

**《JAVA程序设计》——实验报告**

**使用TCP协议的Socket网络程序设计**

**学 院**  计算机科学与工程学院

**专 业 12级**计算机科学与技术

**组 员**  陈伟航

**学 号**  201230600095

**提交日期**  2013年12月12日

**使用TCP协议的Socket网络程序设计**

**一、实验目的**

1. 掌握Socket通讯机制；

2、掌握Socket和ServerSocket类和相关方法；

**二、实验要求**

通过Socket编程，掌握网络应用程序的开发方法，掌握利用java提供的基本组件进行网络传输，掌握Java提供的多线程机制，异常处理机制和对底层协议的通信机制，通过Socket编程，掌握网络应用程序的开发方法。

**三、实验任务**

1、创建服务器和客户程序，在运行客户程序的计算机上输入的内容，可以在服务器屏幕上看到。

2、使用ServerSocket类和Socket类实现按如下协议通信的服务器端和客户端程序。服务器程序的处理规则如下：

1）向客户端程序发送Verifying Server!。

2）若读口令次数超过3次，则发送Illegal User!给客户端，程序退出。否则向下执行步骤3）。

3）读取客户端程序提供的口令。

4）若口令不正确，则发送PassWord Wrong!给客户端，并转步骤2），否则向下执行步骤5）。

5）发送Registration Successful!给客户端程序。并打开另一个多人聊天室程序。

3、客户端程序的处理规则如下：

1）读取服务器反馈信息。

2）若反馈信息不是Verifying Server!，则提示Server Wrong!，程序退出。否则向下执行步骤3） 。

3） 提示输入PassWord并将输入的口令发送给服务器。

4） 读取服务器反馈信息。

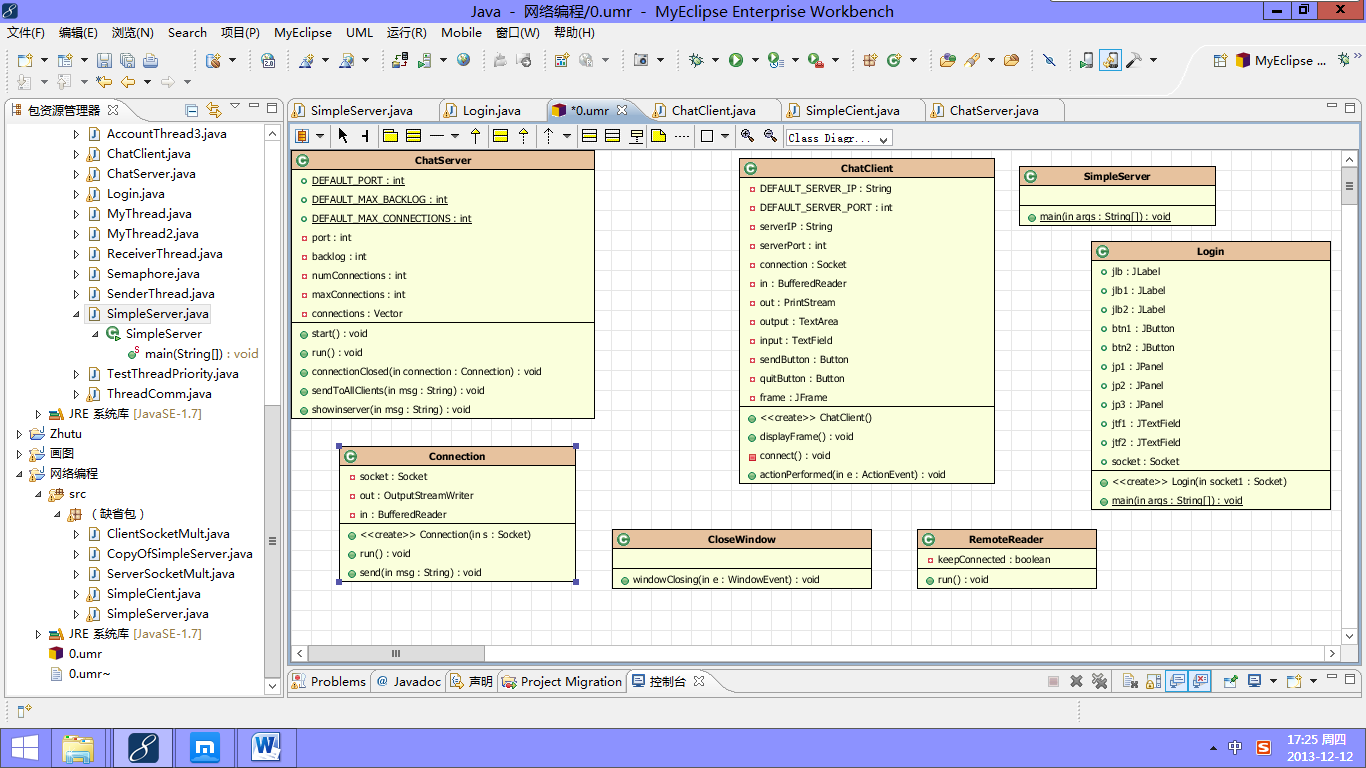
5）若反馈信息是Illegal User!，则提示Illegal User!，程序退出。否则向下执行步骤6） 。

6）若反馈信息是PassWord Wrong!，则提示PassWord Wrong!，并转步骤3），否则向下执行步骤。

7）输出Registration Successful!。

**四、实验结果**

**UML图**



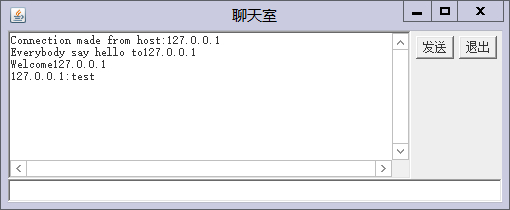
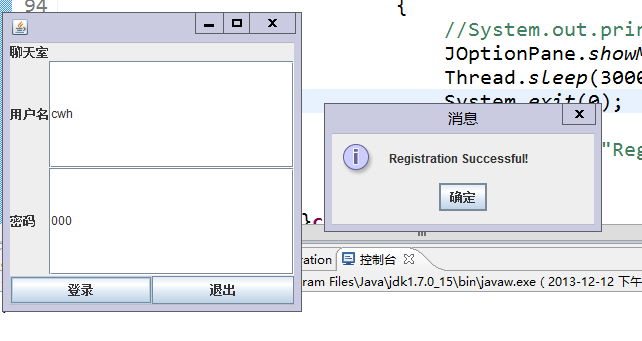
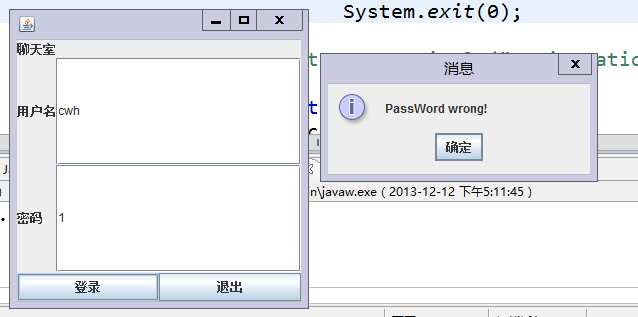
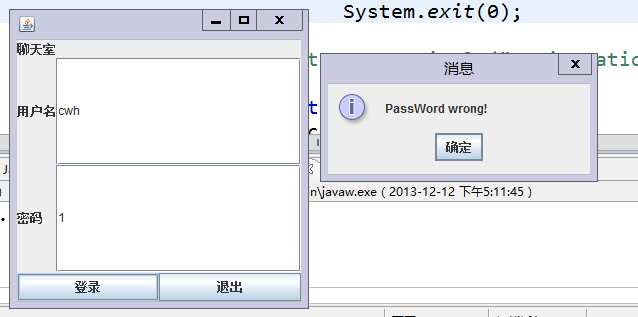
**实验中：1）用户名验证，三次不正确退出程序**

**2）密码验证，三次不正确退出程序**

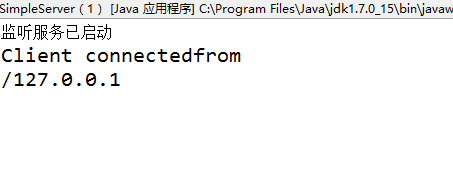
**注：用户名是cwh，密码是 000**

**（1）用户名和密码都是第三次输入正确，前两次输入错误，最后顺利通过验证。**

**客户端：**

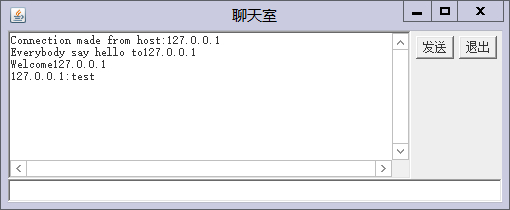
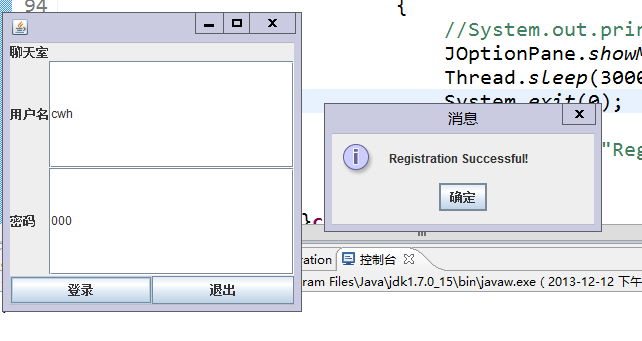


**服务端：**

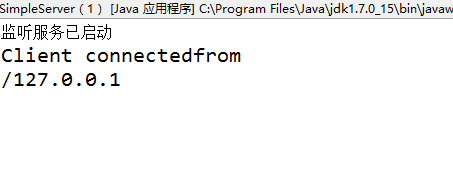


**（2）用户名和密码都是第一次输入正确，顺利通过验证。**

**客户端：**



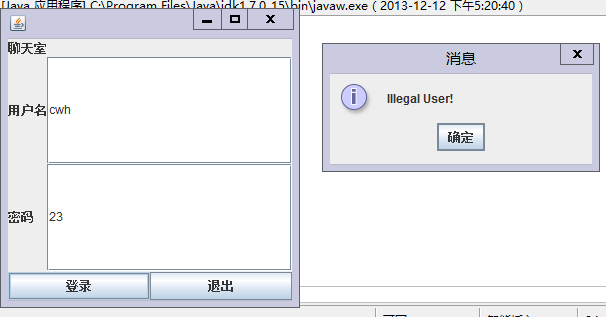
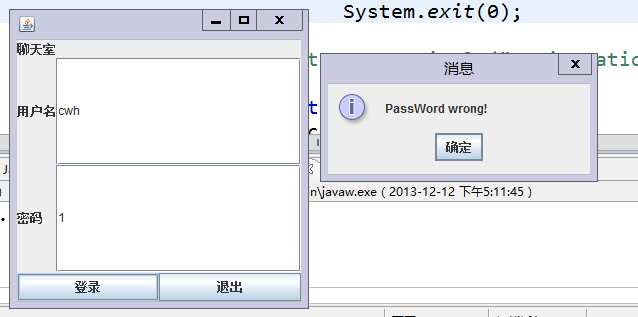
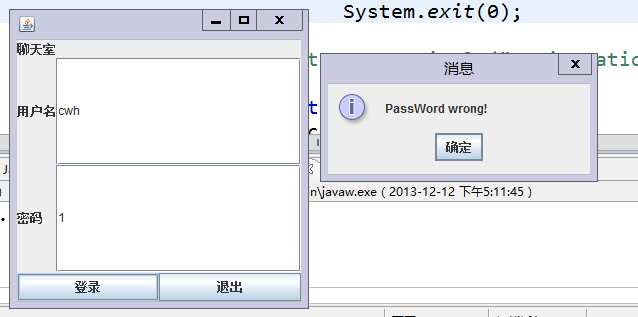
**服务端：**



**（3）用户名和密码都是三次输入不正确，强制关闭程序。**

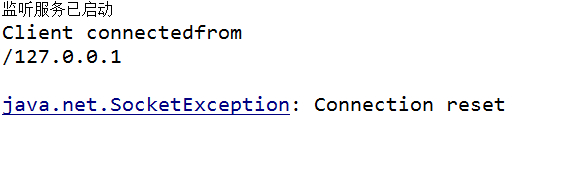
**客户端：**

**（三次密码错误）**



**服务端：**

**（三次用户名错误）**



**五、实验代码**

**验证密码客户端代码：**

**import** java.awt.\*;

**import** java.awt.event.\*;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.OutputStreamWriter;

**import** java.io.PrintWriter;

**import** java.net.InetAddress;

**import** java.net.Socket;

**import** java.util.Scanner;

**import** javax.swing.\*;

**public** **class** Login **extends** JFrame

{

JLabel jlb=**new** JLabel("聊天室");

JLabel jlb1=**new** JLabel("用户名");

JLabel jlb2=**new** JLabel("密码");

JButton btn1=**new** JButton("登录");

//JButton btn3=new JButton("连接服务器");

JButton btn2=**new** JButton("退出");

JPanel jp1=**new** JPanel();

JPanel jp2=**new** JPanel();

JPanel jp3=**new** JPanel();

JTextField jtf1=**new** JTextField();

JTextField jtf2=**new** JTextField();

Socket socket=**new** Socket();

**public** Login(Socket socket1)

{

socket=socket1;

setSize(300,300);

setLocation(400,400);

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

setLayout(**new** BorderLayout());

add(jlb,BorderLayout.*NORTH*);

add(jp1,BorderLayout.*WEST*);

add(jp2,BorderLayout.*CENTER*);

add(jp3,BorderLayout.*SOUTH*);

jp1.setLayout(**new** GridLayout(2,1));

jp1.add(jlb1);

jp1.add(jlb2);

jp2.setLayout(**new** GridLayout(2,1));

jp2.add(jtf1);

jp2.add(jtf2);

jp3.setLayout(**new** GridLayout(1,3));

//jp3.add(btn3);

jp3.add(btn1);

jp3.add(btn2);

btn1.addActionListener(**new** ActionListener()

{

**public** **void** actionPerformed(ActionEvent e)

{

JButton btn=(JButton)e.getSource();

{

**try**

{

BufferedReader in=**new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));

PrintWriter out=**new** PrintWriter(**new** OutputStreamWriter(socket.getOutputStream()),**true**);

String s0=**new** String();

**if**((s0=in.readLine()).equals("Verifying Server!"))

{

//s0=in.readLine();//读取服务器端传过来的数据

/\*Scanner cin=new Scanner(System.in);

System.out.println("");\*/

String str1=jtf1.getText();

//System.out.println(str1);

String str2=jtf2.getText();//cin.next();

out.println(str1);//不会输出，但可以传给服务器端输出

out.flush();//似乎必须这样服务器才能检测到out

out.println(str2);

out.flush();

String s1=in.readLine();//读取服务器端传过来的数据

**if**(s1.equals("Illegal User!"))

{

//System.out.println(s1);//输出服务器端输出的Registration Successful!

JOptionPane.*showMessageDialog*(**null**,"Illegal User!");

System.*exit*(0);

}

**else** **if**(s1.equals("PassWord Wrong!"))

{

//System.out.println("PassWord Wrong!");

//JOptionPane.showMessageDialog(null,"PassWord wrong!");

JOptionPane.*showMessageDialog*(**null**,"Illegal User!");

System.*exit*(0);

}

**else** **if**(s1.equals("Registration Successful!"))

{

JOptionPane.*showMessageDialog*(**null**,"Registration Successful!");

ChatServer chatServer=**new** ChatServer();//创建聊天服务器

chatServer.start();//开始运行

ChatClient chatClient=**new** ChatClient();

chatClient.displayFrame();

}

}

**else**

{

//System.out.println("Server Wrong!");

JOptionPane.*showMessageDialog*(**null**,"Server Wrong!");

Thread.*sleep*(3000);

System.*exit*(0);

}

//System.out.println("Registration Successful!");

socket.close();

}**catch**(Exception e1){}

/\*ChatClient chatClient=new ChatClient();

chatClient.displayFrame();\*/

}

}

});

btn2.addActionListener(**new** ActionListener()

{

**public** **void** actionPerformed(ActionEvent e)

{

JButton btn=(JButton)e.getSource();

{

System.*exit*(0);//close

}

}

});

/\*btn3.addActionListener(new ActionListener()

{

public void actionPerformed(ActionEvent e)

{

JButton btn=(JButton)e.getSource();

{

try

{

Socket socket=new Socket(InetAddress.getByName(null),8080);

}catch(Exception e1){}

}

}

});\*/

}

**public** **static** **void** main(String[]args) **throws** Exception

{

/\*SimpleServer server=new SimpleServer();

server.serverbeg();\*/

/\*ChatServer chatServer=new ChatServer();//创建聊天服务器

chatServer.start();//开始运行

\*/

Socket socket=**new** Socket(InetAddress.*getByName*(**null**),8080);

Login l=**new** Login(socket);

l.setVisible(**true**);

//socket.close();

}

}

**验证密码服务端代码：**

import java.net.\*;

import java.io.\*;

public class SimpleServer {

public static void main(String args[])

{

try

{

ServerSocket server=new ServerSocket(8080);

System.out.println("监听服务已启动");

Socket socket=server.accept();

//System.out.println("mark1");

InetAddress clientAddress=socket.getInetAddress();

System.out.println("Client connected"+"from\n"+clientAddress+"\n");

Thread.sleep(3000);

PrintWriter out=new PrintWriter(new OutputStreamWriter(socket.getOutputStream()),true);

BufferedReader in=new BufferedReader(new InputStreamReader(socket.getInputStream()));

int i;

for(i=0;i<3;i++)

{

out.println("Verifying Server!");

out.flush();

String s1=in.readLine();//读取客户端用户名

String s2=in.readLine();//读取客户端密码

if(s1.equals("cwh")&&s2.equals("000"))

{

out.println("Registration Successful!");//不会输出，但是会传给客户端

out.flush();

break;

}

else

{

if(i>1)out.println("Illegal User!");

else out.println("PassWord Wrong!");//不会输出，但是会传给客户端

out.flush();

}

}

//out.flush();

/\*out.close();//关闭out,输出缓冲区内容

in.close();//关闭in

server.close();//关闭socketserver,不再监听

\*/ }catch(Exception e)

{

System.out.println(e);

}

}

}

聊天室服务端

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class ChatServer implements Runnable

{

//定义常数

public final static int DEFAULT\_PORT=8080;//默认端口

public final static int DEFAULT\_MAX\_BACKLOG=5;//默认等待队列长度

public final static int DEFAULT\_MAX\_CONNECTIONS=20;//默认最大连接数

//定义属性

private int port=DEFAULT\_PORT;//端口号

private int backlog=DEFAULT\_MAX\_BACKLOG;//等待队列长度

private int numConnections=0;//当前连接数

private int maxConnections=DEFAULT\_MAX\_CONNECTIONS;//最大连接数

private Vector connections=null;//保存连接

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

{

ChatServer chatServer=new ChatServer();//创建聊天服务器

chatServer.start();//开始运行

}\*/

public void start()

{

connections=new Vector(maxConnections);//创建向量，保存连接

Thread t=new Thread(this);//使用本实例创建一个线程

t.start();//启动该线程

}

public void run()//线程的主办法

{

ServerSocket server=null;

Socket socket;

try

{

System.out.println("服务器开始启动……");

server=new ServerSocket(port,backlog);//创建ServerSocket

}catch(IOException e)

{

System.out.println(e);

}

System.out.println("服务器启动成功！端口号为："+port);

//使用死循环一直监控客户端的连接请求

while(true)

{

try

{

socket=server.accept();//监听客户端，等待连接请求

synchronized(this)

{

while(numConnections==maxConnections)wait();//如果达到最大连接数，线程等待

}

numConnections++;//连接数加1

Connection con=new Connection(socket);//创建一个连接对象

Thread t=new Thread(con);//每个连接创建一个线程

t.start();//启动该线程

connections.addElement(con);//连接对象放入vector中，用于发送信息

}catch(Exception e)

{

System.out.println(e);

}//end try

}//end while

}//end run

public synchronized void connectionClosed(Connection connection)//关闭连接

{

connections.removeElement(connection);//从连接队列中删除

numConnections--;//连接数减1

notify();//通知等待的线程

}

public void sendToAllClients(String msg)//发送信息给所有的客户端

{

for(int i=0;i<connections.size();i++)//遍历集合中的连接

((Connection)connections.get(i)).send(msg);

}

public void showinserver(String msg)

{

System.out.println(msg);

}

class Connection implements Runnable//定义内部类实现Runnable接口

{

private Socket socket=null;

private OutputStreamWriter out=null;

private BufferedReader in=null;

public Connection(Socket s)

{

socket=s;

}

public void run()//Connection线程的主方法

{

try

{

out=new OutputStreamWriter(socket.getOutputStream());//转换成字符流

in=new BufferedReader(new InputStreamReader(socket.getInputStream()));

InetAddress address=socket.getInetAddress();//获取客户端地址

String hostname=address.getHostName();//获取客户机名

String welcome="Connection made from host:"+hostname+"\nEverybody say hello to"+hostname+"\n";//欢迎词

sendToAllClients(welcome);//向所有客户端发送欢迎词

showinserver(welcome);

send("Welcome"+hostname+"\n");//向本连接发送欢迎信息

showinserver("Welcome"+hostname+"\n");

String input=null;//用于接收客户端信息

while((input=in.readLine())!=null)//接受该客户端的信息

{

sendToAllClients(hostname+":"+input+"\n");//发送给所有的客户端

showinserver(hostname+":"+input+"\n");

}

}catch(Exception e)

{

System.out.println(e);

}finally

{//关闭所有流

try

{

if(in!=null)in.close();

if(out!=null)out.close();

socket.close();

}catch(Exception e)

{

System.out.println(e);

}

connectionClosed(this);//调用属主类的方法关闭连接

}

}

public void send(String msg)//向本连接发送信息

{

try

{

out.write(msg);

out.flush();

}catch(Exception e)

{

System.out.println(e);

}

}//end sendMessage

}//end connection

}//end ChatServer

**聊天室客户端**

import java.awt.\*;

import java.awt.event.\*;

import java.net.\*;

import java.io.\*;

import javax.swing.\*;

import javax.swing.JButton;

public class ChatClient implements ActionListener

{

private final String DEFAULT\_SERVER\_IP="127.0.0.1";//默认服务器地址

private final int DEFAULT\_SERVER\_PORT=8080;//默认服务器端口

private String serverIP=DEFAULT\_SERVER\_IP;//服务器IP地址

private int serverPort=DEFAULT\_SERVER\_PORT;//服务器端口

private Socket connection=null;

private BufferedReader in=null;

private PrintStream out=null;

private TextArea output;//文本域输出信息

private TextField input;//文本行输入信息

private Button sendButton;

private Button quitButton;

private JFrame frame=new JFrame("聊天室");

public ChatClient()//构造方法

{

output=new TextArea(10,50);//文本域10行50列

input=new TextField(50);//文本行50列

sendButton=new Button("发送");

quitButton=new Button("退出");

}

public void displayFrame()

{

frame.setLayout(new BorderLayout());

frame.add("Center",output);

frame.add("South",input);

Panel pl=new Panel();

pl.add(sendButton);

pl.add(quitButton);

frame.add("East",pl);

frame.addWindowListener(new CloseWindow());//注册window事件监听器

sendButton.addActionListener(this);//注册按钮监听器

quitButton.addActionListener(this);//注册按钮监听器

connect();//连接服务器并发送信息

frame.setSize(510,210);

frame.setVisible(true);

}

private void connect()

{

try

{

connection=new Socket(serverIP,serverPort);//连接服务器

in=new BufferedReader(new InputStreamReader(connection.getInputStream()));

out=new PrintStream(connection.getOutputStream());

Thread t=new Thread(new RemoteReader());//创建一个线程来进行通信

t.start();//启动线程

}catch(Exception e)

{

System.out.println(e);

}

}

public void actionPerformed(ActionEvent e)//处理按钮事件

{

if(e.getActionCommand().equals("发送"))

{

String text=input.getText();

out.println(text);//向服务器中输出信息

input.setText("");

}

else

{

frame.dispose();

}

}

private class CloseWindow extends WindowAdapter

{

public void windowClosing(WindowEvent e)//处理窗口关闭事件

{

frame.dispose();

}

}

private class RemoteReader implements Runnable//内部Runnable类完成接收信息

{

private boolean keepConnected=true;

public void run()

{

while(keepConnected==true)//持续接收从服务器端传来的信息

{

try

{

String str=in.readLine();//读入一行信息

output.append(str+"\n");//放在output中

}catch(Exception e)

{

keepConnected=false;

System.out.println(e);

}

}

}

}

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

{

ChatClient chatClient=new ChatClient(jtf1.getText());

chatClient.displayFrame();

}\*/

}