package chat;

import java.awt.event.ActionListener;

import java.awt.event.ItemEvent;

import java.awt.event.ItemListener;

import java.awt.event.KeyEvent;

import java.awt.event.KeyListener;

import java.awt.event.WindowAdapter;

import java.awt.event.WindowEvent;

import java.io.\*;

import java.net.Socket;

import java.net.UnknownHostException;

import java.util.ArrayList;

import java.util.Vector;

import java.awt.event.ActionEvent;

import javax.swing.\*;

import java.awt.\*;

public class client extends JFrame implements KeyListener, Runnable{

public static int mark=0;

private JLabel lb1,lb2,lb3,lb4,lb5,lb6;

private JButton go,change,groupchat,send1,send2,add;

private JPanel jp1, jp2,jp3,jp4,jp5,jp6;

private Socket socket = null;

private DataInputStream inputStream = null;

private DataOutputStream outputStream = null;

public String state,name;

private JTextField id, cypher,name1,msg1,msg2,friend;

public static Vector<String> list = new Vector<String>();

public static ArrayList<client> friendlist = new ArrayList<>();

private JScrollPane sp1,sp2;

private JTextArea jt1,jt2;

private static client view;

JFrame jf=new JFrame("chat");

JFrame jf1=new JFrame("singlechat");

JFrame jf2=new JFrame("groupchat");

private void initview() {

lb1=new JLabel("账号:");

id=new JTextField(8);

id.addKeyListener(this);

lb2=new JLabel("密码:");

cypher =new JTextField(8);

cypher.addKeyListener(this);

go=new JButton("登录");

go.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

onview();

//System.out.println("按钮被点击");

}});

jp1 = new JPanel();

jp1.add(lb1);

jp1.add(id);

jp1.add(lb2);

jp1.add(cypher);

jp1.add(go);

jp1.setLayout(new FlowLayout(FlowLayout.CENTER));

add(jp1, BorderLayout.CENTER);

setTitle("聊天室");

setSize(500, 500);

setLocation(450, 150);

setVisible(true);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

public client() {

initview();

state="在线";

name="用户1";

try {

socket = new Socket("localhost", 9090);//连接本地服务器

outputStream = new DataOutputStream(socket.getOutputStream());

outputStream.writeUTF(name );

outputStream.flush();

} catch (UnknownHostException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

}

}

private void onview() {

setVisible(false);

jf.setSize(250, 250);

jf.setLocationRelativeTo(null);

jf.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

lb3=new JLabel("昵称");

name1=new JTextField(8);

name1.addKeyListener(this);

name1.setText(name);

change=new JButton("change");

change.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

name=name1.getText();

}});

list.add(" ");

lb4=new JLabel("好友列表:");

jp2=new JPanel();

JComboBox<String> comboBox = new JComboBox<String>( list);

comboBox.addItemListener((ItemListener) new ItemListener() {

public void itemStateChanged(ItemEvent e) {

// 只处理选中的状态

if (e.getStateChange() == ItemEvent.SELECTED&& comboBox.getSelectedIndex()!=0) {

mark=0;

int index = comboBox.getSelectedIndex();

String label = comboBox.getItemAt(index).toString();

String[] m = label.split(" ");

singleview(m[0]);

//System.out.println("选中: " + comboBox.getSelectedIndex() + " = " + comboBox.getSelectedItem());

}

}

});

comboBox.setSelectedIndex(0);

friend=new JTextField(8);

friend.addKeyListener(this);

add=new JButton("添加");

add.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

String mn;

/\*if(find(friend.getText())==null)

mn=friend.getText()+" 下线";

else mn=friend.getText()+" "+find(friend.getText());\*/

mn=friend.getText()+" "+find(friend.getText());

list.add(mn);

System.out.println("添加好友");

comboBox.setModel(new DefaultComboBoxModel<String>(list));

}});

groupchat=new JButton("群聊");

groupchat.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

mark=1;

groupview();

}});

jp2.add(lb3);

jp2.add(name1);

jp2.add(change);

jp2.add(lb4);

jp2.add(comboBox);

jp2.add(groupchat);

jp2.add(friend);

jp2.add(add);

jf.setContentPane(jp2);

jf.setVisible(true);

jf.addWindowListener(new WindowAdapter() { //窗口关闭后断开连接

public void windowClosing(WindowEvent e) {

state="下线";

// System.exit(0);

}

});

}

private static String find(String a) {

for(client b:service.clientViews) {

//System.out.println(b.name);

if(b.name.equals(a))

return b.state;

}

return "未知";

}

private void singleview(String a) {

//jf.setVisible(false);

lb5=new JLabel(a);

jf1.setSize(250, 250);

jf1.setLocationRelativeTo(null);

jf1.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

jp3=new JPanel();

jp4=new JPanel();

jp3.add(lb5);

jt1= new JTextArea(20, 20);

jt1.setEditable(false);

sp1 = new JScrollPane(jt1);

msg1=new JTextField(15);

msg1.addKeyListener(this);

//sendmsg a1=new sendmsg(msg1,jt1);

send1=new JButton("发送");

send1.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (e.getSource() == send1) {

sendmsg1(a);

}

}});

jp4.add(msg1);

jp4.add(send1);

jp4.setLayout(new FlowLayout(FlowLayout.CENTER));

jf1.add(jp3, BorderLayout.NORTH);

jf1.add(sp1, BorderLayout.CENTER);

jf1.add(jp4, BorderLayout.SOUTH);

jf1.addWindowListener(new WindowAdapter() { //窗口关闭后断开连接

public void windowClosing(WindowEvent e) {

try {

jf.setVisible(true);

jt1.setText("");

if (socket != null)

socket.close();

if (inputStream!= null)

inputStream.close();

if (outputStream != null)

outputStream.close();

} catch (IOException e1) {

e1.printStackTrace();

}

}

});

jf1.setVisible(true);

}

private void groupview() {

//jf.setVisible(false);

lb6=new JLabel("groupchat");

jf2.setSize(250, 250);

jf2.setLocationRelativeTo(null);

jf2.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);

jp5=new JPanel();

jp6=new JPanel();

jp5.add(lb6);

jt2= new JTextArea(20, 20);

jt2.setEditable(false);

sp2 = new JScrollPane(jt2);

msg2=new JTextField(15);

msg2.addKeyListener(this);

//sendmsg a2=new sendmsg(msg2,jt2);

send2=new JButton("发送");

send2.addActionListener(new ActionListener() {

public void actionPerformed(ActionEvent e) {

if (e.getSource() == send2) {

sendmsg2();

}

}});

jp6.add(msg2);

jp6.add(send2);

jp6.setLayout(new FlowLayout(FlowLayout.CENTER));

jf2.add(jp5, BorderLayout.NORTH);

jf2.add(sp2, BorderLayout.CENTER);

jf2.add(jp6, BorderLayout.SOUTH);

jf2.addWindowListener(new WindowAdapter() { //窗口关闭后断开连接

public void windowClosing(WindowEvent e) {

try {

jt2.setText("");

if (socket != null)

socket.close();

if (inputStream!= null)

inputStream.close();

if (outputStream != null)

outputStream.close();

} catch (IOException e1) {

e1.printStackTrace();

}

}

});

jf2.setVisible(true);

// run(jt2);

}

private void sendmsg2() {

try {

String s = msg2.getText();

if (!s.equals("")) { //发送数据

msg2.setText("");

jt2.append(name+"(我)"+":\r\n" + s + "\r\n");

outputStream = new DataOutputStream(socket.getOutputStream());

outputStream.writeUTF(name + "#" + s+"#"+"1");

outputStream.flush();

}

} catch (IOException e) {

e.printStackTrace();

}

}

private void sendmsg1(String a) {

try {

String s = msg1.getText();

if (!s.equals("")) { //发送数据

msg1.setText("");

jt1.append(name+"(我)"+":\r\n" + s + "\r\n");

outputStream = new DataOutputStream(socket.getOutputStream());

outputStream.writeUTF(name + "#" + s+"#"+"0"+"#"+a);

outputStream.flush();

}

} catch (IOException e) {

e.printStackTrace();

}

}

public void keyTyped(KeyEvent arg0) {

}

public void keyPressed(KeyEvent arg0) {

if (arg0.getKeyCode() == KeyEvent.VK\_ENTER) {

/\*sendmsg2();

sendmsg1();\*/

}

}

@Override

public void keyReleased(KeyEvent arg0) {

}

public void run() {

try {

inputStream = new DataInputStream(socket.getInputStream());

while (true) {

//String m=inputStream.readUTF()+"接受成功";

String[] s = inputStream.readUTF().split("#");

if(mark==1)

jt2.append(s[0] + ":\r\n" + s[1] + "\r\n");

else

jt1.append(s[0] + ":\r\n" + s[1] + "\r\n");

}}

catch (Exception e) {

//jf2.setVisible(false);

System.out.println("无法正常接收");

}

}

public DataInputStream getInputStream() {

return inputStream;

}

public DataOutputStream getOutputStream() {

return outputStream;

}

public static void main(String[] args) {

// TODO 自动生成的方法存根

view =new client();

/\*service.clientViews.add(view);

for(client b:service.clientViews) {

System.out.println(b.name+"3333");

}\*/

Thread thread = new Thread(view);

thread.start();

}

/\*public class sendmsg implements Runnable{

private JTextField a;

private JTextArea b;

public sendmsg(JTextField m, JTextArea n) {

a=new JTextField();

b=new JTextArea();

this.a=m;

this.b=n;

}

public void run() {

// TODO 自动生成的方法存根

while(true) {

String s = a.getText();

if (!s.equals("")) { //发送数据

a.setText("");

b.append(name+"(我)"+":\r\n" + s + "\r\n");

try {

outputStream = new DataOutputStream(socket.getOutputStream());

outputStream.writeUTF(name + "#" + s);

outputStream.flush();

} catch (IOException e1) {

// TODO 自动生成的 catch 块

e1.printStackTrace();

}

}

}

}

}\*/

}

package chat;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.io.\*;

import java.net.ServerSocket;

import java.net.Socket;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.Vector;

import java.util.concurrent.CopyOnWriteArrayList;

import javax.swing.\*;

import java.awt.\*;

public class service extends JFrame implements ActionListener{

private JButton btnOpen, btnStop;

private JLabel label;

private Service service = null;

public static ArrayList<client> clientViews = new ArrayList<>();

private static service view;

public static service getView() {

return view;

}

public static void main(String[] args) {

view = new service();

}

public service() {

initView();

}

private void initView() {

btnOpen = new JButton("start");

btnStop = new JButton("stop");

btnStop.setEnabled(false);

btnOpen.addActionListener(this);

btnStop.addActionListener(this);

label = new JLabel("服务器停止工作");

add(label);

add(btnOpen);

add(btnStop);

setTitle("服务器");

setLayout(new GridLayout(3, 1, 0, 10));

setSize(300, 300);

setLocation(450, 150);

setVisible(true);

setDefaultCloseOperation(EXIT\_ON\_CLOSE);

}

@Override

public void actionPerformed(ActionEvent e) {

if (e.getSource() == btnOpen) {

open();

} else {

stop();

}

}

public void open() { //开启服务器

service = new Service();

Thread thread = new Thread(service);

thread.start();

label.setText("服务器正在运行");

btnOpen.setEnabled(false);

btnStop.setEnabled(true);

}

public void stop() { //关闭服务器

label.setText("服务器已关闭");

btnOpen.setEnabled(true);

btnStop.setEnabled(false);

try {

synchronized (ClientMannager.sockets) { //关闭各个连接

for (ChatSocket socket : ClientMannager.sockets) {

socket.getInputStream().close();

socket.getOutputStream().close();

}

/\* for(ChatSocket socket : ClientMannager.sockets){

ClientMannager.sockets.remove(socket);

} \*/

ClientMannager.sockets.removeAllElements();

}

for (client view : clientViews) {

view.getInputStream().close();

view.getOutputStream().close();

}

service.getServerSocket().close();

} catch (IOException e) {

e.printStackTrace();

}

}

public class ChatSocket implements Runnable{

private Socket socket = null;

private DataInputStream inputStream = null;

private DataOutputStream outputStream = null;

private String id;

public DataInputStream getInputStream() {

return inputStream;

}

public DataOutputStream getOutputStream() {

return outputStream;

}

public ChatSocket(Socket socket,String a) {

this.socket = socket;

this.id=a;

System.out.println(id);

try {

inputStream = new DataInputStream(socket.getInputStream());

outputStream = new DataOutputStream(socket.getOutputStream());

} catch (IOException e) {

e.printStackTrace();

}

}

public void send(String send) { //向客户端发送数据

try {

outputStream.writeUTF(send);

outputStream.flush();

System.out.println(send+"服务器接收到");

} catch (IOException e) {

e.printStackTrace();

}

}

@Override

public void run() { //循环读取客户端发来的数据

while (true) {

try {

// accept = inputStream.readUTF();

String[] s = inputStream.readUTF().split("#");

if(s[2].equals("1"))

{ ClientMannager.sendAll(this, s[0]+"#"+s[1]);}

else {

// System.out.println(s[3]+"00000");

ClientMannager.sendone( s[0]+"#"+s[1],s[3]);

}

} catch (IOException e) {

ClientMannager.sockets.remove(this);

}

}

}

}

public static class ClientMannager {

private ClientMannager() {

}

//public static CopyOnWriteArrayList<ChatSocket> sockets = new CopyOnWriteArrayList<>();

public static Vector<ChatSocket> sockets = new Vector<>();

//public static HashMap<String, Socket> socketList = new HashMap<>();

//向其他客户端发送数据

public static void sendAll(ChatSocket chatSocket, String send) {

for (ChatSocket socket : sockets) {

if (!chatSocket.equals(socket)) {

System.out.println(socket.id);

socket.send(send);

}

}

}

/\* public static void find(String a) {

for (ChatSocket socket : sockets) {

if (socket.id.equals(a)) {

}

}

}\*/

public static void sendone( String send,String a) {

for (ChatSocket socket : sockets) {

if (socket.id.equals(a)) {

System.out.println(a);

socket.send(send);

}

}

}

}

public class Service implements Runnable{

private ServerSocket serverSocket = null;

public ServerSocket getServerSocket() {

return serverSocket;

}

@Override

public void run() {

try {

serverSocket = new ServerSocket(9090);

while (true) { //循地接收客户端的连接

Socket socket = serverSocket.accept();

/\* JOptionPane.showMessageDialog(getView(), "客户端连接端口", "TIP", JOptionPane.INFORMATION\_MESSAGE);\*/

DataInputStream inputStream = new DataInputStream(socket.getInputStream());

String m=inputStream.readUTF();

System.out.println(m);

ChatSocket chatSocket = new ChatSocket(socket,m);

//新客户端连接

ClientMannager.sockets.add(chatSocket); //往客户端管理器里添加客户

Thread thread = new Thread(chatSocket); //启用线程使服务器开始不断接收客户端信息

thread.start();

}

} catch (IOException e) {

e.printStackTrace();

System.out.println("服务器关闭");

}

}

}

}