1. 工程

本案例使用的是 C/S 结构，项目分为 ChatServer 和 ChatClient 两个工程

- **ChatServer：实现了局域网通信服务器功能（服务端）**
- **ChatClient：实现了局域网通信客户端功能（客户端）**

本案例为纯 Java 项目，没有数据库支持，没有数据持久化要求

JDK：JDK17

开发工具：Eclipse

使用其他版本 JDK 或者开发工具，可能需要手动复制代码至工程中

先启动 ChatServer，点击“启动服务”按钮

再启动 ChatClient，进行连接

2. 功能说明

（1） 登入系统

用户在客户端登入界面填写一个任意的“用户名”登入系统，在“登入系统”时服务器会根据所有当前“在线用户”进行检测，保证用户名是唯一的。

（2） 在线用户列表

用户登入系统后，会看到所有“在线用户”的列表，”在线用户“列表每隔一段时间会自动刷新

（3） 发送消息

用户双击“在线用户“列表的某个用户名时，会打开与该在线用户的“聊天窗口”，在聊天窗口中可以发送文本消息给对方

（4） 接收消息

用户在“聊天窗口”与其他用户通信时，也可以接收到对方发来的消息，该消息会

显示在聊天窗口的界面中。

如果其他用户发来消息时，“聊天窗口”没有打开，则需要“在线用户列表”中该用户的名字给出高亮显示

（5） 退出系统

用户关闭“登入界面”，“用户列表界面”界面时，会退出系统

用户关闭“聊天窗口”界面时，不会退出系统

3. 任务

（1） 请求码（已完成）

参考 ChatServer 和 ChatClient 工程 com.neusoft.util.CodeDefine 类中定义和注释

```
package com.neusoft.util;

//定义了客户端与服务端请求操作的代码
//客户端发起请求时指定代码，服务器端接收后解析代码进行相应的处理
public class CodeDefine {

    public static final int SIGN_IN = 1001;    //进入系统
    public static final int ONLINE_LIST = 1002; //更新在线列表和获取新的消息
    public static final int SEND_MESSAGE = 1003; //客户端发送消息
    public static final int GET_MESSAGE = 1004; //客户端接收消息
    public static final int SIGN_OFF = 1005; //退出系统

}
```

（2） 客户端连接工具类（已完成）

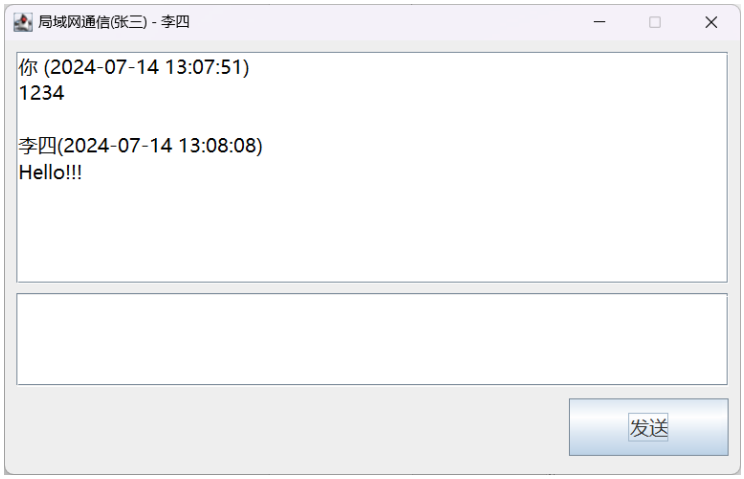
参考 ChatClient 工程 com.neusoft.service.SocketService 类中定义和注

（3） 界面（已完成）

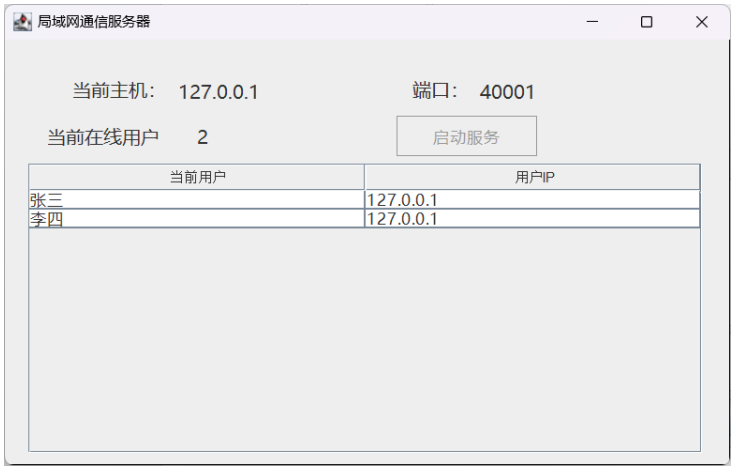
客户端 - 登入界面（com.neusoft.view.SignInWindow）

客户端 - 用户列表界面（com.neusoft.view.MainWindow）

客户端 – 聊天窗口界面（com.neusoft.view.ChatWindow）



服务器 – 控制台界面（com.neusoft.view.ServerListenerWindow）



（4） 登入请求（已完成：做为参考案例）

客户端:

在 com.neusoft.view.SignInWindow 类的“进入系统”按钮的点击事件中实现登入功能。(大约在第 72-99 行)

```
signIn.addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        //按钮点击事件
        //获取输入框内的文本
        String uname = userName.getText();
        //第一次向服务器请求，需要初始化Socket，连接服务器
        //（重要！！！）这个方法仅在第一次请求时调用，后续请求服务器无需调用(仅在此处调用！！)
        SocketService.init();

        //准备发送的请求消息
        //格式：请求码,登入系统的用户名
        String send = CodeDefine.SIGN_IN+","+uname;
        //将“请求消息（send）”发送给服务器，并接到服务器的处理结果
        String rec = SocketService.sendAndReciveInfo(send);
        //根据服务器的响应做出处理
        if(rec.equals("OK")) {
            //如果登入的用户名没有其他人使用
            //打开“主界面（MainWindow）”
            new MainWindow(uname);
            //隐藏当前界面（SignInWindow）
            frame.dispose();
        }else {
            //如果登入的用户名已被他人使用
```

服务器:

在 com.neusoft.service.UserThread 类处理登录请求（大约在第 56-71 行）

```
if(code == CodeDefine.SIGN_IN) {
    //进入系统
    //消息格式： "1001,用户名"
    uname = ss[1];
    if(ServerListener.registSocket(uname, this)) {
        //用户名没有被使用，允许进入系统
        //记录连接IP
        host = socket.getInetAddress().getHostAddress();
        dos.writeUTF("OK"); //回应客户端OK
    }else {
        //用户名被使用
        dos.writeUTF("NG"); //回应客户端NG
        //本次连接结束
        break;
    }
```

（5） 刷新在线用户列表

客户端：

在 com.neusoft.view.MainWindow 中**构造方法中**编写并启动一个**线程**，每隔 10s 中自动向服务器发送请求

请求码：CodeDefine.ONLINE_LIST

请求格式：请求码,　　(请求码加一个英文逗号)

服务器响应格式：

用户名 1,用户名 2,……用户名 n　　（字符串类型，用户名间以英文逗号分隔）

```
public MainWindow(String uname) {
    //当前登录用户名（唯一标识）
    this.uname = uname;
    //初始化界面
    initialize();

    //更新用户列表线程：每隔10s向服务器申请获取在线列表
    //请在此处新建一个线程处理更新用户列表功能
    //please code here
```

客户端需要将服务器响应的数据解析成字符串数组，然后刷新界面。

参考代码如下，其中 rec 为服务器响应的字符串

```
//将数据组织成字符串数组
users = rec.split(",");
//更新在线用户列表
synchronized (userList) {
    userList.setListData(users);
    userList.repaint();
}
```

服务端：

在 com.neusoft.service.UserThread 的 run 方法中设置相应的处理分支进行处理

在线用户的数据可以通过下方代码获取

```
Set<String> set = ServerListener.getOnlineSet();
```

（6） 发送消息

客户端：

在 com.neusoft.view.ChatWindow 中“发送”按钮的单机事件中（大约 81 行）进行处理

请求码：CodeDefine.SEND_MESSAGE

请求格式：请求码,接收者用户名,发送的消息　　(请求码加一个英文逗号)

服务器响应格式：OK　　（客户端接收到响应后无需处理）

获得文本区域的内容，参考代码如下，其中 inputArea 为输入框

```
String s = inputArea.getText();
```

设置文本区域的内容，参考代码如下，其中 inputArea 为输入框

```
inputArea.setText("");
```

界面中有两个文本区域

- **messageArea：上方用于显示对话信息的只读文本区域**
- **inputArea：下方用于输入通信内容的文本区域**

服务端：

在 com.neusoft.service.UserThread 的 run 方法中设置相应的处理分支进行处理

在解析消息后需要将消息封装成 Message 对象，存入未读消息列表中，

参考代码如下，uname 是发送者，message 是消息内容，to_user 是接收者

```
//生成消息对象（发送者,消息内容,服务器时间）
Message m = new Message(uname, message, TimeUtil.getCurrentTimeString());
//获取"接收者"的未读消息列表
List<Message> list = ServerListener.getUserMessageList(to_user);
//将消息存入未读消息列表中
list.add(m);
```

（7） 接收消息

客户端：

在 com.neusoft.view.MainWindow 中**已经编写好了相关线程**，每隔 4s 中自动向服务器发送请求

请求码：CodeDefine.GET_MESSAGE

请求格式：请求码,　　(请求码加一个英文逗号)

服务器响应格式：消息 1;消息 2;…　(多个消息间以英文分号分隔)

单个消息格式：来自用户,消息内容,发送时间　（单个消息内以英文逗号分隔）

客户端相应的解析程序已经编写完毕，只需要接收到正确格式服务端响应即可

```
//获取未读消息线程：每隔4s向服务器申请获取自己的未读消息
new Thread(new Runnable() {

    @Override
    public void run() {
        while(true) {
            //向服务器请求所有他人发送给自己的未读消息
            //请在此处发起请求向服务器读取自己的未读消息列表，返回的字符串使用变量mes接收
            //please code here
            String mes = "";
            //解析服务端的响应
            if(mes != null && !mes.isEmpty()) {
                String[] ms = mes.split(";");
                if(ms != null && ms.length > 0) {
                    for(String ss : ms) {
                        if(ss != null && !ss.isEmpty()) {
                            String[] xs = ss.split(",");
                            String from_user = xs[0];  //来自用户（消息发送者）
```

服务端：

在 com.neusoft.service.UserThread 的 run 方法中设置相应的处理分支进行处理

com.neusoft.service.UserThread 类中的 mlist 属性即为所有未读的消息

（8） 退出系统

客户端：

在 com.neusoft.view.MainWindow 中 **initialize()**，中，为 frame 窗体对象设置了 windowClosing 事件，在客户端点击右上角关闭按钮时，通知服务器结束。否则服务器因为客户端单方失联而无法接收服务线程，造成不断的抛出异常。

请求码：CodeDefine.SIGN_OFF

请求格式：请求码,　　(请求码加一个英文逗号)

服务器响应格式：OK

代码编写位置：

```
frame.addWindowListener(new WindowAdapter() {
    @Override
    public void windowClosing(WindowEvent e) {
        //退出系统
        //please code here
    }
});
```

服务端：

在 com.neusoft.service.UserThread 的 run 方法中设置相应的处理分支进行处理

服务端在退出服务线程前，需要在“用户-线程”映射中注销用户，参考代码如下

```
ServerListener.removeSocket(uname);
```

给出客户端响应后，通过 break 结束死循环，服务线程 UserThread 会自然结束，