软件设计说明书

关键词:GUI 模块、多线程、Socket 通讯、监听器、回车驱动
摘　要:该简易聊天 APP 能实现不同的客户端之间发送文字消息的功能

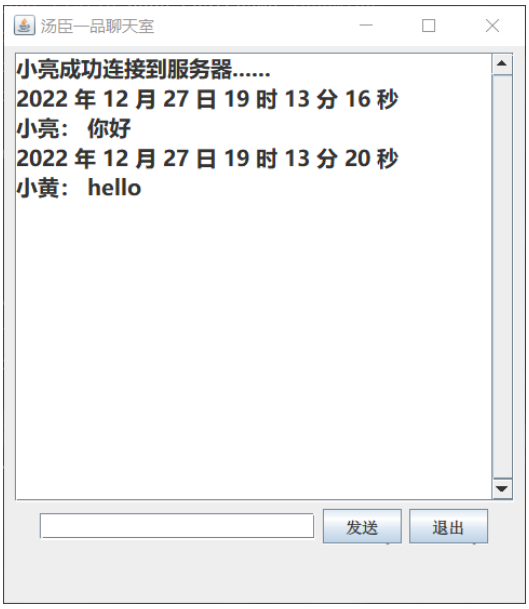
目录

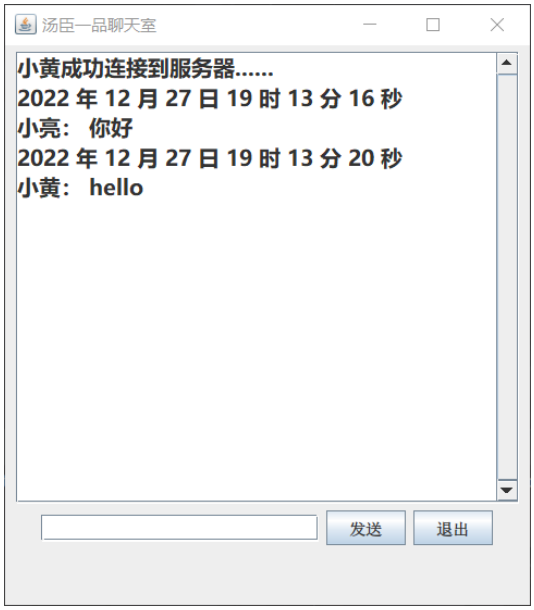
1. 软件功能描述

1.1. 功能说明

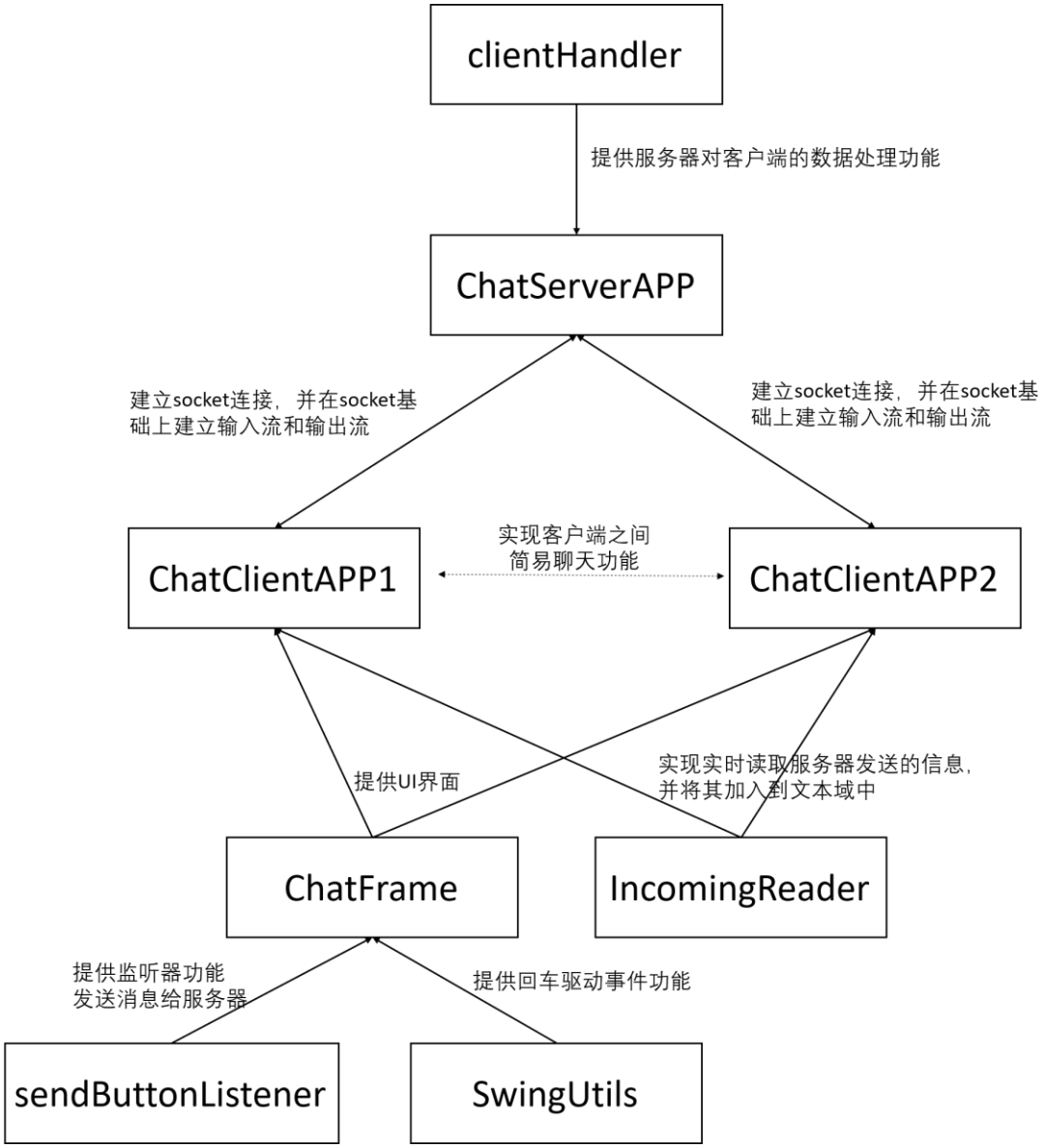
不同客户端之间能够实现基本的简易聊天功能
●聊天记录显示框可以显示客户端之间的聊天记录
●聊天记录包括发送的信息的人、时间以及发送的信息内容
●聊天记录可以正确的显示发送信息的顺序
●点击发送按钮可以将输入的信息发送并显示出来
●输入框信息为空时，点击发送按钮，程序不作反应
●键 Enter 能够实现发送功能
●关闭应用程序或点击退出按钮可以正确地结束程序运行

1.2. 产品界面





1.3.功能框图



2.类规格说明

类	说明
ChatServerAPP	聊天室服务器端口，需要保持后台运行
ChatClientAPP1	聊天室客户端口 1，能够和其他客户端口聊天
ChatClientAPP2	聊天室客户端口 2，能够和其他客户端口聊天
ChatFrame	聊天软件 UI 界面模块
clientHandler	服务器的客户端处理模块
IncomingReader	不断读取服务器发送的信息，并将其加入到文本域中
sendButtonListener	监听器与发送消息给服务器模块
SwingUtils	回车驱动事件功能模块

3.技术方案

3.1.聊天室服务器模块

- 通过 JPanel 容器和 JFrame 面板以及退出按钮 Button exit 实现基本的服务器 UI 界面
- 通过 socket 与客户端 client 进行通信，端口号：8888
- 通过多线程实现与客户端通信，同时使服务器一直处于接受消息并转发给所有人

3.2.聊天室客户端模块

- 通过 socket 与服务器建立连接，地址："127.0.0.1"，端口号：8888
- 利用 swing 图形化相关技术，设计聊天室的 UI 界面，详见 3.3
- 与 socket 建立字符缓冲输入流，用于接受发送框中的消息
- 与 socket 建立字符缓冲输出流，用于发送消息

3.3.聊天室 UI 界面模块

- 创建一个文本域 incoming，用来显示消息
- 创建一个输入框 outgoing，用来输入消息
- 创建发送按钮并创建监听器 sendButtonListener，详见 3.6
- 创建退出按钮，使得单击后退出程序
- 创建键盘监听器实现 enter 键发送与退出，详见 3.7
- 按钮实现，需要将焦点切换到按钮上，文本输入框中也实现键盘监听
- 将所有组件放入 JPanel 容器中，并将 JPanel 容器放入 JFrame 面板中

3.4.服务器的客户端处理模块

- 通过 socket 与客户端建立通信并定义接受数据的输入流
- 创建线程，不断从 socket 上读取数据，同时将数据发送给每一个客户端

3.5.实时更新文本域模块

- 创建字符串 messag 用于一个接受从服务器与客户端建立的输入流中读出的信息
- 将读出的信息实时添加到文本域中

3.6.监听器与发送消息给服务器模块

- 创建一个输出字符流 writer，用于给服务器发消息
- 当发送按钮点击，触发事件，将输入框中的文字获取，并发送给服务器
- 通过 if else 实现当输入为空时，点击无反应
- 设置聊天记录格式：时间+用户+内容
- 时间格式通过创建 SimpleDateForma 对象实现，格式："yyyy 年 MM 月 dd 日 HH 时 mm 分 ss 秒"。
- 按照构造方法中指定的模式，把 Date 解析为符合模式的字符串（文本）
- 当发送结束后，输入框文字清空

3.7.回车驱动事件功能模块

●创建一个方法 enterPressesWhenFocused（）实现传入 JButton 对指定的按钮 button 添加回车驱动事件的功能，传入 JTextField 和 ActionListener 实现对文本域 textField 添加回车触发事件 actionListener

4.源文件列表

4.1.聊天室服务器模块

ChatServerAPP

```
package code;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionListener;
import java.io.*;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.ArrayList;

/**
 * 聊天室服务器端口，需要保持后台运行
 * @author ZHL
 * @author HZX
 * @date 2022.12.25 下午 15:18
 */
public class ChatServerAPP {
    static ArrayList<PrintWriter> clientOutPutStreams = new ArrayList<PrintWriter>();

    public static void main(String[] args) {
        new ChatServerAPP().go();
    }

    /**
     * 服务端 UI 界面和连接客户端的 socket
     */
    public void go() {
        try {
            //server 端创建出与 client 端通信的 socket
            ServerSocket serverSocket = new ServerSocket(8888);
            System.out.println("服务器启动成功！");

            //创建服务器面板，便于观察运行状态
```

```
            JFrame jFrame = new JFrame("汤臣一品聊天室服务器");
            JPanel jPanel = new JPanel();

            //创建退出按钮，使得单击后退出程序
            JButton exit = new JButton("退出服务器");

            ActionListener al = e -> System.exit(0);
            exit.addActionListener(al);

            //将退出组件放入 JPanel 容器中
            jPanel.add(exit);

            //将 JPanel 容器放入 JFrame 面板中
            jFrame.getContentPane().add(BorderLayout.CENTER, jPanel);
            jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            jFrame.setSize(400, 200);
            jFrame.setVisible(true);

            while (true) {
                //服务器创建出与客户端通信的 socket
                Socket clientSocket = serverSocket.accept();
                String ip=clientSocket.getInetAddress().getHostAddress();
                System.out.println(ip+"加入了聊天室！");//控制台打印进入到聊天室的用户地址
并提示

                //与客户端建立输出流，为以后给客户端写数据做准备
                PrintWriter printWriter = new PrintWriter(clientSocket.getOutputStream());
                clientOutPutStreams.add(printWriter);

                //开启一个线程用于与一个客户端进行通信
                Thread thread = new Thread(new clientHandler(clientSocket));

                //开启线程，使服务器一直处于接受消息并转发给所有人
                thread.start();
                System.out.println("有新连接加入");

            }

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

    /**
```

```
     * 给每一个客户端发送消息的方法
     * @param message 从 socket 上读取到的数据
     */
    public static void sendEveryone(String message){
        for (PrintWriter COPStreams: clientOutPutStreams) {
            try {
                PrintWriter writer = COPStreams;
                writer.println(message);
                writer.flush();
            }catch (Exception e){
                e.printStackTrace();
            }
        }
    }
}
```

4.2.聊天室客户端模块

ChatClientAPP1

```
package code;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

import static code.ChatFrame.incoming;
import static code.sendButtonListener.writer;
import static code.IncomingReader.reader;

/**
 * 聊天室客户端口 1，能够和其他客户端口聊天
 * @author ZHL
 * @author HZX
 * @date 2022.12.25 下午 15:19
 */
public class ChatClientAPP1 {
    /**创建准备与服务器连接的套接字*/
    public static Socket socket;
    /**创建聊天用户的名字*/
    public static String name;

    public static void main(String[] args) {
        new ChatClientAPP1().go();
    }
```

```
    /**
     * 客户端 UI 界面和与服务器连接
     */
    public void go() {

        //创建 swing 图形化界面
        ChatFrame.chatFrame();

        //与服务器建立连接
        setUpNetworking();

        //开启一个线程，并执行其中 run()方法
        Thread thread = new Thread(new IncomingReader());
        thread.start();

    }

    /**
     * 与服务器建立 socket 连接，并在 socket 基础上建立输入流和输出流
     */
    private void setUpNetworking() {
        try {

            //与服务器 socket 建立连接
            socket = new Socket("127.0.0.1", 8888);

            //自定义用户名
            name = "小亮";

            //与 socket 建立字符缓冲输入流，用于接受发送框中的消息
            InputStreamReader inputStreamReader = new
InputStreamReader(socket.getInputStream());
            reader = new BufferedReader(inputStreamReader);

            //与 socket 建立字符缓冲输出流，用于发送消息
            writer = new PrintWriter(socket.getOutputStream());
            incoming.append(name+"成功连接到服务器....." + "\n");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

ChatClientAPP2

```
package code;

import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.Socket;

import static code.ChatFrame.incoming;
import static code.sendButtonListener.writer;
import static code.IncomingReader.reader;
import static code.ChatClientAPP1.name;

/**
 * 聊天室客户端口 2，能够和其他客户端口聊天
 * @author ZHL
 * @author HZX
 * @date 2022.12.25 下午 15:19
 */
public class ChatClientAPP2 {
    /**创建准备与服务器连接的套接字*/
    public static Socket socket;

    public static void main(String[] args) {
        new ChatClientAPP2().go();
    }

    /**
     * 客户端 2UI 界面和与服务器连接
     */
    public void go() {

        //创建 swing 图形化界面
        ChatFrame.chatFrame();

        //与服务器建立连接
        setUpNetworking();

        //开启一个线程，并执行其中 run()方法
        Thread thread = new Thread(new IncomingReader());
        thread.start();

    }
```

```
    /**
     * 与服务器建立 socket 连接，并在 socket 基础上建立输入流和输出流
     */
    private void setUpNetworking() {
        try {

            //与服务器 socket 建立连接
            socket = new Socket("127.0.0.1", 8888);

            //自定义用户名
            name = "小黄";

            //与 socket 建立字符缓冲输入流，用于接受发送框中的消息
            InputStreamReader inputStreamReader = new
InputStreamReader(socket.getInputStream());
            reader = new BufferedReader(inputStreamReader);

            //与 socket 建立字符缓冲输出流，用于发送消息
            writer = new PrintWriter(socket.getOutputStream());
            incoming.append(name+"成功连接到服务器....." + "\n");

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4.3.聊天室 UI 界面模块

ChatFrame

```
package code;

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

/**
 * 聊天软件 UI 界面模块
 * @author ZHL
 * @author HZX
 * @date 2022.12.25 下午 15:20
 */
public class ChatFrame {
```

```
    /**输入框*/
    public static JTextField outgoing;
    /**文本域，用来显示消息*/
    public static JTextArea incoming;

    /**
     * 客户端UI界面设计
     */
    public static void chatFrame() {

        JFrame jFrame = new JFrame("汤臣一品聊天室");
        JPanel jPanel = new JPanel();

        //创建文本域
        incoming = new JTextArea(15, 22);
        incoming.setFont(new Font("微软雅黑",Font.BOLD, 16));
        incoming.setLineWrap(true);         //设置在行过长时自动换行
        incoming.setWrapStyleWord(true);    //设置在单词过长时，将单词移到下一行
        incoming.setEditable(false);    //设置选项不可用,防止误操作

        //创建滚动面板，并将文本域放入其中
        JScrollPane jScrollPane = new JScrollPane(incoming);

jScrollPane.setVerticalScrollBarPolicy(ScrollPaneConstants.VERTICAL_SCROLLBAR_ALWAYS);

jScrollPane.setHorizontalScrollBarPolicy(ScrollPaneConstants.HORIZONTAL_SCROLLBAR_NEVER);

        //创建输入框
        outgoing = new JTextField(20);

        //创建发送按钮并创建监听器
        JButton sendButton = new JButton("发送");
        sendButton.addActionListener(new sendButtonListener());

        //创建退出按钮，使得单击后退出程序
        JButton exitButton = new JButton("退出");
        ActionListener al = e -> System.exit(0);
        exitButton.addActionListener(al);

        //创建键盘监听器实现enter键发送与退出
        //按钮实现，需要将焦点切换到按钮上
        SwingUtils.enterPressesWhenFocused(sendButton);
        SwingUtils.enterPressesWhenFocused(exitButton);
        //文本输入框中实现键盘监听
```

```
        SwingUtils.enterPressesWhenFocused(outgoing,new sendButtonListener());

        //将上述定义组件放入 JPanel 容器中
        jPanel.add(jScrollPane);
        jPanel.add(outgoing);
        jPanel.add(sendButton);
        jPanel.add(exitButton);

        //将 JPanel 容器放入 JFrame 面板中
        jFrame.getContentPane().add(BorderLayout.CENTER, jPanel);
        jFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        jFrame.setSize(400, 450);
        jFrame.setVisible(true);
    }
}
```

4.4.服务器的客户端处理模块

clientHandler

```
package code;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;

/**
 * 服务器的客户端处理模块
 * @author ZHL
 * @author HZX
 * @date 2022/12/26/下午 12:44
 */
public class clientHandler implements Runnable {
    BufferedReader reader;
    Socket socket;

    /**
     * clientHandler 的构造方法
     * @param socket 客户端建立通信
     */
    public clientHandler(Socket socket) {
        //与客户端建立通信并定义接受数据的输入流
        try {
            this.socket = socket;
            InputStreamReader             inputStreamReader             =             new
```

```
InputStreamReader(socket.getInputStream());
            reader = new BufferedReader(inputStreamReader);
        } catch (IOException e) {
            e.printStackTrace();
        }

    }

    /**开启线程之后要执行的 run()方法*/
    public void run() {
        //定义准备接受消息变量
        String message = null;

        try {
            //使线程不断从 socket 上读取数据的状态
            while ((message= reader.readLine())!=null){
                System.out.println("MESSAGE_Receive: "+message);
                //将数据发送给每一个客户端
                ChatServerAPP.sendEveryone(message);
            }
        } catch (IOException e) {
            e.printStackTrace();
        }

    }
}
```

4.5.实时更新文本域模块

IncomingReader

```
package code;

import java.io.BufferedReader;

import static code.ChatFrame.incoming;

/**
 * 该线程任务是不断读取服务器发送的信息，并将其加入到文本域中
 * @author ZHL
 * @author HZX
 * @date 2022.12.25 下午 15:20
 */
public class IncomingReader implements Runnable {
    /**缓冲字符输入流，用于读取服务器转发的消息*/
```

```
    public static BufferedReader reader;

    public void run() {
        String message;        //用于接受从服务器与客户端建立的输入流中读出的信息
        try {
            while ((message = reader.readLine()) != null) {
                System.out.println("接收到消息： " + message);
                incoming.append(message + "\n");      //将读出的信息添加到文本域中
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

4.6.监听器与发送消息给服务器模块

sendButtonListener

```
package code;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.PrintWriter;
import java.text.SimpleDateFormat;
import java.util.Date;

import static code.ChatClientAPP1.name;
import static code.ChatFrame.outgoing;

/**
 * 监听器与发送消息给服务器模块
 * @author ZHL
 * @author HZX
 * @date 2022.12.25 下午 15:20
 */
public class sendButtonListener implements ActionListener {
    /**输出字符流，用于给服务器发消息*/
    public static PrintWriter writer;

    public void actionPerformed(ActionEvent e) {
        //当发送按钮点击，触发事件，将输入框中的文字获取，并发送给服务器
        //当输入为空时，点击无反应
        if ("".equals(outgoing.getText())) {
            outgoing.requestFocus();
        } else {
```

```
            // 创建 SimpleDateFormat 对象，在 SimpleDateFormat(String pattern)构造方法中传入指定的模式
            SimpleDateFormat simpleDateFormat = new SimpleDateFormat("yyyy 年 MM 月 dd 日 HH 时 mm 分 ss 秒");

            // 调用 DateFormat 的实现类 SimpleDateFormat 中的 format 方法，
            // 按照构造方法中指定的模式，把 Date 解析为符合模式的字符串（文本）
            Date date = new Date();
            String dateFormat = simpleDateFormat.format(date);

            //聊天记录格式：时间+用户+内容
            writer.println(dateFormat);
            writer.println(name+"： " + outgoing.getText());
            writer.flush();

            //输入框文字清空
            outgoing.setText("");
            outgoing.requestFocus();
        }
    }
}
```

4.7.回车驱动事件功能模块

SwingUtils

```
package code;

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

/**
 * 回车驱动事件功能模块
 * @author ZHL
 * @author YXZ
 * @date 2022.12.25 下午 15:21
 */
public class SwingUtils {
    /**
     * 对指定的 button 添加回车驱动事件的功能
     * @param button 需要添加回车驱动功能的按钮
     */
    public static void enterPressesWhenFocused(JButton button) {
```

```
button.registerKeyboardAction(button.getActionForKeyStroke(KeyStroke.getKeyStroke(KeyEvent.VK_SPACE, 0, false)),
                KeyStroke.getKeyStroke(KeyEvent.VK_ENTER,             0,             false),
JComponent.WHEN_FOCUSED);


button.registerKeyboardAction(button.getActionForKeyStroke(KeyStroke.getKeyStroke(KeyEvent.VK_SPACE, 0, true)),
                KeyStroke.getKeyStroke(KeyEvent.VK_ENTER,             0,             true),
JComponent.WHEN_FOCUSED);
    }

    /**
     * 在文本输入框中回车触发事件
     * @param textField 需要添加回车触发事件的文本域
     * @param actionListener 需要触发的事件
     */
    public static void enterPressesWhenFocused(JTextField textField, ActionListener
actionListener) {
        textField.registerKeyboardAction(actionListener,
KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0, false),
                JComponent.WHEN_FOCUSED);

        textField.registerKeyboardAction(actionListener,
KeyStroke.getKeyStroke(KeyEvent.VK_ENTER, 0, true),
                JComponent.WHEN_FOCUSED);
    }
}
```

5.相关参考资料及文档

[1] https://blog.csdn.net/m0_55376623/article/details/121644245

[2] https://blog.csdn.net/jsjboss/article/details/743135

[3] https://wenku.baidu.com/view/922254fc700abb68a982fb24

[4] https://blog.csdn.net/m0_61933976/article/details/127021176

项目完整代码：https://gitee.com/muye_zhl/Simple_chat_software