《程序设计实践报告（Java）》

- 题　　目: 基于 Java 的即使通讯系统的开发
- 学　　院：计算机学院
- 专　　业：计算机大类
- 学生姓名：付尧
- 班　　级：计类 1506 班
- 学　　号：2015011297
- 指导老师：秦奕青

目　录

摘　要 I
1　引言 1
2　设计任务与目的 1
3　需求分析 1
3.1　系统用例图 1
3.2　主要用例分析 1
4　系统设计 3
4.1　总体设计 3
4.1.1　系统类图 3
4.2　详细设计 4
4.2.1　ChatServer 类 5
4.2.2　ServerListen 类 5
4.2.3　ServerReceive 类 6
4.2.4　PortConf 类 6
4.2.5　Help 类 7
4.2.6　UserLinkList 类 7
4.2.7　Node 类 8
4.2.8　ChatClient 类 8
4.2.9　ConnectConf 类 9
4.2.10　UserConf 类 9
4.2.11　Help 类 10
4.2.12　ClientReceive 类 10
5　系统实现 11
5.1　系统主要用例的实现 11
5.1.1　聊天室服务端用例运行效果 11
5.1.2　聊天室客户端用例运行效果 12
5.2　运行方法 12
6　结束语 12
附录 A：系统源程序 13
1、ChatServer 类 13
2、ServerListen 类 24
3、ServerReceive 类 26
4、UserLinkList 类 30
5、Node 类 33
6、PortConf 类 33
7、Help 类 36
8、ChatClient 类 38
9、ClientReceive 类 50
10、ConnectConf 类 52
11、UserConf 类 55
12、Help 类 58

摘　要

这次程序设计我通过局域网即时通讯系统的代码实现，体会到将理论知识与具体实现相结合，巩固 Java 相关方法。聊天室共分为服务器端和客户端两部分，服务器端程序主要负责侦听客户端发来的消息，客户端需登陆到服务器才可以实现正常的聊天功能。本聊天系统以聊天交流为主，实现局域网内各个用户间的通讯，相比互联网实现更加安全和简便。

关键词：

聊天室；多线程；C/S 模式；GUI 编程

JavaChat

1 引言

随着互联网逐步普及，人们的生活和工作也越来越离不开信息网络的支持，而聊天是人们最常见，最直接的网上交流的方式。本聊天系统以聊天交流为主，为局域网内用户提供即时通讯的服务。本文所介绍的网络聊天系统是基于开源的 JAVA 应用程序开发设计的，其主要特性是能动态、实时的完成信息的传递，且具有交互性，能有效地处理客户请求，且具有脱离数据库的技术方法、易于维护和更新的特点。

2 设计任务与目的

本聊天系统基于 C/S 模式，聊天室共分为服务器端和客户端两部分，服务器端程序主要负责侦听客户端发来的消息，客户端需登陆到服务器才可以实现正常的聊天功能。

通过本项目的设计实践，力求熟练掌握 GUI 程序设计、多线程技术、基于 TCP 的 Socket 通信编程。

3 需求分析

本系统所要实现的主要功能是局域网中的即时通讯。因为服务器端采用多线程，所以性能有很大提升。因为服务端和客户端之间采用 TCP 协议进行网络通信，建立的是可靠的、端到端的网络连接，所以可靠性和安全性都很好。

3.1 系统用例图

图 3-1　系统用例图

3.2 主要用例分析

服务器端的主要功能如下：

1，在特定端口上进行侦听，等待客户端连接

2，用户可以配置服务端的侦听端口，默认端口为 8888.

3，向已经连接到服务端的用户发送系统消息。

4，统计在线人数

5，当停止服务时，断开所有的用户连接。

客户端的主要功能如下：

1，连接到已经开启的聊天服务的服务端。

2，用户可以配置要连接服务器端的 IP 地址和端口号。

3，用户可以配置连接后显示的用户名。

4，当服务器端开启的时候，用户可以随时登录和注销。

5，用户可以向所有人或者某一个人发送消息。

4 系统设计

4.1 总体设计

4.1.1 系统类图

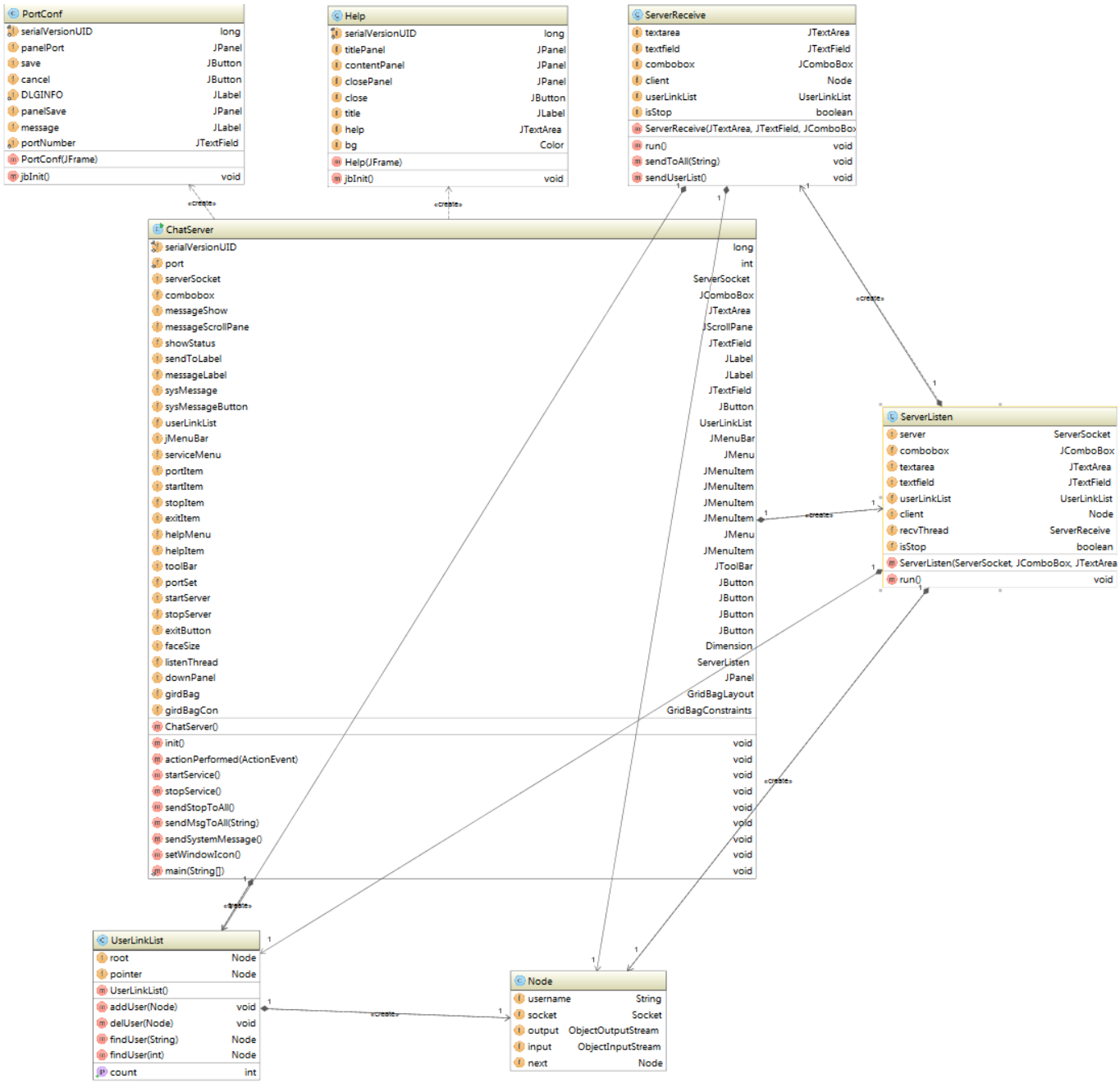

图 4-1-1 服务端系统类图

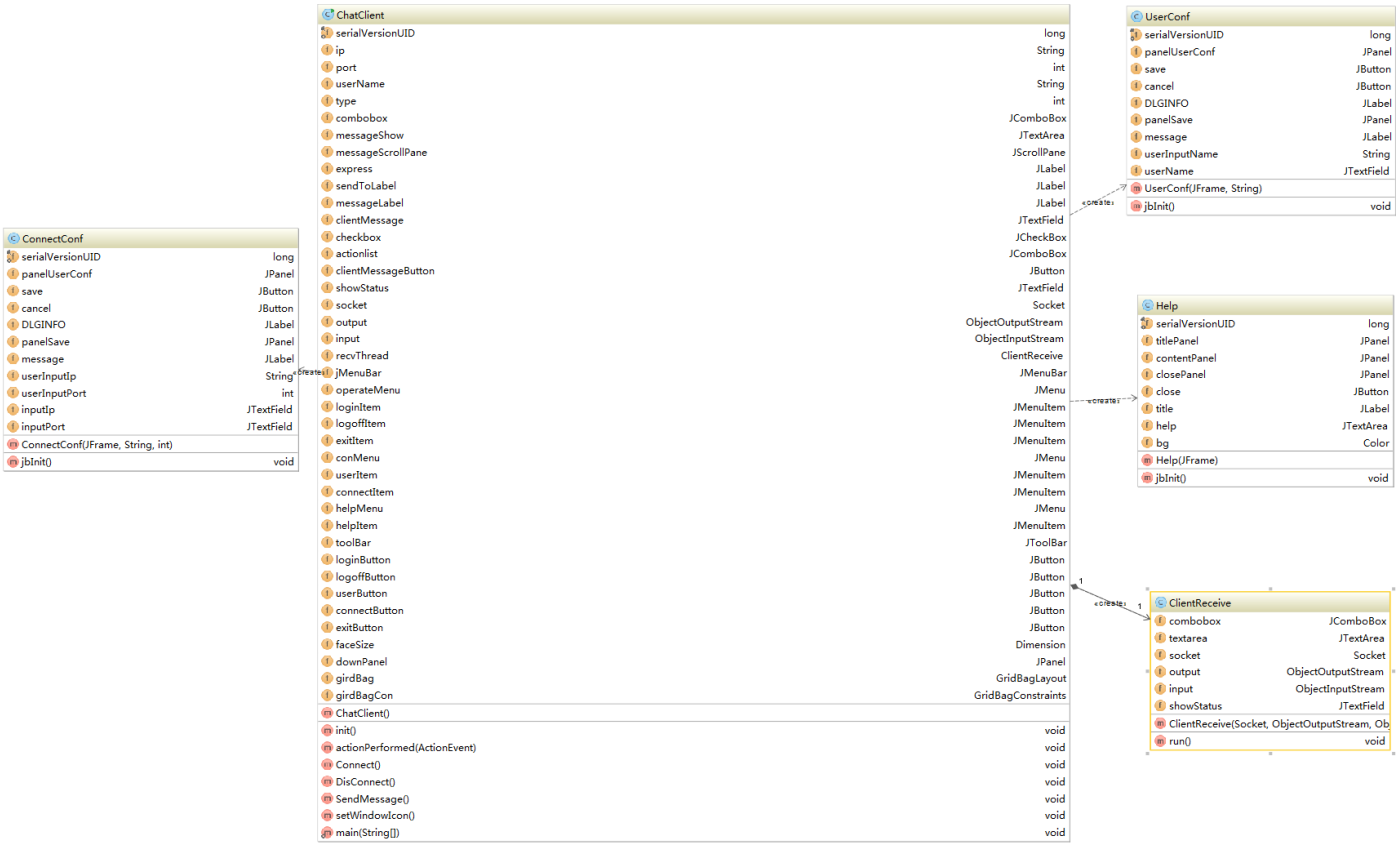

图 4-1-2　客户端系统类图

4.2　详细设计

聊天室服务器端的设计

聊天室服务端主要包括 7 个类，放在 server 包中：

4.2.1 ChatServer 类

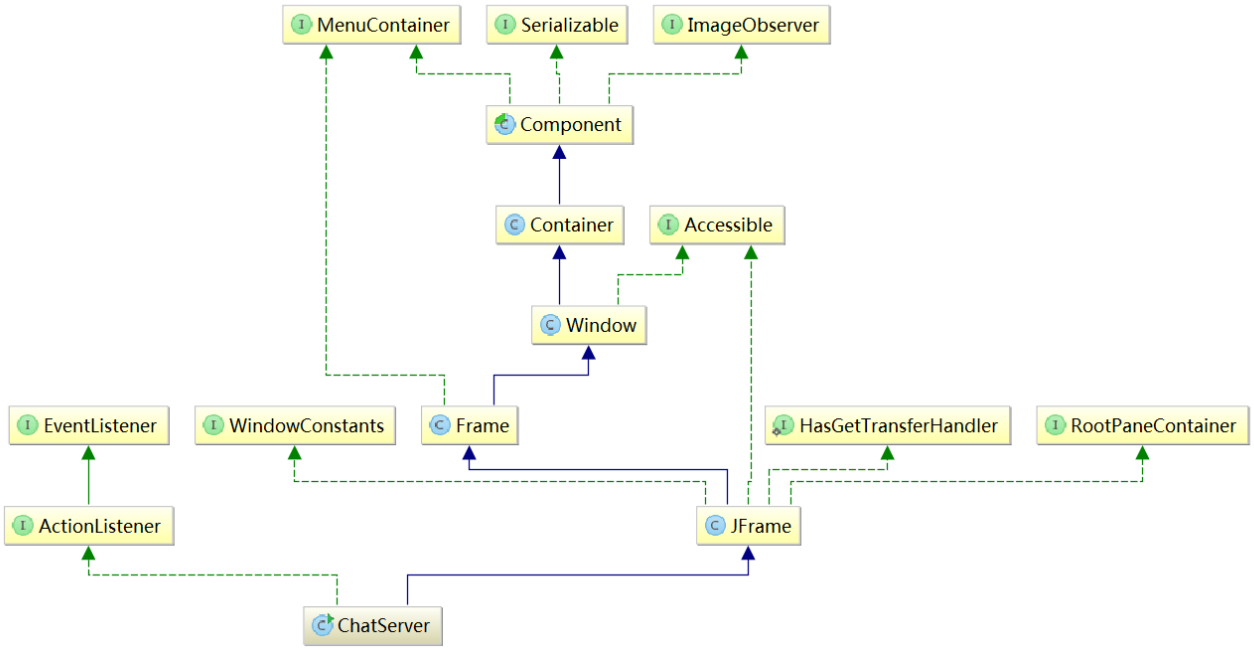

图 4-3　ChatServer 类图

ChatServer.java

包含名为 ChatServer 的 public 类，其主要功能为定义服务器端的界面，添加时间监听与时间处理。调用 ServerListen 类来实现服务端用户上线与下线的监听，调用 ServerReceive 类来实现服务器端的消息收发。

4.2.2 ServerListen 类

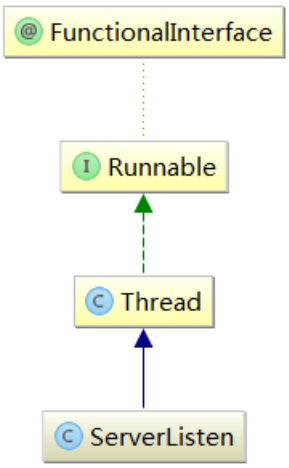

图 4-4　ServerListen 类图

ServerListen.java

该类实现服务器用户上线与下线的监听。该类对用户上线下线的监听是通过调用用户链表类（UserLinkList）来实现的。当用户上线与下线情况发生变化时，该类会对主类的界面进行相应的修改。

4.2.3 ServerReceive 类

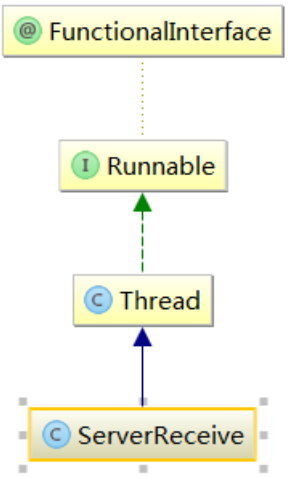

图 4-5 ServerReceive 类图

ServerReceive.java

该类是实现服务器消息收发的类，该类分别定义了向某用户及所有人发送消息的方法，发送的消息会显示在主界面类的界面上。

4.2.4 PortConf 类

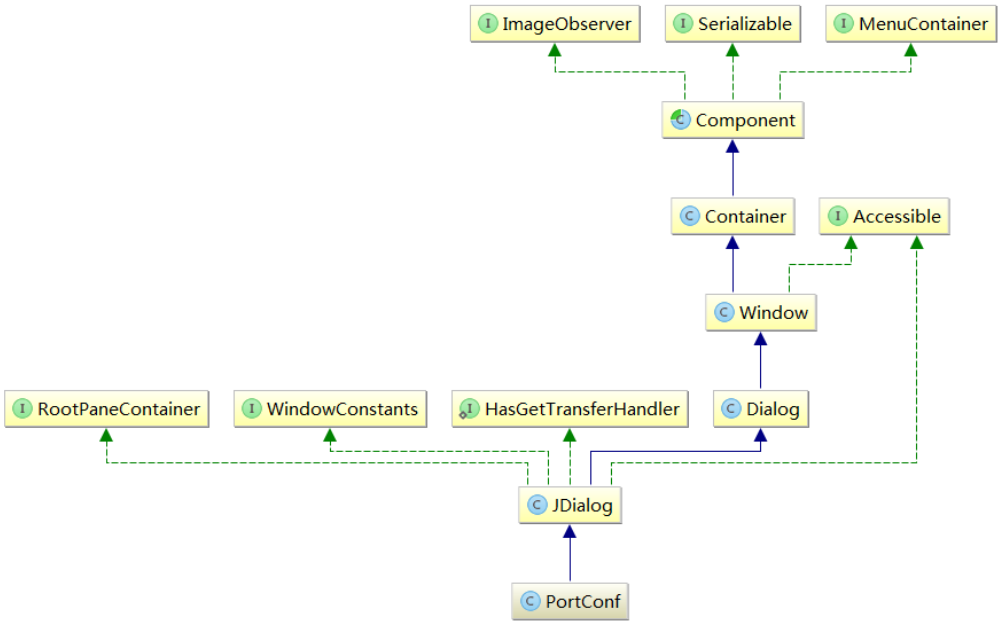

图 4-6 PortConf 类图

PortConf.java

该类继承自 Jdialog，是用户对服务器端监听端口进行修改配置的类。

4.2.5 Help 类

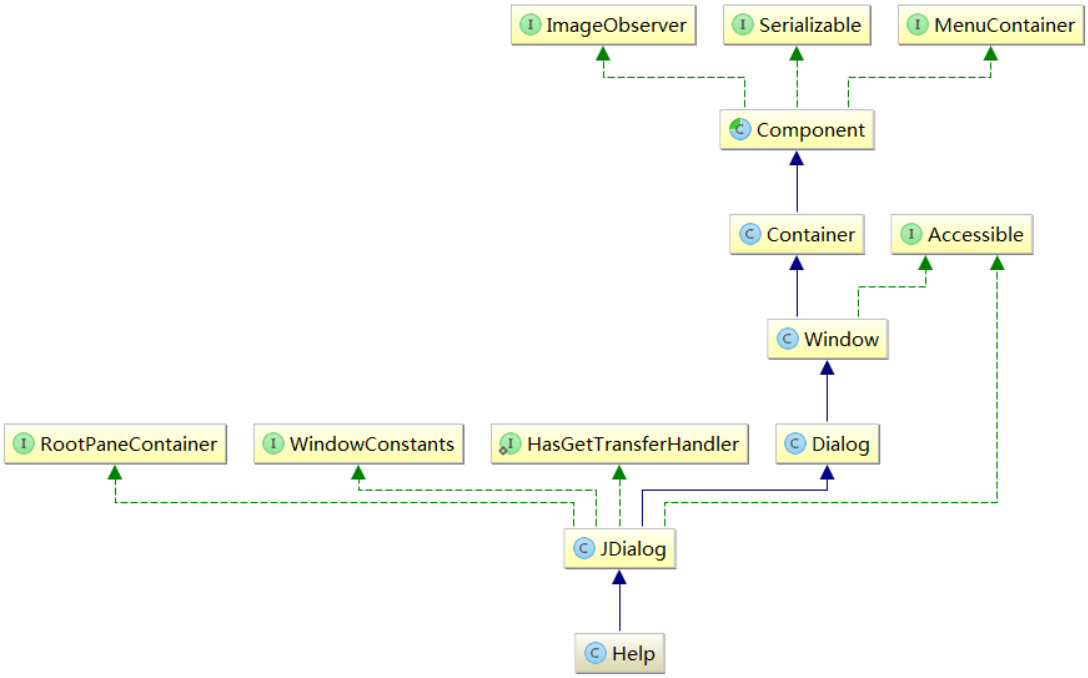

图 4-7 Help 类图

Help.java

服务端程序帮助类。

4.2.6 UserLinkList 类

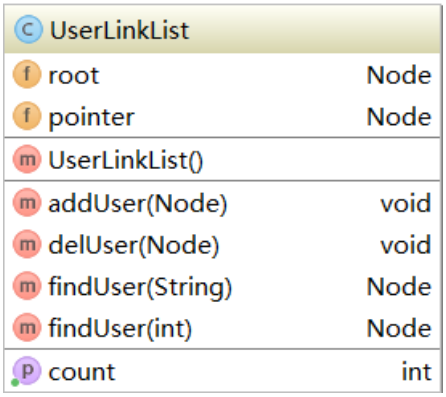

图 4-8 UserLinkList 类图

UserLinkList.java

用户链表节点的具体实现类。该类通过构造函数构造用户链表，定义了添加用户、删除用户、返回用户数、根据用户名查找用户、根据 id 查找用户这 5 个方法。

4.2.7 Node 类

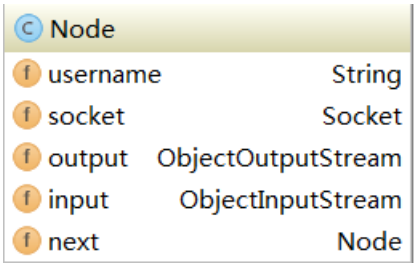

图 4-9 Node 类图

Node.java

用户链表的节点类，定义了链表中的用户。该类与前面所讲的链表节点 Node 类的功能相当。

聊天室客户端设计

聊天室客户端主要包括 5 个类，放在 client 包中：

4.2.8 ChatClient 类

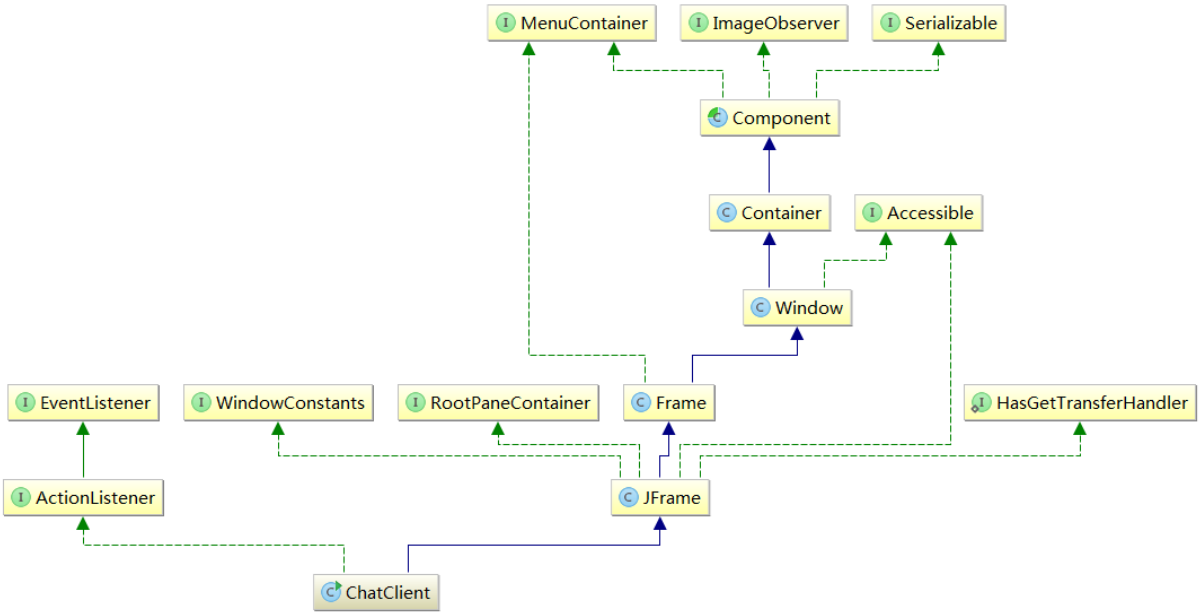

图 4-10 ChatClient 类图

ChatClient.java

包含名为 ChatClient 的 public 类，其主要功能为定义客户端的界面，添加时间监听与事件处理。该类定义了 Connect（）与 DisConnect（）方法实现与客户端的连接与断开连接。当登陆到指定的服务器时，调用 ClientReceive 类实现消息收发，同时该类还定义了 SendMessage（）方法来向其他用户发送带有表情的消息或悄悄话。

4.2.9 ConnectConf 类

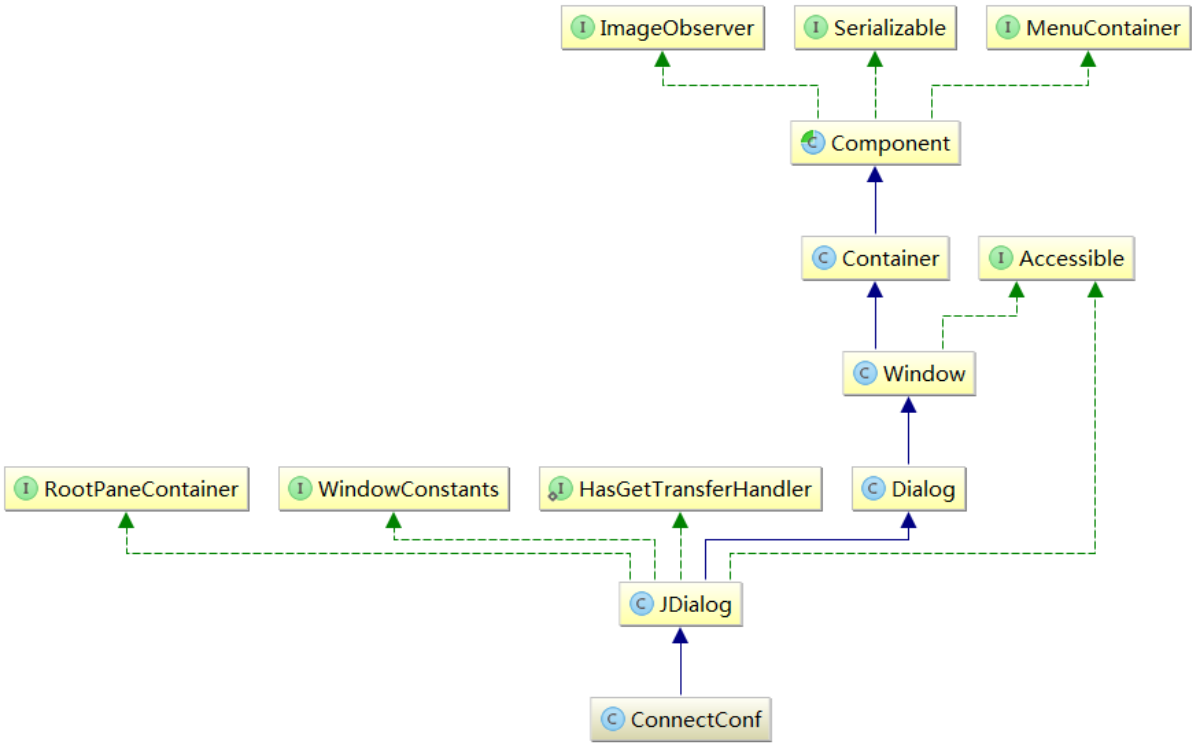

图 4-11　ConnectConf 类图

ConnectConf.java

该类继承自 Jdialog，是用户对所有要连接的服务器 IP 及监听端口进行修改配置的类。

4.2.10 UserConf 类

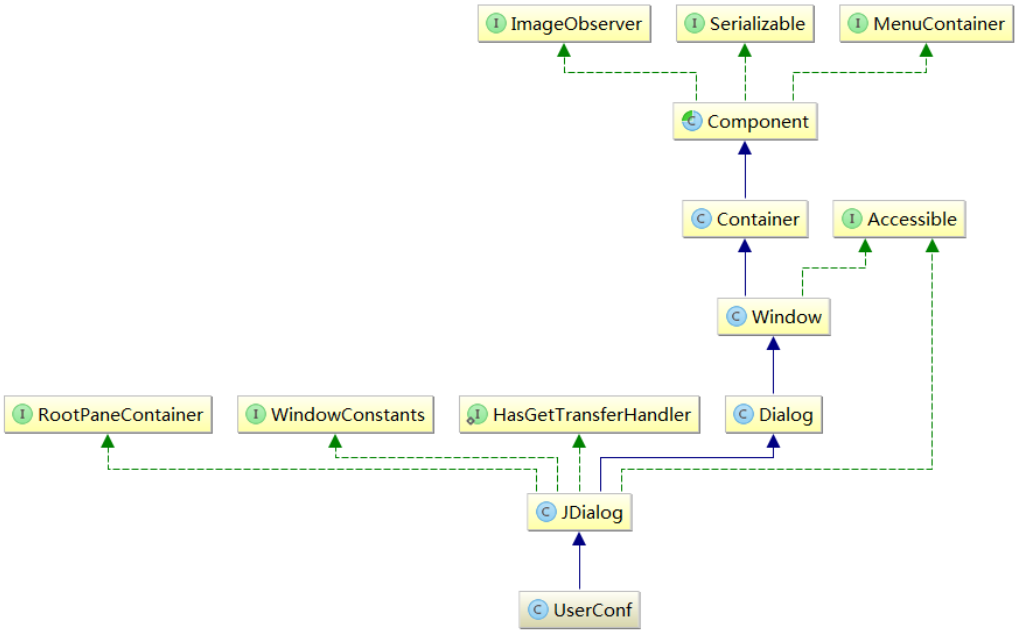

图 4-12　UserConf 类图

UserConf.java

该类继承自 Jdialog，是用户对链接到服务器时所显示的用户名进行修改配置的类。

4.2.11 Help 类

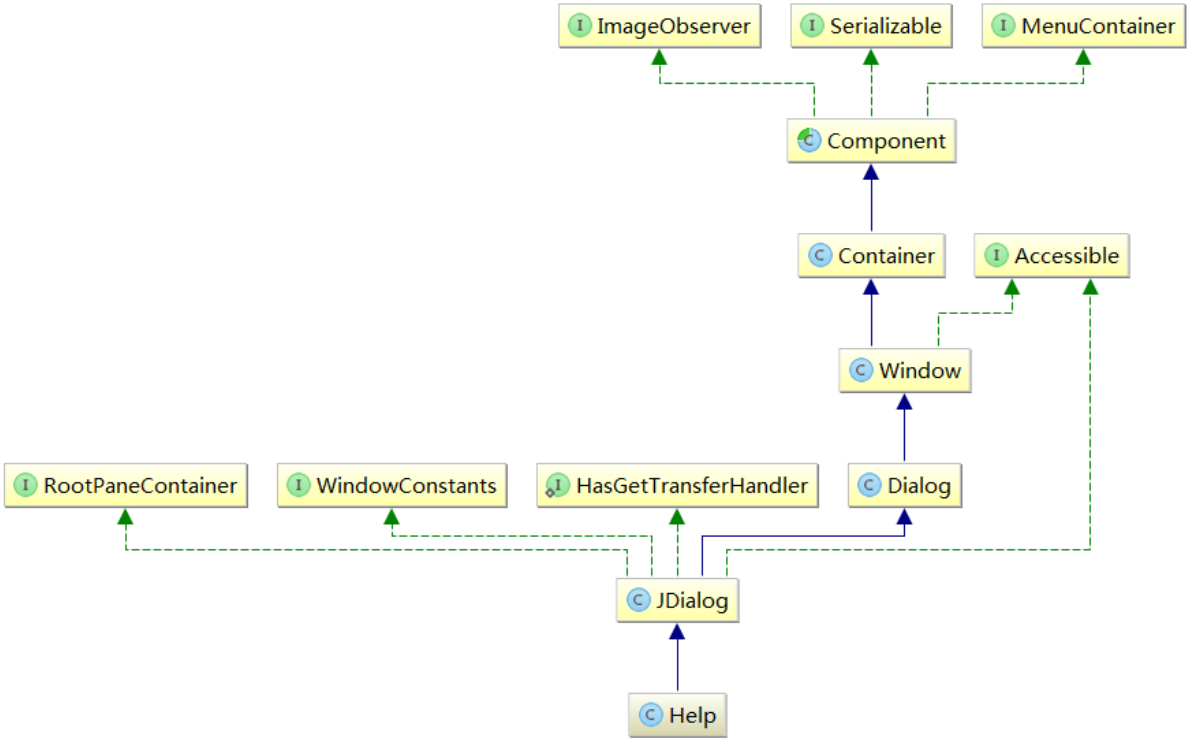

图 4-13 Help 类图

Help.java

客户端程序的帮助类。

4.2.12 ClientReceive 类

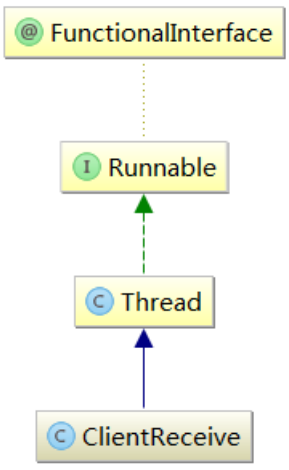

图 4-14 ClientReceive 类图

ClientReceive.java

该类是实现服务器端与客户端消息收发的类。

5 系统实现

5.1 系统主要用例的实现

5.1.1 聊天室服务端用例运行效果

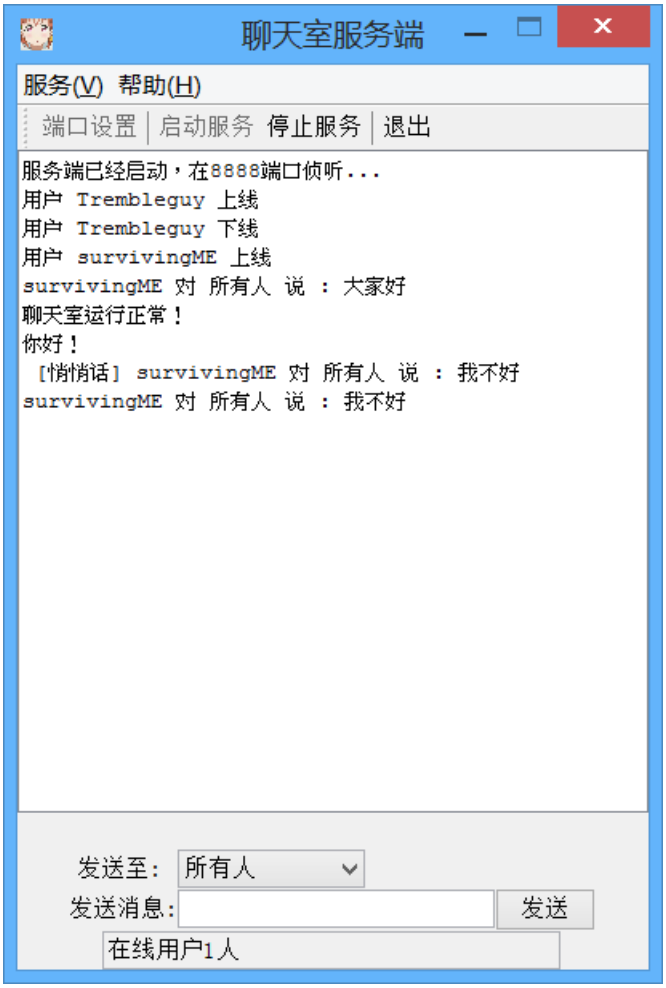

图 5-1 聊天室服务端用例运行效果图

5.1.2 聊天室客户端用例运行效果

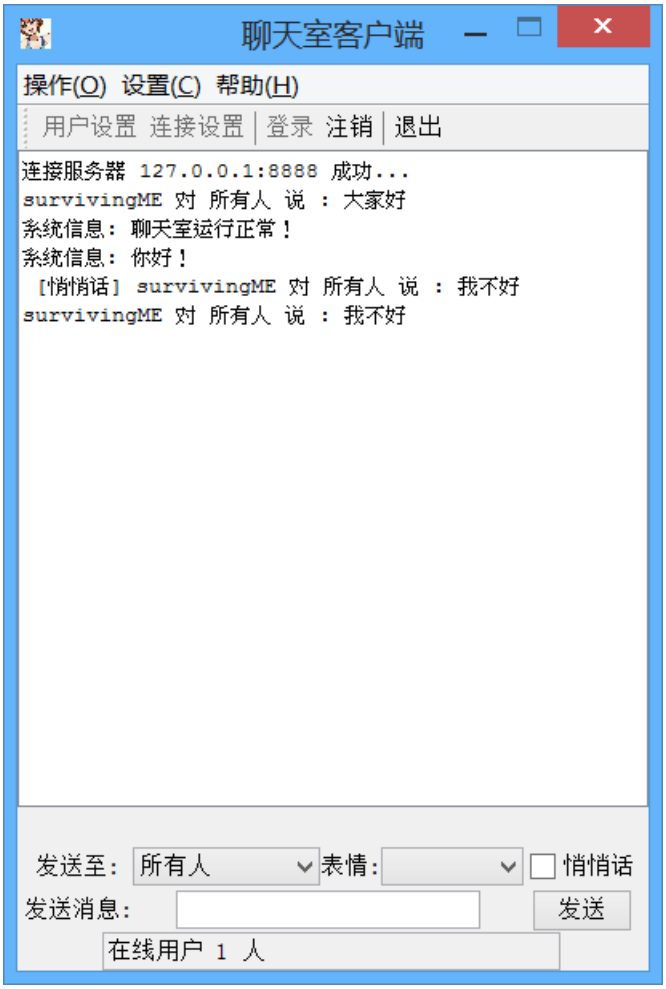

图 5-2 聊天室客户端用例运行效果图

5.2 运行方法

首先运行 ChatServer 的 main 函数，启动服务端侦听服务；再运行 ChatClient 的 main 函数，点击登录按钮后即可发送消息给在线用户/服务器端

6 结束语

Java是一门应用广泛并且实用性极强的高级程序设计语言。通过本次课程设计，我对面向对象的基本思想有了进一步的理解，对项目的各个阶段的任务也有了一定的了解。设计开始阶段必须明确设计的目的与需求分析，总体设计要全面分析聊天系统的架构。通过这次设计，我认识到体系的架构是最重要的，只有准确的系统设计、很好的定义各个模块及模块之间的关系，才能在编码阶段更轻松的的实现。

这次课程设计，对我的逻辑思维能力是一个很大的锻炼，它不仅加强了我的系统思

考问题的能力，而且还改变了我以前不好的编程习惯，虽然花费了不少的时间，但是我学到了丰富的知识。

这次程序设计也是一个毅力的考验过程。有时候往往只是一个小小的错误，却要花出几小时甚至是一两天的时间才可能发现它。所以在这个过程不能过于急躁，要很有耐心将程序反复调试，

附录 A：系统源程序

1、ChatServer 类

功能：为定义服务器端的界面，添加时间监听与时间处理。调用 ServerListen 类来实现服务端用户上线与下线的监听，调用 ServerListen 来实现服务器端的消息收发。

代码：

```
package server;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.net.*;

/*
 * 聊天服务端的主框架类
 */
public class ChatServer extends JFrame implements ActionListener{

    /**
     *
     */
    private static final long serialVersionUID = 1L;

    public static int port = 8888;//服务端的侦听端口

    ServerSocket serverSocket;//服务端 Socket
    JComboBox combobox;//选择发送消息的接受者
    JTextArea messageShow;//服务端的信息显示
    JScrollPane messageScrollPane;//信息显示的滚动条
    JTextField showStatus;//显示用户连接状态
    JLabel sendToLabel,messageLabel;
    JTextField sysMessage;//服务端消息的发送
    JButton sysMessageButton;//服务端消息的发送按钮
```

```
UserLinkList userLinkList;//用户链表

//建立菜单栏
JMenuBar jMenuBar = new JMenuBar();
//建立菜单组
JMenu serviceMenu = new JMenu ("服务(V)");
//建立菜单项
JMenuItem portItem = new JMenuItem ("端口设置(P)");
JMenuItem startItem = new JMenuItem ("启动服务(S)");
JMenuItem stopItem=new JMenuItem ("停止服务(T)");
JMenuItem exitItem=new JMenuItem ("退出(X)");

JMenu helpMenu=new JMenu ("帮助(H)");
JMenuItem helpItem=new JMenuItem ("帮助(H)");

//建立工具栏
JToolBar toolBar = new JToolBar();

//建立工具栏中的按钮组件
JButton portSet;//启动服务端侦听
JButton startServer;//启动服务端侦听
JButton stopServer;//关闭服务端侦听
JButton exitButton;//退出按钮

//框架的大小
Dimension faceSize = new Dimension(400, 600);

ServerListen listenThread;

JPanel downPanel ;
GridBagLayout girdBag;
GridBagConstraints girdBagCon;


/**
 * 服务端构造函数
 */
public ChatServer(){
    init();//初始化程序

    //添加框架的关闭事件处理
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.pack();
    //设置框架的大小
```

```
        this.setSize(faceSize);
        //设置程序图标
        this.setWindowIcon();

        //设置运行时窗口的位置
        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        this.setLocation( (int) (screenSize.width - faceSize.getWidth())
/ 2,
                (int) (screenSize.height - faceSize.getHeight()) / 2);
        this.setResizable(false);

        this.setTitle("聊天室服务端"); //设置标题

        setVisible(true);

        //为服务菜单栏设置热键'V'
        serviceMenu.setMnemonic('V');

        //为端口设置快捷键为ctrl+p
        portItem.setMnemonic ('P');
        portItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_P,InputEvent.CTRL_MASK));

        //为启动服务快捷键为ctrl+s
        startItem.setMnemonic ('S');
        startItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_S,InputEvent.CTRL_MASK));

        //为端口设置快捷键为ctrl+T
        stopItem.setMnemonic ('T');
        stopItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_T,InputEvent.CTRL_MASK));

        //为退出设置快捷键为ctrl+x
        exitItem.setMnemonic ('X');
        exitItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_X,InputEvent.CTRL_MASK));

        //为帮助菜单栏设置热键'H'
        helpMenu.setMnemonic('H');

        //为帮助设置快捷键为ctrl+p
        helpItem.setMnemonic ('H');
```

```
        helpItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_H,InputEvent.CTRL_MASK));

    }

    /**
     * 程序初始化函数
     */
    public void init(){

        Container contentPane = getContentPane();
        contentPane.setLayout(new BorderLayout());

        //添加菜单栏
        serviceMenu.add (portItem);
        serviceMenu.add (startItem);
        serviceMenu.add (stopItem);
        serviceMenu.add (exitItem);
        jMenuBar.add (serviceMenu);
        helpMenu.add (helpItem);
        jMenuBar.add (helpMenu);
        setJMenuBar (jMenuBar);

        //初始化按钮
        portSet = new JButton("端口设置");
        startServer = new JButton("启动服务");
        stopServer = new JButton("停止服务" );
        exitButton = new JButton("退出" );
        //将按钮添加到工具栏
        toolBar.add(portSet);
        toolBar.addSeparator();//添加分隔栏
        toolBar.add(startServer);
        toolBar.add(stopServer);
        toolBar.addSeparator();//添加分隔栏
        toolBar.add(exitButton);
        contentPane.add(toolBar,BorderLayout.NORTH);

        //初始时，令停止服务按钮不可用
        stopServer.setEnabled(false);
        stopItem .setEnabled(false);

        //为菜单栏添加事件监听
        portItem.addActionListener(this);
        startItem.addActionListener(this);
```

```
stopItem.addActionListener(this);
exitItem.addActionListener(this);
helpItem.addActionListener(this);

//添加按钮的事件侦听
portSet.addActionListener(this);
startServer.addActionListener(this);
stopServer.addActionListener(this);
exitButton.addActionListener(this);

combobox = new JComboBox();
combobox.insertItemAt("所有人",0);
combobox.setSelectedIndex(0);

messageShow = new JTextArea();
messageShow.setEditable(false);
//添加滚动条
messageScrollPane = new JScrollPane(messageShow,
        JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
        JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
messageScrollPane.setPreferredSize(new Dimension(400,400));
messageScrollPane.revalidate();

showStatus = new JTextField(35);
showStatus.setEditable(false);

sysMessage = new JTextField(24);
sysMessage.setEnabled(false);
sysMessageButton = new JButton();
sysMessageButton.setText("发送");

//添加系统消息的事件侦听
sysMessage.addActionListener(this);
sysMessageButton.addActionListener(this);

sendToLabel = new JLabel("发送至:");
messageLabel = new JLabel("发送消息:");
downPanel = new JPanel();
girdBag = new GridBagLayout();
downPanel.setLayout(girdBag);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 0;
girdBagCon.gridy = 0;
```

```
girdBagCon.gridwidth = 3;
girdBagCon.gridheight = 2;
girdBagCon.ipadx = 5;
girdBagCon.ipady = 5;
JLabel none = new JLabel("    ");
girdBag.setConstraints(none,girdBagCon);
downPanel.add(none);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 0;
girdBagCon.gridy = 2;
girdBagCon.insets = new Insets(1,0,0,0);
girdBagCon.ipadx = 5;
girdBagCon.ipady = 5;
girdBag.setConstraints(sendToLabel,girdBagCon);
downPanel.add(sendToLabel);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx =1;
girdBagCon.gridy = 2;
girdBagCon.anchor = GridBagConstraints.LINE_START;
girdBag.setConstraints(combobox,girdBagCon);
downPanel.add(combobox);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 0;
girdBagCon.gridy = 3;
girdBag.setConstraints(messageLabel,girdBagCon);
downPanel.add(messageLabel);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 1;
girdBagCon.gridy = 3;
girdBag.setConstraints(sysMessage,girdBagCon);
downPanel.add(sysMessage);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 2;
girdBagCon.gridy = 3;
girdBag.setConstraints(sysMessageButton,girdBagCon);
downPanel.add(sysMessageButton);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 0;
```

```
        girdBagCon.gridy = 4;
        girdBagCon.gridwidth = 3;
        girdBag.setConstraints(showStatus,girdBagCon);
        downPanel.add(showStatus);

        contentPane.add(messageScrollPane,BorderLayout.CENTER);
        contentPane.add(downPanel,BorderLayout.SOUTH);

        //关闭程序时的操作
        this.addWindowListener(
                new WindowAdapter(){
                    public void windowClosing(WindowEvent e){
                        stopService();
                        System.exit(0);
                    }
                }
        );
    }

    /**
     * 事件处理
     */
    public void actionPerformed(ActionEvent e) {
        Object obj = e.getSource();
        if (obj == startServer || obj == startItem) { //启动服务端
            startService();
        }
        else if (obj == stopServer || obj == stopItem) { //停止服务端
            int j=JOptionPane.showConfirmDialog(
                    this,"真的停止服务吗?","停止服务",

JOptionPane.YES_OPTION,JOptionPane.QUESTION_MESSAGE);

            if (j == JOptionPane.YES_OPTION){
                stopService();
            }
        }
        else if (obj == portSet || obj == portItem) { //端口设置
            //调出端口设置的对话框
            PortConf portConf = new PortConf(this);
            portConf.setVisible(true);
        }
        else if (obj == exitButton || obj == exitItem) { //退出程序
            int j=JOptionPane.showConfirmDialog(
```

```
                    this,"真的要退出吗?","退出",
JOptionPane.YES_OPTION,JOptionPane.QUESTION_MESSAGE);

            if (j == JOptionPane.YES_OPTION){
                stopService();
                System.exit(0);
            }
        }
        else if (obj == helpItem) { //菜单栏中的帮助
            //调出帮助对话框
            Help helpDialog = new Help(this);
            helpDialog.setVisible(true);
        }
        else if (obj == sysMessage || obj == sysMessageButton) { //发
送系统消息
            sendSystemMessage();
        }
    }

    /**
     * 启动服务端
     */
    public void startService(){
        try{
            serverSocket = new ServerSocket(port,10);
            messageShow.append("服务端已经启动，在"+port+"端口侦
听...\n");

            startServer.setEnabled(false);
            startItem.setEnabled(false);
            portSet.setEnabled(false);
            portItem.setEnabled(false);

            stopServer .setEnabled(true);
            stopItem .setEnabled(true);
            sysMessage.setEnabled(true);
        }
        catch (Exception e){
            //System.out.println(e);
        }
        userLinkList = new UserLinkList();

        listenThread = new ServerListen(serverSocket,combobox,
```

```
            messageShow,showStatus,userLinkList);
    listenThread.start();
}

/**
 * 关闭服务端
 */
public void stopService(){
    try{
        //向所有人发送服务器关闭的消息
        sendStopToAll();
        listenThread.isStop = true;
        serverSocket.close();

        int count = userLinkList.getCount();

        int i =0;
        while( i < count){
            Node node = userLinkList.findUser(i);

            node.input .close();
            node.output.close();
            node.socket.close();

            i ++;
        }

        stopServer .setEnabled(false);
        stopItem .setEnabled(false);
        startServer.setEnabled(true);
        startItem.setEnabled(true);
        portSet.setEnabled(true);
        portItem.setEnabled(true);
        sysMessage.setEnabled(false);

        messageShow.append("服务端已经关闭\n");

        combobox.removeAllItems();
        combobox.addItem("所有人");
    }
    catch(Exception e){
        //System.out.println(e);
    }
}
```

```
/**
 * 向所有人发送服务器关闭的消息
 */
public void sendStopToAll(){
    int count = userLinkList.getCount();

    int i = 0;
    while(i < count){
        Node node = userLinkList.findUser(i);
        if(node == null) {
            i ++;
            continue;
        }

        try{
            node.output.writeObject("服务关闭");
            node.output.flush();
        }
        catch (Exception e){
            //System.out.println("$$$"+e);
        }

        i++;
    }
}

/**
 * 向所有人发送消息
 */
public void sendMsgToAll(String msg){
    int count = userLinkList.getCount();//用户总数

    int i = 0;
    while(i < count){
        Node node = userLinkList.findUser(i);
        if(node == null) {
            i ++;
            continue;
        }

        try{
            node.output.writeObject("系统信息");
            node.output.flush();
```

```
                node.output.writeObject(msg);
                node.output.flush();
            }
            catch (Exception e){
                //System.out.println("@@@"+e);
            }

            i++;
        }

        sysMessage.setText("");
    }

    /**
     * 向客户端用户发送消息
     */
    public void sendSystemMessage(){
        String toSomebody = combobox.getSelectedItem().toString();
        String message = sysMessage.getText() + "\n";

        messageShow.append(message);

        //向所有人发送消息
        if(toSomebody.equalsIgnoreCase("所有人")){
            sendMsgToAll(message);
        }
        else{
            //向某个用户发送消息
            Node node = userLinkList.findUser(toSomebody);

            try{
                node.output.writeObject("系统信息");
                node.output.flush();
                node.output.writeObject(message);
                node.output.flush();
            }
            catch(Exception e){
                //System.out.println("!!!"+e);
            }
            sysMessage.setText("");//将发送消息栏的消息清空
        }
    }
    public void setWindowIcon()
    {
```

```
        ImageIcon imageIcon = new
ImageIcon(getClass().getResource("/img/servericon.png"));
        this.setIconImage(imageIcon.getImage());
    }

    public static void main(String[] args) {
        try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InstantiationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (UnsupportedLookAndFeelException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        new ChatServer();
    }
}
```

2、ServerListen 类

功能：实现服务器用户上线与下线的监听。该类对用户上线下线的监听是通过调用用户链表类（UserLinkList）来实现的。当用户上线与下线情况发生变化时，该类会对主类的界面进行相应的修改。

代码：

```
package server;

import javax.swing.*;
import java.io.*;
import java.net.*;

/*
 * 服务端的侦听类
 */
public class ServerListen extends Thread {
    ServerSocket server;
```

```
    JComboBox combobox;
    JTextArea textarea;
    JTextField textfield;
    UserLinkList userLinkList;//用户链表

    Node client;
    ServerReceive recvThread;

    public boolean isStop;

    /*
     * 聊天服务端的用户上线于下线侦听类
     */
    public ServerListen(ServerSocket server,JComboBox combobox,
                        JTextArea textarea,JTextField
textfield,UserLinkList userLinkList){

        this.server = server;
        this.combobox = combobox;
        this.textarea = textarea;
        this.textfield = textfield;
        this.userLinkList = userLinkList;

        isStop = false;
    }

    public void run(){
        while(!isStop && !server.isClosed()){
            try{
                client = new Node();
                client.socket = server.accept();
                client.output = new
ObjectOutputStream(client.socket.getOutputStream());
                client.output.flush();
                client.input  = new
ObjectInputStream(client.socket.getInputStream());
                client.username = (String)client.input.readObject();

                //显示提示信息
                combobox.addItem(client.username);
                userLinkList.addUser(client);
                textarea.append("用户 " + client.username + " 上线" +
"\n");
                textfield.setText("在线用户" + userLinkList.getCount() +
```

```
"人\n");

                recvThread = new ServerReceive(textarea,textfield,
                        combobox,client,userLinkList);
                recvThread.start();
            }
            catch(Exception e){
            }
        }
    }
}
```

3、ServerReceive 类

功能：实现服务器消息收发的类，该类分别定义了向某用户及所有人发送消息的方法，发送的消息会显示在主界面类的街面上。

代码：

```
package server;

import javax.swing.*;

/*
 * 服务器收发消息的类
 */
public class ServerReceive extends Thread {
    JTextArea textarea;
    JTextField textfield;
    JComboBox combobox;
    Node client;
    UserLinkList userLinkList;//用户链表

    public boolean isStop;

    public ServerReceive(JTextArea textarea,JTextField textfield,
                         JComboBox combobox,Node client,UserLinkList
userLinkList){

        this.textarea = textarea;
        this.textfield = textfield;
        this.client = client;
        this.userLinkList = userLinkList;
        this.combobox = combobox;

        isStop = false;
```

```
    }

    public void run(){
        //向所有人发送用户的列表
        sendUserList();

        while(!isStop && !client.socket.isClosed()){
            try{
                String type = (String)client.input.readObject();

                if(type.equalsIgnoreCase("聊天信息")){
                    String toSomebody =
(String)client.input.readObject();
                    String status  = (String)client.input.readObject();
                    String action  = (String)client.input.readObject();
                    String message = (String)client.input.readObject();

                    String msg = client.username
                            +" "+ action
                            + "对 "
                            + toSomebody
                            + " 说 : "
                            + message
                            + "\n";
                    if(status.equalsIgnoreCase("悄悄话")){
                        msg = " [悄悄话] " + msg;
                    }
                    textarea.append(msg);

                    if(toSomebody.equalsIgnoreCase("所有人")){
                        sendToAll(msg);//向所有人发送消息
                    }
                    else{
                        try{
                            client.output.writeObject("聊天信息");
                            client.output.flush();
                            client.output.writeObject(msg);
                            client.output.flush();
                        }
                        catch (Exception e){
                            //System.out.println("###"+e);
                        }

                        Node node = userLinkList.findUser(toSomebody);
```

```
                    if(node != null){
                        node.output.writeObject("聊天信息");
                        node.output.flush();
                        node.output.writeObject(msg);
                        node.output.flush();
                    }
                }
            }
            else if(type.equalsIgnoreCase("用户下线")){
                Node node = userLinkList.findUser(client.username);
                userLinkList.delUser(node);

                String msg = "用户 " + client.username + " 下线\n";
                int count = userLinkList.getCount();

                combobox.removeAllItems();
                combobox.addItem("所有人");
                int i = 0;
                while(i < count){
                    node = userLinkList.findUser(i);
                    if(node == null) {
                        i ++;
                        continue;
                    }

                    combobox.addItem(node.username);
                    i++;
                }
                combobox.setSelectedIndex(0);

                textarea.append(msg);
                textfield.setText("在线用户" +
userLinkList.getCount() + "人\n");

                sendToAll(msg);//向所有人发送消息
                sendUserList();//重新发送用户列表,刷新

                break;
            }
        }
        catch (Exception e){
            //System.out.println(e);
        }
```

```
    }
}

/*
 * 向所有人发送消息
 */
public void sendToAll(String msg){
    int count = userLinkList.getCount();

    int i = 0;
    while(i < count){
        Node node = userLinkList.findUser(i);
        if(node == null) {
            i ++;
            continue;
        }

        try{
            node.output.writeObject("聊天信息");
            node.output.flush();
            node.output.writeObject(msg);
            node.output.flush();
        }
        catch (Exception e){
            //System.out.println(e);
        }

        i++;
    }
}

/*
 * 向所有人发送用户的列表
 */
public void sendUserList(){
    String userlist = "";
    int count = userLinkList.getCount();

    int i = 0;
    while(i < count){
        Node node = userLinkList.findUser(i);
        if(node == null) {
            i ++;
            continue;
```

```
            }

            userlist += node.username;
            userlist += '\n';
            i++;
        }

        i = 0;
        while(i < count){
            Node node = userLinkList.findUser(i);
            if(node == null) {
                i ++;
                continue;
            }

            try{
                node.output.writeObject("用户列表");
                node.output.flush();
                node.output.writeObject(userlist);
                node.output.flush();
            }
            catch (Exception e){
                //System.out.println(e);
            }
            i++;
        }
    }
}
```

4、UserLinkList 类

功能：用户链表节点的具体实现类。该类通过构造函数构造用户链表，定义了添加用户、删除用户、返回用户数、根据用户名查找用户、根据 id 查找用户这 5 个方法。

代码：

```
package server;

/**
 * 用户链表
 */
public class UserLinkList {
    Node root;
    Node pointer;
    int count;
```

```
/**
 * 构造用户链表
 */
public UserLinkList(){
    root = new Node();
    root.next = null;
    pointer = null;
    count = 0;
}

/**
 * 添加用户
 */
public void addUser(Node n){
    pointer = root;

    while(pointer.next != null){
        pointer = pointer.next;
    }

    pointer.next = n;
    n.next = null;
    count++;

}

/**
 * 删除用户
 */
public void delUser(Node n){
    pointer = root;

    while(pointer.next != null){
        if(pointer.next == n){
            pointer.next = n.next;
            count --;

            break;
        }

        pointer = pointer.next;
    }
}
```

```java
/**
 * 返回用户数
 */
public int getCount(){
    return count;
}

/**
 * 根据用户名查找用户
 */
public Node findUser(String username){
    if(count == 0) return null;

    pointer = root;

    while(pointer.next != null){
        pointer = pointer.next;

        if(pointer.username.equalsIgnoreCase(username)){
            return pointer;
        }
    }

    return null;
}

/**
 * 根据索引查找用户
 */
public Node findUser(int index){
    if(count == 0) {
        return null;
    }

    if(index <  0) {
        return null;
    }

    pointer = root;

    int i = 0;
    while(i < index + 1){
        if(pointer.next != null){
            pointer = pointer.next;
```

```
            }
            else{
                return null;
            }

            i++;
        }

        return pointer;
    }
}
```

5、Node 类

功能：用户链表的节点类，定义了链表中的用户。该类与前面所讲的链表节点 Node 类的功能相当。

代码：

```
package server;

import java.net.*;
import java.io.*;

/**
 * 用户链表的结点类
 */
public class Node {
    String username = null;
    Socket socket = null;
    ObjectOutputStream output = null;
    ObjectInputStream input = null;

    Node next = null;
}
```

6、PortConf 类

功能：该类继承自 Jdialog，是用户对服务器端监听端口进行修改配置的类。

代码：

```
package server;

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
```

```
/**
 * 生成端口设置对话框的类
 */
public class PortConf extends JDialog {
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    JPanel panelPort = new JPanel();
    JButton save = new JButton();
    JButton cancel = new JButton();
    public static JLabel DLGINFO=new JLabel(
            "          默认端口号为:8888");

    JPanel panelSave = new JPanel();
    JLabel message = new JLabel();

    public static JTextField portNumber ;

    public PortConf(JFrame frame) {
        super(frame, true);
        try {
            jbInit();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        //设置运行位置，使对话框居中
        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        this.setLocation( (int) (screenSize.width - 400) / 2 + 50,
                (int) (screenSize.height - 600) / 2 + 150);
        this.setResizable(false);
    }

    private void jbInit() throws Exception {
        this.setSize(new Dimension(300, 120));
        this.setTitle("端口设置");
        message.setText("请输入侦听的端口号:");
        portNumber = new JTextField(10);
        portNumber.setText(""+ChatServer.port);
        save.setText("保存");
        cancel.setText("取消");
```

```
        panelPort.setLayout(new FlowLayout());
        panelPort.add(message);
        panelPort.add(portNumber);

        panelSave.add(new Label("             "));
        panelSave.add(save);
        panelSave.add(cancel);
        panelSave.add(new Label("             "));

        Container contentPane = getContentPane();
        contentPane.setLayout(new BorderLayout());
        contentPane.add(panelPort, BorderLayout.NORTH);
        contentPane.add(DLGINFO, BorderLayout.CENTER);
        contentPane.add(panelSave, BorderLayout.SOUTH);

        //保存按钮的事件处理
        save.addActionListener(
                new ActionListener() {
                    public void actionPerformed (ActionEvent a) {
                        int savePort;
                        try{


savePort=Integer.parseInt(PortConf.portNumber.getText());

                            if(savePort<1 || savePort>65535){
                                PortConf.DLGINFO.setText("侦听端口必须是
0-65535 之间的整数!");
                                PortConf.portNumber.setText("");
                                return;
                            }
                            ChatServer.port = savePort;
                            dispose();
                        }
                        catch(NumberFormatException e){
                            PortConf.DLGINFO.setText("错误的端口号,端口
号请填写整数!");
                            PortConf.portNumber.setText("");
                            return;
                        }
                    }
                }
        );
```

```
        //关闭对话框时的操作
        this.addWindowListener(
                new WindowAdapter(){
                    public void windowClosing(WindowEvent e){
                        DLGINFO.setText("          默认端口号为:8888");
                    }
                }
        );

        //取消按钮的事件处理
        cancel.addActionListener(
                new ActionListener(){
                    public void actionPerformed(ActionEvent e){
                        DLGINFO.setText("          默认端口号为:8888");
                        dispose();
                    }
                }
        );
    }
}
```

7、Help 类

功能：服务端程序帮助类。

代码：

```
package server;

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

/**
 * 生成设置对话框的类
 */
public class Help extends JDialog {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    JPanel titlePanel = new JPanel();
    JPanel contentPanel = new JPanel();
    JPanel closePanel = new JPanel();
```

```
    JButton close = new JButton();
    JLabel title = new JLabel("聊天室服务端帮助");
    JTextArea help = new JTextArea();

    Color bg = new Color(255,255,255);

    public Help(JFrame frame) {
        super(frame, true);
        try {
            jbInit();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        //设置运行位置，使对话框居中
        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        this.setLocation( (int) (screenSize.width - 400) / 2,
                (int) (screenSize.height - 320) / 2);
        this.setResizable(false);
    }

    private void jbInit() throws Exception {
        this.setSize(new Dimension(400, 200));
        this.setTitle("帮助");

        titlePanel.setBackground(bg);;
        contentPanel.setBackground(bg);
        closePanel.setBackground(bg);

        help.setText("1、设置服务端的侦听端口（默认端口为 8888）。\n"+
                "2、点击 启动服务 按钮便可在指定的端口启动服务。\n"+
                "3、选择需要接受消息的用户，在消息栏中写入消息，之后便可发送
消息。\n"+
                "4、信息状态栏中显示服务器当前的启动与停止状态、"+
                "用户发送的消息和\n      服务器端发送的系统消息。");
        help.setEditable(false);

        titlePanel.add(new Label("              "));
        titlePanel.add(title);
        titlePanel.add(new Label("              "));

        contentPanel.add(help);
```

```
        closePanel.add(new Label("              "));
        closePanel.add(close);
        closePanel.add(new Label("              "));

        Container contentPane = getContentPane();
        contentPane.setLayout(new BorderLayout());
        contentPane.add(titlePanel, BorderLayout.NORTH);
        contentPane.add(contentPanel, BorderLayout.CENTER);
        contentPane.add(closePanel, BorderLayout.SOUTH);

        close.setText("关闭");
        //事件处理
        close.addActionListener(
                new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        dispose();
                    }
                }
        );
    }
}
```

8、ChatClient 类

功能：定义客户端的界面，添加时间监听与事件处理。该类定义了 Connect（）与 DisConnect（）方法实现与客户端的连接与断开连接。当登陆到指定的服务器时，调用 ClientReceive 类实现消息收发，同时该类还定义了 SendMessaga（）方法来其他用户发送带有表情的消息或悄悄话。

代码：

```
package client;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import java.io.*;
import java.net.*;

/*
 * 聊天客户端的主框架类
 */
public class ChatClient extends JFrame implements ActionListener{

    /**
     *
```

```
 */
private static final long serialVersionUID = 1L;
String ip = "127.0.0.1";//连接到服务端的 ip 地址
int port = 8888;//连接到服务端的端口号
String userName = "匿名君";//用户名
int type = 0;//0 表示未连接，1 表示已连接

JComboBox combobox;//选择发送消息的接受者
JTextArea messageShow;//客户端的信息显示
JScrollPane messageScrollPane;//信息显示的滚动条

JLabel express,sendToLabel,messageLabel ;

JTextField clientMessage;//客户端消息的发送
JCheckBox checkbox;//悄悄话
JComboBox actionlist;//表情选择
JButton clientMessageButton;//发送消息
JTextField showStatus;//显示用户连接状态

Socket socket;
ObjectOutputStream output;//socket 输出流
ObjectInputStream input;//socket 输入流

ClientReceive recvThread;

//建立菜单栏
JMenuBar jMenuBar = new JMenuBar();
//建立菜单组
JMenu operateMenu = new JMenu ("操作(O)");
//建立菜单项
JMenuItem loginItem = new JMenuItem ("用户登录(I)");
JMenuItem logoffItem = new JMenuItem ("用户注销(L)");
JMenuItem exitItem=new JMenuItem ("退出(X)");

JMenu conMenu=new JMenu ("设置(C)");
JMenuItem userItem=new JMenuItem ("用户设置(U)");
JMenuItem connectItem=new JMenuItem ("连接设置(C)");


JMenu helpMenu=new JMenu ("帮助(H)");
JMenuItem helpItem=new JMenuItem ("帮助(H)");

//建立工具栏
JToolBar toolBar = new JToolBar();
```

```
    //建立工具栏中的按钮组件
    JButton loginButton;//用户登录
    JButton logoffButton;//用户注销
    JButton userButton;//用户信息的设置
    JButton connectButton;//连接设置
    JButton exitButton;//退出按钮

    //窗口的大小
    Dimension faceSize = new Dimension(400, 600);

    JPanel downPanel ;
    GridBagLayout girdBag;
    GridBagConstraints girdBagCon;

    public ChatClient(){
        init();//初始化程序

        //添加框架的关闭事件处理
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.pack();
        //设置框架的大小
        this.setSize(faceSize);
        //设置程序图标
        this.setWindowIcon();

        //设置运行时窗口的位置
        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        this.setLocation( (int) (screenSize.width - faceSize.getWidth())
/ 2,
                (int) (screenSize.height - faceSize.getHeight()) / 2);
        this.setResizable(false);
        this.setTitle("聊天室客户端"); //设置标题

        setVisible(true);

        //为操作菜单栏设置热键'V'
        operateMenu.setMnemonic('O');

        //为用户登录设置快捷键为ctrl+i
        loginItem.setMnemonic ('I');
        loginItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_I,InputEvent.CTRL_MASK));
```

```
        //为用户注销快捷键为ctrl+l
        logoffItem.setMnemonic ('L');
        logoffItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_L,InputEvent.CTRL_MASK));

        //为退出快捷键为ctrl+x
        exitItem.setMnemonic ('X');
        exitItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_X,InputEvent.CTRL_MASK));

        //为设置菜单栏设置热键'C'
        conMenu.setMnemonic('C');

        //为用户设置设置快捷键为ctrl+u
        userItem.setMnemonic ('U');
        userItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_U,InputEvent.CTRL_MASK));

        //为连接设置设置快捷键为ctrl+c
        connectItem.setMnemonic ('C');
        connectItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_C,InputEvent.CTRL_MASK));

        //为帮助菜单栏设置热键'H'
        helpMenu.setMnemonic('H');

        //为帮助设置快捷键为ctrl+p
        helpItem.setMnemonic ('H');
        helpItem.setAccelerator (KeyStroke.getKeyStroke
(KeyEvent.VK_H,InputEvent.CTRL_MASK));
    }

    /**
     * 程序初始化函数
     */
    public void init(){

        Container contentPane = getContentPane();
        contentPane.setLayout(new BorderLayout());

        //添加菜单栏
        operateMenu.add (loginItem);
        operateMenu.add (logoffItem);
        operateMenu.add (exitItem);
```

```
jMenuBar.add (operateMenu);
conMenu.add (userItem);
conMenu.add (connectItem);
jMenuBar.add (conMenu);
helpMenu.add (helpItem);
jMenuBar.add (helpMenu);
setJMenuBar (jMenuBar);

//初始化按钮
loginButton = new JButton("登录");
logoffButton = new JButton("注销");
userButton  = new JButton("用户设置" );
connectButton  = new JButton("连接设置" );
exitButton = new JButton("退出" );
//当鼠标放上显示信息
loginButton.setToolTipText("连接到指定的服务器");
logoffButton.setToolTipText("与服务器断开连接");
userButton.setToolTipText("设置用户信息");
connectButton.setToolTipText("设置所要连接到的服务器信息");
//将按钮添加到工具栏
toolBar.add(userButton);
toolBar.add(connectButton);
toolBar.addSeparator();//添加分隔栏
toolBar.add(loginButton);
toolBar.add(logoffButton);
toolBar.addSeparator();//添加分隔栏
toolBar.add(exitButton);
contentPane.add(toolBar,BorderLayout.NORTH);

checkbox = new JCheckBox("悄悄话");
checkbox.setSelected(false);

actionlist = new JComboBox();
actionlist.addItem("");
actionlist.addItem("微笑地");
actionlist.addItem("呵呵地");
actionlist.addItem("生气地");
actionlist.addItem("神经病地");
actionlist.addItem("悲伤地");
actionlist.setSelectedIndex(0);

//初始时
loginButton.setEnabled(true);
logoffButton.setEnabled(false);
```

```
//为菜单栏添加事件监听
loginItem.addActionListener(this);
logoffItem.addActionListener(this);
exitItem.addActionListener(this);
userItem.addActionListener(this);
connectItem.addActionListener(this);
helpItem.addActionListener(this);

//添加按钮的事件侦听
loginButton.addActionListener(this);
logoffButton.addActionListener(this);
userButton.addActionListener(this);
connectButton.addActionListener(this);
exitButton.addActionListener(this);

//添加发送对象
combobox = new JComboBox();
combobox.insertItemAt("所有人",0);
combobox.setSelectedIndex(0);

//添加消息显示框
messageShow = new JTextArea();
messageShow.setEditable(false);

//添加滚动条
messageScrollPane = new JScrollPane(messageShow,
        JScrollPane.VERTICAL_SCROLLBAR_AS_NEEDED,
        JScrollPane.HORIZONTAL_SCROLLBAR_AS_NEEDED);
messageScrollPane.setPreferredSize(new Dimension(400,400));
messageScrollPane.revalidate();

//添加发送消息框与发送按钮
clientMessage = new JTextField(23);
clientMessage.setEnabled(false);
clientMessageButton = new JButton();
clientMessageButton.setText("发送");

//添加系统消息的事件侦听
clientMessage.addActionListener(this);
clientMessageButton.addActionListener(this);

sendToLabel = new JLabel("发送至:");
express = new JLabel("表情:");
```

```
messageLabel = new JLabel("发送消息:");
downPanel = new JPanel();
girdBag = new GridBagLayout();
downPanel.setLayout(girdBag);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 0;
girdBagCon.gridy = 0;
girdBagCon.gridwidth = 5;
girdBagCon.gridheight = 2;
girdBagCon.ipadx = 5;
girdBagCon.ipady = 5;
JLabel none = new JLabel("    ");
girdBag.setConstraints(none,girdBagCon);
downPanel.add(none);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 0;
girdBagCon.gridy = 2;
girdBagCon.insets = new Insets(1,0,0,0);
//girdBagCon.ipadx = 5;
//girdBagCon.ipady = 5;
girdBag.setConstraints(sendToLabel,girdBagCon);
downPanel.add(sendToLabel);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx =1;
girdBagCon.gridy = 2;
girdBagCon.anchor = GridBagConstraints.LINE_START;
girdBag.setConstraints(combobox,girdBagCon);
downPanel.add(combobox);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx =2;
girdBagCon.gridy = 2;
girdBagCon.anchor = GridBagConstraints.LINE_END;
girdBag.setConstraints(express,girdBagCon);
downPanel.add(express);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 3;
girdBagCon.gridy = 2;
girdBagCon.anchor = GridBagConstraints.LINE_START;
//girdBagCon.insets = new Insets(1,0,0,0);
```

```
//girdBagCon.ipadx = 5;
//girdBagCon.ipady = 5;
girdBag.setConstraints(actionlist,girdBagCon);
downPanel.add(actionlist);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 4;
girdBagCon.gridy = 2;
girdBagCon.insets = new Insets(1,0,0,0);
//girdBagCon.ipadx = 5;
//girdBagCon.ipady = 5;
girdBag.setConstraints(checkbox,girdBagCon);
downPanel.add(checkbox);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 0;
girdBagCon.gridy = 3;
girdBag.setConstraints(messageLabel,girdBagCon);
downPanel.add(messageLabel);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 1;
girdBagCon.gridy = 3;
girdBagCon.gridwidth = 3;
girdBagCon.gridheight = 1;
girdBag.setConstraints(clientMessage,girdBagCon);
downPanel.add(clientMessage);

girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 4;
girdBagCon.gridy = 3;
girdBag.setConstraints(clientMessageButton,girdBagCon);
downPanel.add(clientMessageButton);

showStatus = new JTextField(35);
showStatus.setEditable(false);
girdBagCon = new GridBagConstraints();
girdBagCon.gridx = 0;
girdBagCon.gridy = 5;
girdBagCon.gridwidth = 5;
girdBag.setConstraints(showStatus,girdBagCon);
downPanel.add(showStatus);

contentPane.add(messageScrollPane,BorderLayout.CENTER);
```

```
        contentPane.add(downPanel,BorderLayout.SOUTH);

        //关闭程序时的操作
        this.addWindowListener(
                new WindowAdapter(){
                    public void windowClosing(WindowEvent e){
                        if(type == 1){
                            DisConnect();
                        }
                        System.exit(0);
                    }
                }
        );
    }

    /**
     * 事件处理
     */
    public void actionPerformed(ActionEvent e) {
        Object obj = e.getSource();

        if (obj == userItem || obj == userButton) { //用户信息设置
            //调出用户信息设置对话框
            UserConf userConf = new UserConf(this,userName);
            userConf.setVisible(true);
            userName = userConf.userInputName;
        }
        else if (obj == connectItem || obj == connectButton) { //连接
服务端设置
            //调出连接设置对话框
            ConnectConf conConf = new ConnectConf(this,ip,port);
            conConf.setVisible(true);
            ip = conConf.userInputIp;
            port = conConf.userInputPort;
        }
        else if (obj == loginItem || obj == loginButton) { //登录
            Connect();
        }
        else if (obj == logoffItem || obj == logoffButton) { //注销
            DisConnect();
            showStatus.setText("");
        }
        else if (obj == clientMessage || obj == clientMessageButton) { //
发送消息
```

```
            SendMessage();
            clientMessage.setText("");
        }
        else if (obj == exitButton || obj == exitItem) { //退出
            int j=JOptionPane.showConfirmDialog(
                    this,"真的要退出吗?","退出",
JOptionPane.YES_OPTION,JOptionPane.QUESTION_MESSAGE);
            if (j == JOptionPane.YES_OPTION){
                if(type == 1){
                    DisConnect();
                }
                System.exit(0);
            }
        }
        else if (obj == helpItem) { //菜单栏中的帮助
            //调出帮助对话框
            Help helpDialog = new Help(this);
            helpDialog.setVisible(true);
        }
    }


    public void Connect(){
        try{
            socket = new Socket(ip,port);
        }
        catch (Exception e){
            JOptionPane.showConfirmDialog(
                    this,"不能连接到指定的服务器。\n 请确认连接设置是否正确。
","提示",
JOptionPane.DEFAULT_OPTION,JOptionPane.WARNING_MESSAGE);
            return;
        }

        try{
            output = new ObjectOutputStream(socket.getOutputStream());
            output.flush();
            input  = new ObjectInputStream(socket.getInputStream() );

            output.writeObject(userName);
            output.flush();
```

```
            recvThread = new
ClientReceive(socket,output,input,combobox,messageShow,showStatus);
            recvThread.start();

            loginButton.setEnabled(false);
            loginItem.setEnabled(false);
            userButton.setEnabled(false);
            userItem.setEnabled(false);
            connectButton.setEnabled(false);
            connectItem.setEnabled(false);
            logoffButton.setEnabled(true);
            logoffItem.setEnabled(true);
            clientMessage.setEnabled(true);
            messageShow.append("连接服务器 "+ip+":"+port+" 成功...\n");
            type = 1;//标志位设为已连接
        }
        catch (Exception e){
            System.out.println(e);
            return;
        }
    }

    public void DisConnect(){
        loginButton.setEnabled(true);
        loginItem.setEnabled(true);
        userButton.setEnabled(true);
        userItem.setEnabled(true);
        connectButton.setEnabled(true);
        connectItem.setEnabled(true);
        logoffButton.setEnabled(false);
        logoffItem.setEnabled(false);
        clientMessage.setEnabled(false);

        if(socket.isClosed()){
            return ;
        }

        try{
            output.writeObject("用户下线");
            output.flush();

            input.close();
            output.close();
```

```
            socket.close();
            messageShow.append("已经与服务器断开连接...\n");
            type = 0;//标志位设为未连接
        }
        catch (Exception e){
            //
        }
    }

    public void SendMessage(){
        String toSomebody = combobox.getSelectedItem().toString();
        String status  = "";
        if(checkbox.isSelected()){
            status = "悄悄话";
        }

        String action = actionlist.getSelectedItem().toString();
        String message = clientMessage.getText();

        if(socket.isClosed()){
            return ;
        }

        try{
            output.writeObject("聊天信息");
            output.flush();
            output.writeObject(toSomebody);
            output.flush();
            output.writeObject(status);
            output.flush();
            output.writeObject(action);
            output.flush();
            output.writeObject(message);
            output.flush();
        }
        catch (Exception e){
            //
        }
    }
    public void setWindowIcon()
    {
        ImageIcon imageIcon = new
ImageIcon(getClass().getResource("/img/clienticon.png"));
        this.setIconImage(imageIcon.getImage());
```

```
    }

    public static void main(String[] args) {
        try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        } catch (ClassNotFoundException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (InstantiationException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (IllegalAccessException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        } catch (UnsupportedLookAndFeelException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        new ChatClient();

    }
}
```

9、ClientReceive 类

功能：实现服务器端与客户端消息收发的类。

代码：

```
package client;

import javax.swing.*;
import java.io.*;
import java.net.*;

/*
 * 聊天客户端消息收发类
 */
public class ClientReceive extends Thread {
    private JComboBox combobox;
    private JTextArea textarea;

    Socket socket;
    ObjectOutputStream output;
    ObjectInputStream  input;
```

```
    JTextField showStatus;

    public ClientReceive(Socket socket,ObjectOutputStream output,
                         ObjectInputStream  input,JComboBox
combobox,JTextArea textarea,JTextField showStatus){

        this.socket = socket;
        this.output = output;
        this.input = input;
        this.combobox = combobox;
        this.textarea = textarea;
        this.showStatus = showStatus;
    }

    public void run(){
        while(!socket.isClosed()){
            try{
                String type = (String)input.readObject();

                if(type.equalsIgnoreCase("系统信息")){
                    String sysmsg = (String)input.readObject();
                    textarea.append("系统信息: "+sysmsg);
                }
                else if(type.equalsIgnoreCase("服务关闭")){
                    output.close();
                    input.close();
                    socket.close();

                    textarea.append("服务器已关闭! \n");

                    break;
                }
                else if(type.equalsIgnoreCase("聊天信息")){
                    String message = (String)input.readObject();
                    textarea.append(message);
                }
                else if(type.equalsIgnoreCase("用户列表")){
                    String userlist = (String)input.readObject();
                    String usernames[] = userlist.split("\n");
                    combobox.removeAllItems();

                    int i =0;
                    combobox.addItem("所有人");
                    while(i < usernames.length){
```

```
                    combobox.addItem(usernames[i]);
                    i ++;
                }
                combobox.setSelectedIndex(0);
                showStatus.setText("在线用户 " + usernames.length + "
人");
            }
        }
        catch (Exception e ){
            System.out.println(e);
        }
    }
}
}
```

10、ConnectConf 类

功能：用户对所有要连接的服务器 IP 及监听端口进行修改配置的类。

代码：

```
package client;

import java.awt.*;
import java.net.*;
import javax.swing.*;
import java.awt.event.*;

/**
 * 生成连接信息输入的对话框
 * 让用户输入连接服务器的 IP 和端口
 */
public class ConnectConf extends JDialog {
    /**
     *
     */
    private static final long serialVersionUID = 1L;
    JPanel panelUserConf = new JPanel();
    JButton save = new JButton();
    JButton cancel = new JButton();
    JLabel DLGINFO=new JLabel(
            "默认连接设置为 127.0.0.1:8888");

    JPanel panelSave = new JPanel();
    JLabel message = new JLabel();
```

```
    String userInputIp;
    int userInputPort;

    JTextField inputIp;
    JTextField inputPort;

    public ConnectConf(JFrame frame,String ip,int port) {
        super(frame, true);
        this.userInputIp = ip;
        this.userInputPort = port;
        try {
            jbInit();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        //设置运行位置，使对话框居中
        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        this.setLocation( (int) (screenSize.width - 400) / 2 + 50,
                (int) (screenSize.height - 600) / 2 + 150);
        this.setResizable(false);
    }

    private void jbInit() throws Exception {
        this.setSize(new Dimension(300, 130));
        this.setTitle("连接设置");
        message.setText("请输入服务器的 IP:");
        inputIp = new JTextField(10);
        inputIp.setText(userInputIp);
        inputPort = new JTextField(4);
        inputPort.setText(""+userInputPort);
        save.setText("保存");
        cancel.setText("取消");

        panelUserConf.setLayout(new GridLayout(2,2,1,1));
        panelUserConf.add(message);
        panelUserConf.add(inputIp);
        panelUserConf.add(new JLabel("请输入服务器的端口:"));
        panelUserConf.add(inputPort);

        panelSave.add(new Label("              "));
        panelSave.add(save);
        panelSave.add(cancel);
```

```
        panelSave.add(new Label("              "));

        Container contentPane = getContentPane();
        contentPane.setLayout(new BorderLayout());
        contentPane.add(panelUserConf, BorderLayout.NORTH);
        contentPane.add(DLGINFO, BorderLayout.CENTER);
        contentPane.add(panelSave, BorderLayout.SOUTH);

        //保存按钮的事件处理
        save.addActionListener(
                new ActionListener() {
                    public void actionPerformed (ActionEvent a) {
                        int savePort;
                        //判断端口号是否合法
                        try{
                            userInputIp = "" +
InetAddress.getByName(inputIp.getText());
                            userInputIp = userInputIp.substring(1);
                        }
                        catch(UnknownHostException e){
                            DLGINFO.setText(
                                    "错误的 IP 地址！");

                            return;
                        }
                        //userInputIp = inputIP;

                        //判断端口号是否合法
                        try{
                            savePort =
Integer.parseInt(inputPort.getText());

                            if(savePort<1 || savePort>65535){
                                DLGINFO.setText("侦听端口必须是 0-65535 之
间的整数!");
                                inputPort.setText("");
                                return;
                            }
                            userInputPort = savePort;
                            dispose();
                        }
                        catch(NumberFormatException e){
                            DLGINFO.setText("错误的端口号,端口号请填写整
数!");
```

```
                        inputPort.setText("");
                        return;
                    }
                }
            }
        );

        //关闭对话框时的操作
        this.addWindowListener(
                new WindowAdapter(){
                    public void windowClosing(WindowEvent e){
                        DLGINFO.setText("默认连接设置为 127.0.0.1:8888");
                    }
                }
        );

        //取消按钮的事件处理
        cancel.addActionListener(
                new ActionListener(){
                    public void actionPerformed(ActionEvent e){
                        DLGINFO.setText("默认连接设置为 127.0.0.1:8888");
                        dispose();
                    }
                }
        );
    }
}
```

11、UserConf 类

功能：用户对链接到服务器时所显示的用户名进行修改配置的类。

代码：

```
package client;

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

/**
 * 生成用户信息输入对话框的类
 * 让用户输入自己的用户名
 */
public class UserConf extends JDialog {
    /**
```

```
     *
     */
    private static final long serialVersionUID = 1L;
    JPanel panelUserConf = new JPanel();
    JButton save = new JButton();
    JButton cancel = new JButton();
    JLabel DLGINFO=new JLabel(
            "          默认用户名为：匿名君");

    JPanel panelSave = new JPanel();
    JLabel message = new JLabel();
    String userInputName;

    JTextField userName ;

    public UserConf(JFrame frame,String str) {
        super(frame, true);
        this.userInputName = str;
        try {
            jbInit();
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        //设置运行位置，使对话框居中
        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        this.setLocation( (int) (screenSize.width - 400) / 2 + 50,
                (int) (screenSize.height - 600) / 2 + 150);
        this.setResizable(false);
    }

    private void jbInit() throws Exception {
        this.setSize(new Dimension(300, 120));
        this.setTitle("用户设置");
        message.setText("请输入用户名:");
        userName = new JTextField(10);
        userName.setText(userInputName);
        save.setText("保存");
        cancel.setText("取消");

        panelUserConf.setLayout(new FlowLayout());
        panelUserConf.add(message);
        panelUserConf.add(userName);
```

```
        panelSave.add(new Label("                "));
        panelSave.add(save);
        panelSave.add(cancel);
        panelSave.add(new Label("                "));

        Container contentPane = getContentPane();
        contentPane.setLayout(new BorderLayout());
        contentPane.add(panelUserConf, BorderLayout.NORTH);
        contentPane.add(DLGINFO, BorderLayout.CENTER);
        contentPane.add(panelSave, BorderLayout.SOUTH);

        //保存按钮的事件处理
        save.addActionListener(
                new ActionListener() {
                    public void actionPerformed (ActionEvent a) {
                        if(userName.getText().equals("")){
                            DLGINFO.setText(
                                    "          用户名不能为空！");
                            userName.setText(userInputName);
                            return;
                        }
                        else if(userName.getText().length() > 15){
                            DLGINFO.setText("          用户名长度不能大于
15 个字符！");
                            userName.setText(userInputName);
                            return;
                        }
                        userInputName = userName.getText();
                        dispose();
                    }
                }
        );

        //关闭对话框时的操作
        this.addWindowListener(
                new WindowAdapter(){
                    public void windowClosing(WindowEvent e){
                        DLGINFO.setText("          默认用户名为：匿名君");
                    }
                }
        );

        //取消按钮的事件处理
```

```
        cancel.addActionListener(
                new ActionListener(){
                    public void actionPerformed(ActionEvent e){
                        DLGINFO.setText("          默认用户名为：匿名君");
                        dispose();
                    }
                }
        );
    }
}
```

12、Help 类

功能：客户端程序的帮助类。

代码：

```
package client;

import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

/**
 * 生成设置对话框的类
 */
public class Help extends JDialog {

    /**
     *
     */
    private static final long serialVersionUID = 1L;
    JPanel titlePanel = new JPanel();
    JPanel contentPanel = new JPanel();
    JPanel closePanel = new JPanel();

    JButton close = new JButton();
    JLabel title = new JLabel("聊天室客户端帮助");
    JTextArea help = new JTextArea();

    Color bg = new Color(255,255,255);

    public Help(JFrame frame) {
        super(frame, true);
        try {
            jbInit();
```

```
        }
        catch (Exception e) {
            e.printStackTrace();
        }
        //设置运行位置，使对话框居中
        Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
        this.setLocation( (int) (screenSize.width - 400) / 2 + 25,
                (int) (screenSize.height - 320) / 2);
        this.setResizable(false);
    }

    private void jbInit() throws Exception {
        this.setSize(new Dimension(350, 270));
        this.setTitle("帮助");

        titlePanel.setBackground(bg);;
        contentPanel.setBackground(bg);
        closePanel.setBackground(bg);

        help.setText("1、设置所要连接服务端的 IP 地址和端口"+
                "（默认设置为\n      127.0.0.1:8888）。\n"+
                "2、输入你的用户名（默认设置为:匆匆过客）。\n"+
                "3、点击“登录”便可以连接到指定的服务器；\n"+
                "      点击“注销”可以和服务器端开连接。\n"+
                "4、选择需要接受消息的用户，在消息栏中写入消息，\n"+
                "      同时选择表情，之后便可发送消息。\n");
        help.setEditable(false);

        titlePanel.add(new Label("              "));
        titlePanel.add(title);
        titlePanel.add(new Label("              "));

        contentPanel.add(help);

        closePanel.add(new Label("              "));
        closePanel.add(close);
        closePanel.add(new Label("              "));

        Container contentPane = getContentPane();
        contentPane.setLayout(new BorderLayout());
        contentPane.add(titlePanel, BorderLayout.NORTH);
        contentPane.add(contentPanel, BorderLayout.CENTER);
        contentPane.add(closePanel, BorderLayout.SOUTH);
```

```
        close.setText("关闭");
        //事件处理
        close.addActionListener(
                new ActionListener() {
                    public void actionPerformed(ActionEvent e) {
                        dispose();
                    }
                }
        );
    }
}
```