浙江大学实验报告

课程名称: Java应用技术 实验类型: 综合型

实验项目名称: 多客户端的纯文本聊天服务器

学生姓名: 管嘉瑞 专业: 计算机科学与技术 学号: 3200102557

电子邮件地址: <u>3200102557@zju.edu.cn</u> 手机: 13588084334

实验日期: 2022年 11月 25日

一、功能需求

实现一个多客户端的纯文本聊天服务器,能同时接受多个客户端的连接,并将任意一个客户端发送文本向所有客户端(包括发送方)转发。

本实验使用了Java的swing和awt组件,结合socket编程,提供了一个带有GUI界面的多客户端聊天器。支持换行输出、支持中文输入。

二、环境配置

• Java版本:

openjdk 18.0.2.1 2022-08-18

OpenJDK Runtime Environment (build 18.0.2.1+1-1)

OpenJDK 64-Bit Server VM (build 18.0.2.1+1-1, mixed mode, sharing)

• Linux/MacOS平台运行指令:

参考ReadMe.md

三、实验内容

● 服务器端

代码文件: src/Server.java

功能:与客户端建立连接,监听客户端发送的信息,并广播发送给所有客户端,主要代码模块如下

连接建立:

```
void startServe() throws IOException {
    ServerSocket clientSocket = new ServerSocket(PORT);
    Server.clientNum = 0;
    System.out.println("FreeChat Server Started");
    try
    {
        while (true)
        {
            Socket socket = clientSocket.accept();
        }
}
```

```
10
                     Server.clientNum++;
11
                     System.out.println("[Server] Client " + Server.clientNum + "
    has connected");
12
                     PrintWriter clientWriter = new PrintWriter(new BufferedWriter(
13
                                                  new
    OutputStreamWriter(socket.getOutputStream())),true);
14
                     clientWriter.println("[Server] Client " + Server.clientNum + "
    has connected");
15
                     clientWriter.flush();
                     clientWriterList.add(clientWriter);
16
                     ServerHandler reader = this.new ServerHandler(socket);
17
18
                     Thread clientHandler = new Thread(reader);
19
                     clientHandler.start();
20
                }
21
            }
            finally
2.2
23
            {
24
                clientSocket.close();
25
            }
26
        }
```

监听并转发客户端信息

```
1
             @Override
             public void run() {
 2.
 3
                 try {
 4
                     while ((msg = client_input.readLine()) != null) {
                          for (PrintWriter writer : clientWriterList) {
 5
 6
                              writer.println(msg);
 7
                              writer.flush();
 8
                          }
 9
                     }
                 } catch (Exception exc) {
1.0
11
                     exc.printStackTrace();
12
                 }
13
             }
```

● 客户端

代码文件: src/Client.java

功能:接受用户输入,发送文本信息给服务器,接受服务器的信息(来自其他客户端或自己)并显示在聊天框中,主要代码模块如下

连接服务器并创建通信渠道:

```
1
            try {
 2
                 socket = new Socket("localhost", PORT);
 3
 4
                outWriter = new PrintWriter(new BufferedWriter(new
    OutputStreamWriter(socket.getOutputStream())),true);
                inReader = new BufferedReader( new
    InputStreamReader(socket.getInputStream()));
 6
            }
 7
            catch (IOException exc)
 8
9
                JOptionPane.showMessageDialog(null, "Cannot connect to server!");
1.0
                if (socket != null)
11
                     socket.close();
12
            }
```

向服务器发送用户输入信息(send按钮触发):

```
1
            ActionListener sendListener = new ActionListener() {
 2
                 @Override
                public void actionPerformed(ActionEvent e) {
 4
                     String message = inputField.getText();
                     if (message != null && !message.equals("")) {
 5
 6
                         try {
 7
                             outWriter.println("Client " + ID + ": " + message);
 8
                             outWriter.flush();
9
                         } catch (Exception exc) {
10
                             exc.printStackTrace();
11
                         }
12
                     } else {
                         JOptionPane.showMessageDialog(null, "Message should not be
13
    empty!");
14
15
                     inputField.setText("");
16
                 }
17
            };
```

接受服务器信息并输入到聊天框:

```
1
            @Override
            public void run() {
 2
 3
                try {
 4
                     while ((message = inReader.readLine()) != null) {
5
                         if (!ID_set) {
                             String[] str = message.split(" ");
 6
7
                             if(str[0].equals("[Server]"))
8
9
                                 ID = Integer.parseInt(str[2]);
10
                                 setTitle("FreeChat Client " + ID);
```

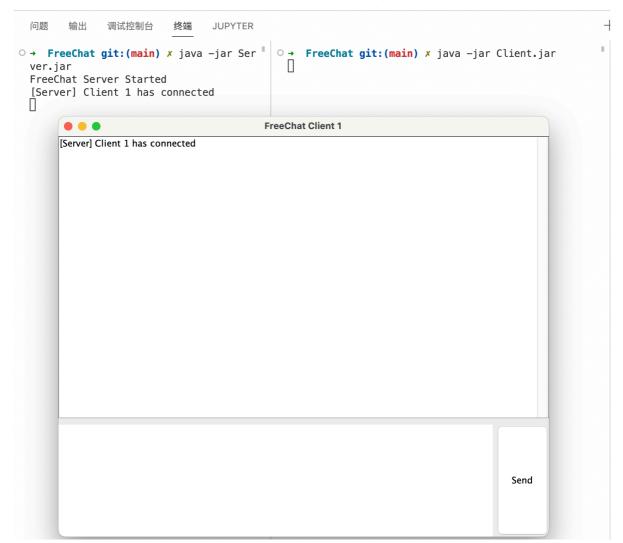
```
11
                                  ID set = true;
12
                              }
13
                          }
                         msgArea.append(message + "\n");
14
15
16
                 } catch (Exception exc) {
17
                     exc.printStackTrace();
18
                 }
             }
19
```

四、实验结果和分析

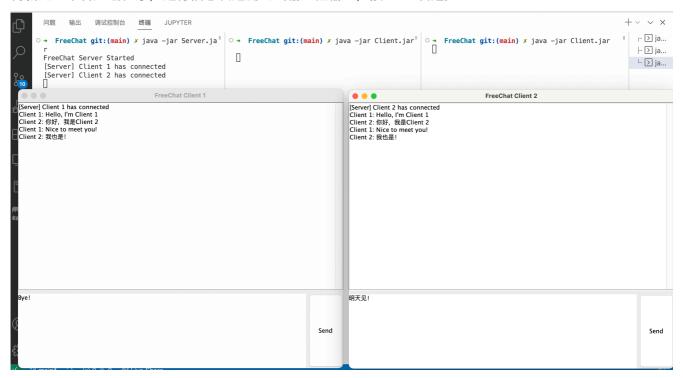
● 运行服务器程序

```
→ FreeChat git:(main) x java -jar Server.jar
FreeChat Server Started
```

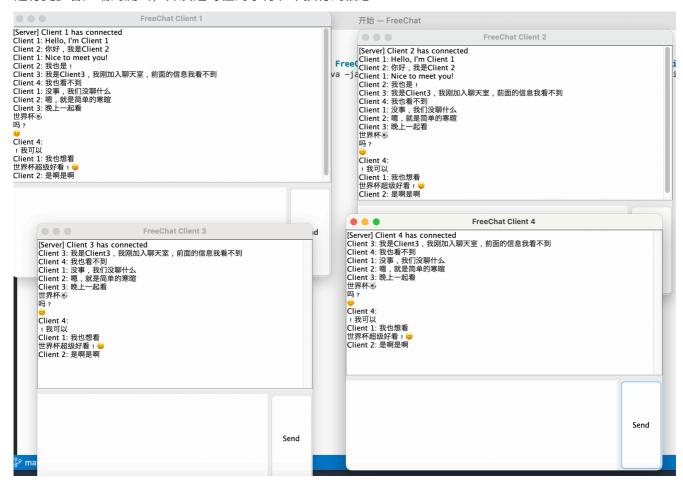
● 新建终端,运行客户端程序。可以看到服务器有广播信息打印出来,客户端窗口弹出。



● 再新建一个客户端程序,进行信息发送测试(输入框输入,按Send发送)



● 进行更多客户端的测试,并发送奇怪的字符、带换行的信息



注意新进入聊天室的用户无法获得之前的聊天内容,并且一旦退出就无法以同样的id号再次进入聊天室,只能获得递增的新id号。

五、实验心得

本实验主要涉及的知识点是线程、Socket编程。

难点主要在于如何设计客户端和用户端的信息交流方式,比如客户端需要从服务器端获取自己应有的ID号,这里通过特定格式字符串的单词提取完成。有一个未解决的问题是我想要能够接受客户端断开连接的信息,然后额外开辟一个记录在线人员名单的窗口,还没找到有效的方法。

不过总体而言设计这么一个聊天室程序还是挺有意思的。业界真正的应用还要考虑更多的问题,如并发、性能优化、跟数据库的结合等等。