****

Java实验报告

|  |  |
| --- | --- |
| 学　　院： | 计算机学院 |
| 专业班级： |  |
| 指导教师： |  |
| 学　　号： |  |
| 学生姓名： |  |

2020年1月

**目录**

[一、实验概述 1](#_Toc43499168)

[1.1 实验目的 1](#_Toc43499169)

[1.2 内容及要求 1](#_Toc43499170)

[1.3实验任务 1](#_Toc43499171)

[二、实验成果展示 2](#_Toc43499172)

[2.1 启动服务端 2](#_Toc43499173)

[2.2 启动客户端，注册账号、并登录 3](#_Toc43499174)

[2.3 云笔记主界面及笔记自动保存功能 5](#_Toc43499175)

[2.5创建新笔记，打开已有笔记，以及自动恢复功能 7](#_Toc43499176)

[2.6主动保存，以及复制、黏贴、全选、剪切四个基本编辑功能 8](#_Toc43499177)

[2.7打开作者博客 9](#_Toc43499178)

[三、核心代码讲解 11](#_Toc43499179)

[3.1 服务端与客户端之间的数据约定 11](#_Toc43499180)

[3.1.1 登录操作 11](#_Toc43499181)

[3.1.2 接收最近编辑的文档 11](#_Toc43499182)

[3.1.3 创建新的笔记 11](#_Toc43499183)

[3.1.4 创建新的账号 11](#_Toc43499184)

[3.1.5 打开以存在的文件 12](#_Toc43499185)

[3.1.3 客户端退出 12](#_Toc43499186)

[3.2 服务端核心数据结构 12](#_Toc43499187)

[3.2.1 /data/account 12](#_Toc43499188)

[3.2.2 文件夹/data/accountName 13](#_Toc43499189)

[3.3 服务器核心代码 14](#_Toc43499190)

[3.3.1 总览 14](#_Toc43499191)

[3.3.2 NoterServer.initServer() 14](#_Toc43499192)

[3.4 客户端核心代码 18](#_Toc43499193)

[四、实验总结 20](#_Toc43499194)

[五、代码 21](#_Toc43499195)

[5.1 服务端代码 21](#_Toc43499196)

[5.2 客户端代码 29](#_Toc43499197)

# 一、实验概述

## 1.1 实验目的

掌握Java语言与系统设计的基本思路和方法。

利用所学的基本知识和技能，解决简单的Java语言与系统设计问题。

## 1.2 内容及要求

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。

## 1.3实验任务

用JavaSocket编程开发云笔记本

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

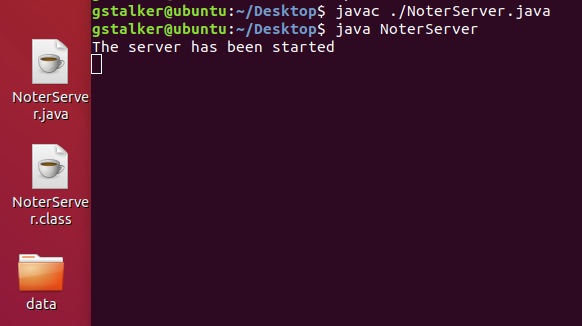
# 二、实验成果展示

## 2.1 启动服务端

编译源文件NoterServer.java

然后java NoterServer,启动客户端

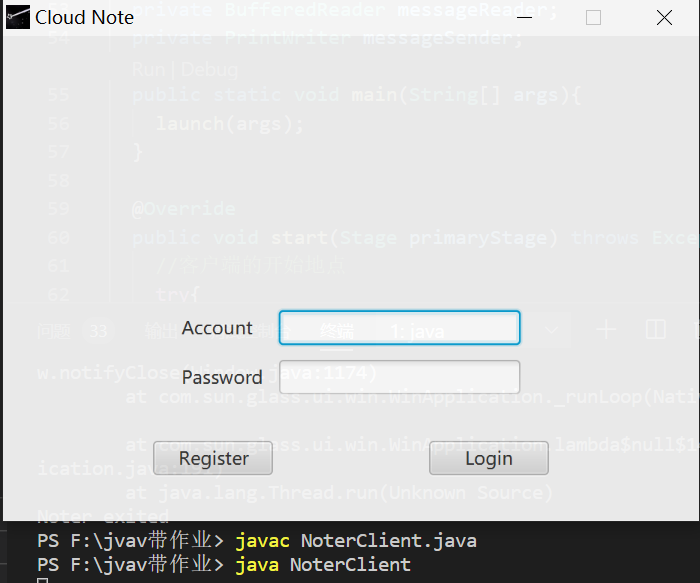
启动客户端前需保证当前data文件家与NoterServer在同一文件夹下



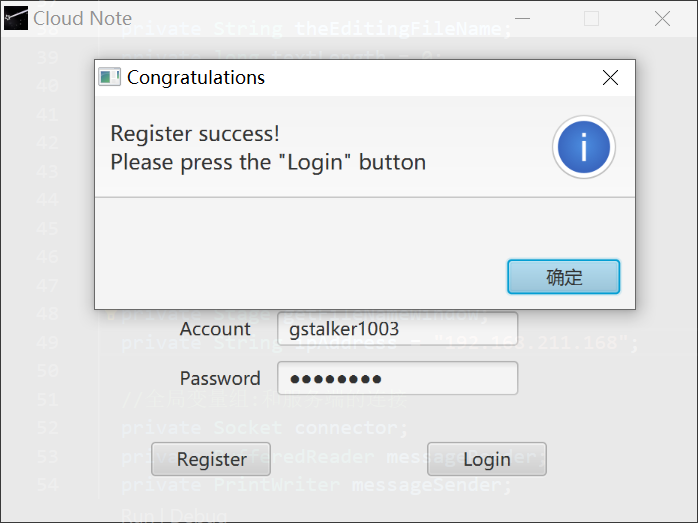
## 2.2 启动客户端，注册账号、并登录

编译NoterClient.java

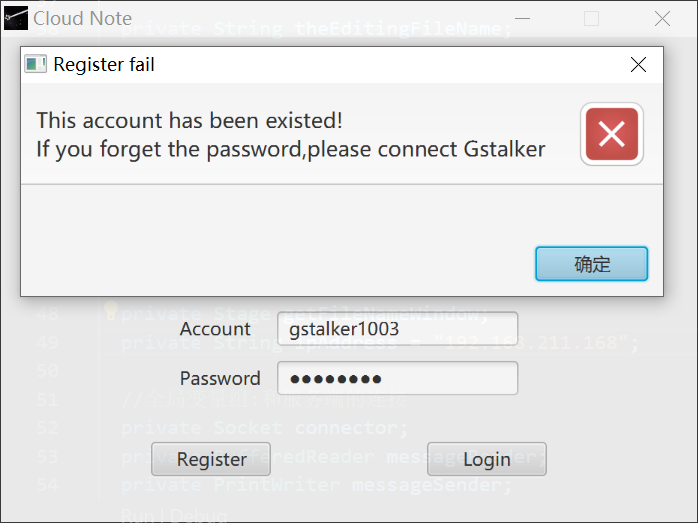
Java NoterClient，运行客户端



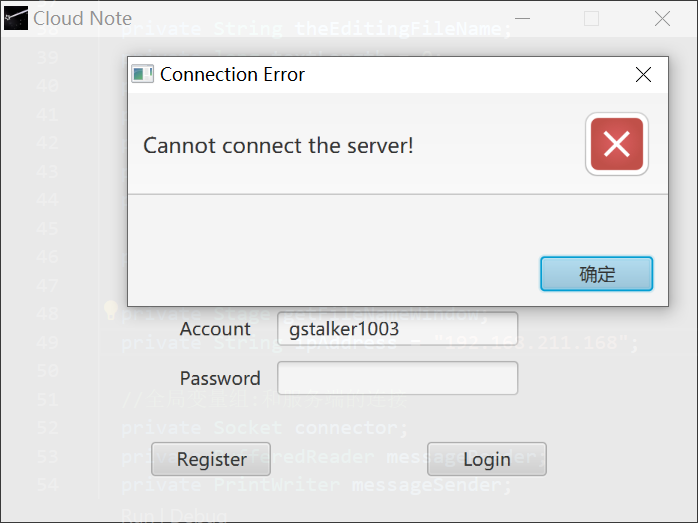
输入用户预期的账号及密码，点击Register，即可注册账号



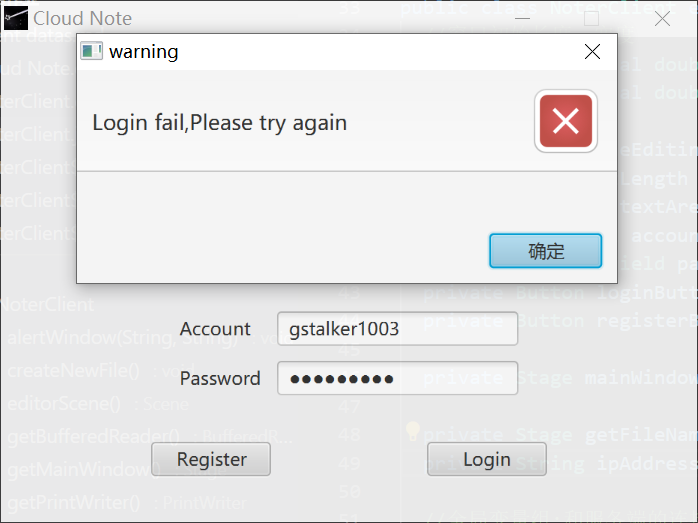
如果账号已存在，则会提示



如果无法连接服务器，则会提示



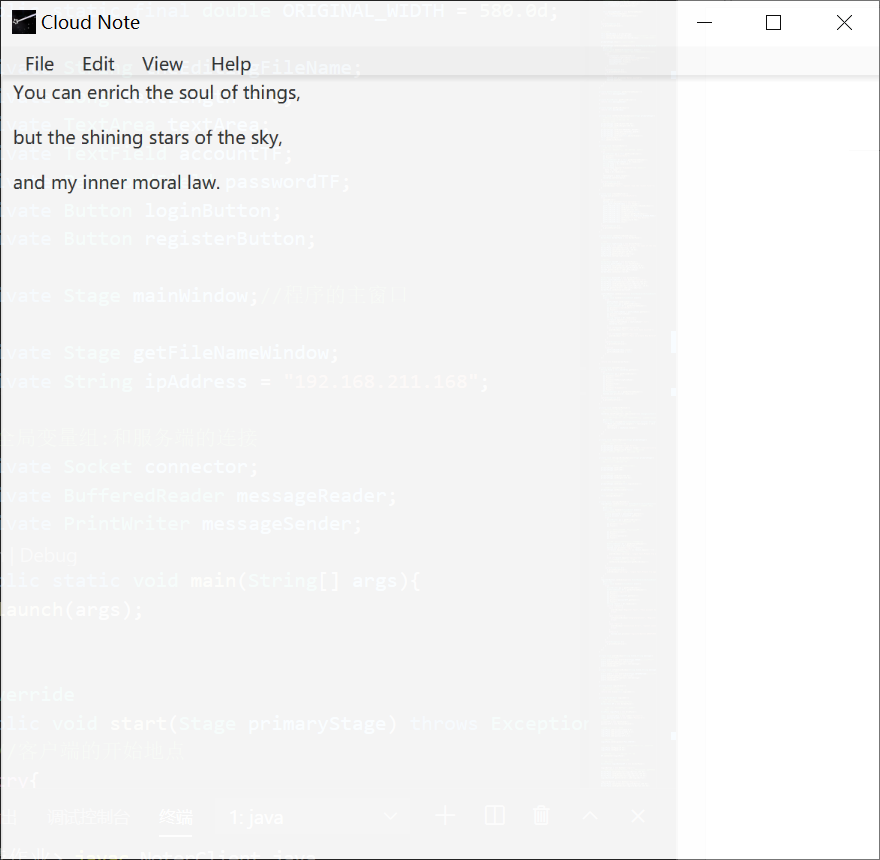
输入刚才注册的账号及密码，点击Login按钮。如果密码错误或者账号不存在，则会提示



如果账号和密码对服务端./data/account文件夹内记录的内容，则成功登录

## 2.3 云笔记主界面及笔记自动保存功能

登录成功后进入文本编辑器主界面



云笔记会自动保存用户输入的内容。当文本编辑框中字符的数量与上一次保存时的数量之差大于20，则客户端会自动上传当前编辑框中的笔记到服务器进行保存。这项功能通过javafx提供的文本监听接口完成

  private void setEditorMathod(){

    //函数setEditorMathod

    //功能：设定编辑模式的基本设置

    textArea.textProperty().addListener(new ChangeListener<String>() {

      //设定textArea中的文本监听器

      //当用户修改文本超过20个字符的时候，上传新的文本信息。

      @Override

      public void changed(ObservableValue<? extends String> observable,String oldValue,String newValue){

        if(Math.abs(newValue.length() - textLength) > 20){

          uploadFile();

          textLength = newValue.length();

        }

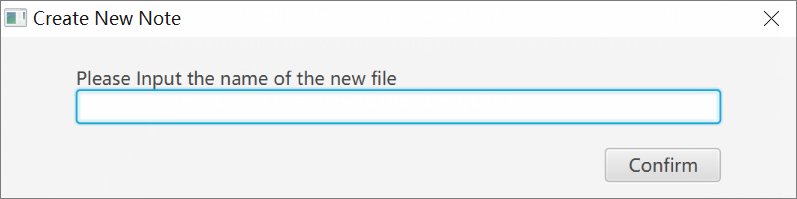
      }

    });

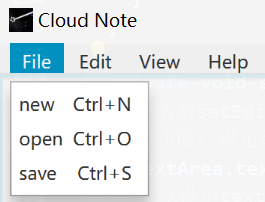
  }

## 2.5创建新笔记，打开已有笔记，以及自动恢复功能

选中云笔记主窗口，按下CTRL+N，即可进入创建新笔记的界面，只需要输入笔记名称即可创建新笔记



也可以在主窗口的左上角的菜单栏中找到创建新笔记的选项卡。CTRL+N实质上是我为该选项卡绑定的快捷键，通过javafx的快捷键接口实现



MenuItem newFile = new MenuItem("new");

    newFile.setOnAction(new EventHandler<ActionEvent>() {

      @Override

      public void handle(ActionEvent event){

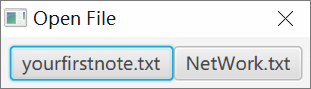
        createNewFile();

      }

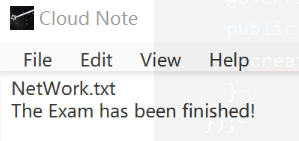
    });

    newFile.setAccelerator(KeyCombination.valueOf("CTRL+N"));

按下CTRL+O，可以进入打开现有笔记的窗口，这个窗口列举了当前用户现有的所有笔记。



关闭主窗口，再次登录。客户端会自动显示上一次用户编辑的笔记内容。



该功能通过以下代码实现

private void recvLastData(){

    //函数 recvLastData()

    //功能：从客户端接受上一次用户编辑过的数据

    try{

      PrintWriter pw = this.getPrintWriter();

      pw.println("recvLastData");

      pw.flush();

      BufferedReader br = this.getBufferedReader();

      this.theEditingFileName = br.readLine();

      String temp = br.readLine();

      String data = "";

      while(!temp.equals("0xbadbeef")){//文件结束

        data += temp+'\n';

        temp = br.readLine();

      }

      textLength = data.length();

      textArea.setText(data);

    }

    catch(Exception e){

      e.printStackTrace();

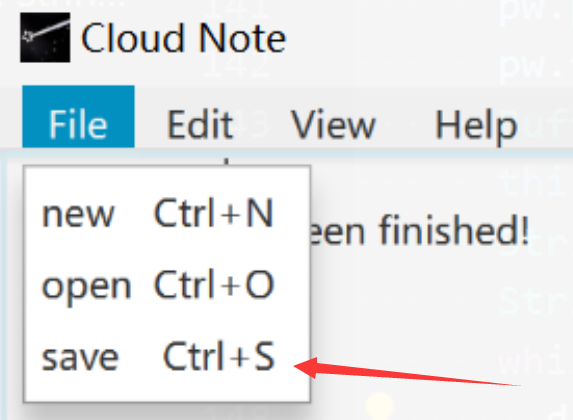
      alertWindow("fail", "Cannot read the recent file!");

    }

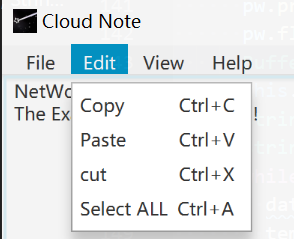
  }

## 2.6主动保存，以及复制、黏贴、全选、剪切四个基本编辑功能

除了自动保存以外，云笔记还提供主动保存功能。只需要像使用其它大多数文本编辑软件那样，按下熟悉的CTRL+S键即可完成保存当前文件的操作。也可以通过菜单栏点击选项卡的方式完成该操作



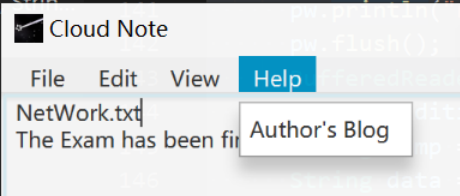
常用的四种编辑操作：



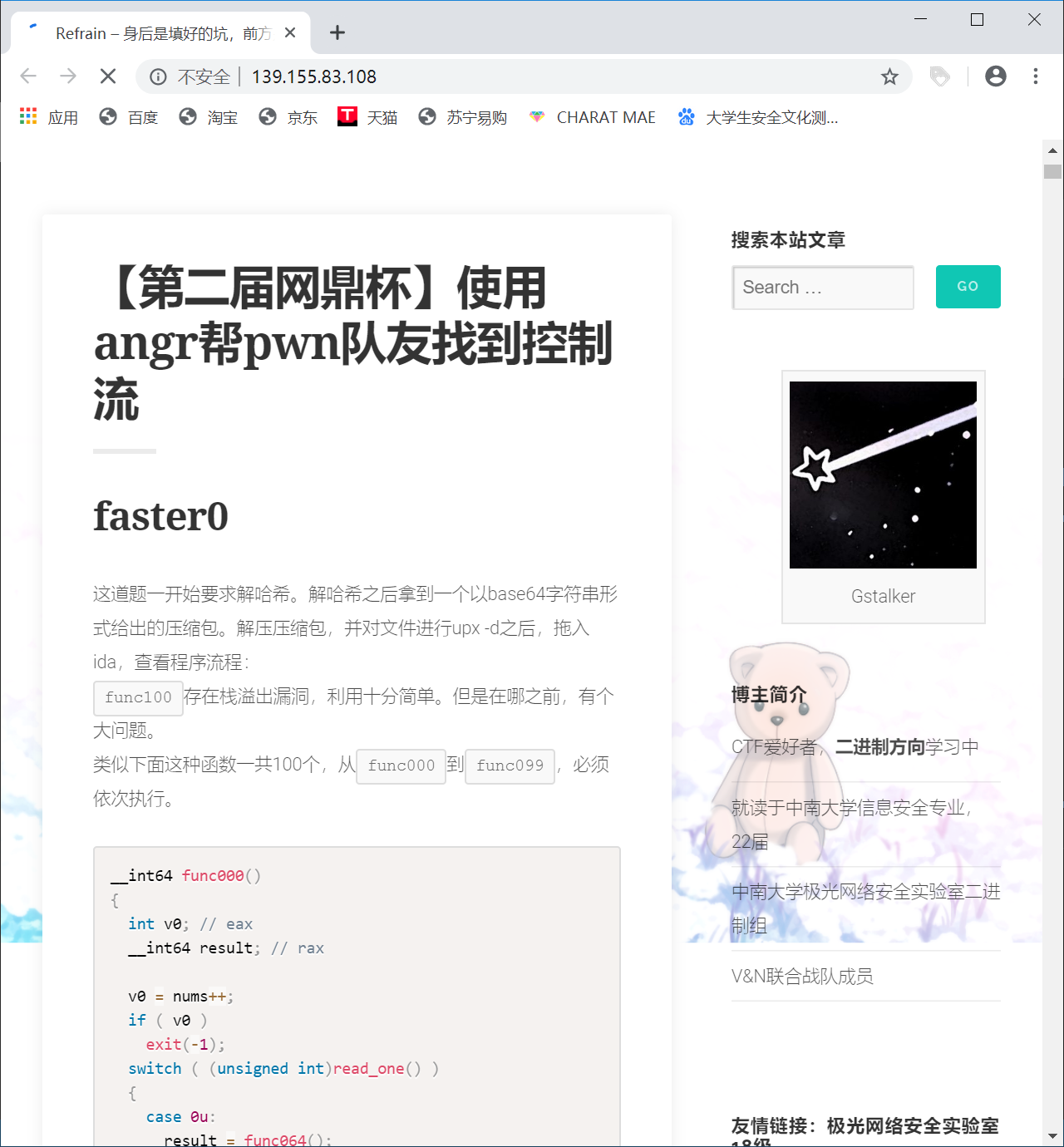
其效果不再赘述。快捷键实现原理同2.5节中new操作。

## 2.7打开作者博客

算是彩蛋。菜单栏最后一条Help中只有一个选项卡



单击这个选项卡，打开云笔记作者的博客



实现这项功能的代码如下:

//打开作者博客

    authorHelp.setOnAction(new EventHandler<ActionEvent>(){

      @Override

      public void handle(ActionEvent event) {

        HostServices host = getHostServices();

        host.showDocument("http://139.155.83.108/");

      }

    });

# 三、核心代码讲解

## 3.1 服务端与客户端之间的数据约定

### 3.1.1 登录操作

客户端发送字符串 “login”,账户，密码各一行给服务器

服务端发送：如果登录成功，则返回"success\n",失败则返回"fail\n"

如果登录成功，服务端进入登录状态

### 3.1.2 接收最近编辑的文档

客户端发送字符串 “recvLastData”

服务端发送：上一次编辑的文件名，和上一次编辑的文件

### 3.1.3 创建新的笔记

客户端发送字段"createNewNote"

客户端发送filename

服务器创建filename.txt

服务器创建成功后服务器返回success

存在同名文件则服务器返回“exist”

创建失败则服务器返回“fail“

### 3.1.4 创建新的账号

发送字段：

1. "register"

2. account

3. password

接收字段：

“exist:注册失败”,同名账号已存在

“success“:注册成功

“fail”：注册失败，网络问题

### 3.1.5 打开以存在的文件

客户端发送字段 “OpenFile"

服务器接收该字段后，发送文件总数，分行发送所有的文件名给客户端

客户端接收完成后，弹出窗口，等待用户选择文件名

客户端发送字段 filename（用户选择要打开的笔记的名称）

服务器接收该字段后，向客户端发送文件内容

客户端接收完毕后向服务器答复Success

### 3.1.3 客户端退出

客户端发送字符串 “exit”

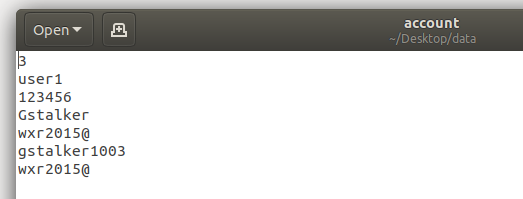
服务端退出登录状态

## 3.2 服务端核心数据结构

### 3.2.1 /data/account

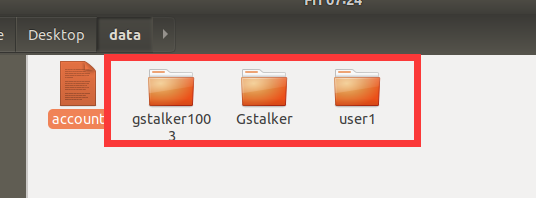
文件内容如图。

第一行的数字代表账户总数。1、3、5…行代表账户名，2、4、6…代表上一行的账户对应的密码



### 3.2.2 文件夹/data/accountName

服务端将用户数据储存在用户专属文件夹中



文件夹中有两类文件

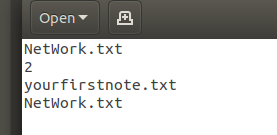
第一类：userdata.dat

Userdata.dat中储存的数据如图。

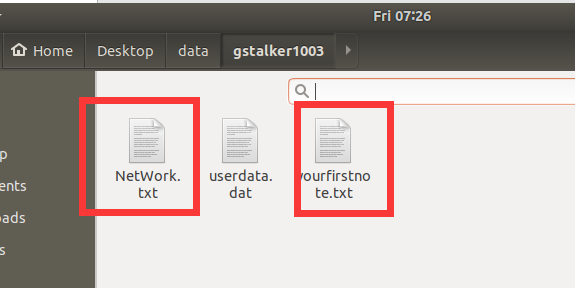
第一行是用户上一次退出前修改的笔记名称

第二行代表用户的笔记总数量

第三行、第四行往后是用户各个笔记的文件名称。



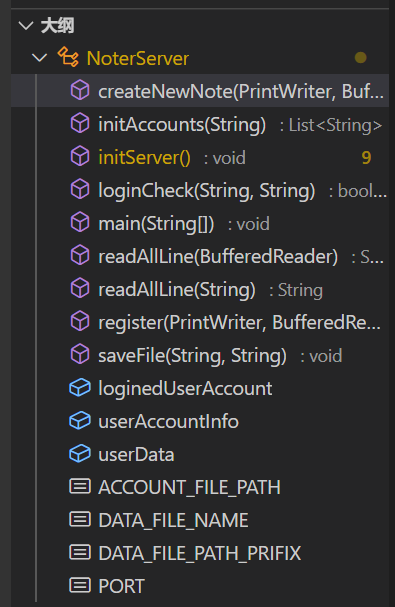
第二类：笔记文件



所有的笔记都以.txt的形式储存在这里

## 3.3 服务器核心代码

### 3.3.1 总览



图中 蓝色的是类成员变量，白色的是声明为private static final的类成员宏，紫色的是类成员函数

其中核心代码是NoterServer.initServer()

### 3.3.2 NoterServer.initServer()

这个函数是服务器端的核心函数。

这个函数首先创建了服务端Socket，然后通过菜单式监听实现响应客户端发送的不同请求，完成了服务端提供服务的任务

  public void initServer(){

    userAccountInfo = this.initAccounts(ACCOUNT\_FILE\_PATH);

    try(

      ServerSocket server = new ServerSocket(PORT);

      Socket socket = server.accept();

      InputStream is = socket.getInputStream();

      BufferedReader br = new BufferedReader(new InputStreamReader(is));

      OutputStream os = socket.getOutputStream();

      PrintWriter pw = new PrintWriter(os);

    ){

      System.out.println("The server has been started");

      while(true){

        String userRequestCode = br.readLine();

        System.out.println("new Request:    "+userRequestCode);

        if(userRequestCode == null){

          try{

            Thread.sleep(500);

          }

          catch(Exception e){

            e.printStackTrace();

          }

          continue;

        }

        switch(userRequestCode){

          case "login":{

            //login

            System.out.println("request recived:  login");

            //接收用户发送的账户和密码

            String account = br.readLine();

            System.out.println("account :    "+account);

            String password = br.readLine();

            System.out.println("password:    "+password);

            //登录检测

            if(account != null && password!= null && this.loginCheck(account,password)){

              //登录成功

              System.out.println("Login Success!,user:    "+account);

              this.loginedUserAccount = account;

              //告诉客户端，登录成功

              pw.println("success");

              pw.flush();

            }

            else{

              System.out.println("Login Fail!");

              pw.println("fail");

              pw.flush();

            }

            break;

          }

          case "recvLastData":{

            //recvLastData

            System.out.println("request recived:  recvLastData");

            this.userData = this.initAccounts(this.DATA\_FILE\_PATH\_PRIFIX+"/"+this.loginedUserAccount+this.DATA\_FILE\_NAME);

            String file = this.readAllLine(this.DATA\_FILE\_PATH\_PRIFIX+"/"+this.loginedUserAccount+"/"+this.userData.get(0));

            pw.println((String)this.userData.get(0));//发送文件名

            pw.flush();

            pw.write(file);//发送文件

            pw.flush();

            pw.println("0xbadbeef");

            pw.flush();

            break;

          }

          case "OpenFile":{

            System.out.println("request recived:  OpenFile");

            pw.println(userData.get(1));

            for(int i = 2 ; i < userData.size() ;++i){

              pw.println(userData.get(i));

              pw.flush();

            }

            String fileName = br.readLine();

            System.out.println("    OpenFile: "+fileName);

            String file = readAllLine(DATA\_FILE\_PATH\_PRIFIX+"/"+loginedUserAccount+"/"+fileName);

            pw.println(file);

            pw.flush();

            pw.println("0xbadbeef");

            pw.flush();

            System.out.println("    Send File "+fileName+' '+br.readLine());

            userData.set(0,fileName);

            String tmp = "";

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

              tmp += userData.get(i)+'\n';

            }

            saveFile(DATA\_FILE\_PATH\_PRIFIX+'/'+loginedUserAccount+'/'+"/userdata.dat",tmp);

            break;

          }

          case "update":{

            //update

            System.out.println("request recived:  update");

            String fileName = br.readLine();

            String file = this.readAllLine(br);

            this.saveFile(this.DATA\_FILE\_PATH\_PRIFIX+"/"+this.loginedUserAccount+"/"+fileName,file);

            pw.println("Auto Save Complete!");

            pw.flush();

            break;

          }

          case "register":{

            System.out.println("request recived:  register");

            register(pw,br);

            break;

          }

          case "createNewNote":{

            System.out.println("request recived:  createNewNote");

            createNewNote(pw,br);

            break;

          }

          case "exit":{

            //exit

            String newUserData = "";

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

              newUserData += userData.get(i) + '\n';

            }

            saveFile(DATA\_FILE\_PATH\_PRIFIX+'/'+loginedUserAccount+'/'+"/userdata.dat" , newUserData);

            System.out.println("request recived:  exit");

            return;

          }

          default :{

            //dead

            System.out.println("request recived:  UNDIFINED");

            break;

          }

        }

      }

    }

    catch(IOException e){

      e.printStackTrace();

    }

  }

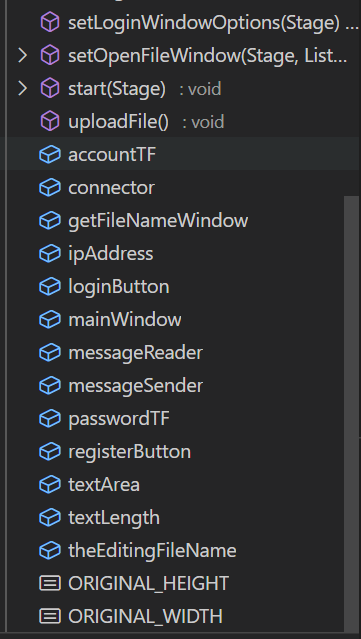
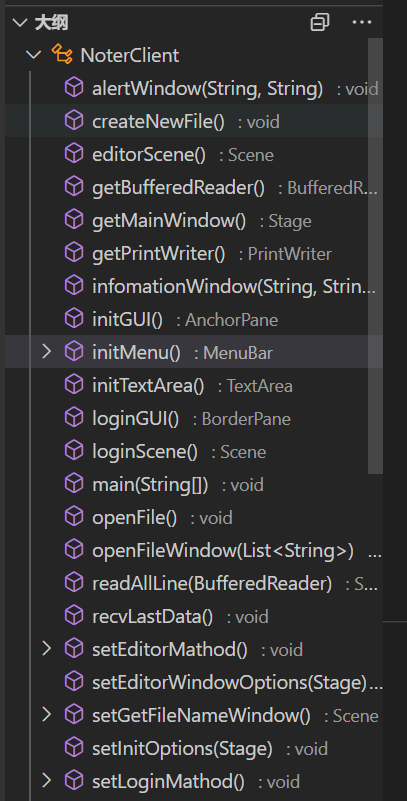
## 3.4 客户端核心代码

客户端使用javafx实现gui。类成员函数分为两类：

第一类：初始化gui窗口

第二类：设定gui窗口的功能，以及核心功能类代码

大纲如下：



其中NoterClient.start()是本程序的入口点。

@Override

  public void start(Stage primaryStage) throws Exception{

    //客户端的开始地点

    try{

      //创建并保留和服务器的连接

      this.connector = new Socket(ipAddress,19613);

      this.messageSender = new PrintWriter(

        this.connector.getOutputStream()

      );

      this.messageReader = new BufferedReader(

        new InputStreamReader(

          this.connector.getInputStream()

        )

      );

    }

    catch(Exception e){

      e.printStackTrace();

    }

    //设定GUI

    this.mainWindow = primaryStage;

    this.setInitOptions(mainWindow);//设定title，图标，透明度

    this.setLoginWindowOptions(mainWindow);//设定窗口组件及其功能

    primaryStage.show();//展示登录窗口

    primaryStage.setOnCloseRequest(new EventHandler<WindowEvent>() {//退出事件

      @Override

      public void handle(WindowEvent event){

        //关闭程序的时候，保存当前正在编辑的文件，并且切断和服务器的连接

        try{

          if(connector.isConnected()){

            if(theEditingFileName!=null)uploadFile();

            messageSender.println("exit");

            messageSender.flush();

            connector.close();

          }

        }

        catch(Exception e){

          e.printStackTrace();

        }

        finally{

          System.out.println("Noter exited");

          System.exit(0);

        }

      }

    });

  }

# 四、实验总结

通过此次课程设计，使我更加扎实的掌握了有关javaSocket编程和javafx图形化框架方面的知识，在设计过程中虽然遇到了一些问题，但经过一次又一次的思考，一遍又一遍的检查终于找出了原因所在，也暴露出了前期我在这方面的知识欠缺和经验不足。实践出真知，通过亲自动手制作，使我们掌握的知识不再是纸上谈兵。 过而能改，善莫大焉。在课程设计过程中，我们不断发现错误，不断改正，不断领悟，不断获取。最终的检测调试环节，本身就是在践行“过而能改，善莫大焉”的知行观。这次课程设计终于顺利完成了，在设计中遇到了很多问题，最后在老师的指导下，终于游逆而解。在今后社会的发展和学习实践过程中，一定要不懈努力，不能遇到问题就想到要退缩，一定要不厌其烦的发现问题所在，然后一一进行解决，只有这样，才能成功的做成想做的事，才能在今后的道路上劈荆斩棘，而不是知难而退，那样永远不可能收获成功，收获喜悦，也永远不可能得到社会及他人对你的认可！ 课程设计诚然是一门专业课，给我很多专业知识以及专业技能上的提升，同时又是一门讲道课，一门辩思课，给了我许多道，给了我很多思，给了我莫大的空间。同时，设计让我感触很深。使我对抽象的理论有了具体的认识。通过这次课程设计，我掌握了javaSocket编程和javafx图形化框架的识别和测试；熟悉了javaSocket编程和javafx图形化框架；了解了javaSocket编程和javafx图形化框架方法；以及如何提高javaSocket编程和javafx图形化框架的性能等等，掌握了javaSocket编程和javafx图形化框架的方法和技术，通过查询资料，也了解了javaSocket编程和javafx图形化框架原理。 我认为，在这学期的实验中，不仅培养了独立思考、动手操作的能力，在各种其它能力上也都有了提高。更重要的是，在实验课上，我们学会了很多学习的方法。而这是日后最实用的，真的是受益匪浅。要面对社会的挑战，只有不断的学习、实践，再学习、再实践。这对于我们的将来也有很大的帮助。以后，不管有多苦，我想我们都能变苦为乐，找寻有趣的事情，发现其中珍贵的事情。就像中国提倡的艰苦奋斗一样，我们都可以在实验结束之后变的更加成熟，会面对需要面对的事情。 回顾起此课程设计，至今我仍感慨颇

多，从理论到实践，在这段日子里，可以说得是苦多于甜，但是可以学到很多很多的东西，同时不仅可以巩固了以前所学过的知识，而且学到了很多在书本上所没有学到过的知识。通过这次课程设计使我懂得了理论与实际相结合是很重要的，只有理论知识是远远不够的，只有把所学的理论知识与实践相结合起来，从理论中得出结论，才能真正为社会服务，从而提高自己的实际动手能力和独立思考的能力。在设计的过程中遇到问题，可以说得是困难重重，但可喜的是最终都得到了解决。 实验过程中，也对团队精神的进行了考察，让我们在合作起来更加默契，在成功后一起体会喜悦的心情。果然是团结就是力量，只有互相之间默契融洽的配合才能换来最终完美的结果

# 五、代码

## 5.1 服务端代码

import java.net.\*;

import java.util.ArrayList;

import java.util.List;

import java.io.\*;

public class NoterServer{

  private List<String> userAccountInfo;//1，3，5，7...是账户名

                               //2,4,6,7...是账户密码

                               //第0项是账户数量。

  private List<String> userData;//保存了用户的文件名

                        //0:上一次打开的文本

                        //1:用户的笔记总数

                        //2~:用户的笔记名

  private static final String ACCOUNT\_FILE\_PATH = "./data/account";//储存文件的路径地址

  private static final String DATA\_FILE\_PATH\_PRIFIX = "./data";

  private static final String DATA\_FILE\_NAME = "/userdata.dat";

  private static final int PORT = 19613;//开放端口

  private String loginedUserAccount = null;

  public static void main(String args[]){

    NoterServer server = new NoterServer();

    while(true)server.initServer();

  }

  public void initServer(){

    userAccountInfo = this.initAccounts(ACCOUNT\_FILE\_PATH);

    try(

      ServerSocket server = new ServerSocket(PORT);

      Socket socket = server.accept();

      InputStream is = socket.getInputStream();

      BufferedReader br = new BufferedReader(new InputStreamReader(is));

      OutputStream os = socket.getOutputStream();

      PrintWriter pw = new PrintWriter(os);

    ){

      System.out.println("The server has been started");

      while(true){

        String userRequestCode = br.readLine();

        System.out.println("new Request:    "+userRequestCode);

        if(userRequestCode == null){

          try{

            Thread.sleep(500);

          }

          catch(Exception e){

            e.printStackTrace();

          }

          continue;

        }

        switch(userRequestCode){

          case "login":{

            //login

            System.out.println("request recived:  login");

            //接收用户发送的账户和密码

            String account = br.readLine();

            System.out.println("account :    "+account);

            String password = br.readLine();

            System.out.println("password:    "+password);

            //登录检测

            if(account != null && password!= null && this.loginCheck(account,password)){

              //登录成功

              System.out.println("Login Success!,user:    "+account);

              this.loginedUserAccount = account;

              //告诉客户端，登录成功

              pw.println("success");

              pw.flush();

            }

            else{

              System.out.println("Login Fail!");

              pw.println("fail");

              pw.flush();

            }

            break;

          }

          case "recvLastData":{

            //recvLastData

            System.out.println("request recived:  recvLastData");

            this.userData = this.initAccounts(this.DATA\_FILE\_PATH\_PRIFIX+"/"+this.loginedUserAccount+this.DATA\_FILE\_NAME);

            String file = this.readAllLine(this.DATA\_FILE\_PATH\_PRIFIX+"/"+this.loginedUserAccount+"/"+this.userData.get(0));

            pw.println((String)this.userData.get(0));//发送文件名

            pw.flush();

            pw.write(file);//发送文件

            pw.flush();

            pw.println("0xbadbeef");

            pw.flush();

            break;

          }

          case "OpenFile":{

            System.out.println("request recived:  OpenFile");

            pw.println(userData.get(1));

            for(int i = 2 ; i < userData.size() ;++i){

              pw.println(userData.get(i));

              pw.flush();

            }

            String fileName = br.readLine();

            System.out.println("    OpenFile: "+fileName);

            String file = readAllLine(DATA\_FILE\_PATH\_PRIFIX+"/"+loginedUserAccount+"/"+fileName);

            pw.println(file);

            pw.flush();

            pw.println("0xbadbeef");

            pw.flush();

            System.out.println("    Send File "+fileName+' '+br.readLine());

            userData.set(0,fileName);

            String tmp = "";

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

              tmp += userData.get(i)+'\n';

            }

            saveFile(DATA\_FILE\_PATH\_PRIFIX+'/'+loginedUserAccount+'/'+"/userdata.dat",tmp);

            break;

          }

          case "update":{

            //update

            System.out.println("request recived:  update");

            String fileName = br.readLine();

            String file = this.readAllLine(br);

            this.saveFile(this.DATA\_FILE\_PATH\_PRIFIX+"/"+this.loginedUserAccount+"/"+fileName,file);

            pw.println("Auto Save Complete!");

            pw.flush();

            break;

          }

          case "register":{

            System.out.println("request recived:  register");

            register(pw,br);

            break;

          }

          case "createNewNote":{

            System.out.println("request recived:  createNewNote");

            createNewNote(pw,br);

            break;

          }

          case "exit":{

            //exit

            String newUserData = "";

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

              newUserData += userData.get(i) + '\n';

            }

            saveFile(DATA\_FILE\_PATH\_PRIFIX+'/'+loginedUserAccount+'/'+"/userdata.dat" , newUserData);

            System.out.println("request recived:  exit");

            return;

          }

          default :{

            //dead

            System.out.println("request recived:  UNDIFINED");

            break;

          }

        }

      }

    }

    catch(IOException e){

      e.printStackTrace();

    }

  }

  private void createNewNote(PrintWriter messageWriter,BufferedReader messageReader){

    try{

      String newFileName = messageReader.readLine()+".txt";

      int totalFileCount = Integer.parseInt(userData.get(1));

      for(int i = 2;i < totalFileCount+2;++i){

        if(newFileName.equals(userData.get(i))){

          messageWriter.println("exist");

          messageWriter.flush();

          return;

        }

      }

      String userPath = DATA\_FILE\_PATH\_PRIFIX+'/'+loginedUserAccount;

      String emptyFile = "empty";

      saveFile(userPath+'/'+newFileName,emptyFile);

      userData.set(0,newFileName);

      userData.set(1,String.valueOf(totalFileCount+1));

      userData.add(newFileName);

      String newUserData = "";

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

        newUserData += userData.get(i) + '\n';

      }

      saveFile(userPath+"/userdata.dat" , newUserData);

      messageWriter.println("success");

      messageWriter.flush();

    }

    catch(Exception e){

      e.printStackTrace();

    }

  }

  private void register(PrintWriter messageWriter,BufferedReader messageReader){

    try{

      String account = messageReader.readLine();

      String password = messageReader.readLine();

      System.out.println("Account: "+account);

      System.out.println("Password: "+password);

      if(account.equals("account") || password.equals("")){

        messageWriter.println("fail");

        messageWriter.flush();

        return;

      }

      for(int i = 1 ; i <= Integer.parseInt(userAccountInfo.get(0))\*2;i+=2){

        if(account.equals(userAccountInfo.get(i))){

          messageWriter.println("exist");

          messageWriter.flush();

          return;

        }

      }

      userAccountInfo.add(account);

      userAccountInfo.add(password);

      userAccountInfo.set(0, String.valueOf(Integer.parseInt(userAccountInfo.get(0))+1));

      messageWriter.println("success");

      messageWriter.flush();

      String userFilePath = DATA\_FILE\_PATH\_PRIFIX+'/'+account;

      File userDir = new File(userFilePath);

      if(!userDir.exists()){

        userDir.mkdir();

      }

      String userDat = "yourfirstnote.txt\n1\nyourfirstnote.txt";

      String firstFile = "You can enrich the soul of things,\n\nbut the shining stars of the sky,\n\nand my inner moral law.";

      saveFile(userFilePath + DATA\_FILE\_NAME , userDat);

      saveFile(userFilePath + "/yourfirstnote.txt" , firstFile);

      String newAccountFile = String.valueOf(Integer.parseInt(userAccountInfo.get(0)))+'\n';

      for(int i = 1 ; i < userAccountInfo.size() ; ++i){

        newAccountFile += userAccountInfo.get(i)+'\n';

      }

      saveFile(ACCOUNT\_FILE\_PATH,newAccountFile);

    }

    catch(Exception e){

      e.printStackTrace();

    }

  }

  private boolean loginCheck(String account,String password){

    //有几个账户

    int total = Integer.parseInt(this.userAccountInfo.get(0));

    //对所有储存着的账户进行一对一检测

    for(int i = 1;i<=total\*2;i+=2){

      if(

        account.equals(this.userAccountInfo.get(i)) &&

        password.equals(this.userAccountInfo.get(i+1))

      ){

        return true;

      }

    }

    return false;

  }

  private void saveFile(String filePath,String usrInput){

    //功能：向位于filePath位置的文件写入usrInput

    try(

      FileWriter fwriter = new FileWriter(filePath);

    ){

      fwriter.write(usrInput);

      fwriter.flush();

      fwriter.close();

    }catch(Exception e){

      e.printStackTrace();

    }

  }

  private List<String> initAccounts(String filePath){

  // 函数：initAccounts

  //功能：返回一个列表，其中第一项是账户个数

  //1，3，5，7...是账户名

  //2,4,6,7...是账户密码

    List<String> result = new ArrayList<>();

    try(

      BufferedReader file = new BufferedReader(new FileReader(filePath));

    ){

      String temp;

      while((temp = file.readLine()) != null){

        result.add(temp);

      }

    }

    catch(Exception e){

      e.printStackTrace();

    }

    return result;

  }

  private String readAllLine(String filePath){

    //全文读取一个文件并且以String形式返回

    String result = "";

    try(

      BufferedReader file = new BufferedReader(new FileReader(filePath));

    ){

      String temp;

      while((temp = file.readLine()) != null){

        result += temp + '\n';

      }

    }

    catch(Exception e){

      e.printStackTrace();

    }

    return result;

  }

  private String readAllLine(BufferedReader messageReader){

    //全文读取一个文件并且以String形式返回

    String result = "";

    try{

      String temp = messageReader.readLine();

      while(!temp.equals("0xbadbeef")){//文件结束

        result += temp+'\n';

        temp = messageReader.readLine();

      }

    }

    catch(Exception e){

      e.printStackTrace();

    }

    return result;

  }

}

## 5.2 客户端代码

import javafx.application.Application;

import javafx.application.HostServices;

import javafx.beans.value.\*;

import javafx.event.ActionEvent;

import javafx.event.EventHandler;

import javafx.geometry.Orientation;

import javafx.geometry.Pos;

import javafx.scene.Scene;

import javafx.scene.control.Alert;

import javafx.scene.control.Button;

import javafx.scene.control.Label;

import javafx.scene.control.Menu;

import javafx.scene.control.MenuBar;

import javafx.scene.control.MenuItem;

import javafx.scene.control.PasswordField;

import javafx.scene.control.TextArea;

import javafx.scene.control.TextField;

import javafx.scene.control.Alert.AlertType;

import javafx.scene.image.Image;

import javafx.scene.input.KeyCombination;

import javafx.scene.layout.AnchorPane;

import javafx.scene.layout.BorderPane;

import javafx.scene.layout.FlowPane;

import javafx.scene.layout.GridPane;

import javafx.stage.Modality;

import javafx.stage.Stage;

import javafx.stage.WindowEvent;

import java.net.\*;

import java.util.ArrayList;

import java.util.List;

import java.io.\*;

public class NoterClient extends Application{

  //窗口初始长度，宽度

  public static final double ORIGINAL\_HEIGHT = 600.0d;

  public static final double ORIGINAL\_WIDTH = 580.0d;

  private String theEditingFileName;

  private long textLength = 0;

  private TextArea textArea;

  private TextField accountTF;

  private PasswordField passwordTF;

  private Button loginButton;

  private Button registerButton;

  private Stage mainWindow;//程序的主窗口

  private Stage getFileNameWindow;

  private String ipAddress = "192.168.211.168";

  //全局变量组:和服务端的连接

  private Socket connector;

  private BufferedReader messageReader;

  private PrintWriter messageSender;

  public static void main(String[] args){

    launch(args);

  }

  @Override

  public void start(Stage primaryStage) throws Exception{

    //客户端的开始地点

    try{

      //创建并保留和服务器的连接

      this.connector = new Socket(ipAddress,19613);

      this.messageSender = new PrintWriter(

        this.connector.getOutputStream()

      );

      this.messageReader = new BufferedReader(

        new InputStreamReader(

          this.connector.getInputStream()

        )

      );

    }

    catch(Exception e){

      e.printStackTrace();

    }

    //设定GUI

    this.mainWindow = primaryStage;

    this.setInitOptions(mainWindow);//设定title，图标，透明度

    this.setLoginWindowOptions(mainWindow);//设定窗口组件及其功能

    primaryStage.show();//展示登录窗口

    primaryStage.setOnCloseRequest(new EventHandler<WindowEvent>() {//退出事件

      @Override

      public void handle(WindowEvent event){

        //关闭程序的时候，保存当前正在编辑的文件，并且切断和服务器的连接

        try{

          if(connector.isConnected()){

            if(theEditingFileName!=null)uploadFile();

            messageSender.println("exit");

            messageSender.flush();

            connector.close();

          }

        }

        catch(Exception e){

          e.printStackTrace();

        }

        finally{

          System.out.println("Noter exited");

          System.exit(0);

        }

      }

    });

  }

  private BufferedReader getBufferedReader(){

    return this.messageReader;

  }

  private PrintWriter getPrintWriter(){

    return this.messageSender;

  }

  private Stage getMainWindow(){

    return this.mainWindow;

  }

  private void setEditorWindowOptions(Stage primaryStage){

    //函数：setLoginWindowOptions

    //功能：设定editor窗口的参数

    //窗口最小的大小

    primaryStage.setMinWidth(400.0d);

    primaryStage.setMinHeight(200.0d);

    //Editor窗口的大小

    primaryStage.setWidth(ORIGINAL\_HEIGHT);

    primaryStage.setHeight(ORIGINAL\_WIDTH);

    primaryStage.setResizable(true);

    primaryStage.setScene(this.editorScene());

    this.recvLastData();

    this.setEditorMathod();

  }

  private void recvLastData(){

    //函数 recvLastData()

    //功能：从客户端接受上一次用户编辑过的数据

    try{

      PrintWriter pw = this.getPrintWriter();

      pw.println("recvLastData");

      pw.flush();

      BufferedReader br = this.getBufferedReader();

      this.theEditingFileName = br.readLine();

      String temp = br.readLine();

      String data = "";

      while(!temp.equals("0xbadbeef")){//文件结束

        data += temp+'\n';

        temp = br.readLine();

      }

      textLength = data.length();

      textArea.setText(data);

    }

    catch(Exception e){

      e.printStackTrace();

      alertWindow("fail", "Cannot read the recent file!");

    }

  }

  private void createNewFile(){

    System.out.println("createNewFile");

    try{

      uploadFile();

      System.out.println("file uploaded!");

      this.getFileNameWindow = new Stage();

      getFileNameWindow.setScene(setGetFileNameWindow());

      getFileNameWindow.setWidth(533.34);

      getFileNameWindow.setHeight(133.34);

      getFileNameWindow.setResizable(false);

      getFileNameWindow.initOwner(getMainWindow());

      getFileNameWindow.initModality(Modality.WINDOW\_MODAL);

      getFileNameWindow.setTitle("Create New Note");

      getFileNameWindow.setOpacity(0.90d);

      getFileNameWindow.show();

    }

    catch(Exception e){

      e.printStackTrace();

    }

  }

  private Scene setGetFileNameWindow(){

    BorderPane borderPane = new BorderPane();

    //填充

    AnchorPane topFilling = new AnchorPane();

    Label tip = new Label("Please Input the name of the new file");

    AnchorPane.setTopAnchor(tip, 20.0d);

    AnchorPane.setLeftAnchor(tip, 50.0d);

    topFilling.setPrefHeight(30.0d);

    topFilling.getChildren().add(tip);

    borderPane.setTop(topFilling);

    //文本框

    AnchorPane textAP = new AnchorPane();

    TextField getFileName = new TextField();

    AnchorPane.setLeftAnchor(getFileName,50.0);

    AnchorPane.setRightAnchor(getFileName,50.0);

    textAP.getChildren().add(getFileName);

    borderPane.setCenter(textAP);

    //按钮

    AnchorPane buttonAP = new AnchorPane();

    Button confirmButton = new Button("Confirm");

    AnchorPane.setLeftAnchor(confirmButton,402.44);

    AnchorPane.setRightAnchor(confirmButton,50.0);

    AnchorPane.setBottomAnchor(confirmButton, 10.0);

    buttonAP.getChildren().add(confirmButton);

    borderPane.setBottom(buttonAP);

    confirmButton.setOnAction(new EventHandler<ActionEvent>() {

      @Override

      public void handle(ActionEvent event){

        try{

          getFileName.setDisable(true);

          PrintWriter pw = getPrintWriter();

          BufferedReader br = getBufferedReader();

          pw.println("createNewNote");

          pw.flush();

          String newFileName = getFileName.getText();

          pw.println(newFileName);

          pw.flush();

          String result = br.readLine();

          if(result.equals("success")){

            theEditingFileName = newFileName+".txt";

            textArea.clear();

          }

          else if(result.equals("exist")){

            alertWindow("Fail!","The note have existed");

          }

          else if(result.equals("fail")){

            alertWindow("Fail!","Fail in Create New Note\nTry again!");

          }

        }

        catch(Exception e){

          e.printStackTrace();

        }

        finally{

          getFileNameWindow.close();

          getFileNameWindow = null;

        }

      }

    });

    return new Scene(borderPane);

  }

  private void uploadFile(){

    String file = this.textArea.getText();

    try{

      PrintWriter pw = getPrintWriter();

      pw.println("update");

      pw.flush();

      pw.println(theEditingFileName);

      pw.flush();

      pw.println(file);

      pw.flush();

      pw.println("0xbadbeef");

      pw.flush();

      BufferedReader br = getBufferedReader();

      System.out.println(br.readLine());

    }

    catch(Exception e){

      e.printStackTrace();

    }

  }

  private void setEditorMathod(){

    //函数setEditorMathod

    //功能：设定编辑模式的基本设置

    textArea.textProperty().addListener(new ChangeListener<String>() {

      //设定textArea中的文本监听器

      //当用户修改文本超过20个字符的时候，上传新的文本信息。

      @Override

      public void changed(ObservableValue<? extends String> observable,String oldValue,String newValue){

        if(Math.abs(newValue.length() - textLength) > 20){

          uploadFile();

          textLength = newValue.length();

        }

      }

    });

  }

  private void setInitOptions(Stage primaryStage){

    //函数setInitOptions

    //功能:为窗体设定初始参数

    primaryStage.setTitle("Cloud Note");

    primaryStage.getIcons().add(new Image("./sources/icon/white.jpg"));//设置图标

    primaryStage.setOpacity(0.95d);//设置透明度

    //窗口出现时的位置，坐标为左上角的位置

  }

  private void setLoginWindowOptions(Stage primaryStage){

    //函数：setLoginWindowOptions

    //功能：设定登录窗口的参数

    //设定窗口出现时距离屏幕左上角的距离

    primaryStage.setX(100);

    primaryStage.setY(100);

    //设定窗口的长 和 宽

    primaryStage.setWidth(468.0d);

    primaryStage.setHeight(350.0d);

    //设定窗口具体的组件配置

    primaryStage.setScene(this.loginScene());

    //设置窗口是否可以调整大小

    primaryStage.setResizable(false);

    //设定登录窗口的登录功能

    this.setLoginMathod();

  }

  private void setLoginMathod(){

    loginButton.setOnAction(new EventHandler<ActionEvent>() {

      //监听这个组件，当他被点击的时候，启用下面这些代码

      @Override

      public void handle(ActionEvent event){

        //获取账号/密码栏中的字符串

        String password = passwordTF.getText();

        String account = accountTF.getText();

        try{

          PrintWriter pw = getPrintWriter();

          //告诉服务器，客户端要干什么

          pw.println("login");

          pw.flush();

          //把用户输入的账户和密码发送给服务器

          pw.println(account);

          pw.flush();

          pw.println(password);

          pw.flush();

          //从服务端那边接收消息，登陆成功与否

          BufferedReader br = getBufferedReader();

          String result = br.readLine();

          System.out.println("login "+result);

          if(result.equals("fail") || result.equals("")){//登陆失败时的处理

            //登陆失败，弹窗提示

            alertWindow("warning", "Login fail,Please try again");

          }

          else if(result.equals("success")){

            //登陆成功以后，进入主界面

            setEditorWindowOptions(getMainWindow());

          }

        }

        catch(Exception e){

          e.printStackTrace();

          alertWindow("warning", "Login fail,Please try again");

        }

      }

    });

    registerButton.setOnAction(new EventHandler<ActionEvent>() {

      @Override

      public void handle(ActionEvent event){

        try{

          PrintWriter pw = getPrintWriter();

          BufferedReader br = getBufferedReader();

          pw.println("register");

          pw.flush();

          pw.println(accountTF.getText());

          pw.flush();

          pw.println(passwordTF.getText());

          pw.flush();

          String result = br.readLine();

          switch (result){

            case "exist":{

              alertWindow("Register fail", "This account has been existed!\nIf you forget the password,please connect Gstalker");

              break;

            }

            case "success":{

              infomationWindow("Congratulations", "Register success!\nPlease press the \"Login\" button");

              break;

            }

            case "fail":{

              alertWindow("Connection Error", "Cannot connect the server!");

              break;

            }

            default:{

              System.out.println("regisiterButton:UNDEFINED");

            }

          }

        }

        catch(Exception e){

          e.printStackTrace();

        }

      }

    });

  }

  private void alertWindow(String title,String message){

    //弹出一个警告窗口

    Alert alert = new Alert(AlertType.ERROR);//设定警告图标类型

    alert.titleProperty().set(title);

    alert.headerTextProperty().set(message);

    alert.showAndWait();

  }

  private void infomationWindow(String title,String message){

    //弹出一个警告窗口

    Alert alert = new Alert(AlertType.INFORMATION);//设定警告图标类型

    alert.titleProperty().set(title);

    alert.headerTextProperty().set(message);

    alert.showAndWait();

  }

  private Scene loginScene(){

    //函数：loginScene

    //功能：返回登录窗口的场景

    return new Scene(this.loginGUI());

  }

  private BorderPane loginGUI(){

    //函数 loginGUI

    //功能 返回登陆窗口的BorderPane

    BorderPane BP = new BorderPane();

    //使用窗格布局来布置密码框和账号框

    //位置：中部

    GridPane loginPane = new GridPane();

    //标签：Account和Password

    Label accountLable = new Label("Account");

    Label passwordLabel = new Label("Password");

    //账号栏和密码栏

    accountTF = new TextField();

    passwordTF = new PasswordField();

    //向窗格布局中添加组件

    loginPane.add(accountLable,0,0);

    loginPane.add(passwordLabel,0,1);

    loginPane.add(accountTF,1,0);

    loginPane.add(passwordTF,1,1);

    //设置对齐方式

    loginPane.setAlignment(Pos.CENTER);

    //网格布局模式中，各个网格之间的间隙，垂直和水平间隙

    loginPane.setHgap(10.0d);

    loginPane.setVgap(10.0d);

    //把这个窗格布局放在窗口的中间位置

    BP.setCenter(loginPane);

    //使用锚定布局来拉伸登录按键

    //位置：下方

    AnchorPane loginButtonAP = new AnchorPane();

    loginButton = new Button("Login");

    //设置该组件在AnchorPane布局中距离某个方向的像素点数量

    AnchorPane.setLeftAnchor(loginButton,284.0d);

    AnchorPane.setRightAnchor(loginButton,100.0d);

    AnchorPane.setBottomAnchor(loginButton,30.0d);

    loginButtonAP.getChildren().add(loginButton);

    registerButton = new Button("Register");

    AnchorPane.setLeftAnchor(registerButton,100.0d);

    AnchorPane.setRightAnchor(registerButton,284.0d);

    AnchorPane.setBottomAnchor(registerButton,30.0d);

    loginButtonAP.getChildren().add(registerButton);

    BP.setBottom(loginButtonAP);

    //使用锚定布局填充上方

    AnchorPane topFiller = new AnchorPane();

    topFiller.setPrefHeight(150.0d);

    BP.setTop(topFiller);

    return BP;

  }

  private Scene editorScene(){

    //函数 editorScene

    //功能：返回文本编辑器的Scene

    return new Scene(this.initGUI());

  }

  private AnchorPane initGUI(){

    //函数 initGUI

    //功能 返回文本编辑器的AnchorPane

    AnchorPane noterPane = new AnchorPane();

    //设置上方菜单栏

    noterPane.getChildren().add(this.initMenu());

    //添加文本编辑区域

    noterPane.getChildren().add(this.initTextArea());

    return noterPane;

  }

  private MenuBar initMenu(){

    //函数：initMenu

    //功能：返回设置好的菜单栏对象

    MenuBar menuBar = new MenuBar();

    //菜单：文件操作

    Menu fileOpt = new Menu("File");

    MenuItem openFile = new MenuItem("open");

    openFile.setOnAction(new EventHandler<ActionEvent>() {

      @Override

      public void handle(ActionEvent event){

        openFile();

      }

    });

    openFile.setAccelerator(KeyCombination.valueOf("CTRL+O"));

    MenuItem newFile = new MenuItem("new");

    newFile.setOnAction(new EventHandler<ActionEvent>() {

      @Override

      public void handle(ActionEvent event){

        createNewFile();

      }

    });

    newFile.setAccelerator(KeyCombination.valueOf("CTRL+N"));

    MenuItem saveFile = new MenuItem("save");

    saveFile.setOnAction(new EventHandler<ActionEvent>() {

      @Override

      public void handle(ActionEvent event){

        uploadFile();

      }

    });

    //设置快捷键

    saveFile.setAccelerator(KeyCombination.valueOf("CTRL+S"));

    //把组件添加到Menu中

    fileOpt.getItems().addAll(newFile,openFile,saveFile);

    //菜单：编辑操作

    Menu editOpt = new Menu("Edit");

    MenuItem copyEdit = new MenuItem("Copy");

    copyEdit.setOnAction(new EventHandler<ActionEvent>() {

      @Override

      public void handle(ActionEvent event){

        textArea.copy();

      }

    });

    copyEdit.setAccelerator(KeyCombination.valueOf("CTRL+C"));

    MenuItem pasteEdit = new MenuItem("Paste");

    pasteEdit.setOnAction(new EventHandler<ActionEvent>() {

      @Override

      public void handle(ActionEvent event){

        textArea.paste();

      }

    });

    pasteEdit.setAccelerator(KeyCombination.valueOf("CTRL+V"));

    MenuItem cutEdit = new MenuItem("cut");

    cutEdit.setOnAction(new EventHandler<ActionEvent>() {

      @Override

      public void handle(ActionEvent event){

        textArea.cut();

      }

    });

    cutEdit.setAccelerator(KeyCombination.valueOf("CTRL+X"));

    MenuItem selectAllEdit = new MenuItem("Select ALL");

    selectAllEdit.setOnAction(new EventHandler<ActionEvent>() {

      @Override

      public void handle(ActionEvent event){

        textArea.selectAll();

      }

    });

    selectAllEdit.setAccelerator(KeyCombination.valueOf("CTRL+A"));

    editOpt.getItems().addAll(

      copyEdit,

      pasteEdit,

      cutEdit,

      selectAllEdit

    );

    //菜单：视觉样式

    Menu viewOpt = new Menu("View");

    //菜单：帮助

    Menu helpOpt = new Menu("Help");

    MenuItem authorHelp = new MenuItem("Author's Blog");

    helpOpt.getItems().addAll(authorHelp);

    //打开作者博客

    authorHelp.setOnAction(new EventHandler<ActionEvent>(){

      @Override

      public void handle(ActionEvent event) {

        HostServices host = getHostServices();

        host.showDocument("http://139.155.83.108/");

      }

    });

    //菜单栏:添加所有控件

    menuBar.getMenus().addAll(fileOpt,editOpt,viewOpt,helpOpt);

    //设置菜单栏的位置

    AnchorPane.setLeftAnchor(menuBar, 0.0d);

    AnchorPane.setRightAnchor(menuBar, 0.0d);

    AnchorPane.setTopAnchor(menuBar, 0.0d);

    AnchorPane.setBottomAnchor(menuBar, 20.0d);

    return menuBar;

  }

  private TextArea initTextArea(){

    //函数：initTextArea

    //功能：返回设定好的TextArea

    textArea = new TextArea();

    //设置提示

    textArea.setPromptText("Write down your knowladge here ~");

    //是否默认聚焦在该部件上:否

    textArea.setFocusTraversable(false);

    //文本自动换行

    textArea.setWrapText(true);

    //去除边框默认样式

    textArea.setStyle(

      "-fx-background-insets: 0;"+

      "-fx-focus-color: transparent;"+

      "-fx-padding: 0;"

    );

    //左右自动拉伸

    AnchorPane.setLeftAnchor(textArea, 0.0d);

    AnchorPane.setRightAnchor(textArea,0.0d);

    //menuBar采用默认高度20.0像素点

    AnchorPane.setTopAnchor(textArea, 20.0d);

    //底部提示区长度20.0个像素点

    AnchorPane.setBottomAnchor(textArea, 20.0d);

    return textArea;

  }

  private void openFile(){

    PrintWriter pw = getPrintWriter();

    BufferedReader br = getBufferedReader();

    uploadFile();

    try{

      pw.println("OpenFile");

      pw.flush();

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

      List<String> files = new ArrayList<>();

      System.out.println("Available:");

      for(int i = 0; i < totalFileCount ; ++i){

        files.add(br.readLine());

        System.out.println("    "+files.get(i));

      }

      openFileWindow(files);

    }

    catch(Exception e){

      e.printStackTrace();

    }

  }

  private void openFileWindow(List<String> files){

    Stage openFileStage = new Stage();

    System.out.println("file uploaded!");

    openFileStage.setScene(setOpenFileWindow(openFileStage,files));

    openFileStage.setResizable(false);

    openFileStage.setX(100.0d);

    openFileStage.setY(100.0d);

    openFileStage.initOwner(getMainWindow());

    openFileStage.initModality(Modality.WINDOW\_MODAL);

    openFileStage.setTitle("Open File");

    openFileStage.setOpacity(0.90d);

    openFileStage.show();

  }

  private Scene setOpenFileWindow(Stage stage,List<String> files){

    FlowPane fp = new FlowPane();

    fp.setPrefHeight(0.0d);

    fp.setOrientation(Orientation.VERTICAL);

    fp.setVgap(10.0d);

    fp.setAlignment(Pos.CENTER);

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

      Button file = new Button(files.get(i));

      fp.getChildren().add(file);

      file.setOnAction(new EventHandler<ActionEvent>() {

        @Override

        public void handle(ActionEvent event){

          try{

            PrintWriter pw = getPrintWriter();

            BufferedReader br = getBufferedReader();

            pw.println(file.getText());

            pw.flush();

            theEditingFileName = file.getText();

            String file = readAllLine(br);

            System.out.println("Open file "+theEditingFileName);

            pw.println("success!");

            pw.flush();

            textArea.setText(file);

            stage.close();

          }

          catch(Exception e){

            e.printStackTrace();

          }

        }

      });

    }

    return new Scene(fp);

  }

  private String readAllLine(BufferedReader messageReader){

    //全文读取一个文件并且以String形式返回

    String result = "";

    try{

      String temp = messageReader.readLine();

      while(!temp.equals("0xbadbeef")){//文件结束

        result += temp+'\n';

        temp = messageReader.readLine();

      }

    }

    catch(Exception e){

      e.printStackTrace();

    }

    return result;

  }

}