**Java程序设计课程设计报告**

院 系：

专 业：

学生姓名：

学 号：

指导老师：

2018 年 7 月 9 日

烟台大学计算机与控制工程学院

1. **项目背景**

通讯技术应用渗透到各个领域，在Java本身的有点就可以实现这些功能。对运用Java中的多线程机制、Java网络编程图形用户界面的设计以及Access对账户信息的处理技术使开发一个通讯技术会的更加有优势。在系统的最终目标是实现可以多用户之间的通讯和多用户之间的通讯

1. **需求分析**

## 2.1系统简要说明

使用的工具：NetBeans IDE 7.0.1

平台：Window 10

## 2.2 主要功能图

退

出

注

册

登录

显示主机IP

输入密码

输入用户名

连接

服务器

退出客户端窗口

输入信息

字体颜色选择

私聊复选

客户端窗口

登陆

服务器

在线用户列表

发送信息按钮

接受信息

显示登陆

|  |
| --- |
| 服务器 |

显示用户

踢出用户

警告用户

注册

|  |
| --- |
| 服务器连接 |

Access数据库

保存用户密码

保存用户名称

## 2.3 系统需求分析

系统在登录时能在连接面板能够通过服务器和Access数据库通讯对输入的用户和密码进行判断用户名是否记录在数据库中以及已经登陆与否，对密码判断是否为空，以及和相应的用户名是否一致。对新注册的用户可以通过连接面板进行和数据库通讯，记录在数据库中，还可以打开userlist\_DB数据源直接进行录入。账户

1. 用户管理：可以管理用户的基本信息
2. 数据的处理：对发送接收的数据可以进行处理。
3. 用户登录：实现多用户的登录
4. 对聊天人员的选择和满足字体颜色的变化以及一些基本的退出操作操作处理。
5. 对言论不合法的用户可以给予警告和剔除操作。

**3、****总体设计**

## 3.1 系统体系结构

系统采用了C/S和B/C两种模式，实现了通讯的三层结构，由客户端程序ChatClient\_DB.java、服务器端程序 ChatServer\_DB.java 和数据库程序 ServerTaber\_DB.java ChatClient\_DB.html四个文件组成。

系统体系结构图

警告用户

用户层

用户n

Database服务器

Access数据库

服务器层

用户与Access数据库通讯

用户1

踢除用户

聊天室系统用户信息库

登陆、注册、取消、接收数据、发送数据、选择聊天对象、选择字体颜色、选择私聊

用户与Access数据库通讯

显示登陆用户

连接

Access

用户n

Database服务器

Access数据库

|  |
| --- |
| Yonghuceng  用户层 |

服务器层

## 3.2 功能模块划分

根据系统的需求，实现设计大致可分为以下几个方面：

1. 发送数据，当前用户发送信息给对方
2. 接收数据，接受对方的信息
3. 选择聊天对象（在线人员），选择不同的在线用户进行对话
4. 选择字体颜色。
5. 私聊复选，用户可以选择是否和多个对象聊天。

（6）退出客户端，点击退出按钮客户端窗口会退出。

（7）警告或者剔除用户，处理一些言论不合法的用户。

**4、****详细设计**

## 4.1 系统类图

1

N

1

ConnectDialog

app:ChatClient\_DB DialogPane:JPanel

host : JTextField user : JTextField

cancel : JButton cd：ConnectDialog

ConnectDialog(ChatClient\_DB,Frame,String,boolean )

cancle\_mouseClicked(MouseEvent) ：void register\_mouseClicked(MouseEvent ) ：void

connect\_mouseClicked(MouseEvent )：void

b\_tf(String ,String )： boolean

N

N

N

N

1

ChatClient\_DB

textField1: JTextField send :JButton

exit : JButton userList : JComboBox

mychoice :JComboBoxsl : JCheckBox

ta : JTextArea SC\_ta ：JScrollPane

clientsocket ：Socket

login : LoginClient

app: ChatClient\_DB frame ： Frame

LoginClient

client: ChatClient\_DB

fromServer1 : String

LoginClient(ChatClient\_DB client)

run(): void

ServerTaber\_DB

main(String args[]) : void

init() ： void itemStateChanged(ItemEvent ）：void

actionPerformed(ActionEvent evt) : void sendData(String ) ： boolean

getData() : String

closeSocket() : void

connectHost(String hostName) ： boolean display(String str) : void

YHM(String str) ： void

out\_println(String str) ：void

display1(String str) ：void

1

1

stmt : Statement con : Connection

pstmt : PreparedStatement rs :ResultSet URL : String

ServerTaber\_DB() void createTable(String sql)

insertRec(String sql,String name,String passwd) : void

queryTable(String sql) : String deleteRec(String sql) : void

dispResultSet : String

N

1

ChatServer\_DB

1

1

Connection1

cs ：ChatServer\_DB s ： Socket

dis : DataInputStream

dos : DataOutputStream

username : String talker : String

whisper : boolean

PORT : int user\_list

ss : ServerSocket

users : Vector

connections : Vector

username : String

passwd : String

Connection1(Socket s,ChatServer\_DB cs)

run() ： void

close() ： void

readData() ： String

writeData(String str) ： void

main(String) : void

ChatServer\_DB()

run() : void

serverWriter(String str,boolean w,String S\_user,boolean b\_delete) synchronized void

## 4.2主要模块功能

<1>、客户端功能介绍：在定义一个面板和和各种标签、按钮、等组件后，由一个放置在一个窗体中进行显示出来，由app.init()来进行对容器和窗体进行初始化，再由app.start()进行线程调度来对在线程调度里面执行两个方法首先client.getData()调用 ，从is输出流里面is.read3UTF()读取出来的内容有#&@#的时候LoginClient 内生明的一个 ChatClient\_DB client对象就client.display1(fromServer1)来显示选择的字体颜色并且client.display(fromServer)在数据显示区域显示出读取的数据内容。

没有#&@#的时候就只client.display(fromServer)显示出读取的套接字的信息。

app.init()中带调用ConnectDialog类带有构造函数的一个对象进行注册、登陆、取消功能ConnectDialog内设置的一个窗口，进行不同的操作会触发不同的监听器来引发各个功能模块

<2>、注册功能:当监听器侦听到来自注册按钮的MouseEvent事件时，调用register\_mouseClicked(MouseEvent )进行注册功能在用户名和密码注册符合条件的时候就调用ServerTaber\_DB 类的insertRec(String sql,String name,String passwd)函数把注册信息插入到数据库中当再次启动客户聊天线程的时候就可以对刚进行注册的用户进行号登录功能

<3>、登录功能：监听器侦听到来自登录按钮的MouseEvent事件时，调用connect\_mouseClicked(MouseEvent )对输入的数据进行处理，输入没有非法字符后，则调用connectHost(String)来连接对方主机和设置好的主机通道，连接之后调用sendData(String)向服务器发送数据（对应的密码和用户名）返回成功后

调用getDate() 服务器端数据看对应账户的密码和账户名看是否一致。

<4>、发送数据：客户聊天线程LoginClient类中发送信息到数据流中对象从触发事件的发生，调用sendData(String)向服务器发送数据信息。

<5>、接收数据：客户聊天线程LoginClient类中从字节流中获取数据信息，选择对应的用户名称，用display函数显示在ta中

<6>、选择聊天对象：在actionPerformed(获取触发事件的对象时候，用getSelectedItem() 定义选择用户数组 ，在穷举所有的对象，当符合选择的聊天对象时，显示在在线人员列表。

<7>、选择字体颜色：从客户端数据流中获取带有“#&@#"字符的时候，说名是更改颜色信息，并调用 display1在用户端设置颜色变化。

<8>、警告或者剔除用户:服务器端的serverWriter(String str,boolean w,String S\_user,boolean b\_delete){方法调用对是删除或警告的请求向通过数据流向用客户端发出对应的信息。

## 4.3 系统数据库的设计

数据库设置的一张userlist\_DB表用来存储用户的信息包括用户的密码和用户名。

用户

E-R图

### 4.3.1 数据库表的结构

列名 数据类型 说明 可否为空

Notnull

主键

name

Varchar(50)

Notnull

passwd

Varchar(50)

### 4.2.2 数据库选择及理由

选择Access数据库

其优点是：

1.集成环境、处理多种数据信息。

2.界面友好、易操作。

3.面向对象ACCESS是个面向对象的开发工具，利用面向对象的方式将数据库系统中的各种功能对象化，将数据库管理的各种功能封装在各类对象中。

4.存储方式简单，易于维护管理。

**5、****实施与程序编码**

## 5.1 系统组件图

登录

服务器连接

***. . . .***

客户端

客户端

## 5.2 系统部署图

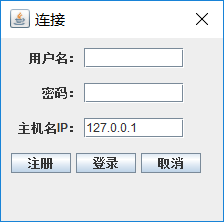
客户端

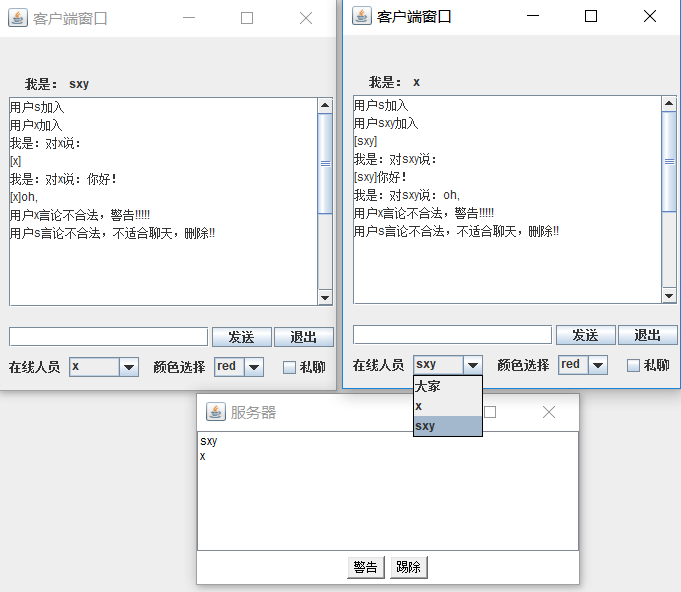
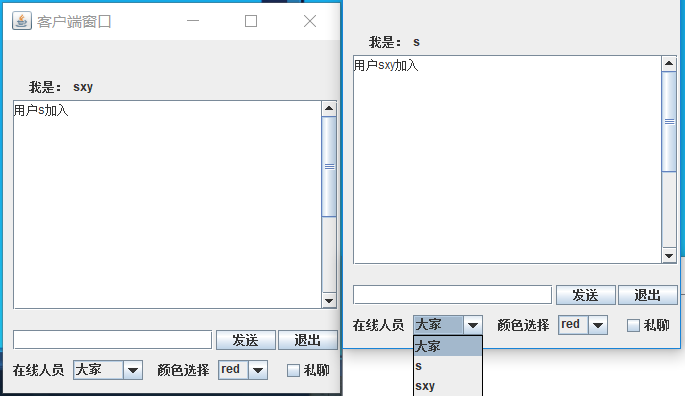
服务器

数据库

**6、****综合测试**

客户端和服务器端运行结果：





**7、****总结**

**8、****参考文献**

**[1]耿祥义、张跃平，《Java程序设计实用教程》第2版：人民邮电出版社 9、****源代码**

ChatClient\_DB.html

<HTML>

<HEAD>

<TITLE> 客户端</title>

</head>

<body>

<applet codebase="."

code="ChatClient\_DB.class"

name="客户端"

width=360

height=350

hspace=0

vspace=0

align=middle

>

</applet>

</body>

</html>

客户端程序（ChatClient\_DB.java）

/////////////////////////////////////////////////////////////////////////////////////////////////////

//客户端程序 //

//多用户聊天室 //

//是Application，也是Applet ，在单机和浏览器上都可运行 //

//ChatClient\_DB.java //

//姓名： //

//20XX 年XX月XX日 //

/////////////////////////////////////////////////////////////////////////////////////////////////

import java.awt.\*;

import java.awt.event.\*;

import java.applet.\*;

import java.util.\*;

import java.io.\*;

import java.net.\*;

import javax.swing.\*;

import javax.swing.JLabel.\*;

import javax.swing.JApplet;

import javax.swing.event.\*;

public class ChatClient\_DB extends JApplet implements ActionListener,ItemListener

{

static int n=0;

JLabel lb1=new JLabel("在线人员");

JLabel lb2=new JLabel("颜色选择");

JLabel lb3=new JLabel("我是：",JLabel.RIGHT);//用户名称

JLabel lb4=new JLabel("用户名称:",JLabel.LEFT);

JTextField textField1=new JTextField(30);//用于数据输入

JButton send=new JButton("发送");

JButton exit=new JButton("退出");

//Image myImage;

JComboBox userList=new JComboBox(); //定义用户列表

JComboBox mychoice;

JCheckBox sl=new JCheckBox("私聊"); //定义私聊复选

JTextArea ta=new JTextArea(null,20,50); //用于显示数据

JScrollPane SC\_ta=new JScrollPane(ta);

Socket clientsocket=null; //用于通信的套接字

DataOutputStream os;

DataInputStream is;

String ss3="";

String S\_items[]={"red","blue","green","yellow","black","orange","pink"};

int t; //次整形变量为颜色信息长度

boolean boo;//私聊状态 true

LoginClient login=null; //将聊天的数据显示在 ta中的一个线程

public static void main(String args[]) { // 一个 application的 main()

Frame frame = new Frame("客户端窗口"); // 创建窗口frame to house the applet

ChatClient\_DB app = new ChatClient\_DB(); // Create an instance of the class (applet)

n=30;

app.init(); //Call the applet methods

//Add it to the center of the frame

frame.add("Center",app);

frame.setSize(355,400);

frame.validate();

frame.setVisible(true);

frame.addWindowListener(new WindowAdapter()

{ public void windowClosing(WindowEvent e)

{ // frame.dispose();

System.exit(0); }

});

app.start(); //###########先是由这个启动登陆对话框的线程#########

}

public void init(){

mychoice=new JComboBox(S\_items);//定义颜色

userList.addItem("大家");

ta.setLineWrap(true);//设置换行方式

setLayout(null);

setBounds(5,8,355,370);

Panel p1=new Panel(); //定义容器

p1.setLayout(null);

lb3.setBounds(new Rectangle(5,n+5,55,15));

lb4.setBounds(new Rectangle(65,n+5,100,15));

SC\_ta.setBounds(new Rectangle(5,n+25,325,210));//5,60,325,210

ta.setEditable(false); //设容器位置

//lb3.setVisible(true);

//lb4.setVisible(true);

textField1.setBounds(new Rectangle(5,285,200,20));

send.setBounds(new Rectangle(208,285,60,20));

exit.setBounds(new Rectangle(270,285,60,20));

lb1.setBounds(new Rectangle(5,315,55,20));

userList.setBounds(new Rectangle(65,315,70,20));

lb2.setBounds(new Rectangle(150,315,55,20));

mychoice.setBounds(new Rectangle(210,315,50,20));

sl.setBounds(new Rectangle(275,315,55,20));

//setVisible(true);

p1.setBounds(5,5,355,360);

p1.add(lb3);

p1.add(lb4);

p1.add(SC\_ta);

p1.add(textField1); //把控件添加在容器中

p1.add(send);

p1.add(exit);

p1.add(lb1); p1.add(userList);

p1.add(lb2); p1.add(mychoice);

p1.add(sl);

add(p1);

send.addActionListener(this); //设置监听器

exit.addActionListener(this);

textField1.addActionListener(this);

mychoice.addItemListener(this);

userList.addItemListener(this);

userList.addActionListener(this);

setSize(355,370);

ConnectDialog cd=new ConnectDialog(this,new Frame(),"连接",true); //输入用户名和主机，以便登录

cd.setVisible(true);

}//init()

public void itemStateChanged(ItemEvent e){

Object obj=e.getSource();

if(obj.equals(mychoice) ){

//设置发送端颜色 18

ss3=(String)mychoice.getSelectedItem();

if(ss3=="red")

textField1.setForeground(Color.red);

if(ss3=="blue")

textField1.setForeground(Color.blue);

if(ss3=="yellow")

textField1.setForeground(Color.yellow);

if(ss3=="black")

textField1.setForeground(Color.black);

if(ss3=="orange")

textField1.setForeground(Color.orange);

if(ss3=="pink")

textField1.setForeground(Color.pink);

if(ss3=="green")

textField1.setForeground(Color.green);

t=ss3.length();

ss3="#&@#"+ss3;

}

}

public void actionPerformed(ActionEvent evt) {//获取触发事件的对象

String sendString;

String sList=(String)userList.getSelectedItem();

Object obj=evt.getSource();

boo=(sList.indexOf("大家")!=-1)?false:true;//boolean

if((obj.equals(textField1)||obj.equals(send))&&!boo){//公聊

sendString="对"+sList+"说："+textField1.getText()+ss3;

sendData(sendString);

}else if (!obj.equals(exit)&&boo){//私聊，被聊天用户名前后加"#"

sendString="#"+sList+"#"+textField1.getText()+ss3;

sendData(sendString);

}else if(obj.equals(exit)){//退出按钮

sendString="用户"+lb4.getText()+"退出";//+textField1.getText()

sendData(sendString);

closeSocket();//本聊客退出

//this.dispose();

System.exit(0);

}

this.display(lb3.getText()+"对"+sList+"说："+textField1.getText()+"\n");

textField1.setText(""); //发送后清空

textField1.requestFocus();//焦点 19

}

boolean sendData(String toServer){ //自定义向服务器端发送数据方法

try{ if(toServer.length()>0) {//判断语句是否为空

if(toServer.charAt(0)!='#')//无"#"号，则前面加"\*"号

toServer='\*'+toServer;//

os.writeUTF(toServer); //从面板中读到的数据写到服务器中

os.flush();

}

}catch(Exception e){

System.out.println("Exception in sendData:"+e);

return false;

} return true;

}

String getData(){//服务器端得到数据

int i,j;

String user;

String fromServer=null;

out\_println("fromServer0：");

try{fromServer=is.readUTF();//从服务器端接收信息，从其它用户下拉列表中加入或删除

out\_println("fromServer1："+fromServer);

if (fromServer.indexOf("用户")!=-1)

{

//"警告"、"退出"、"删除"和"加入"操作

if (fromServer.indexOf("退出")!=-1||fromServer.indexOf("删除")!=-1)

{//在用户端列表中删除

if (fromServer.indexOf(lb3.getText())!=-1)

{//"删除"自己

userList.removeAllItems();//"删除"操作：被剔除的用户下拉列表中清空

}

else

{//其余用户的"退出"和"删除"操作：删除此用户

j=userList.getItemCount();

for (int k=1; k<=j;++k )

{

user=(String)userList.getItemAt(k-1);

if (fromServer.indexOf(user)!=-1)

{//

userList.removeItem(user);//其余的将"退出"和"删除"用户删除

--j;

out\_println("userList0："+j+";k="+k);

}

out\_println("userList1："+j+";k="+k);

}//for

}//if lb3

}//if"退出"，"删除"

//"加入"操作，包括自己

if (fromServer.endsWith("加入\n")){//在用户端添加列表

i=fromServer.length();

user=fromServer.substring(2,i-3);//2 指"用户"

userList.addItem(user);

}//if加入 20

}

else

{ //"聊天"和"警告"操作，获取用户名

i=fromServer.length();

user=fromServer.substring(4,i);

//userList.addItem(user);//userList.add(user);

} //if用户

}catch(IOException e)

{

return null;

} //try

return fromServer;

}//get

void closeSocket(){//本聊客退出，关闭服务器连接

try{

os.close();

is.close();

clientsocket.close();//套接字

//userList.removeAllItems();//清除所有

}catch(Exception e){

System.out.println("Exception in closeData:"+e);

}

}

boolean connectHost(String hostName){//建立服务器连接

if(clientsocket==null){

try{clientsocket=new Socket(hostName,8888);

is=new DataInputStream(clientsocket.getInputStream());

os=new DataOutputStream(clientsocket.getOutputStream());

}catch(UnknownHostException e){

System.out.println("Trying to connect unknown Host"+e);

return false;

}catch(IOException e){

System.out.println(e.toString());

return false;

}

}

return true;

}

public void display(String str){//在ta中显示收到的数据

ta.append(str);

}

public void YHM(String str){//在lb4中显示用户名

lb4.setText(str);

}

public void out\_println(String str){//显示

System.out.println(str);

}

public void display1(String str){ //在客户端显示颜色

if (str.indexOf("pink")!=-1){

ta.setForeground(Color.pink);

}else if (str.indexOf("orange")!=-1){ //判断颜色

ta.setForeground(Color.orange);

}else if (str.indexOf("green")!=-1){

ta.setForeground(Color.green);

}else if (str.indexOf("black")!=-1){

ta.setForeground(Color.black);

}else if (str.indexOf("red")!=-1){

ta.setForeground(Color.red);

}else if (str.indexOf("yellow")!=-1){

ta.setForeground(Color.yellow);

}else if (str.indexOf("blue")!=-1){

ta.setForeground(Color.blue);

}

}

}

//客户聊天线程

class LoginClient extends Thread{

ChatClient\_DB client=null;

String fromServer1;

//将客户对象client与客户线程捆绑在一起

public LoginClient(ChatClient\_DB client){

this.client=client;

}

public void run(){

String fromServer=null;

while((fromServer=client.getData())!=null){//始终循环

if (fromServer.indexOf("#&@#")!=-1)

{// 取颜色信息并在用户端设置颜色

int i=fromServer.lastIndexOf("#&@#"); fromServer1=fromServer.substring(fromServer.indexOf("#&@#"));//获取 #$@#

fromServer=fromServer.substring(0,fromServer.length()-fromServer1.length());

fromServer1=fromServer1.substring(4,fromServer1.length()-1);//获取颜色名称

client.display(fromServer+"\n"); //在 ta中显示收到的数据

client.display1(fromServer1); //在客户端显示颜色

}

else client.display(fromServer); //在 ta中显示收到的数据

}

}//run()

}//LoginClient

class ConnectDialog extends JDialog{ //建立登录信息对话框

ChatClient\_DB app;

JPanel DialogPane= new JPanel();

JTextField host=new JTextField("127.0.0.1",20); //本地主机名

JTextField user=new JTextField(20);

JTextField passwd=new JTextField(20);

JButton register=new JButton("注册");

JButton connect=new JButton("登录");//连接

JButton cancel=new JButton("取消");

JLabel L\_user=new JLabel("用户名：", JLabel.RIGHT);

JLabel L\_host=new JLabel("主机名IP：", JLabel.RIGHT);

JLabel L\_passwd=new JLabel("密码：", JLabel.RIGHT);

ConnectDialog(ChatClient\_DB ap,Frame f1,String str,boolean b1){

super(f1,str,b1); //Dialog(Dialog owner, String title, boolean modal) 创建一个具有指定标题、模式和指定所有者 Dialog 的对话框。

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*两个super不理解

app=ap;

app.out\_println("登录对话框 0");

DialogPane.setLayout(null);

L\_user.setBounds(new Rectangle(10,10,70,20));

user.setBounds(new Rectangle(83,10,100,20));

L\_passwd.setBounds(new Rectangle(10,45,70,20));

passwd.setBounds(new Rectangle(83,45,100,20));

L\_host.setBounds(new Rectangle(10,80,70,20));

host.setBounds(new Rectangle(83,80,100,20));

register.setBounds(new Rectangle(10,115,60,20));

connect.setBounds(new Rectangle(75,115,60,20));

cancel.setBounds(new Rectangle(140,115,60,20));

DialogPane.add(L\_user);

DialogPane.add(user);

DialogPane.add(L\_passwd);

DialogPane.add(passwd);

DialogPane.add(L\_host);

DialogPane.add(host);

DialogPane.add(register);

DialogPane.add(connect);

DialogPane.add(cancel);

add(DialogPane);

app.out\_println("登录对话框 1");

setBounds(5,5,240,230);

register.addMouseListener(new java.awt.event.MouseAdapter(){

public void mouseClicked(MouseEvent e){

register\_mouseClicked(e); }//登录、注册

});

connect.addMouseListener(new java.awt.event.MouseAdapter(){

public void mouseClicked(MouseEvent e){

connect\_mouseClicked(e);}//登录、连接

});

cancel.addMouseListener(new java.awt.event.MouseAdapter(){

public void mouseClicked(MouseEvent e){

cancle\_mouseClicked(e);

}

});

}

//放弃连接服务器

void cancle\_mouseClicked(MouseEvent e){

app.closeSocket();

app.destroy();

dispose();

System.exit(0);

}

//注册。可判断空、可识别用户名是否已经被注册

void register\_mouseClicked(MouseEvent e){

String S\_user=user.getText().trim();

String S\_passwd=passwd.getText().trim();

// ServerTaber\_DB SDB=new ServerTaber\_DB();

app.out\_println("注册 S\_passwd："+S\_passwd);

if(b\_tf(S\_user,S\_passwd)){ //b\_tf()字符串是否为空？不空返回true

if(app.connectHost(host.getText())){

app.sendData("%%%%%"+S\_user);//用户名前端系统加"%%%%%"，表示客户端为登录、注册操作

app.sendData(S\_passwd);

if(app.getData().equals("error"))

user.setText("用户名已经被注册，请换一个");

else{

// SDB.insertRec("INSERT INTO userlist(name,passwd) VALUES("+S\_user+','+S\_passwd+");", S\_user, S\_passwd);

//在这儿写入到数据库。。。。。。。。public void insertRec(String sql,String name,String passwd)

user.setText("注册成功");//"right"

app.out\_println("用户："+S\_user+"注册成功！");

}

}else host.setText("不能连接服务器");

}//if

}//register\_mouseClicked

//登录、连接

void connect\_mouseClicked(MouseEvent e){ //连接服务器

String S\_user=user.getText().trim();

String S\_passwd=passwd.getText().trim();

app.out\_println("登录 S\_passwd0："+S\_passwd);

if(b\_tf(S\_user,S\_passwd)){ //b\_tf()字符串是否为空？不空返回true

//app.out\_println("S\_passwd1："+S\_passwd);

if(app.connectHost(host.getText())){ //连接主机

app.sendData(S\_user); //发送用户名

app.sendData(S\_passwd);

//app.out\_println("S\_passwd2："+S\_passwd);

if(app.getData().equals("error1"))//出现异常！ 24

user.setText("密码错误或数据库没有连上！");

else if(app.getData().equals("error2")){//出现异常！（//有重名或密码错误或数据库没有连上等）

user.setText("有重名或以登录！");

}else{ //"right"

app.out\_println("“"+S\_user+"”用户名登录成功！");

app.YHM(S\_user); //在lb4中显示用户名

app.login=new LoginClient(app);

app.login.start(); //登录线程 ############上面的app.start()线程启动成功后对密码账户处理后才进入这个线程

app.setName("Client-user["+S\_user+"]:");

dispose();//使面板消失的方法，不然会登陆不上

}

}else{ host.setText("不能连接主机");

}

}//if

}//connect\_mouseClicked

//字符串是否为空？空返回 false，不空返回true

boolean b\_tf(String S\_u,String S\_p){

if(S\_u.equals("")||S\_u.equals("用户名不能为空!")){

user.setText("用户名不能为空!");

return false;

}else if(S\_p.equals("")||S\_p.equals("密码不能为空!")){

passwd.setText("密码不能为空!");

return false;

}else{ if(!S\_u.equals("")&&!S\_p.equals(""))

return true;

else return false;

}//if

}//b\_tf

}//end ConnectDialog

//////////////////////////////////////////////

//服务器端程序 //

//多用户聊天室 //

// ChatServer\_DB.java //

//是Application在单机运行 //

//姓名： //

//20XX 年XX月XX日 //

///////////////////////////////////////////////

//ChatServer\_DB.java

import java.awt.\*;

import java.io.\*;

import java.net.\*;

import java.util.\*;

import java.awt.event.\*;

public class ChatServer\_DB extends Thread{

public final static int PORT=8888;

ServerSocket ss;

java.awt.List user\_list;//生成用户列表；

Vector users;

Vector connections;

String username;//当前登录用户

String passwd; //当前登录用户密码

public synchronized void removeUser(String S\_user,Connection1 con){ //线程同步 从服服务器中删除

user\_list.remove(S\_user);

users.remove(S\_user);

connections.remove(con);

}

public void out\_println(String str){//将跟踪信息显示在 DOS 界面上

System.out.println(str);

}

public ChatServer\_DB()

{//构造函数初始化服务端界面；

super("Server");//

try

{

ss=new ServerSocket(PORT);

}

catch(IOException e)

{

e.printStackTrace();

}

Frame f=new Frame("服务器");

user\_list=new java.awt.List();

Button warn=new Button("警告");

Button delete=new Button("踢除");

// warn.setBounds(new Rectangle(108,85,60,20));

// delete.setBounds(new Rectangle(290,85,60,20));

Panel p2=new Panel();

p2.add("West",warn);

p2.add("East",delete);

warn.addMouseListener(new java.awt.event.MouseAdapter(){

public void mouseClicked(MouseEvent e){

//8、"警告"操作，言论不合法，警告!!,并公布给大家 false,不删除false

serverWriter("用户"+user\_list.getSelectedItem()+"言论不合法，警告!!!!!",false,null,false);

}

});

delete.addMouseListener(new java.awt.event.MouseAdapter(){

public void mouseClicked(MouseEvent e){

String S\_user=user\_list.getSelectedItem();//选中删除用户 26

//7、"踢除"("删除")操作，公布给大家false,并删除!!true，删除者S\_user

serverWriter("用户"+S\_user+"言论不合法，不适合聊天，删除!!",false,S\_user,true);

//在serverWriter中，通过识别"删除!!"来删除 S\_user聊客

}

});

f.add("South",p2);

f.add("Center",user\_list);

f.setSize(400,200);

f.setVisible(true);

// f.addWindowListener(new closeWin());

f.addWindowListener(new WindowAdapter()

{ public void windowClosing(WindowEvent e)

{ //f.dispose();

System.exit(0); }

});

users=new Vector();

connections=new Vector();

start();

out\_println("服务器运行成功！");

}//public ChatServer\_DB()

public void run()

{//用户登录线程启动（主线程启动）

try{

while(true)

{

out\_println("主线程启动！\n");

Socket s=ss.accept();//侦听并接受到此套接字的连接

out\_println("侦听启动！\n");

Connection1 c=new Connection1(s,this);

out\_println("主线程c："+c+"\n");

ServerTaber\_DB js=new ServerTaber\_DB();//创建数据库操作对象

out\_println("创建数据库操作对象\n");

username=c.readData(); //否则会处于等待状态

passwd=c.readData();

out\_println("主线程启动！"+username+"\n");

String str2=new String();

//注册：用户名前端系统加"%%%%%"，表示客户端为注册操作

if(username.startsWith("%%%%%"))

{

String b=username.substring(5);

try

{//数据库操作，查询此用户密码

str2=js.queryTable("select passwd from userlist where name='"+b+"';");

str2 = str2.trim();

}

catch(Exception e){}

//如果数据库中没有此用户密码，返回空时则加上"$$$$$$"

//以username 和passwd注册插入更新数据库;

if(str2.equals("$$$$$$"))

{

try

{//

js.insertRec("insert into userlist (name,passwd) values (?,?);",b,passwd);

}

catch(Exception e){}

c.writeData("right");//告知用户注册成功

out\_println(username+"注册成功！\n");

}else c.writeData("error");

}//if

//连接：用户名前端系统没有加"%%%%%"，表示客户端为连接操作

if(!username.startsWith("%%%%%"))

{//无重名 break跳出循环，可以连接

while(true)

{//

try

{//

str2=js.queryTable("select passwd from userlist where name='"+username+"';");

str2 = str2.trim();

}

catch(Exception e){}

if(passwd.equals(""+str2)&&(!users.contains(username)))

break;//跳出循环，可以连接

if(!passwd.equals(""+str2))

c.writeData("error1");//密码错误或数据库没有连上

if(users.contains(username))

c.writeData("error2");//有重名或以登录

username=c.readData();//循环等候接收新的用户名

passwd=c.readData();

}//while(true)

c.writeData("right");//可以连接

c.username=username;

c.start();

synchronized(users)

{//同步实现加载用户

users.addElement(username);

connections.addElement(c);

user\_list.add(username);

}

//1、"加入"告知大家（你对大家），包括自己

//登录成功，信息发布给大家，添加到用户列表中

//登录用户 username告知大家（公聊、不踢除）

serverWriter("用户"+username+"加入",false,null,false);//公聊false

//2、"加入"，大家对你，将已有用户名加入到你的列表框中

//在新用户登录后，列表框中加入已有用户名，不包括自己

Enumeration em=users.elements();//已有用户向量

while(em.hasMoreElements()){

String str=(String)em.nextElement();

if (!str.equals(username)){//不包括自己，大家对你私聊

serverWriter("用户"+str+"加入",true,username,false);//私聊

}

}

}//if

}//while(true)

}catch(IOException e){ e.printStackTrace();}

}//run

public synchronized void serverWriter(String str,boolean w,String S\_user,boolean b\_delete){

Enumeration em=connections.elements();//连接线程用户向量枚举

Connection1 c\_delet=null;

while(em.hasMoreElements()){//枚举循环

Connection1 c=(Connection1)em.nextElement();

if(w)

{ //私聊true

if(c.username.equals(S\_user))//匹配私聊对象S\_user，以连接用户线程 c.username

c.writeData(str+"\n");

}

else

{ //公聊false

//if (!S\_user.equals(c.username)||str.indexOf("加入")!=-1 ){//不包括自己

c.writeData(str+"\n"); //不匹配聊天对象S\_user=null

out\_println("c："+c+";S\_c:"+c.username+";S\_user:"+S\_user+";str:"+str+"\n");

//}

}//if

if(b\_delete)

{//"删除"操作：匹配删除对象 S\_user。不包括"退出"

if(c.username.equals(S\_user))

{//匹配删除对象S\_user

c\_delet=(Connection1)c;

}

}

}//while

if(b\_delete && c\_delet.username.equals(S\_user)){//删除匹配对象S\_user

removeUser(S\_user,c\_delet);//在服务器中删除 29

try{

c\_delet.dis.close();//c\_delet.close();不能用，c\_delet没有传过去

c\_delet.dos.close();

c\_delet.s.close();

}catch(IOException e){ e.printStackTrace();}

out\_println("删除！"+S\_user+";c\_delet:"+c\_delet+"\n");

}

}// serverWriter()

public static void main(String[] args)//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*Main

{

new ChatServer\_DB();

}

}//ChatServer\_DB

class Connection1 extends Thread{

ChatServer\_DB cs;

Socket s;

DataInputStream dis;

DataOutputStream dos;

public String username;

String talker=null;

boolean whisper;

public Connection1(Socket s,ChatServer\_DB cs){

this.cs=cs;

this.s=s;

try{dis=new DataInputStream(s.getInputStream());

dos=new DataOutputStream(s.getOutputStream());

}catch(IOException e){ e.printStackTrace();}

cs.out\_println("输入输出流启动！\n");

}//Connection1()

public void run(){//登录后聊天线程

String temp;

try{

while(true)

{

temp=readData();//返回聊天内容，同时确定username、whisper,talker

if (temp.indexOf("退出")!=-1){

//3、或 4、"退出"操作，接收退出处理(公聊和私聊)

cs.serverWriter(temp,whisper,talker,false);//"["+username+"]"+

cs.out\_println("聊天线程退出："+username+";"+temp+"\n");

close();//"退出"在此服务器中删除

}else{

//5、或 6、"聊天"操作,非退出 30

cs.serverWriter("["+username+"]"+temp,whisper,talker,false);

cs.out\_println("聊天线程启动："+username+"\n");

}

}//while

}catch(IOException e2){}

finally{ //close();

}

}//run()

public void close(){

cs.removeUser(username,this);

try{ dis.close();

dos.close();

s.close();

}catch(IOException e){}

//cs.serverWriter("用户"+username+"退出",false,null,true);//公聊，告知所有聊客，退出

}//close()

//返回聊天内容

public String readData() throws IOException{

String temp=dis.readUTF();

if(temp.charAt(0)=='#')

{ //私聊，确定username、whisper,talker

whisper=true; //私聊

int kk=temp.indexOf('#',1);

talker=temp.substring(1,kk); //私聊对象

return new String(temp.substring(kk+1));//聊天内容

}

whisper=false;//公聊

talker=null;

return new String(temp.substring(1));//聊天内容

}// readData()

public void writeData(String str) {

try{dos.writeUTF(str);

dos.flush();

}catch(IOException e){ e.printStackTrace();}

}//writeData()

}//Connection1

////////////////////////////////////////

//数据库程序 //

//ServerTaber\_DB.java //

//姓名： //

//20xx年 xx月 xx日 //

///////////////////////////////////////

/\*

1、Access 数据库userlist.mdb，数据库中表单名userlist。

2、建成邦定Access 数据库 userlist\_DB.mdb 的 userlist\_DB数据源。

3、本程序功能：创建数据库表单、插入记录、条件查询、删除记录和显示结果。

\*/

import java.sql.\*;

public class ServerTaber\_DB {

Statement stmt=null;

Connection con=null;

PreparedStatement pstmt=null;

ResultSet rs=null;

String URL;

//连接数据库源

public ServerTaber\_DB() {

//userlist\_DB数据源事先已建成（邦定Access 数据库）

URL="jdbc:odbc:userlist\_DB";

//1、加载驱动程序，同时向 DriverManager注册，DriverManager为数据库查询驱动程序对象的类

try{

Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");

} catch(Exception e){

System.out.println("Failed to load JDBC/ODBC driver.");

return;

}

//2、连接数据库源

try{

con=(Connection)DriverManager.getConnection(URL);

//3、创建发送SQL语句的Statement对象

stmt=con.createStatement();

} catch(Exception e){

System.err.println("problems connecting to"+URL);

}

}//end ServerTaber\_DB()

//创建表单

public void createTable(String sql)

{ //4、执行SQL语句

try{

stmt.execute(sql);

}catch(Exception e){

System.out.println("createTabele:"+e);

}

}

//插入记录

public void insertRec(String sql,String name,String passwd){

try{ pstmt=con.prepareStatement(sql);

pstmt.setString(1,name);

pstmt.setString(2,passwd);

pstmt.executeUpdate();

}catch (Exception e){

System.err.println("queryTable:"+e); } }

//条件查询

public String queryTable(String sql)throws SQLException{

try{ rs=stmt.executeQuery(sql);

}catch(Exception e){

System.out.println("queryTable:"+e); }

return(dispResultSet(rs));// 调用 dispResultSet(rs)方法显示结果 }

//删除记录

public void deleteRec(String sql)throws

SQLException{

try{ stmt.executeUpdate(sql);

// dispResultSet(rs)

//System.out.println(dispResultSet(rs));

}catch(Exception e){

System.out.println("detele:"+e); } }

private String dispResultSet(ResultSet rs) throws SQLException{

int i;

ResultSetMetaData rsmd=rs.getMetaData();

int numCols=rsmd.getColumnCount(); //字段数或多少列数

if(rs.next()==false) // 当 前 行 不 存 在 返 回"$$$$$$"可能是用户名或

return("$$$$$$");

else { String str=rs.getString(numCols);//返回当前行中的列值

return(str);

} } }