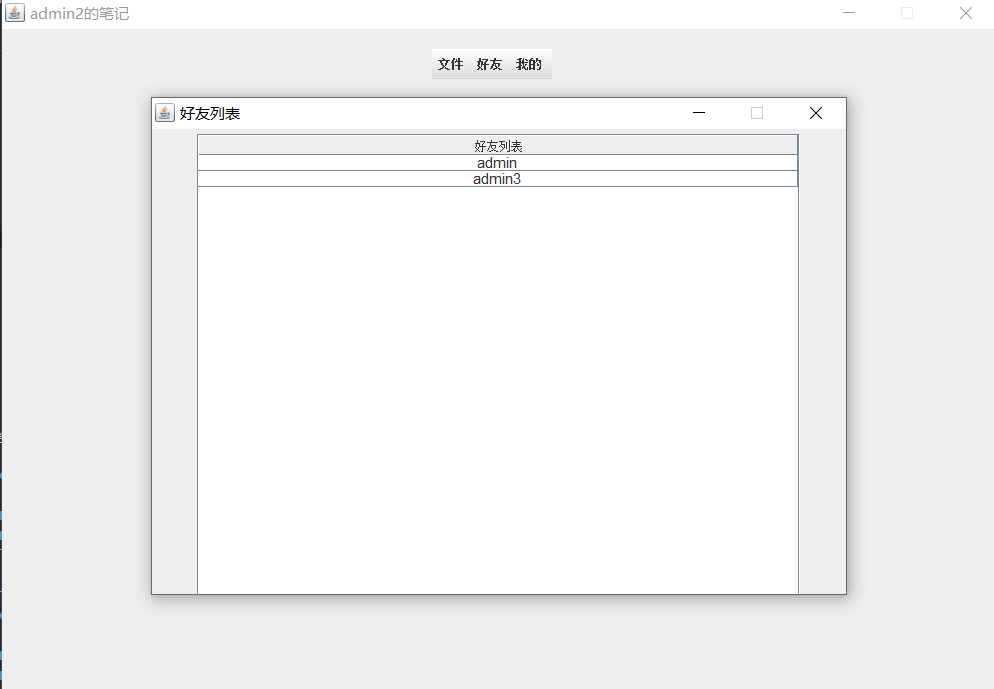
### Client.好友

### Client.好友.添加好友

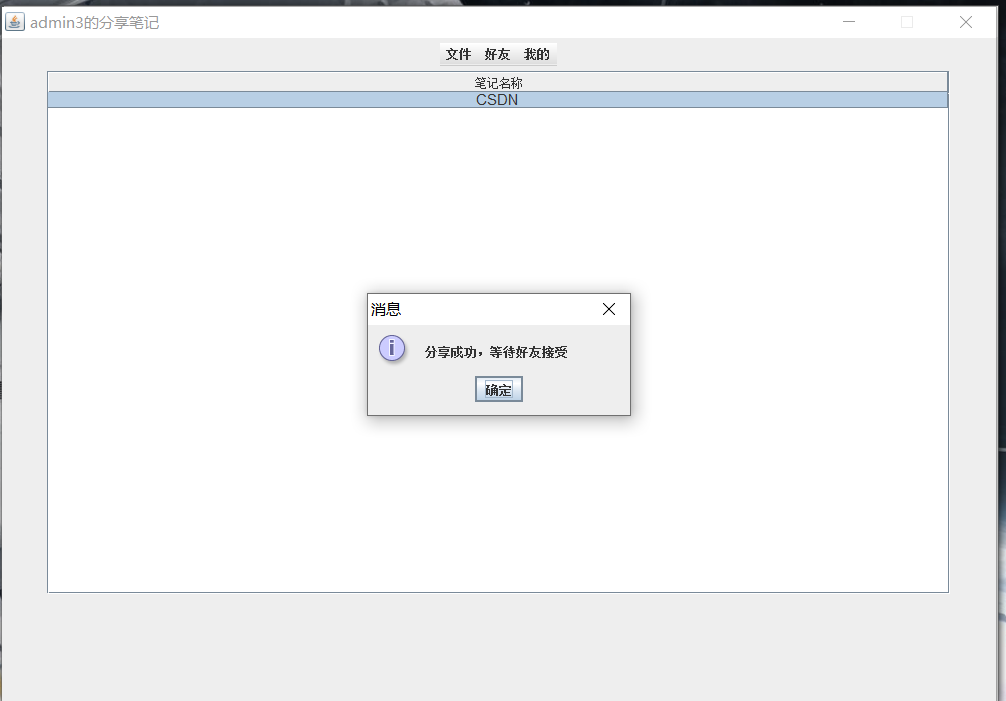
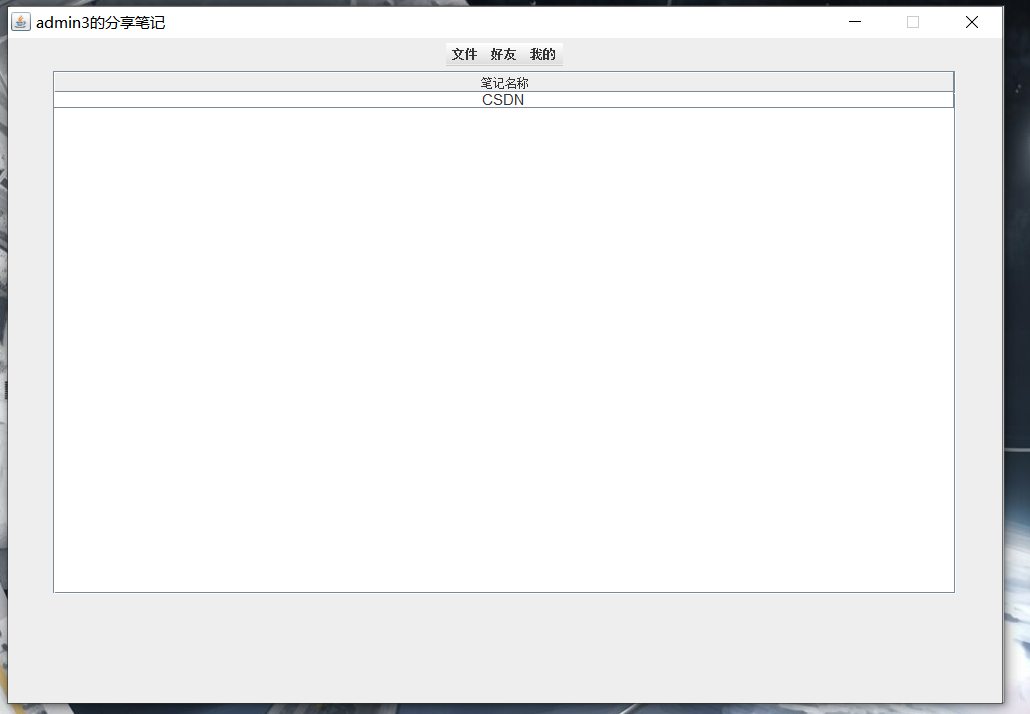




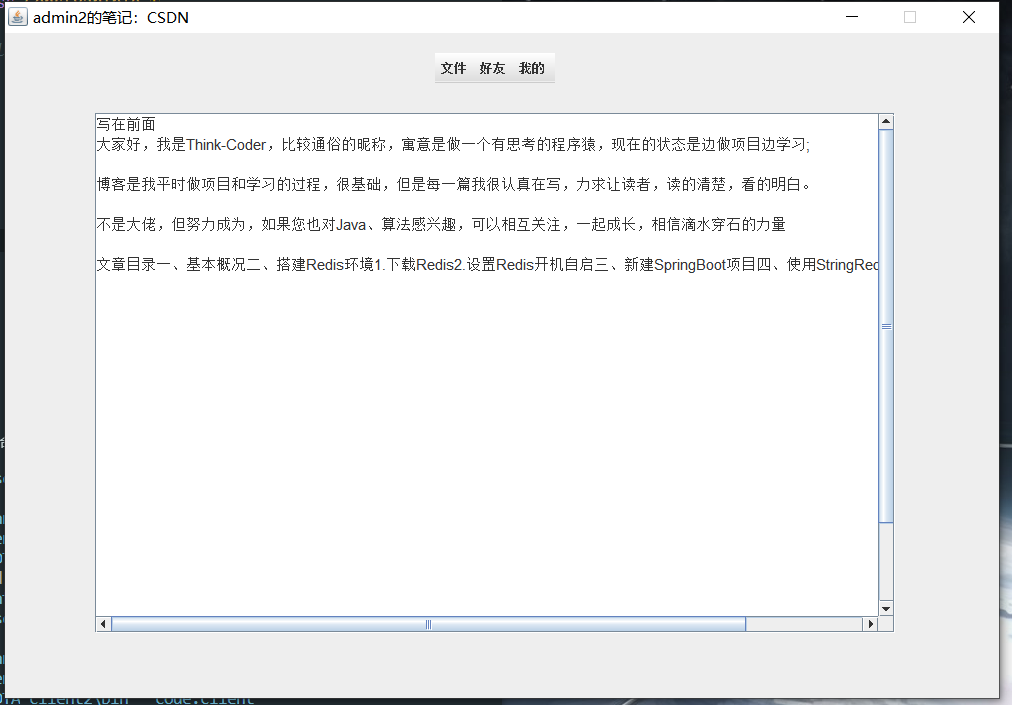
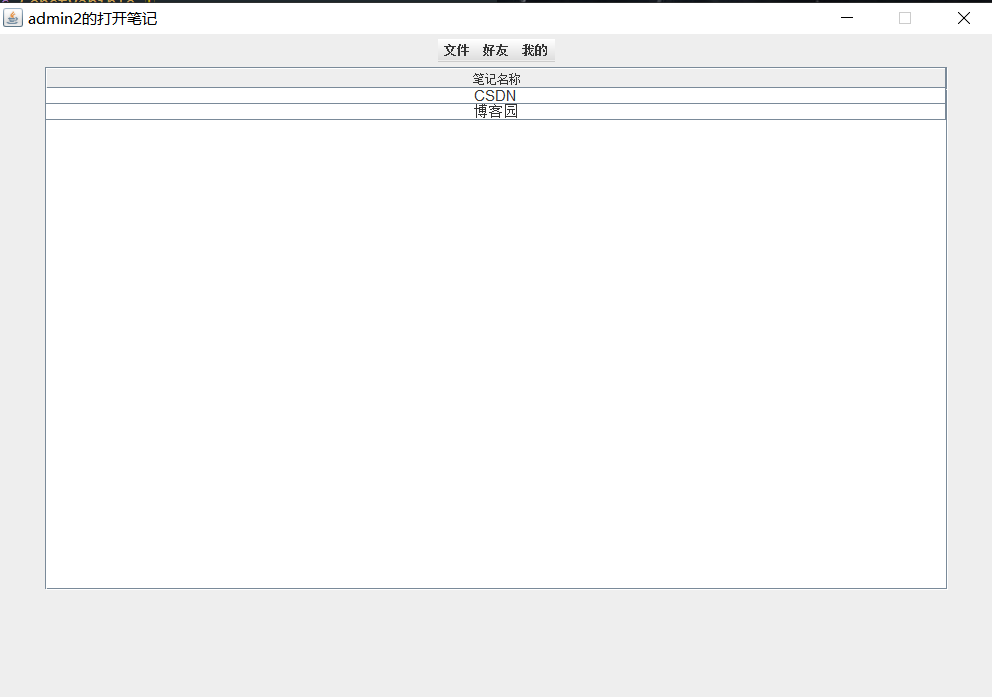
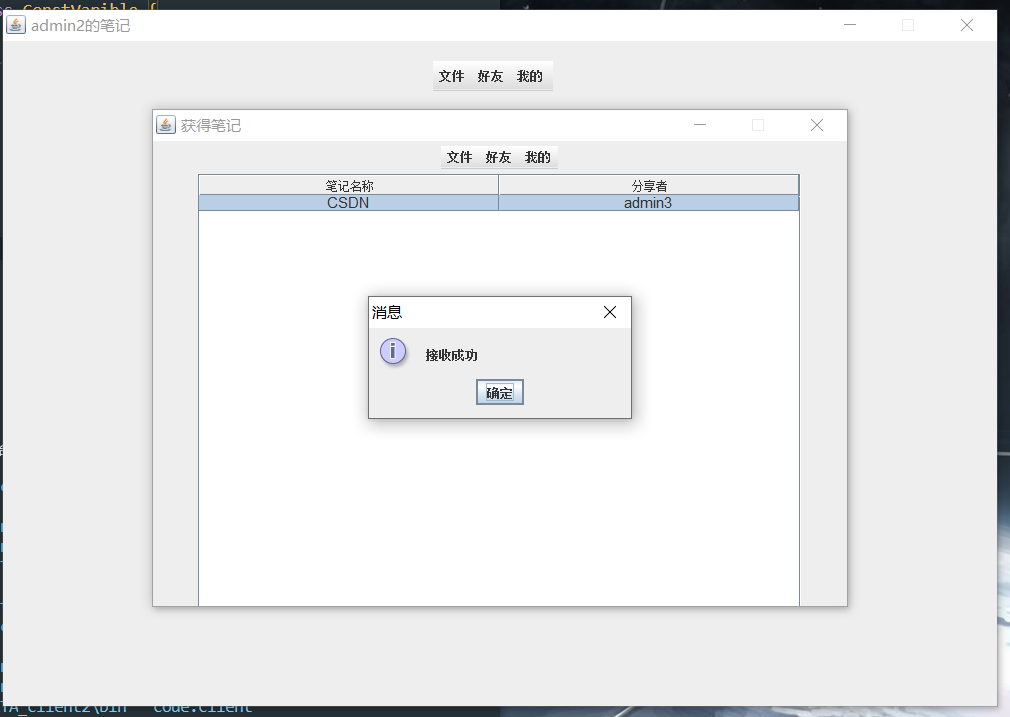
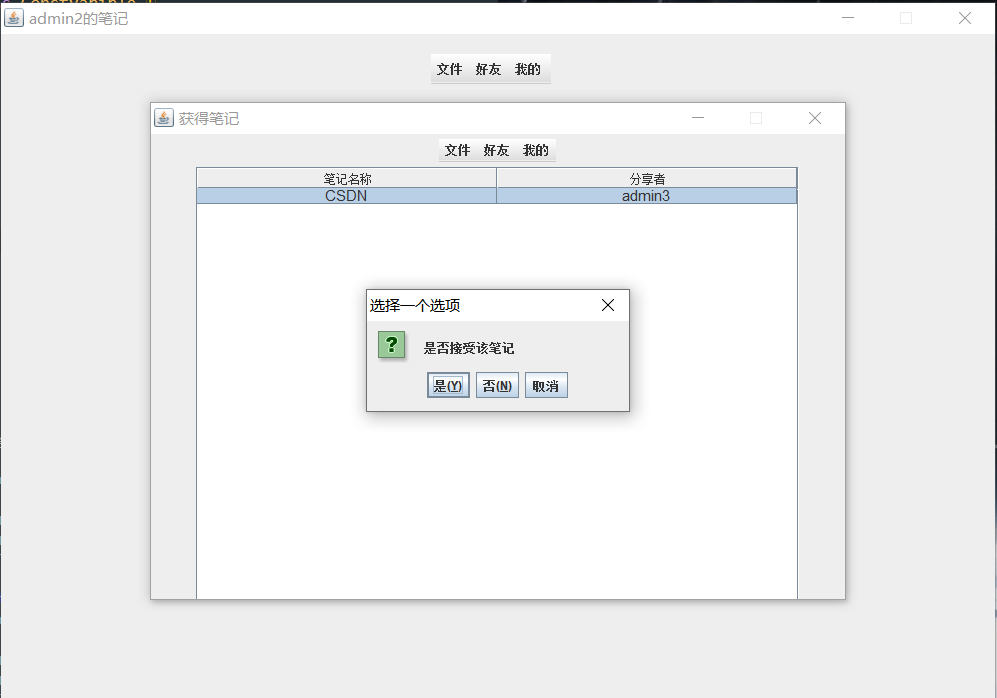
### Client.好友.好友列表



### Client.好友.分享笔记



### Client.好友.获得笔记



### Client.我的.退出



## 五、实验源代码

server.java（服务器）

package code;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import java.awt.event.\*;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.InputStreamReader;

import java.io.ObjectOutputStream;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.Date;

import java.util.Iterator;

import java.util.LinkedHashSet;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Vector;

import java.awt.Font;

import java.awt.Dimension;

public class server {

    public LinkedHashSet<String> linkedHashSet = *new* LinkedHashSet<>();*// 全局变量*

    public String accountString =*new* String();

    public class ancestor extends JFrame {

        protected JPanel jPanel = *new* JPanel();

        protected ServerSocket ss = null;

        protected Socket s = null;

        protected String title = null;

        protected JTable jTable = null;

        protected JScrollPane jScrollPane = null;

        protected Vector<Vector<String>> data = *new* Vector<>();

        protected Vector<String> columnName = *new* Vector<>();

        public ancestor(String title) {

            super(title);

            this.title = title;

            this.setVisible(true);

            this.setLocationRelativeTo(null);

*// this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);*

            this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

        }

    }

    public class mainInterface extends ancestor implements ActionListener, Runnable {

        private JButton all\_users = *new* JButton("管理所有用户");

        private JButton now\_users = *new* JButton("管理当前连接的用户");

        public mainInterface() {

*// GUI*

            super("服务器");

            jPanel.add(all\_users);

            jPanel.add(now\_users);

            this.add(jPanel);

            this.setSize(400, 250);

            jPanel.setLayout(null);

            all\_users.setBounds(115, 30, 150, 50);

            now\_users.setBounds(90, 120, 200, 50);

            this.setLocationRelativeTo(null);

            this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

            all\_users.addActionListener(this);

            now\_users.addActionListener(this);

*// TCP*

*try* {

                ss = *new* ServerSocket(ConstVarible.port);

*new* Thread(this).start();

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

        public void run() {

*try* {

*while* (true) {

                    s = ss.accept();

                    singleClient sc = new singleClient(s);

*new* Thread(sc).start();

                }

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

        public void actionPerformed(ActionEvent ae) {

            JButton jButton = (JButton) ae.getSource();

*if* (jButton.getText().compareTo("管理所有用户") == 0) {

*new* all\_users();

            } *else* *if* (jButton.getText().compareTo("管理当前连接的用户") == 0) {

*new* now\_users();

            }

        }

    }

    public class singleClient implements Runnable {

        private Socket s = null;

        private BufferedReader br = null;*// save,open,inquire,check account and password*

        public PrintStream ps = null;*// download,delete*

        public String account = null;

        private ArrayList<String> arrayList = null;

        public singleClient(Socket s) {

            this.s = s;

*try* {

                br = *new* BufferedReader(*new* InputStreamReader(s.getInputStream(),"UTF-8"));

                ps = *new* PrintStream(s.getOutputStream(),true,"UTF-8");

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

        public void run() {

*try* {

*while* (true) {

                    String str = *new* String();

*try* {

                        str = br.readLine();

                    } *catch* (Exception ex) {

*// ex.printStackTrace();*

                        linkedHashSet.remove(accountString);*// 断开连接抛出java.net.SocketException: Connectionreset，移除该用户连接*

*break*;

                    }

*if* ("login".compareTo(str) == 0) {

                        account = br.readLine();

                        accountString=account;

                        String password = fileOperation.getPasswordByAccount(account);

                        ps.println(password);

                        linkedHashSet.add(account);

                    }

*if* ("register".compareTo(str) == 0) {

                        account = br.readLine();

                        accountString=account;

                        String password = br.readLine();

*if* (fileOperation.check(account) == false) {

                            fileOperation.register(account, password);

                            ps.println("success");

                        } *else* {

                            ps.println("failed");

                        }

                        linkedHashSet.add(account);

                    }

*if* ("save".compareTo(str) == 0) {

                        String filename = br.readLine();

*if* (fileOperation.checkNoteName(filename) == true) {

                            ps.println("exist");

                            String str1 = null;

*try* {

                                str1 = br.readLine();

                            } *catch* (Exception ex) {

                                ex.printStackTrace();

                            }

*if* (str1.compareTo("yes") == 0) {

                                ;*// do nothing,接着执行后面的程序*

                            } *else* {

*new* Thread(*new* singleClient(s)).start();

*return*;

                            }

                        } *else* {

                            ps.println("not exist");

                        }

                        File notebook = *new* File(ConstVarible.dataPath + filename);

                        FileWriter fileWriter = *new* FileWriter(notebook);

*while* (true) {

                            String string = br.readLine();

*if* (string.compareTo(ConstVarible.endFlag) == 0) {

*break*;

                            }

                            fileWriter.write(string + "\n");

                        }

                        fileWriter.close();

                        ps.println("success");

                    }

*if* ("open".compareTo(str) == 0) {

                        account = "str1";

                        arrayList = null;

*try* {

                            account = br.readLine();

                            arrayList = fileOperation.getFileNameByAccount(account);

*for* (int i = 0; i < arrayList.size(); i++) {

                                ps.println(arrayList.get(i));

                            }

                            ps.println(ConstVarible.endFlag);

                        } *catch* (Exception ex) {

                            ex.printStackTrace();

                        }

                        int noteSelected = Integer.parseInt(br.readLine());

                        String notename = arrayList.get(noteSelected);

                        File note = *new* File(notename);

                        FileReader fr = *new* FileReader(note);

                        BufferedReader br = *new* BufferedReader(fr);

*while* (true) {

                            String string = br.readLine();

*if* (string == null) {

*break*;

                            }

                            ps.println(string);

                        }

                        ps.println(ConstVarible.endFlag);

                        br.close();

                    }

*if* ("delete".compareTo(str) == 0) {

                        int noteDelete = Integer.parseInt(br.readLine());

                        File file = *new* File(arrayList.get(noteDelete));

                        String notename = arrayList.get(noteDelete);

                        notename = notename.substring(notename.lastIndexOf("\\") + 1);

*// notename = notename.substring(notename.indexOf("\_") + 1, notename.indexOf("."));*

*// Date date = new Date();*

*// SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy\_mm\_dd\_hh\_mm\_ss");*

*// file.renameTo(new File(ConstVarible.dataPath + simpleDateFormat.format(date) + "\_" + account*

*//         + "\_" + notename + ConstVarible.format\_txt));*

                        file.renameTo(*new* File(ConstVarible.deletedPath+notename));

*if* (fileOperation.checkNoteName(account + "\_" + notename) == false) {

                            ps.println("success");

                        } *else* {

                            ps.println("failed");

                        }

                    }

*if* ("inquire".compareTo(str) == 0) {

                        String keyWord = null;

*try* {

                            account = br.readLine();

                            keyWord = br.readLine();

                        } *catch* (Exception ex) {

                            ex.printStackTrace();

                        }

                        arrayList = fileOperation.getFileNameByAccount(account);

                        File noteInquire = null;

                        Vector<Vector<String>> data = *new* Vector<>();

*for* (int i = 0; i < arrayList.size(); i++) {

                            noteInquire = *new* File(arrayList.get(i));

                            FileReader fileReader = *new* FileReader(noteInquire);

                            BufferedReader bufferedReader = *new* BufferedReader(fileReader);

                            String string = *new* String();

                            int row = 0;*// 行数*

                            Vector<String> noteVector = *new* Vector<>();

                            String notename = arrayList.get(i);

                            notename = notename.substring(notename.lastIndexOf("\\") + 1);

                            notename = notename.substring(0, notename.indexOf("."));

                            noteVector.add(notename);

*try* {

                                noteVector.add(String.valueOf(fileOperation.countWordInFile(keyWord, noteInquire)));

*while* (true) {

                                    row++;

                                    string = bufferedReader.readLine();

*if* (string == null) {

*break*;

                                    }

                                    int column = string.indexOf(keyWord);

*if* (column != -1) {

                                        noteVector.add("(" + row + " , " + column + ")");

*if* (noteVector.size() == 5) {

*break*;

                                        }

                                    }

                                }

                                data.add(noteVector);

                            } *catch* (Exception ex) {

                                ex.printStackTrace();

                            } *finally* {

*try* {

                                    bufferedReader.close();

                                } *catch* (Exception ex) {

                                    ex.printStackTrace();

                                }

                            }

                        }

                        ObjectOutputStream objectOutputStream = *new* ObjectOutputStream(s.getOutputStream());

                        objectOutputStream.writeObject(data);

                    }

*if* ("exit".compareTo(str) == 0) {

                        String string = br.readLine();

                        linkedHashSet.remove(string);

                    }

*if* ((ConstVarible.startFlag + "makefriends").compareTo(str) == 0) {

                        account = br.readLine();

                        String friendAccount = br.readLine();

                        int res = fileOperation.storeFriendsAccount(account, friendAccount);

*if* (res == 0) {

                            ps.println("该好友账户不存在");

                        } *else* *if* (res == 1) {

                            ps.println("您已经添加过该好友");

                        } *else* *if* (res == 2) {

                            ps.println("添加成功");

                        } *else* *if* (res == 3) {

                            ps.println("添加失败");

                        }

                    }

*if* ((ConstVarible.startFlag + "list").compareTo(str) == 0) {

                        account = br.readLine();

                        ObjectOutputStream objectOutputStream = *new* ObjectOutputStream(s.getOutputStream());

                        objectOutputStream.writeObject(fileOperation.getfriend(account));

                    }

*if*((ConstVarible.startFlag+"share").compareTo(str)==0){

                        String account=br.readLine();

                        String friendAccount=br.readLine();

                        String noteName=br.readLine();

                        int res=fileOperation.shareNote(account, friendAccount, noteName);

*if*(res==0){

                            ps.println("success");

                        }*else*{

                            ps.println("failure");

                        }

                    }

*if*((ConstVarible.startFlag+"getshared").compareTo(str)==0){

                        account=br.readLine();

                        ObjectOutputStream oos=*new* ObjectOutputStream(s.getOutputStream());

                        oos.writeObject(fileOperation.getshared(account));

                    }

*if*((ConstVarible.startFlag+"getsharednote").compareTo(str)==0){

                        account=br.readLine();

                        String notename=br.readLine();

                        String writer=br.readLine();

                        int res=fileOperation.moveSharedTo(writer, notename, account);

*if*(res==0){

                            ps.println("success");

                        }*else*{

                            ps.println("failure");

                        }

                    }

                }

            } *catch* (Exception ex) {

                ex.printStackTrace();

            } *finally* {

*try* {

                    br.close();

                } *catch* (Exception ex) {

*// ex.printStackTrace();*

                }

            }

        }

    }

    public class all\_users extends ancestor {

        public all\_users() {

            super("所有用户");

            columnName.add("用户名称");

            columnName.add("笔记数量");

            data = fileOperation.getClientInfo();

            jTable = *new* JTable(data, columnName);

            jTable.setPreferredScrollableViewportSize(*new* Dimension(900, 500));

            Font font = *new* Font(ConstVarible.font\_String, ConstVarible.font\_style, ConstVarible.font\_size);

            jTable.setFont(font);

            jTable.setFillsViewportHeight(true);

            jScrollPane = *new* JScrollPane(jTable);

            jPanel.add(jScrollPane);

            this.add(jPanel);

            this.setSize(1000, 700);

            this.setLocationRelativeTo(null);

        }

    }

    public class now\_users extends ancestor {

        public now\_users() {

            super("当前连接的用户");

            columnName.add("用户名称");

            Iterator<String> iterator = linkedHashSet.iterator();

*while* (iterator.hasNext()) {

                Vector<String> vector = *new* Vector<>();

                vector.add(iterator.next());

                data.add(vector);

            }

            jTable = *new* JTable(data, columnName);

            jTable.setPreferredScrollableViewportSize(*new* Dimension(900, 500));

            Font font = *new* Font(ConstVarible.font\_String, ConstVarible.font\_style, ConstVarible.font\_size);

            jTable.setFont(font);

            jTable.setFillsViewportHeight(true);

            jScrollPane = *new* JScrollPane(jTable);

            jPanel.add(jScrollPane);

            this.add(jPanel);

            this.setSize(1000, 700);

            this.setLocationRelativeTo(null);

        }

    }

    public static void main(String[] args) {

        server s = *new* server();

        s.new mainInterface();

    }

}

fileOperation.java（服务器）

package code;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.PrintStream;

import java.nio.file.Files;

import java.util.ArrayList;

import java.util.Properties;

import java.util.Set;

import java.util.TreeMap;

import java.util.Vector;

import javax.swing.JOptionPane;

public class fileOperation {

    private static Properties pps;

    static {

        pps = *new* Properties();

        FileReader reader = null;

*try* {

            File file=*new* File(ConstVarible.dataPath + "account.inc");

*if*(file.exists()==false){

                file.createNewFile();

            }

            reader = *new* FileReader(file);

            pps.load(reader);

        } *catch* (Exception ex) {

            JOptionPane.showMessageDialog(null, "文件操作异常1");

            System.exit(0);

        } *finally* {

*try* {

                reader.close();

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

    }

    public static void register(String nickname, String password) {

        pps.setProperty(nickname, password);

        PrintStream ps = null;

*try* {

            ps = *new* PrintStream(ConstVarible.dataPath + "account.inc");

            pps.list(ps);

        } *catch* (Exception ex) {

            JOptionPane.showMessageDialog(null, "文件操作异常2");

            System.exit(0);

        } *finally* {

*try* {

                ps.close();

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

    }

    public static String getPasswordByAccount(String account) {

        String password = pps.getProperty(account);

*return* password;

    }

    public static boolean check(String str) {*// 没找到返回false，找到返回true*

*if* (pps.getProperty(str) == null) {

*return* false;

        } *else* {

*return* true;

        }

    }

    public static boolean checkNoteName(String str) {*// filename已存在返回true，否则false*

        File file = *new* File(ConstVarible.dataPath + str);

*if* (file.exists() == true) {

*return* true;

        } *else* {

*return* false;

        }

    }

    public static ArrayList<String> getFileNameByAccount(String account) {

        File file = *new* File(ConstVarible.dataFile);

        File[] files = file.listFiles();

        ArrayList<String> arrayList = *new* ArrayList<String>();

*for* (File f *:* files) {

            String filename = f.getAbsolutePath();

*// System.out.println(filename);*

*if* (filename.startsWith(ConstVarible.dataPath+account + "\_")) {

                arrayList.add(filename);

            }

        }

*return* arrayList;

    }

    public static int countWordInFile(String word,File file)throws Exception{

        FileInputStream fis=*new* FileInputStream(file);

        byte[] data=*new* byte[(int)file.length()];

        fis.read(data);

        fis.close();

        String msg=*new* String(data);

        String aft=msg.replace(word, "");

*return* (msg.length()-aft.length())/word.length();

    }

    public static Vector<Vector<String>> getClientInfo(){

        File file=*new* File(ConstVarible.dataFile);

        File[] files=file.listFiles();

        Vector<Vector<String>> data=*new* Vector<>();

        TreeMap<String,Integer> treeMap=*new* TreeMap<>();

*for*(File f*:*files){

            String filename=f.getName();

            String str=filename.replace("\_", "");

            int numOf\_=(filename.length()-str.length())/"\_".length();

*if*(numOf\_==1){

*if*(treeMap.containsKey(filename)){

                    treeMap.put(filename.substring(0,filename.indexOf("\_")),treeMap.get(filename)+1);

                }

*else*{

                    treeMap.put(filename.substring(0,filename.indexOf("\_")),1);

                }

            }

        }

        Set keySet=treeMap.keySet();

*for*(Object key*:*keySet){

            Vector<String> row=*new* Vector<>();

*// System.out.println(key+" "+treeMap.get(key));*

            row.add(String.valueOf(key));

            row.add(String.valueOf(treeMap.get(key)));

            data.add(row);

        }

*return* data;

    }

    public static Vector<Vector<String>> getfriend(String account) {

        File file = *new* File(ConstVarible.friendsPath+account);

*if*(file.exists()==false)*return* null;

        Vector<Vector<String>> data = *new* Vector<>();

*try*{

            FileReader fileReader=*new* FileReader(file);

            BufferedReader bufferedReader=*new* BufferedReader(fileReader);

*while*(true){

                String string=bufferedReader.readLine();

*// System.out.println(string);*

*if*(string==null)*break*;

                Vector<String> row=*new* Vector<>();

                row.add(string);

                data.add(row);

            }

            bufferedReader.close();

        }*catch*(Exception ex){

            ex.printStackTrace();

        }

*return* data;

    }

    public static int storeFriendsAccount(String account, String friendAccount) {

*if* (check(friendAccount) == false) {*// 先检查好友账户是否存在*

*return* 0;

        } *else* {

            File file = *new* File(ConstVarible.friendsPath + account);

*try* {

*if* (file.exists() == false) {*//检查文件是否存在*

                    file.createNewFile();

                }

*if*(countWordInFile(friendAccount, file)==1){*//检查是否已经添加过该好友*

*return* 1;

                }

                FileWriter fw=*new* FileWriter(file,true);

                fw.append(friendAccount+"\n");

                fw.close();

                storeFriendsAccount(friendAccount, account);*//互为好友*

*return* 2;

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

*return* 3;

        }

    }

    public static int shareNote(String account,String friendAccount,String noteName){

*if*(checkNoteName(account+"\_"+noteName+ConstVarible.format\_txt)==false)*return* -1;

        File folder=*new* File(ConstVarible.dataPath+account+"ShareWith"+friendAccount);

*if*(folder.exists()==false){

            folder.mkdirs();

        }

        File sharedNote=*new* File(ConstVarible.dataPath+account+"\_"+noteName+ConstVarible.format\_txt);

*try*{

            Files.copy(sharedNote.toPath(),*new* File(ConstVarible.dataPath+account+"ShareWith"+friendAccount+"\\"+account+"\_"+noteName+ConstVarible.format\_txt).toPath());

        }*catch*(Exception ex){

            ex.printStackTrace();

        }

*return* 0;

    }

    public static Vector<Vector<String>> getshared(String account){

        File file = *new* File(ConstVarible.dataFile);

        File[] files = file.listFiles();

        Vector<Vector<String>> data=*new* Vector<>();

*//遍历所有文件，查看分享*

*for* (File f *:* files) {

*if*(f.isDirectory()==true && f.getName().endsWith("ShareWith"+account)==true){

                File[] sonfiles=f.listFiles();

*for*(File f2*:*sonfiles){

                    String filename=f2.getName();

                    Vector<String> row=*new* Vector<>();

                    row.add(filename.substring(filename.indexOf("\_")+1,filename.indexOf(".")));

                    row.add(filename.substring(0,filename.indexOf("\_")));

                    data.add(row);

                }

            }

        }

*return* data;

    }

    public static int moveSharedTo(String writer,String notename,String account){

        File file = *new* File(ConstVarible.dataFile);

        File[] files = file.listFiles();

*//遍历所有文件*

*for* (File f *:* files) {

*if*(f.isDirectory()==true && f.getName().endsWith("ShareWith"+account)==true){

                File[] sonfiles=f.listFiles();

*for*(File f2*:*sonfiles){

                    String filename=f2.getName();

*if*(filename.compareTo(writer+"\_"+notename+ConstVarible.format\_txt)==0){

                        f2.renameTo(*new* File(ConstVarible.dataPath+account+"\_"+notename+ConstVarible.format\_txt));

                    }

                }

            }

        }

*return* 0;

    }

}

ConstVarible.java（服务器）

package code;

import java.awt.Font;

public class ConstVarible {

    public static final int port=9999;

    public static final String startFlag="+\_)(\*&^%$#@!~";

    public static final String endFlag="!@#$%^&\*()";

    public static final String friendsPath="C:\\Users\\mashed potato\\Desktop\\vscode\\.vscode\\NOTA\_server\\src\\data\\friends\\";

    public static final String deletedPath="C:\\Users\\mashed potato\\Desktop\\vscode\\.vscode\\NOTA\_server\\src\\data\\deleted\_files\\";

    public static final String dataPath="C:\\Users\\mashed potato\\Desktop\\vscode\\.vscode\\NOTA\_server\\src\\data\\";

    public static final String dataFile="C:\\Users\\mashed potato\\Desktop\\vscode\\.vscode\\NOTA\_server\\src\\data";

    public static final String format\_txt=".txt";

    public static final String font\_String="楷体\_GB2312";

    public static final int font\_style=Font.PLAIN;

    public static final int font\_size=15;

}

client.java

package code;

import javax.swing.\*;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.table.DefaultTableCellRenderer;

import java.awt.\*;

import java.awt.event.\*;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileWriter;

import java.io.InputStream;

import java.io.InputStreamReader;

import java.io.ObjectInputStream;

import java.io.OutputStream;

import java.io.PrintStream;

import java.net.Socket;

import java.util.Vector;

import java.awt.Font;

public class client {

    public class ancestor extends JFrame {

        protected JPanel jPanel = *new* JPanel();

        protected Socket s = null;

        protected OutputStream os = null;

        protected PrintStream ps = null;

        protected InputStream is = null;

        protected BufferedReader br = null;

        protected String title = null;

        public ancestor(String str) {

            super(str);

            title = str;

            this.setVisible(true);

            this.setLocationRelativeTo(null);

            this.setDefaultCloseOperation(JFrame.DISPOSE\_ON\_CLOSE);

            this.setResizable(false);

*try* {

                s = *new* Socket(ConstVarible.address\_home\_wifi, ConstVarible.port);

                os = s.getOutputStream();

                ps = *new* PrintStream(os, true, "UTF-8");

                is = s.getInputStream();

                br = *new* BufferedReader(*new* InputStreamReader(is, "UTF-8"));

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

        }

    }

    public class login extends ancestor implements ActionListener, FocusListener {*// 登录界面*

        private GridLayout gridLayout = *new* GridLayout(3, 2, 20, 20);

        private JLabel label\_name = *new* JLabel("账号");

        private JLabel label\_password = *new* JLabel("密码");

        private JTextField input\_name = *new* JTextField("输入账号");

        private JPasswordField input\_password = *new* JPasswordField("输入密码");

        private JButton registerButton = *new* JButton("去注册");

        private JButton loginButton = *new* JButton("登录");

        public login() {

            super("登录界面");

            jPanel.setLayout(gridLayout);

            jPanel.add(label\_name);

            jPanel.add(input\_name);

            jPanel.add(label\_password);

            jPanel.add(input\_password);

            jPanel.add(loginButton);

            jPanel.add(registerButton);

            this.add(jPanel);

            this.setSize(400, 300);

            this.setLocationRelativeTo(null);

            registerButton.addActionListener(this);

            loginButton.addActionListener(this);

            input\_name.addFocusListener(this);

            input\_password.addFocusListener(this);

            input\_password.addActionListener(this);

        }

        public void actionPerformed(ActionEvent e) {

            JButton jButton = *new* JButton("jButton");

*try* {

                jButton = (JButton) e.getSource();

            } *catch* (Exception ex) {

                log();

*return*;

            }

*if* (jButton.getText().compareTo("去注册") == 0) {

*new* client().new register();

                this.dispose();

            } *else* {

                log();

            }

        }

        public void log() {

            String realPassword = null;

*try* {

                ps.println("login");

                ps.println(input\_name.getText());

*// System.out.println("waiting");*

                realPassword = br.readLine();

*// System.out.println("waiting2");*

            } *catch* (Exception ex) {

                ex.printStackTrace();

            } *finally* {

*try* {

                    br.close();

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

            }

*if* (realPassword.compareTo(String.valueOf(input\_password.getPassword())) == 0) {

                JOptionPane.showMessageDialog(null, "登录成功");

*new* client().new mainInterface(input\_name.getText());

                this.dispose();

            } *else* {

                JOptionPane.showMessageDialog(null, "账号不存在或密码错误");

*new* client().new login();

                this.dispose();

            }

        }

        public void focusGained(FocusEvent fe) {

*if* (input\_name.isFocusOwner() == true) {

                input\_name.setText("");

            }

*if* (input\_password.isFocusOwner() == true) {

                input\_password.setText("");

            }

        }

        public void focusLost(FocusEvent fe) {

*if* (input\_name.isFocusOwner() == false) {

*// input\_name.setText("输入账号");*

            }

*if* (input\_password.isFocusOwner() == false) {

*// input\_password.setText("输入密码");*

            }

        }

    }

    public class register extends ancestor implements ActionListener, FocusListener {

        private GridLayout gridLayout = *new* GridLayout(4, 2, 20, 20);

        private JLabel label\_name = *new* JLabel("账号");

        private JLabel label\_password = *new* JLabel("密码");

        private JLabel label\_password\_again = *new* JLabel("确认密码");

        private JTextField input\_name = *new* JTextField("输入账号");

        private JPasswordField input\_password = *new* JPasswordField("输入密码");

        private JPasswordField input\_password\_again = *new* JPasswordField("重复密码");

        private JButton registerButton = *new* JButton("注册");

        private JButton cancelButton = *new* JButton("取消");

        public register() {

            super("注册界面");

            input\_name.addFocusListener(this);

            input\_password.addFocusListener(this);

            input\_password\_again.addFocusListener(this);

            jPanel.setLayout(gridLayout);

            jPanel.add(label\_name);

            jPanel.add(input\_name);

            jPanel.add(label\_password);

            jPanel.add(input\_password);

            jPanel.add(label\_password\_again);

            jPanel.add(input\_password\_again);

            jPanel.add(registerButton);

            jPanel.add(cancelButton);

            this.add(jPanel);

            this.setSize(500, 400);

            this.setLocationRelativeTo(null);

            registerButton.addActionListener(this);

            cancelButton.addActionListener(this);

        }

        public void actionPerformed(ActionEvent e) {

            JButton jButton = (JButton) e.getSource();

*if* (jButton.getText().compareTo("注册") == 0) {

*if* (String.valueOf(input\_password.getPassword())

                        .compareTo(String.valueOf(input\_password\_again.getPassword())) == 0) {

*try* {

                        os = s.getOutputStream();

                        ps = *new* PrintStream(os, true, "UTF-8");

                        ps.println("register");

                        ps.println(input\_name.getText());

                        ps.println(String.valueOf(input\_password.getPassword()));

                        is = s.getInputStream();

                        br = *new* BufferedReader(*new* InputStreamReader(is, "UTF-8"));

*if* (br.readLine().compareTo("success") == 0) {

                            JOptionPane.showMessageDialog(this, "注册成功");

*new* client().new mainInterface(input\_name.getText());

                            this.dispose();

                        } *else* {

                            JOptionPane.showMessageDialog(this, "此账号已存在");

*new* client().new register();

                            this.dispose();

                        }

                    } *catch* (Exception ex) {

                        ex.printStackTrace();

                    } *finally* {

*try* {

                            br.close();

                        } *catch* (Exception ex) {

                            ex.printStackTrace();

                        }

                    }

                } *else* {

                    JOptionPane.showMessageDialog(null, "密码与确认密码不一致，请重新输入");

                }

            } *else* *if* (jButton.getText().compareTo("取消") == 0) {

*new* client().new login();

                this.dispose();

            }

        }

        public void focusGained(FocusEvent fe) {

*if* (input\_name.isFocusOwner() == true) {

                input\_name.setText("");

            }

*if* (input\_password.isFocusOwner() == true) {

                input\_password.setText("");

            }

*if* (input\_password\_again.isFocusOwner() == true) {

                input\_password\_again.setText("");

            }

        }

        public void focusLost(FocusEvent fe) {

        }

    }

    public class mainInterface extends ancestor implements ActionListener {

        protected JMenuBar jMenuBar = *new* JMenuBar();

        protected JMenu file = *new* JMenu("文件");

        protected JMenuItem newNote = *new* JMenuItem("新建笔记");

        protected JMenuItem saveNote = *new* JMenuItem("保存笔记");

        protected JMenuItem openNote = *new* JMenuItem("打开笔记");

        protected JMenuItem deleteNote = *new* JMenuItem("删除笔记");

        protected JMenuItem inquireNote = *new* JMenuItem("查询笔记");

        protected JMenuItem downloadNote = *new* JMenuItem("下载笔记");

        protected JMenu friends = *new* JMenu("好友");

        protected JMenuItem makefriends = *new* JMenuItem("添加好友");

        protected JMenuItem myfriends = *new* JMenuItem("我的好友");

        protected JMenuItem shareNote = *new* JMenuItem("分享笔记");

        protected JMenuItem getSharedNote = *new* JMenuItem("获得笔记");

        protected JMenu accountJMenu = *new* JMenu("我的");

*// protected JMenuItem modifiyInfo = new JMenuItem("修改账户信息");*

        protected JMenuItem exitItem = *new* JMenuItem("退出");

        protected JTextArea newJTextArea = *new* JTextArea(30, 60);

        protected JScrollPane newJScrollPane = *new* JScrollPane(newJTextArea);

        protected String title = null;

        public mainInterface(String str) {

            super(str + "的笔记");

            title = str;

            newNote.addActionListener(this);

            saveNote.addActionListener(this);

            openNote.addActionListener(this);

            deleteNote.addActionListener(this);

            inquireNote.addActionListener(this);

            downloadNote.addActionListener(this);

            makefriends.addActionListener(this);

            myfriends.addActionListener(this);

            shareNote.addActionListener(this);

            getSharedNote.addActionListener(this);

*// modifiyInfo.addActionListener(this);*

            exitItem.addActionListener(this);

            jPanel.add(jMenuBar);

            jMenuBar.add(file);

            jMenuBar.add(friends);

            jMenuBar.add(accountJMenu);

            file.add(newNote);

            file.add(saveNote);

            file.add(openNote);

            file.add(deleteNote);

            file.add(inquireNote);

            file.add(downloadNote);

            friends.add(makefriends);

            friends.add(myfriends);

            friends.add(shareNote);

            friends.add(getSharedNote);

*// accountJMenu.add(modifiyInfo);*

            accountJMenu.add(exitItem);

            jPanel.setLayout(null);

            this.add(jPanel);

            this.setSize(1000, 700);

            jMenuBar.setBounds(430, 20, 120, 30);

            this.setLocationRelativeTo(null);

        }

        public void actionPerformed(ActionEvent ae) {

            JMenuItem jMenuItem = (JMenuItem) ae.getSource();

            String string = jMenuItem.getText();

*switch* (string) {

*case* "新建笔记"*:*

*new* newNote(title);

*break*;

*case* "保存笔记"*:*

*new* saveNote(title, this);

*break*;

*case* "打开笔记"*:*

*new* openNote(title);

*break*;

*case* "删除笔记"*:*

*new* deleteNote(title);

*break*;

*case* "查询笔记"*:*

*new* inquireNote(title);

*break*;

*case* "下载笔记"*:*

*new* downloadNote(title);

*break*;

*case* "退出"*:*

                    ps.println("exit");

                    ps.println(title);

                    System.exit(0);

*break*;

*case* "添加好友"*:*

*new* makefriend(title);

*break*;

*case* "我的好友"*:*

*new* listfriend(title);

*break*;

*case* "分享笔记"*:*

*new* shareNote(title);

*break*;

*case* "获得笔记"*:*

*new* getSharedNote(title);

*break*;

            }

        }

    }

    public class newNote extends mainInterface {

        public newNote(String str) {

            super(str);

            this.setTitle(str + "的新建笔记");

            jPanel.add(newJScrollPane);

            this.add(jPanel);

            newJScrollPane.setBounds(90, 80, 800, 520);

            Font jTextAreaFont = *new* Font("楷体\_GB2312", Font.PLAIN, 15);

            newJTextArea.setFont(jTextAreaFont);

*// newJTextArea.setLineWrap(true);*

        }

    }

    public class saveNote extends ancestor implements ActionListener {

        private JLabel nameJLabel = *new* JLabel("笔记名称");

        private JTextField nameJTextField = *new* JTextField();

        private JButton saveButton = *new* JButton("保存");

        private JButton cancelButton = *new* JButton("取消");

        private GridLayout gridLayout = *new* GridLayout(2, 2, 20, 20);

        mainInterface anotherInterface = null;

        public saveNote(String str, mainInterface jframe) {

            super(str);

            anotherInterface = jframe;

            this.setTitle(str + "的保存笔记");

            this.setVisible(true);

            this.setSize(300, 170);

            jPanel.setLayout(gridLayout);

            this.setLocationRelativeTo(null);

            jPanel.add(nameJLabel);

            jPanel.add(nameJTextField);

            jPanel.add(saveButton);

            jPanel.add(cancelButton);

            this.add(jPanel);

            saveButton.addActionListener(this);

            cancelButton.addActionListener(this);

        }

        public void actionPerformed(ActionEvent ae) {

            JButton jButton = (JButton) ae.getSource();

*if* (jButton.getText().compareTo("保存") == 0) {

                String account = anotherInterface.title;

                ps.println("save");

                ps.println(account + "\_" + nameJTextField.getText() + ".txt");*// 用户名+name*

                String str = "str";

*try* {

                    str = br.readLine();

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

*if* (str.compareTo("exist") == 0) {

                    int isCover = JOptionPane.showConfirmDialog(this, "该笔记本名称已存在，继续保存将覆盖原有笔记本，是否继续？");

*if* (isCover == 0) {*// 确认覆盖,接着执行后面的程序*

                        ps.println("yes");

                    } *else* {*// 否则，重新输入*

                        ps.println("no");

*return*;

                    }

                }

                ps.println(anotherInterface.newJTextArea.getText());

                ps.println(ConstVarible.endFlag);

*try* {

                    String str1 = br.readLine();

*if* (str1.compareTo("success") == 0) {

                        JOptionPane.showMessageDialog(this, "保存成功");

                    }

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

                this.dispose();

            } *else* *if* (jButton.getText().compareTo("取消") == 0) {

                this.dispose();

            }

        }

    }

    public class openNote extends mainInterface implements ListSelectionListener {

        protected JTable jTable = null;

        protected JScrollPane jScrollPane = null;

        protected Vector<Vector<String>> data = *new* Vector<>();

        protected Vector<String> columnName = *new* Vector<>();

        public openNote(String str) {

            super(str);

            this.setTitle(str + "的打开笔记");

            ps.println("open");

            ps.println(str);

            String notename = null;

*try* {

*while* (true) {

                    notename = br.readLine();

*if* (notename.compareTo(ConstVarible.endFlag) == 0) {

*break*;

                    }

                    notename = notename.substring(notename.lastIndexOf("\\") + 1);

                    notename = notename.substring(notename.indexOf("\_") + 1, notename.indexOf("."));

                    Vector<String> row = *new* Vector<>();

                    row.add(notename);

                    data.add(row);

                }

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

            FlowLayout flowLayout = *new* FlowLayout();

            jPanel.setLayout(flowLayout);

            columnName.add("笔记名称");

            jTable = *new* JTable(data, columnName) {

                public boolean isCellEditable(int row, int column) {

*return* false;

                }

            };

            jTable.setPreferredScrollableViewportSize(*new* Dimension(900, 500));

            Font font = *new* Font(ConstVarible.font\_String, ConstVarible.font\_style, ConstVarible.font\_size);

            jTable.setFont(font);

            jTable.getSelectionModel().addListSelectionListener(this);

            jTable.setFillsViewportHeight(true);

            DefaultTableCellRenderer defaultTableCellRenderer = *new* DefaultTableCellRenderer();

            defaultTableCellRenderer.setHorizontalAlignment(JLabel.CENTER);

            jTable.setDefaultRenderer(Object.class, defaultTableCellRenderer);

            jScrollPane = *new* JScrollPane(jTable);

            jPanel.add(jScrollPane);

            this.add(jPanel);

        }

        public void valueChanged(ListSelectionEvent le) {

            int row = jTable.getSelectedRow();

*if* (le.getValueIsAdjusting() == false) {

                ps.println(row);

                newNote opennote = new newNote(title);

                opennote.setTitle(title + "的笔记：" + jTable.getModel().getValueAt(row, 0));

                this.dispose();

*try* {

*while* (true) {

                        String string = br.readLine();

*if* (string.compareTo(ConstVarible.endFlag) == 0) {

*break*;

                        }

                        opennote.newJTextArea.append(string + "\n");

                    }

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

            }

        }

    }

    public class deleteNote extends openNote {

        public deleteNote(String str) {

            super(str);

            this.setTitle(str + "的删除笔记");

        }

        public void valueChanged(ListSelectionEvent le) {

            int row = jTable.getSelectedRow();

*if* (le.getValueIsAdjusting() == false) {

                ps.println(row);

                newNote opennote = new newNote(title);

                opennote.setTitle(title + "的笔记：" + jTable.getModel().getValueAt(row, 0));

*try* {

*while* (true) {

                        String string = br.readLine();

*if* (string.compareTo(ConstVarible.endFlag) == 0) {

*break*;

                        }

                        opennote.newJTextArea.append(string + "\n");

                    }

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

                int isDeleted = JOptionPane.showConfirmDialog(opennote, "是否删除该笔记？");

*if* (isDeleted == 0) {

                    ps.println("delete");

                    ps.println(row);

*try* {

*if* (br.readLine().compareTo("success") == 0) {

                            opennote.newJTextArea.setText("");

                            opennote.dispose();

                            this.dispose();

                            JOptionPane.showMessageDialog(this, "删除成功");

                        } *else* {

                            JOptionPane.showMessageDialog(this, "删除失败");

                        }

                    } *catch* (Exception ex) {

                        ex.printStackTrace();

                    }

                }

            }

        }

    }

    public class inquireNote extends mainInterface {

        protected JTable jTable = null;

        protected JScrollPane jScrollPane = null;

        public inquireNote(String str) {

            super(str);

            this.setTitle(str + "的查询笔记");

            this.dispose();

            String keyWord = JOptionPane.showInputDialog(this, "请输入您想要查询的关键词");

*if* (keyWord == null) {

*return*;

            }

*if* (keyWord.compareTo("") == 0) {

                JOptionPane.showMessageDialog(this, "未输入任何内容");

*return*;

            }

            this.setVisible(true);

            ps.println("inquire");

            ps.println(title);

            ps.println(keyWord);

            Vector<Vector<String>> data = null;

            ObjectInputStream objectInputStream = null;

*try* {

                objectInputStream = *new* ObjectInputStream(is);

                data = (Vector<Vector<String>>) objectInputStream.readObject();

            } *catch* (Exception ex) {

                System.out.println("objecetInputStream");

                ex.printStackTrace();

            }

            FlowLayout flowLayout = *new* FlowLayout();

            jPanel.setLayout(flowLayout);

            Vector<String> columnName = *new* Vector<>();

            columnName.add("笔记名称");

            columnName.add("出现次数");

*for* (int i = 1; i <= 5; i++) {

                columnName.add("第" + i + "行次出现在(行,列)");

            }

            jTable = *new* JTable(data, columnName) {

                public boolean isCellEditable(int row, int column) {

*return* false;

                }

            };

            jTable.setPreferredScrollableViewportSize(*new* Dimension(900, 500));

            Font font = *new* Font("楷体\_GB2312", Font.PLAIN, 15);

            jTable.setFont(font);

            DefaultTableCellRenderer defaultTableCellRenderer = *new* DefaultTableCellRenderer();

            defaultTableCellRenderer.setHorizontalAlignment(JLabel.CENTER);

            jTable.setDefaultRenderer(Object.class, defaultTableCellRenderer);

            jTable.setFillsViewportHeight(true);

            jScrollPane = *new* JScrollPane(jTable);

            jPanel.add(jScrollPane);

            this.add(jPanel);

            this.validate();

        }

    }

    public class downloadNote extends openNote {

        int row = jTable.getSelectedRow();

        JFileChooser jFileChooser = null;

        File noteDownload = null;

        FileWriter fileWriter = null;

        public downloadNote(String str) {

            super(str);

            this.setTitle(str + "的下载笔记");

        }

        public void valueChanged(ListSelectionEvent le) {

            int row = jTable.getSelectedRow();

*if* (le.getValueIsAdjusting() == false) {

                jFileChooser = *new* JFileChooser();

                jFileChooser.setDialogTitle("选择下载路径");

                jFileChooser.setApproveButtonText("选择");

                jFileChooser.setFileSelectionMode(JFileChooser.DIRECTORIES\_ONLY);

                int returnVal = jFileChooser.showOpenDialog(this);

                String noteDownloadPath = null;

*if* (returnVal == JFileChooser.APPROVE\_OPTION) {

                    noteDownloadPath = jFileChooser.getSelectedFile().getAbsolutePath();

                }

                String notename = JOptionPane.showInputDialog(this, "请输入笔记名称");

                noteDownload = *new* File(noteDownloadPath + "\\" + notename + ConstVarible.format\_txt);

*try* {

                    fileWriter = *new* FileWriter(noteDownload);

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

                ps.println(row);

*try* {

*while* (true) {

                        String string = br.readLine();

*if* (string.compareTo(ConstVarible.endFlag) == 0) {

*break*;

                        }

                        fileWriter.write(string + "\n");

                    }

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                    JOptionPane.showMessageDialog(this, "下载失败");

*return*;

                } *finally* {

*try* {

                        fileWriter.close();

                    } *catch* (Exception ex) {

                        ex.printStackTrace();

                    }

                }

                JOptionPane.showMessageDialog(this, "下载成功");

                this.dispose();

            }

        }

    }

    public class makefriend extends ancestor {

        public makefriend(String account) {

            super("交友");

            this.setVisible(false);

            String friendAccount = JOptionPane.showInputDialog(this, "请输入好友账号");

*if* (friendAccount == null || friendAccount.compareTo("") == 0) {

*return*;

            } *else* {

                ps.println(ConstVarible.startFlag + "makefriends");

                ps.println(account);

                ps.println(friendAccount);

*// JOptionPane.showMessageDialog(this, "hh");*

*try* {

                    JOptionPane.showMessageDialog(this, br.readLine());

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

            }

        }

    }

    public class listfriend extends ancestor {

        protected JTable jTable = null;

        protected JScrollPane jScrollPane = null;

        protected Vector<Vector<String>> data = *new* Vector<>();

        protected Vector<String> columnName = *new* Vector<>();

        public listfriend(String account) {

            super(account);

            this.setTitle("好友列表");

            ps.println(ConstVarible.startFlag + "list");

            ps.println(account);

            ObjectInputStream objectInputStream = null;

*try* {

                objectInputStream = *new* ObjectInputStream(is);

                data = (Vector<Vector<String>>) objectInputStream.readObject();

            } *catch* (Exception ex) {

                System.out.println("objecetInputStream");

                ex.printStackTrace();

            }

            FlowLayout flowLayout = *new* FlowLayout();

            jPanel.setLayout(flowLayout);

            columnName.add("好友列表");

            jTable = *new* JTable(data, columnName) {

                public boolean isCellEditable(int row, int column) {

*return* false;

                }

            };

            jTable.setPreferredScrollableViewportSize(*new* Dimension(600, 450));

            Font font = *new* Font(ConstVarible.font\_String, ConstVarible.font\_style, ConstVarible.font\_size);

            jTable.setFont(font);

            jTable.setFillsViewportHeight(true);

            DefaultTableCellRenderer defaultTableCellRenderer = *new* DefaultTableCellRenderer();

            defaultTableCellRenderer.setHorizontalAlignment(JLabel.CENTER);

            jTable.setDefaultRenderer(Object.class, defaultTableCellRenderer);

            jScrollPane = *new* JScrollPane(jTable);

            jPanel.add(jScrollPane);

            this.add(jPanel);

            this.setSize(700, 500);

            this.setLocationRelativeTo(null);

        }

    }

    public class shareNote extends openNote {

        public shareNote(String account) {

            super(account);

            this.setTitle(account + "的分享笔记");

        }

        public void valueChanged(ListSelectionEvent le) {

            int row = jTable.getSelectedRow();

*if* (le.getValueIsAdjusting() == false) {

                ps.println(row);

                newNote opennote = new newNote(title);

                opennote.setTitle(title + "的笔记：" + jTable.getModel().getValueAt(row, 0));

*try* {

*while* (true) {

                        String string = br.readLine();

*if* (string.compareTo(ConstVarible.endFlag) == 0) {

*break*;

                        }

                        opennote.newJTextArea.append(string + "\n");

                    }

                } *catch* (Exception ex) {

                    ex.printStackTrace();

                }

                String friendAccount = JOptionPane.showInputDialog(opennote, "请输入您想要分享该笔记的好友账号");

*if* (friendAccount != null && friendAccount.compareTo("") != 0) {

                    ps.println(ConstVarible.startFlag + "share");

                    ps.println(title);

                    ps.println(friendAccount);

                    ps.println(data.get(row).get(0));

*try* {

                        String str = br.readLine();

*if* (str.compareTo("success") == 0) {

                            JOptionPane.showMessageDialog(this, "分享成功，等待好友接受");

                        } *else* {

                            JOptionPane.showMessageDialog(this, "分享失败");

                        }

                    } *catch* (Exception ex) {

                        ex.printStackTrace();

                    }

                }

            }

        }

    }

    public class getSharedNote extends mainInterface implements ListSelectionListener {

        protected JTable jTable = null;

        protected JScrollPane jScrollPane = null;

        protected Vector<Vector<String>> data = *new* Vector<>();

        protected Vector<String> columnName = *new* Vector<>();

        public getSharedNote(String account) {

            super(account);

            this.setTitle("获得笔记");

*// 获取表格数据*

            columnName.add("笔记名称");

            columnName.add("分享者");

            ps.println(ConstVarible.startFlag + "getshared");

            ps.println(account);

*try* {

                ObjectInputStream ois = *new* ObjectInputStream(is);

                data = (Vector<Vector<String>>) ois.readObject();

            } *catch* (Exception ex) {

                ex.printStackTrace();

            }

*// 排版*

            FlowLayout flowLayout = *new* FlowLayout();

            jPanel.setLayout(flowLayout);

            jTable = *new* JTable(data, columnName) {

                public boolean isCellEditable(int row, int column) {

*return* false;

                }

            };

            jTable.setPreferredScrollableViewportSize(*new* Dimension(600, 450));

            Font font = *new* Font(ConstVarible.font\_String, ConstVarible.font\_style, ConstVarible.font\_size);

            jTable.setFont(font);

            jTable.getSelectionModel().addListSelectionListener(this);

            jTable.setFillsViewportHeight(true);

            DefaultTableCellRenderer defaultTableCellRenderer = *new* DefaultTableCellRenderer();

            defaultTableCellRenderer.setHorizontalAlignment(JLabel.CENTER);

            jTable.setDefaultRenderer(Object.class, defaultTableCellRenderer);

            jScrollPane = *new* JScrollPane(jTable);

            jPanel.add(jScrollPane);

            this.add(jPanel);

            this.setSize(700, 500);

            this.setLocationRelativeTo(null);

        }

        public void valueChanged(ListSelectionEvent le) {

            int row = jTable.getSelectedRow();

*if* (le.getValueIsAdjusting() == false) {

                int res = JOptionPane.showConfirmDialog(this, "是否接受该笔记");

*if* (res == JOptionPane.YES\_OPTION) {

                    ps.println(ConstVarible.startFlag + "getsharednote");

                    ps.println(title);

                    ps.println(data.get(row).get(0));

                    ps.println(data.get(row).get(1));

*try* {

*if* (br.readLine().compareTo("success") == 0) {

                            JOptionPane.showMessageDialog(this, "接收成功");

                        } *else* {

                            JOptionPane.showMessageDialog(this, "接收失败");

                        }

                    } *catch* (Exception ex) {

                        ex.printStackTrace();

                    }

                }

            }

        }

    }

    public static void main(String[] args) {

        client c = *new* client();

        login l = c.new login();

    }

}

ConstVarible.java

package code;

import java.awt.Font;

public class ConstVarible {

    public static final String address\_home\_wifi="192.168.0.102";

*// public static final String address\_home\_wifi="106.53.197.20";//云服务器*

    public static final int port=9999;

    public static final String startFlag="+\_)(\*&^%$#@!~";

    public static final String endFlag="!@#$%^&\*()";

    public static final String format\_txt=".txt";

    public static final String font\_String="楷体\_GB2312";

    public static final int font\_style=Font.PLAIN;

    public static final int font\_size=15;

}

# 实验总结

通过本次实验对java这项编程语言更加了解了，可以更加熟练的使用java编程语言进行编程。熟悉了java编程语言的swing、socket以及io流的操作。在遇到不懂的问题时通过咨询同学老师或者上网查询资料解决了不少难题，让我了解到在遇到困难时不能知难而退，要勇于面对，没有解决不了的问题。

**参考资料**

《用Java语言描述面向对象》作者：郭克华