//client.java

package client;

import java.io.BufferedReader;

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.event.\*;

import java.awt.\*;

import java.awt.Font;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JMenu;

import javax.swing.JMenuBar;

import javax.swing.JMenuItem;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JPasswordField;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

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 account = null;

public ancestor(String str) {

super(str);

account = str;

this.setTitle(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(ConstVarible.startFlag + "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(ConstVarible.startFlag + "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, Runnable {

private JTextArea taMsg = new JTextArea("以下是聊天记录\n");

private JTextField tfMsg = new JTextField();

private JScrollPane newJScrollPane = new JScrollPane(taMsg);

private JMenuBar jMenuBar = new JMenuBar();

private JMenu friends = new JMenu("好友");

private JMenuItem makefriends = new JMenuItem("添加好友");

private JMenuItem listfriends = new JMenuItem("好友列表");

private JMenuItem chatfriends = new JMenuItem("私聊好友");

private JMenuItem refreshfriends = new JMenuItem("刷新列表");

public mainInterface(String str) {

super(str);

account = str;

BorderLayout borderLayout = new BorderLayout();

jPanel.setLayout(borderLayout);

jPanel.add(newJScrollPane, BorderLayout.CENTER);

jPanel.add(tfMsg, BorderLayout.SOUTH);

jPanel.add(jMenuBar, BorderLayout.NORTH);

jMenuBar.add(friends);

friends.add(makefriends);

friends.add(listfriends);

friends.add(chatfriends);

friends.add(refreshfriends);

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

taMsg.setFont(font);

tfMsg.setFont(font);

taMsg.setEditable(false);

tfMsg.setBackground(Color.yellow);

tfMsg.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent ae) {

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

try {

ps.println(account + "说:" + tfMsg.getText());

tfMsg.setText("");

} catch (Exception ex) {

ex.printStackTrace();

}

}

});

makefriends.addActionListener(this);

listfriends.addActionListener(this);

chatfriends.addActionListener(this);

refreshfriends.addActionListener(this);

this.add(jPanel);

this.setSize(700, 500);

this.setLocationRelativeTo(null);

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

new Thread(this).start();

}

public void run() {

try {

while (true) {

String str = br.readLine();

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

String string=br.readLine();

chatprivate cp= new chatprivate(account, string);

System.out.println("hhh");

}

taMsg.append(str + "\n");

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

public void actionPerformed(ActionEvent ae) {

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

if (jMenuItem.getText().compareTo("添加好友") == 0) {

new makefriend(account);

} else if (jMenuItem.getText().compareTo("私聊好友") == 0) {

new chatfriend(account);

} else if (jMenuItem.getText().compareTo("刷新列表") == 0) {

new refreshfriend(account);

} else if (jMenuItem.getText().compareTo("好友列表") == 0) {

new listfriend(account);

}

}

}

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);

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);

jScrollPane = new JScrollPane(jTable);

jPanel.add(jScrollPane);

this.add(jPanel);

this.setSize(700, 500);

this.setLocationRelativeTo(null);

}

}

public class chatfriend extends ancestor implements ListSelectionListener {

protected JTable jTable = null;

protected JScrollPane jScrollPane = null;

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

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

public chatfriend(String account) {

super(account);

this.setTitle("选择私聊好友");

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

ps.println(account);

columnName.add("在线好友");

while (true) {

try {

String string = br.readLine();

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

break;

}

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

row.add(string);

data.add(row);

} catch (Exception ex) {

ex.printStackTrace();

}

}

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.setFillsViewportHeight(true);

jTable.getSelectionModel().addListSelectionListener(this);

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() == true) {

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

ps.println(account);

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

new chatprivate(account,data.get(row).get(0));

}

}

}

public class chatprivate extends mainInterface{

public chatprivate(String account,String friendAccount){

super(account);

this.setTitle(account+"与"+friendAccount+"的私聊");

}

}

public class refreshfriend extends ancestor {

public refreshfriend(String account) {

super("刷新");

this.setVisible(false);

}

}

public static void main(String[] args) {

new client().new login();

}

}

//client.ConstVarible.java

package client;

import java.awt.Font;

public class ConstVarible {

public static final String address\_home\_wifi="192.168.1.2";

public static final int port=9999;

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

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

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 server;

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 dataPath="C:\\Users\\mashed potato\\Desktop\\vscode\\.vscode\\chatRoom\_server\\src\\data\\";

public static final String dataFile="C:\\Users\\mashed potato\\Desktop\\vscode\\.vscode\\chatRoom\_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;

}

//fileOperation.java

package server;

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.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>> getfriend(String account) {

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

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) {

return 0;

} else {

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

try {

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

file.createNewFile();

}

if(countWordInFile(friendAccount, file)==1){

return 1;

}

FileWriter fw=new FileWriter(file,true);

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

fw.close();

storeFriendsAccount(friendAccount, account);

return 2;

} catch (Exception ex) {

ex.printStackTrace();

}

return 3;

}

}

}

//server.java

package server;

import java.awt.event.\*;

import java.awt.Color;

import java.awt.Dimension;

import java.awt.BorderLayout;

import java.awt.Font;

import java.io.BufferedReader;

import java.io.InputStreamReader;

import java.io.ObjectOutputStream;

import java.io.PrintStream;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.SocketException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.LinkedHashSet;

import java.util.Vector;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JOptionPane;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTable;

import javax.swing.JTextArea;

import javax.swing.JTextField;

import javax.swing.event.ListSelectionEvent;

import javax.swing.event.ListSelectionListener;

import javax.swing.table.DefaultTableCellRenderer;

public class server {

public LinkedHashSet<String> linkedHashSet = new LinkedHashSet<>();

public String accountString = new String();

public ArrayList clients = new ArrayList<>();

public void sendMessage(String msg) {

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

singleClient sc = (singleClient) clients.get(i);

if (sc.ID.compareTo("public") == 0) {

sc.ps.println(msg);

}

}

}

public void sendMessageToSomeone(String account, String message) {

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

singleClient sc = (singleClient) clients.get(i);

if (sc.account.compareTo(account) == 0 && sc.ID.compareTo("private") == 0) {

sc.ps.println(message);

}

}

}

public class ancestor extends JFrame {

protected JPanel jPanel = new JPanel();

protected ServerSocket ss = null;

protected Socket s = null;

protected String title = null;

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 sendButton = new JButton("系统消息");

private JButton force = new JButton("管理当前连接的用户");

public mainInterface() {

super("服务器");

jPanel.add(sendButton);

jPanel.add(force);

this.add(jPanel);

this.setSize(400, 250);

jPanel.setLayout(null);

sendButton.setBounds(135, 30, 100, 50);

force.setBounds(90, 120, 200, 60);

this.setLocationRelativeTo(null);

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

sendButton.addActionListener(this);

force.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();

clients.add(sc);

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

public void actionPerformed(ActionEvent ae) {

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

if (jButton.getText().compareTo("系统消息") == 0) {

new message(s);

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

new force();

}

}

}

public class singleClient implements Runnable {

private Socket s = null;

private BufferedReader br = null;

public PrintStream ps = null;

public String account = "account";

protected String ID = "public";

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 {

ID = br.readLine();

account=br.readLine();

while (true) {

String str = new String();

try {

str = br.readLine();

} catch (SocketException ex) {

if (ex.getMessage().compareTo("Connection reset") == 0) {

linkedHashSet.remove(accountString);

sendMessage("系统消息：" + accountString + "下线");

break;

}

}

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

accountString = account;

String password = fileOperation.getPasswordByAccount(account);

ps.println(password);

linkedHashSet.add(account);

sendMessage("系统消息：" + account + "上线");

}

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

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);

sendMessage("系统消息：" + account + "上线");

}

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

try {

while (true) {

String string = br.readLine();

sendMessage(string);

br.readLine();

br.readLine();

br.readLine();

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

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

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) {

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

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

}

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

Vector<Vector<String>> friends = fileOperation.getfriend(account);

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

String string = friends.get(i).get(0);

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

while (iterator.hasNext()) {

if (string.compareTo(iterator.next()) == 0) {

ps.println(string);

}

}

}

ps.println(ConstVarible.endFlag);

}

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

String friendAccount = br.readLine();

sendMessage("系统消息："+account+"向"+friendAccount+"发起私聊");

try {

while (true) {

String string = br.readLine();

if (string == null)

break;

sendMessageToSomeone(friendAccount, string);

sendMessageToSomeone(account, string);

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

}

} catch (Exception ex) {

ex.printStackTrace();

} finally {

try {

br.close();

} catch (Exception ex) {

ex.printStackTrace();

}

}

}

}

public class message extends ancestor {

protected JTextArea taMsg = new JTextArea("以下是系统消息\n");

protected JTextField tfMsg = new JTextField();

protected JScrollPane newJScrollPane = new JScrollPane(taMsg);

protected ActionListener sendListener = null;

public message(Socket s) {

super("系统消息");

BorderLayout borderLayout = new BorderLayout();

jPanel.setLayout(borderLayout);

jPanel.add(newJScrollPane, BorderLayout.CENTER);

jPanel.add(tfMsg, BorderLayout.SOUTH);

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

taMsg.setFont(font);

tfMsg.setFont(font);

taMsg.setEditable(false);

tfMsg.setBackground(Color.yellow);

sendListener = new ActionListener() {

public void actionPerformed(ActionEvent ae) {

try {

sendMessage("系统消息：" + tfMsg.getText());

taMsg.append("系统消息：" + tfMsg.getText() + "\n");

tfMsg.setText("");

} catch (Exception ex) {

ex.printStackTrace();

}

}

};

tfMsg.addActionListener(sendListener);

this.add(jPanel);

this.setSize(700, 500);

this.setLocationRelativeTo(null);

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

}

}

public class force extends ancestor implements ListSelectionListener {

protected JTable jTable = null;

protected JScrollPane jScrollPane = null;

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

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

public force() {

super("强制下线当前连接的用户");

columnName.add("当前在线用户");

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

while (iterator.hasNext()) {

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

row.add(iterator.next());

data.add(row);

}

BorderLayout borderLayout = new BorderLayout();

jPanel.setLayout(borderLayout);

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);

jTable.getSelectionModel().addListSelectionListener(this);

jScrollPane = new JScrollPane(jTable);

jPanel.add(jScrollPane, BorderLayout.CENTER);

this.add(jPanel);

this.setSize(700, 500);

this.setLocationRelativeTo(null);

}

public void valueChanged(ListSelectionEvent le) {

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

int row = jTable.getSelectedRow();

int option = JOptionPane.showConfirmDialog(this, "是否强制下线该用户？");

if (option == JOptionPane.YES\_OPTION) {

try {

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

singleClient sc = (singleClient) clients.get(i);

if (sc.account.compareTo(data.get(row).get(0)) == 0) {

sc.ps.println(ConstVarible.startFlag + "exit");

}

}

} catch (Exception ex) {

ex.printStackTrace();

}

}

}

}

}

public static void main(String[] args) {

new server().new mainInterface();

}

}