//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);

is = s.getInputStream();

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

} catch (Exception ex) {

ex.printStackTrace();

}

}

}

public class login extends ancestor implements ActionListener, FocusListener {

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());

realPassword = br.readLine();

} 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) {

}

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

}

}

}

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);

ps.println("register");

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

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

is = s.getInputStream();

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

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();

if (jMenuItem.getText().compareTo("新建笔记") == 0) {

new newNote(title);

} else if (jMenuItem.getText().compareTo("保存笔记") == 0) {

new saveNote(title, this);

} else if (jMenuItem.getText().compareTo("打开笔记") == 0) {

new openNote(title);

} else if (jMenuItem.getText().compareTo("删除笔记") == 0) {

new deleteNote(title);

} else if (jMenuItem.getText().compareTo("查询笔记") == 0) {

new inquireNote(title);

} else if (jMenuItem.getText().compareTo("下载笔记") == 0) {

new downloadNote(title);

}else if(jMenuItem.getText().compareTo("退出")==0){

ps.println("exit");

ps.println(title);

System.exit(0);

}

}

}

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);

}

}

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");

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) {

ps.println("yes");

} else {

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;

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

private 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);

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);

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);

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 static void main(String[] args) {

client c = new client();

login l = c.new login();

}

}

//client.ConstVarible.java

package code;

import java.awt.Font;

public class ConstVarible {

public static final String address\_home\_wifi="192.168.0.104";

public static final int port=9999;

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;

}

//server.ConstVarible.java

package code;

import java.awt.Font;

public class ConstVarible {

public static final String address\_home\_wifi="192.168.0.104";

public static final int port=9999;

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

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;

}

//server.fileOperation.java

package code;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileReader;

import java.io.PrintStream;

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) {

if (pps.getProperty(str) == null) {

return false;

} else {

return true;

}

}

public static boolean checkNoteName(String str) {

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();

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<>();

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

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

data.add(row);

}

return data;

}

}

//server.java

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<>();

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.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() {

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);

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;

public PrintStream ps = null;

public String account = null;

private ArrayList<String> arrayList = null;

public singleClient(Socket s) {

this.s = s;

try {

br = new BufferedReader(new InputStreamReader(s.getInputStream()));

ps = new PrintStream(s.getOutputStream());

} catch (Exception ex) {

ex.printStackTrace();

}

}

public void run() {

try {

while (true) {

String str = new String();

try {

str = br.readLine();

} catch (Exception ex) {

linkedHashSet.remove(accountString);

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) {

;

} 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));

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;

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);

}

}

} catch (Exception ex) {

ex.printStackTrace();

} finally {

try {

br.close();

} catch (Exception ex) {

}

}

}

}

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();

}

}