

北京邮电大学

Java 大作业文档



姓 名 谢牧航
学 院 计算机学院
班 级 2022211301
学 号 2022211361

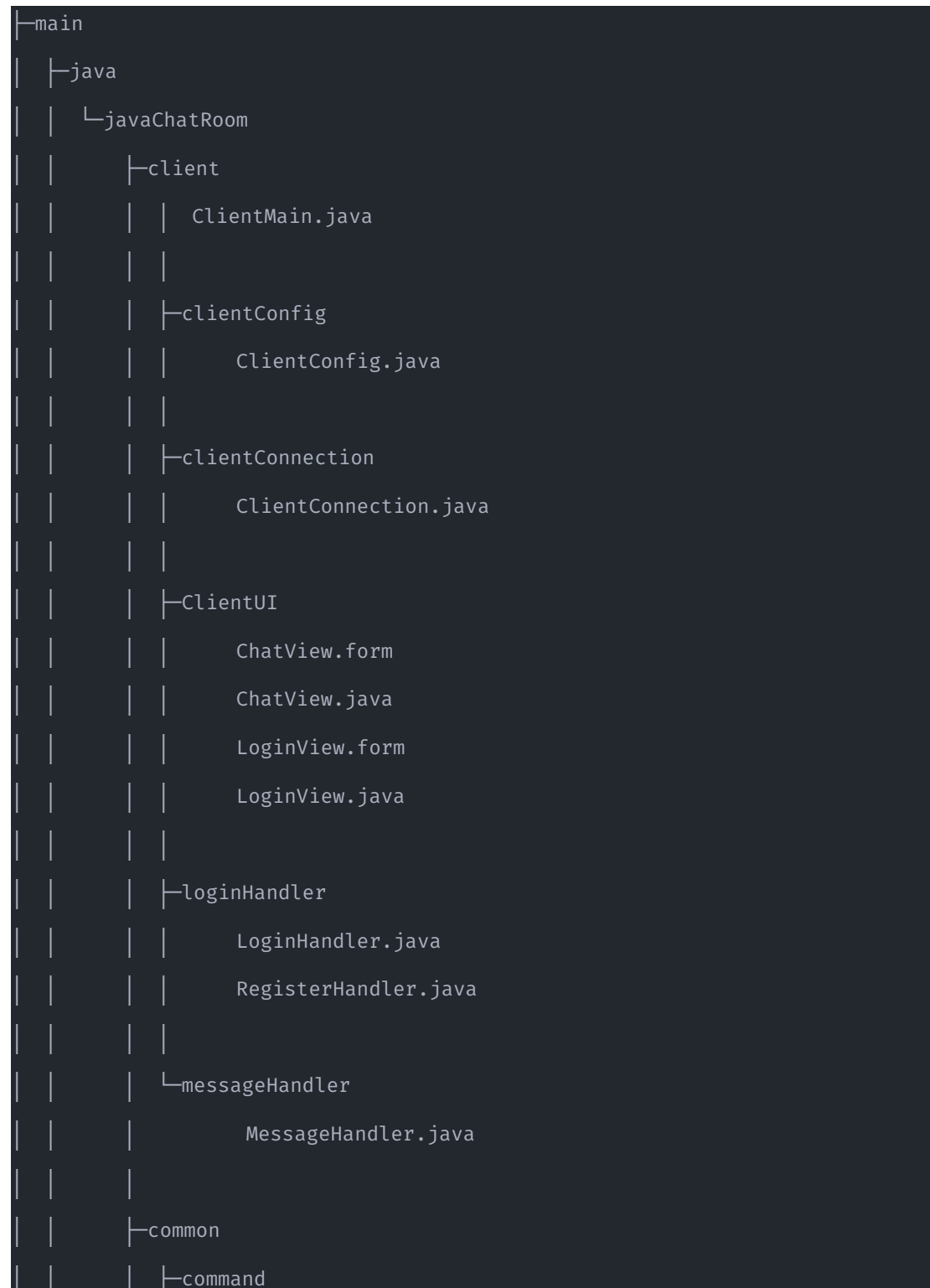
2024 年 7 月

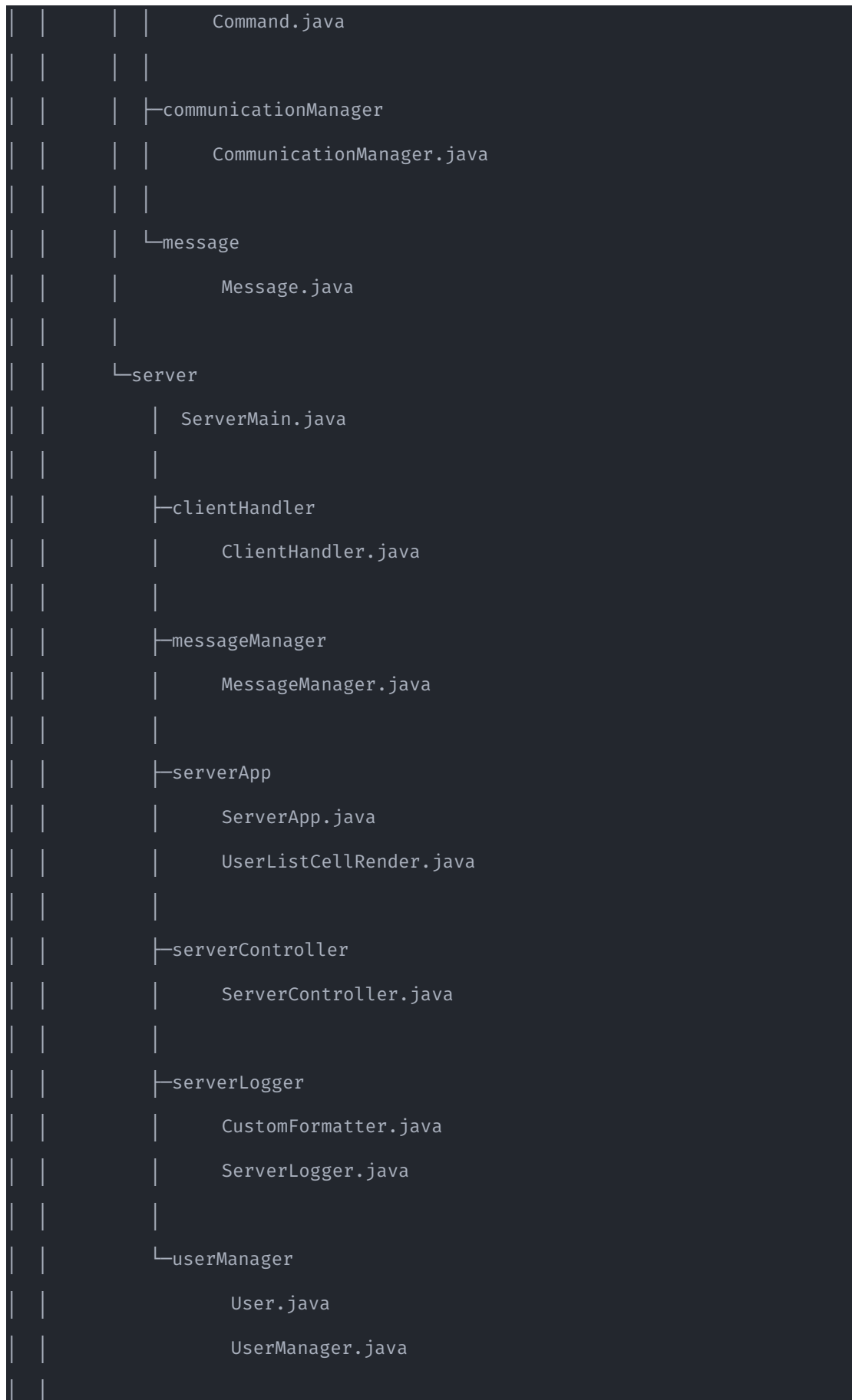
目录

一、设计思路	2
common 包	4
server 包	4
client 包	6
test 包	7
二、使用文档	8
服务器端	8
客户端	11
三、未来改进的部分	20
四、作业经验和心得	21

一、设计思路

聊天室的整体思路遵循 C/S 设计思路，项目源码目录 src 的结构如下：







我们来逐部分分析每部分的设计思路和类的关系。

common 包

这个包包含了服务器端和客户端都需要使用到的类。实际部署中，两端应该都包含这些类。

- command 包

这个包包含的 **Command** 类，规定了指令信息的内容和类型。两方直接通过传输对象来传输命令。

- message 包

这个包包含的 **Message** 类，规定了普通信息的发送人、接收人、内容、是否广播、是否匿名。两方直接通过传输对象来传输信息。

- communicationManager 包

这个包包含的 **CommunicationManager** 类主要包装了 **Socket** 传输的对象输入输出流，使得两端能方便的传输对象，同时捕获和处理错误。

server 包

这个包主要包含了服务器端特有的类。

- userManager 包

这个包包含的 **User** 类规定了用户所包含的内容，包含用户名和密码。

这个包包含的 **UserManager** 类主要是对用户数据的管理，提供了如下方法：从文件中读取用户列表和保存用户列表到文件；提供登录鉴权和注册服务；返回在线用户列表和所有用户列表；返回在线用户用户名和对应的用户端处理类（**ClientHandler**）的映射 **Map**。

- **messageHandler** 包

这个包包含的 **MessageHandler** 类主要作用是加载或保存服务器端所有的信息列表，用于显示历史聊天信息。同时，该类还可以筛选所有与该用户相关的信息，可用于用户登录时显示历史聊天信息。这个功能目前还没有整合进 **ServerMain** 里，但是接口方法已经写好。因为助教老师在群里说暂时不需要这个功能，所有就没有加。

- **clientHandler** 包

这个包包含的 **ClientHandler** 非常重要，是服务器端处理与客户端通信的类。当启动它时，服务器端会新建一个线程与客户端建立 **Socket** 连接。这个类包含了所有需要和客户端通信的功能，包含信息的接受、识别、处理和转发；指令的接受、识别和处理。在处理中，我设置一个登录的用户对应一个 **ClientHandler**，这个映射关系储存在 **UserManager** 里。所以只需要知道用户名，服务器便可以找到对应的连接，方便服务器消息的转发。这里消息的转发都是使用上文提过的 **CommunicationManager** 完成的。

- **serverController** 包

这个包包含的 **ServerController** 类保存了服务器的 **Socket**，和服务器的 **UserManager**，同时提供了服务器启停服务。相当于服务器端的 **UI** 界面都是使用的 **serverController** 里的方法。这个类相当于服务器 **UI** 界面的“后端”。

- **serverApp** 包

这个包包含的 **ServerApp** 类提供了服务器端的 **UI** 界面，架构使用 **JavaSwing**。具体外观见使用文档-服务器端。这个包包含的 **UserListCellRender** 类提供了显示用户列表时的样式。

- **serverLogger** 包

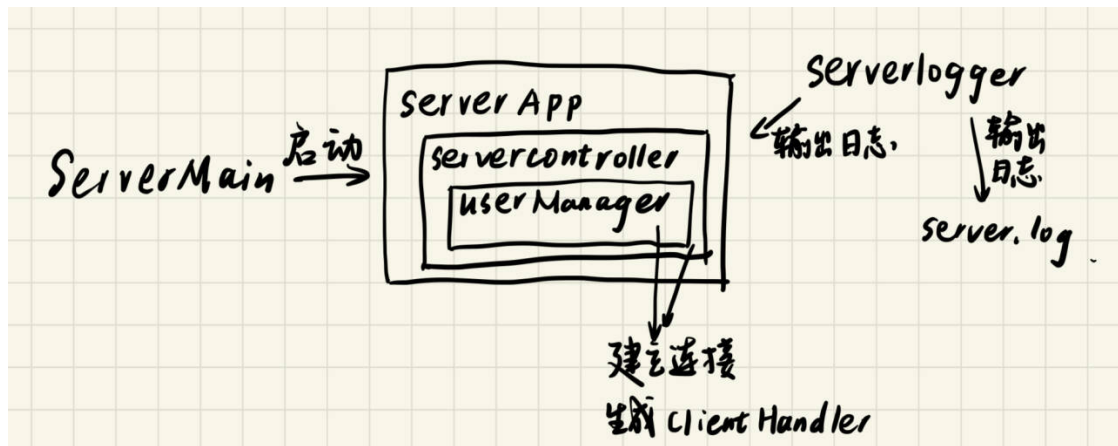
这个包包含的 **ServerLogger** 类主要是处理服务器端的日志生成和保存。这里我选择包装了 **java.util.logging** 包中的 **log** 功能，将日志分为 **Debug**、**Info**、**Warning**、**Error** 四个级别，会显示服务器的启停、用户的登入登出、报错等信息。日志会同时输出至 **UI** 界面对应的 **JTextArea logArea** 和项目根目录下的 **server.log** 中。

这个包包含的 **CustomFormatter** 类规定了日志显示的格式。

- **ServerMain** 类

这是启动服务器端的入口，运行 **main** 方法来启动服务器端程序。

服务器端的类之间的大致关系如下：



client 包

- clientConfig 包

这个包包含的 ClientConfig 类保存了当前客户端的配置状态，包含：当前登录用户的用户名，是否为广播模式，是否为匿名模式。在之后，匿名模式的切换将不会转发到服务器端处理，而是客户端本地处理，即改变 ClientConfig 的状态。

- clientConnection 包

这个包包含的 ClientConnection 类是用于和服务器端 Socket 建立连接并通信的，其包装了 CommunicationManager 的功能，并保存服务器端的 IP 地址和端口。

- loginHandler 包

这个包包含的 LoginHandler 类用于向服务器发送登录请求，并解析服务器返回的值，并得到登录成功或失败的结果

这个包包含的 RegisterHandler 类效果类似，不过这个类是用来处理注册请求的。

- messageHandler 包

这个包包含的 MessageHandler 类主要是客户端本地对信息的处理。在这里，客户端将分为两个线程：一个用于本地处理信息并发送给服务器，一个用于从服务器接受信息。信息和命令经过处理后，将正确的发送给服务器，并通过 ChatView 的 displayMessage() 方法将收到的信息显示或将发出的信息回显在 ChatView 的 messageWindowPanel 中。

- clientUI 包

这个包包含的 LoginView 类产生了登录界面的 UI，并在登录成功时启动 ChatView 和 MessageHandler。

这个包包含的 ChatView 类产生了聊天室界面的 UI。这个类还包含了一个 MessageHandler，用于接下来的信息收发。同时还会根据 MessageHandler 设置的不同消息等级，显示的消息会有不同的颜色。

这里我们发现 MessageHandler 和 ChatView 类互相引用。假如直接在构造函数中引用，会导致在初始化的时候一直互相引用最终爆栈。经查询，我选择了先建立

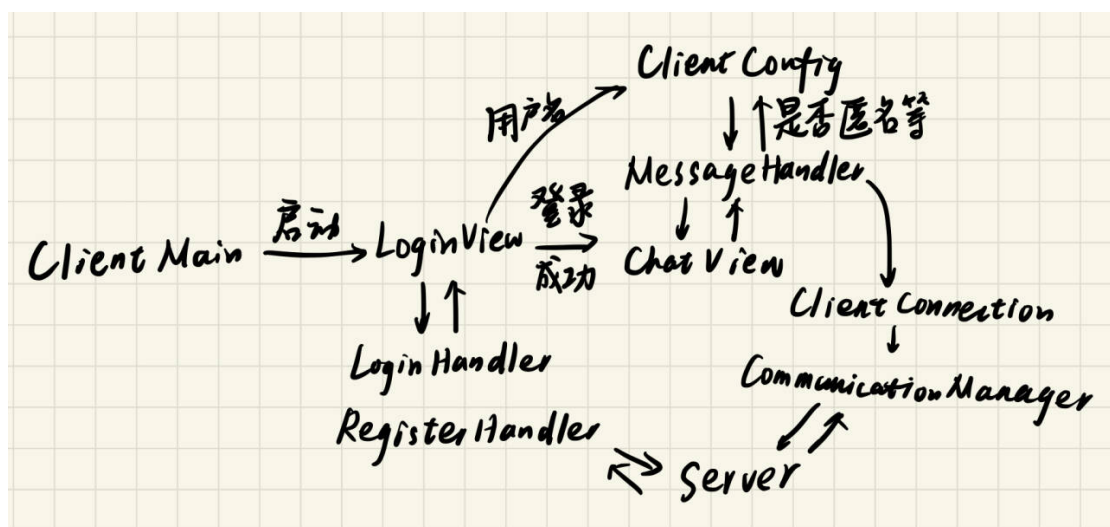
对象，再通过 set 方法来引用对方的方法来解决这个问题，大致流程如下：



● ClientMain 类

这是启动客户端的入口，运行 main 方法来启动客户端程序。

客户端的类的大致关系如下：



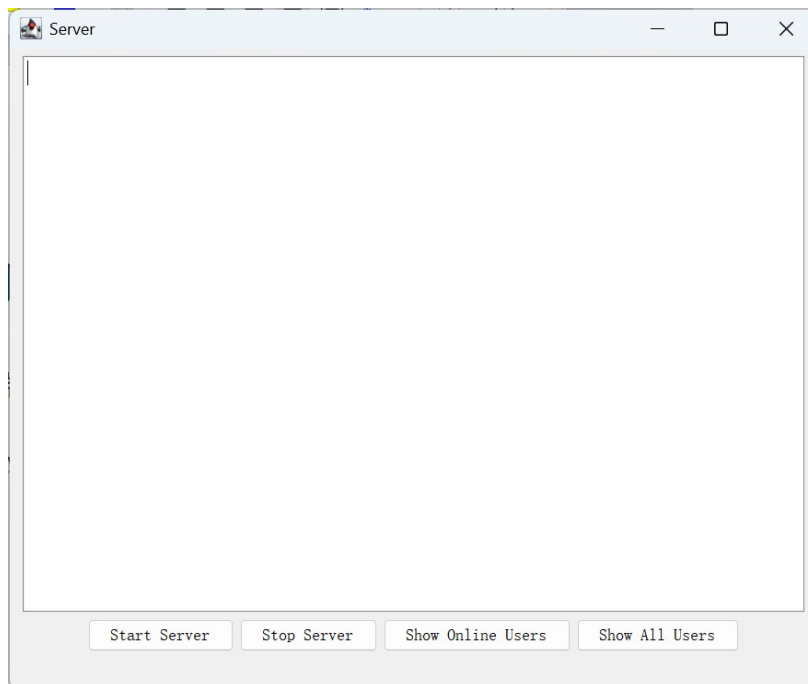
test 包

这个包包含的类主要是用于对程序进行单元测试所用。

二、使用文档

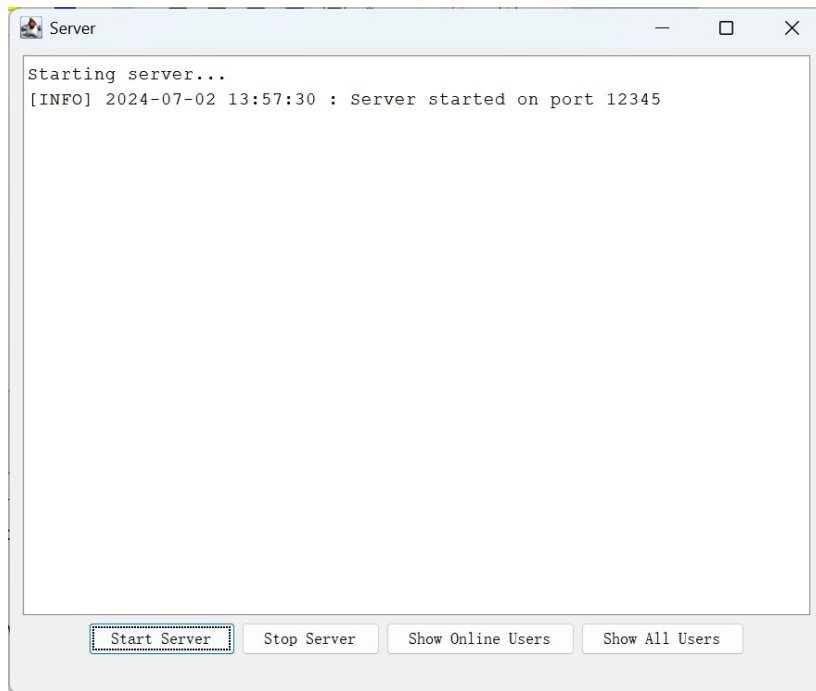
服务器端

在 IDEA 中运行 `ServerMain.java`，出现如下窗口。

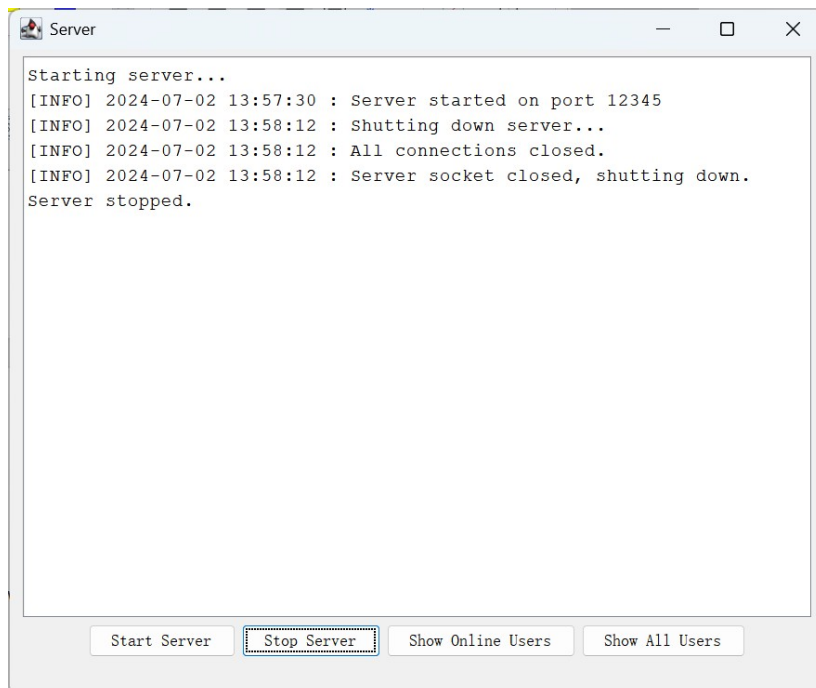


其下四个按钮作用为：

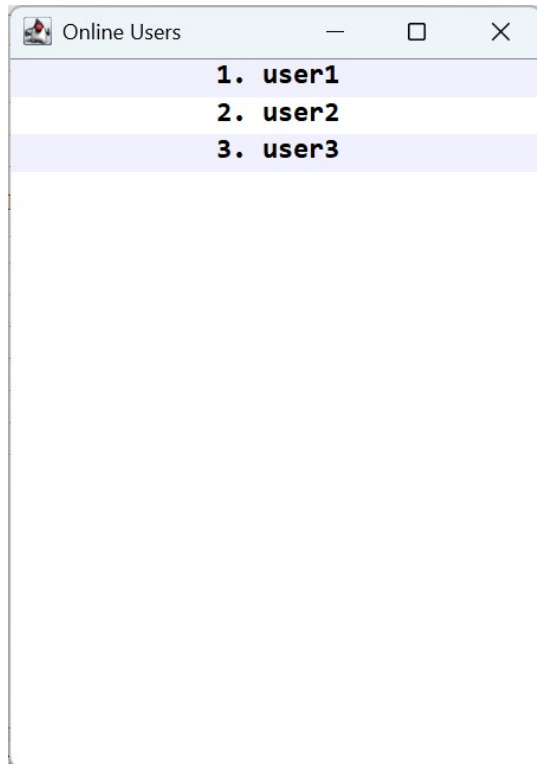
- **Start Server：** 启动服务器



- **Stop Server: 停止服务器**

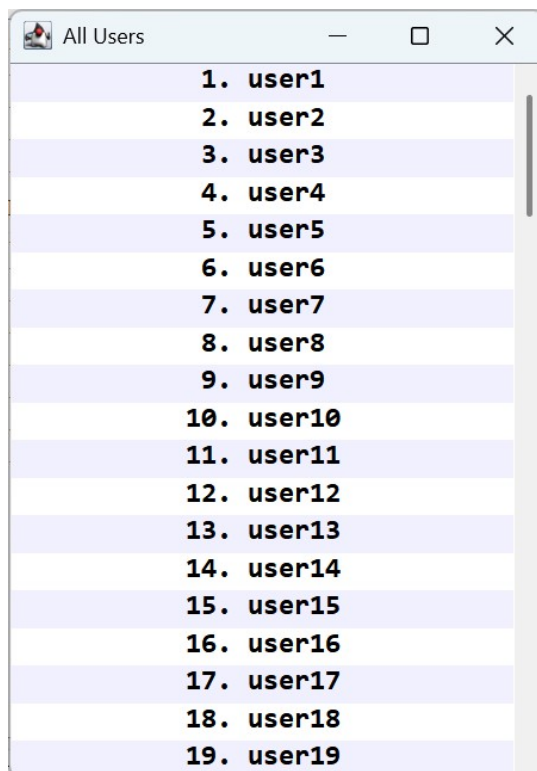


- **Show Online Users: 显示在线用户**



这个功能在服务器未启动的时候无法使用。如果没有用户在线的时候会显示 No user。

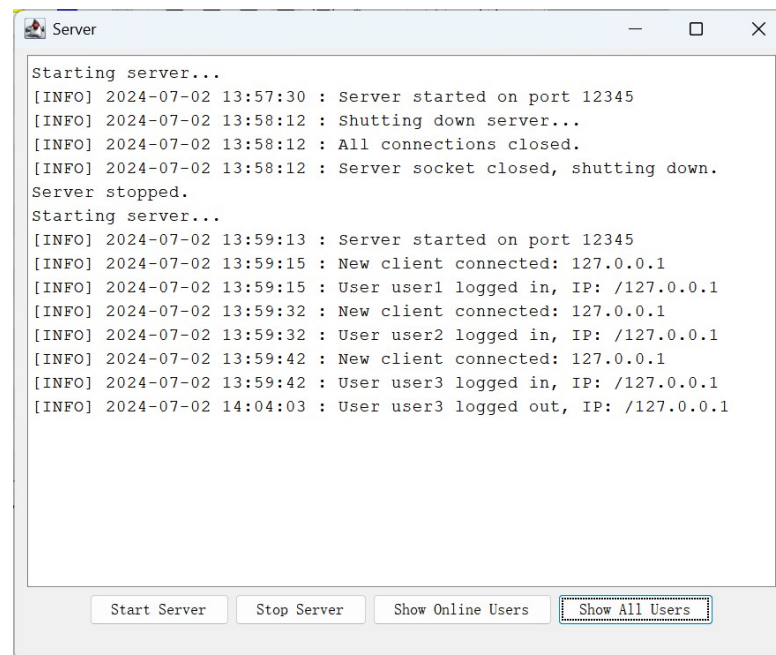
- 显示所有用户



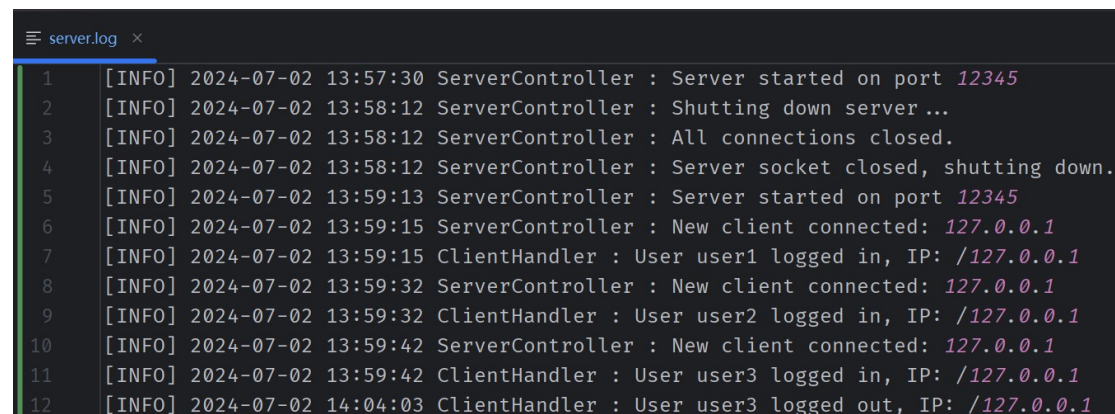
启动服务器的时候，会从项目根目录下的 user.ser 中反序列化读入所有用户名和密码（现有超过 100 个用户）。当然，通过客户端的注册功能也可以添加用户并

序列化保存到 `user.ser` 中。

- 日志功能



日志分为四个级别：**Debug**, **Info**, **Warning** 和 **Error**。包括服务器的启动和关闭，以及客户端的登入登出和错误信息都会在上方的日志区显示。同时，日志也会保存在项目根目录的 `server.log` 文件里。

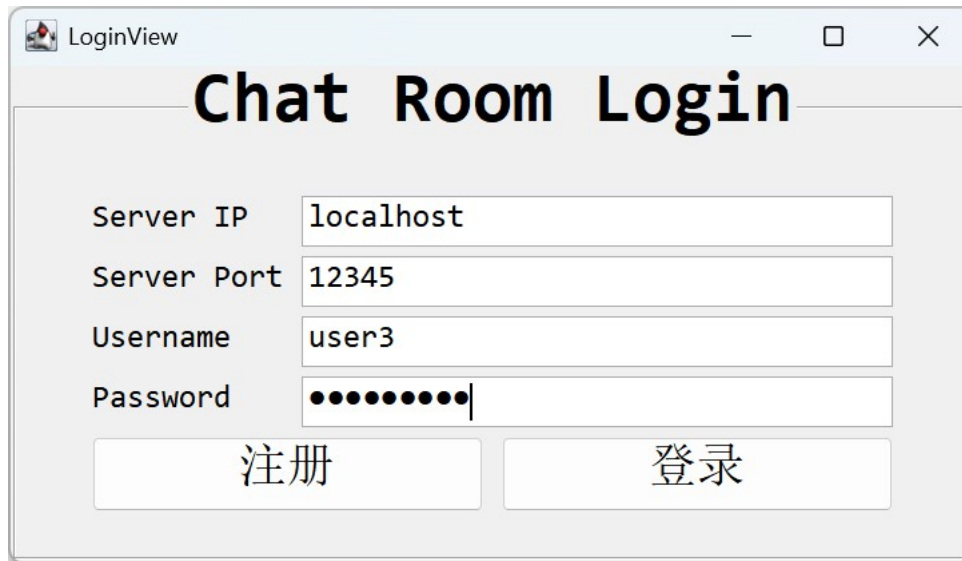


`server.log` 文件中显示的日志还会标出这个日志信息是由哪个类产生的。

客户端

在 IDEA 中运行 `ClientMain.java`，出现以下窗口。

1. 登录与注册



The screenshot shows a window titled "LoginView" with the subtitle "Chat Room Login". It contains four input fields: "Server IP" with the value "localhost", "Server Port" with the value "12345", "Username" with the value "user3", and "Password" with masked characters "••••••••". Below the fields are two buttons: "注册" (Register) and "登录" (Login).

首先需要输入服务器的 IP 和端口号（默认填写 localhost 和 12345）。然后输入用户名和密码。如果接下来点击注册按钮，则会注册新的账号。



The screenshot shows the same "Chat Room Login" window, but with a modal dialog box titled "Register Success" overlaid. The dialog contains an information icon, the text "Register successful.", and a "确定" (OK) button. The "注册" button in the background is highlighted in blue.

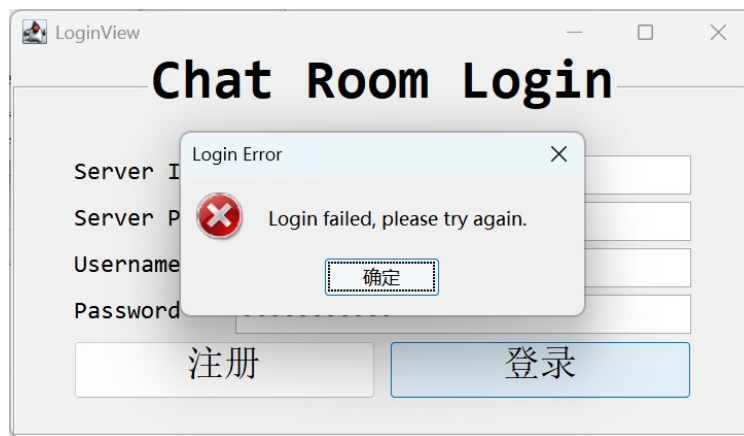
假如注册的用户名已被占用，则无法注册成功。



The screenshot shows the same "Chat Room Login" window, but with a modal dialog box titled "Register Error" overlaid. The dialog contains an error icon, the text "Register failed, please try again.", and a "确定" (OK) button. The "注册" button in the background is highlighted in blue.

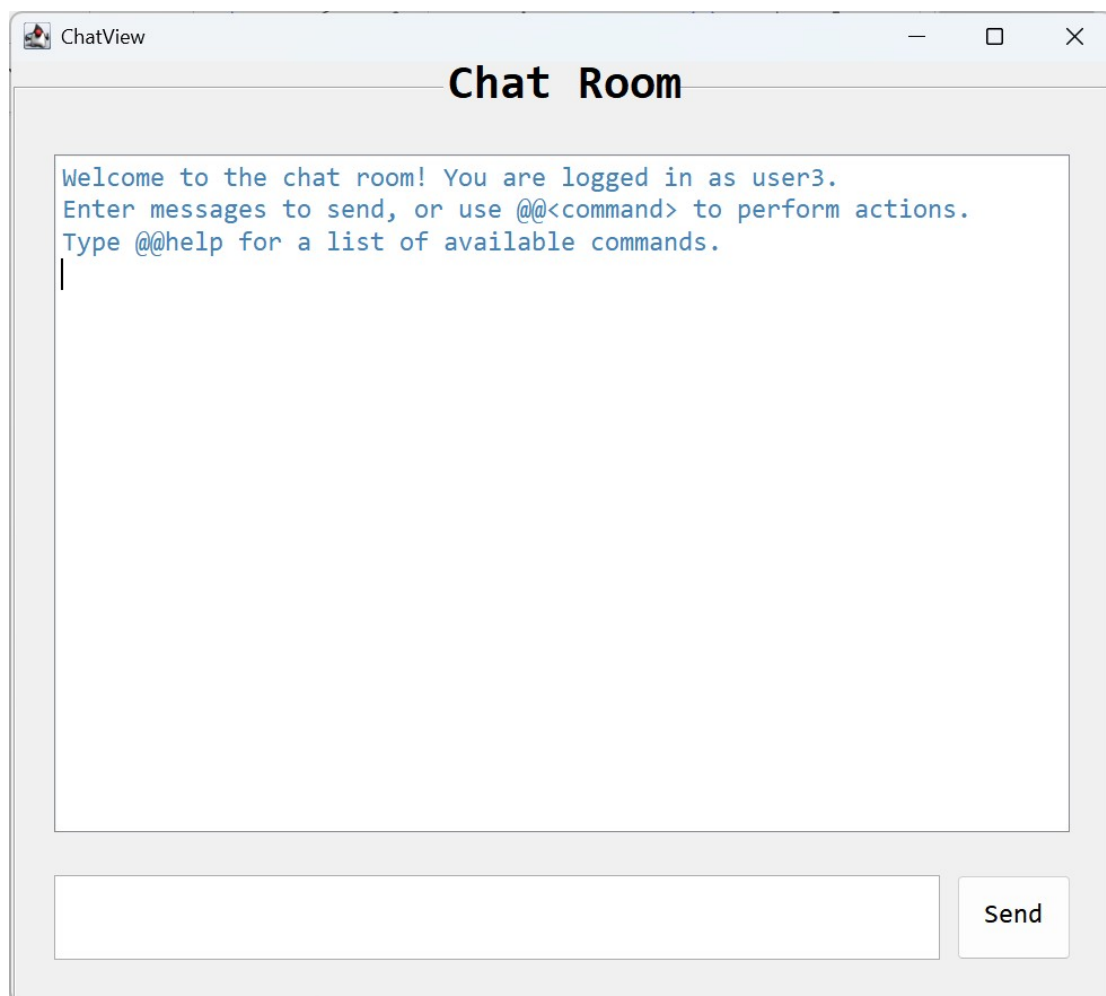
登录功能需要输入正确的用户名和密码，即可登录聊天室。

如果输入的用户已登录或账号密码错误则无法登录成功。



2. 聊天功能

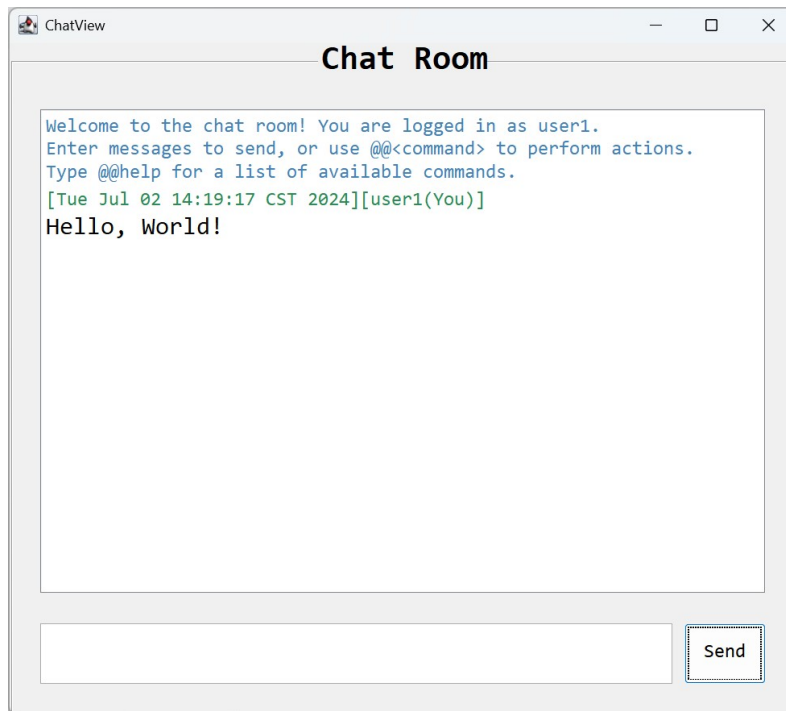
登录成功后，即可见到聊天室窗口。



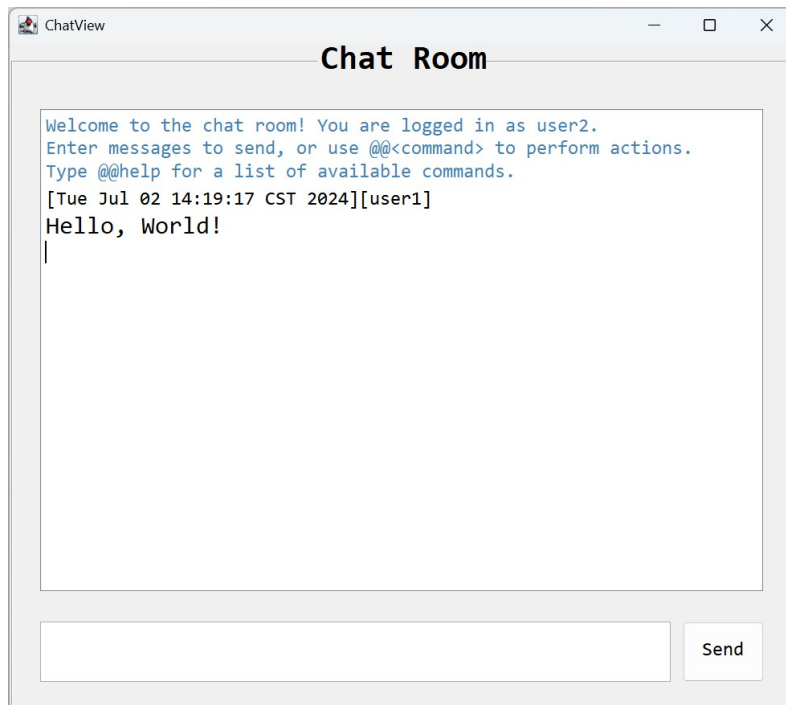
- 广播消息

直接在输入框内输入消息并点击发送键，即可发送广播消息。

发送人视角：



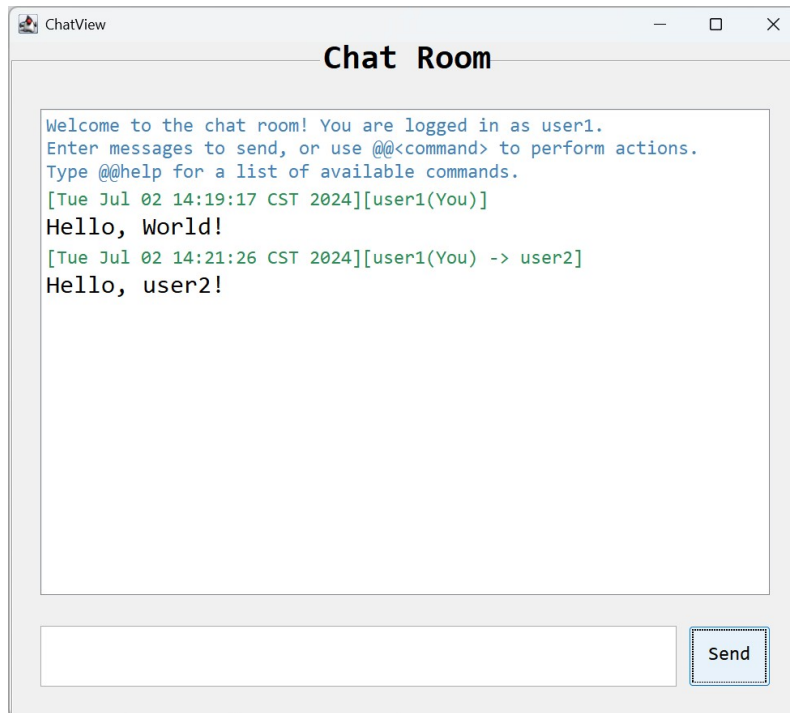
他人视角：



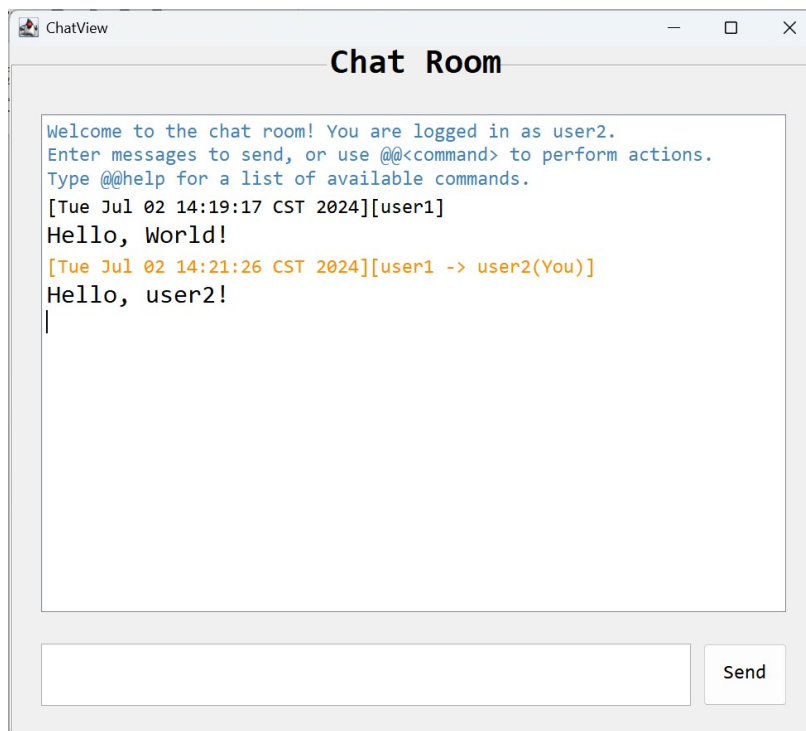
● 私聊消息

输入@<username> message，即可发送私聊消息。

发送人视角：

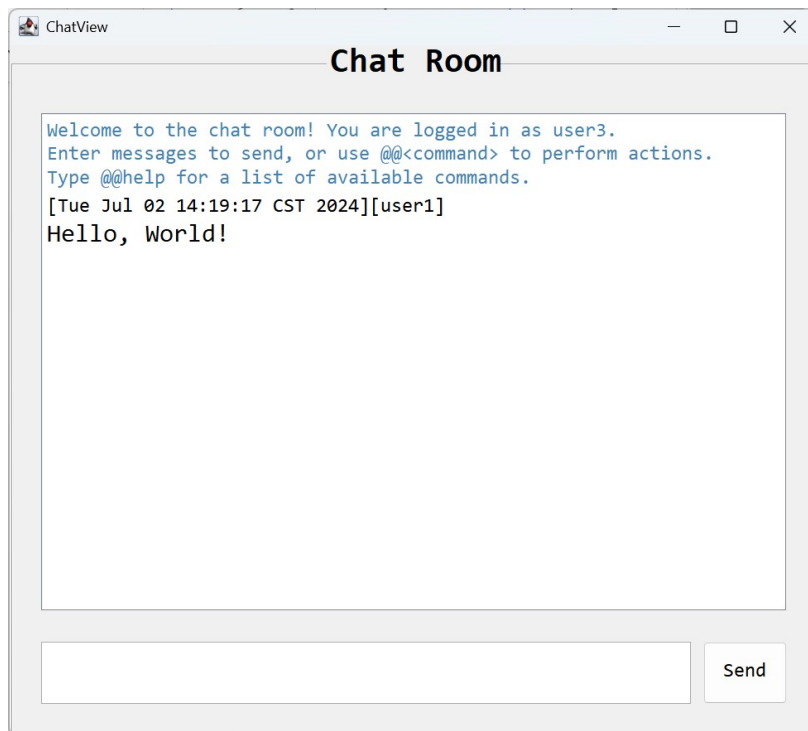


接收人视角:

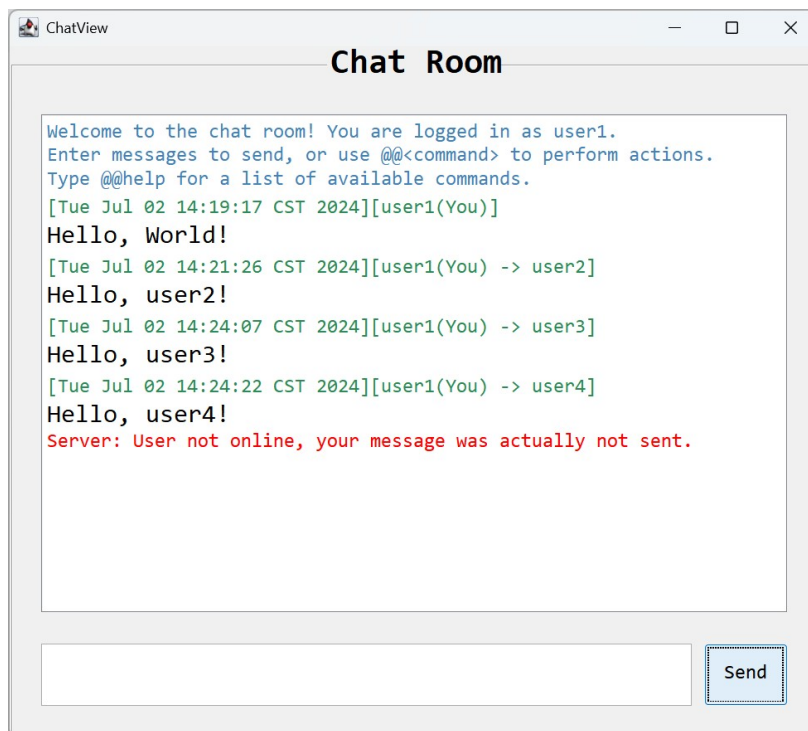


他人视角:

其他非接收人用户则无法看到此条消息。



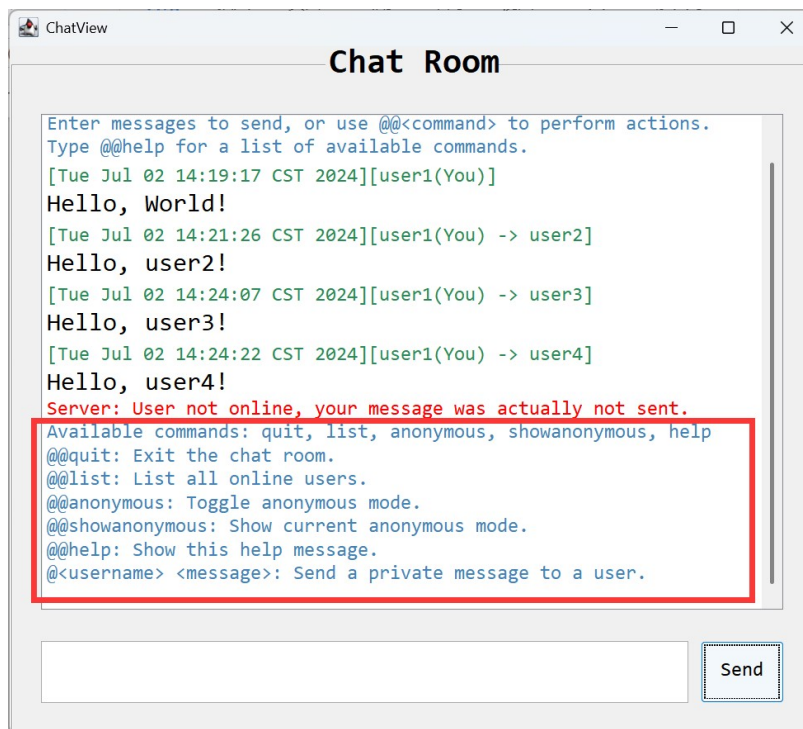
如果@的人不在线，则消息不会被发送并收到服务器的提醒。



3. 指令功能

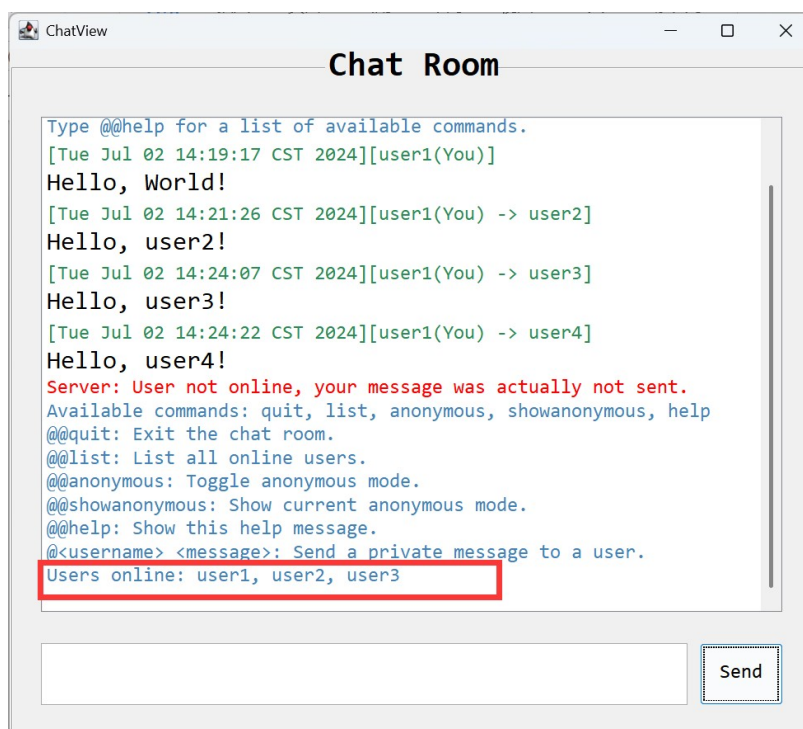
- 帮助指令。

发送 @@help 指令可以查看指令列表。



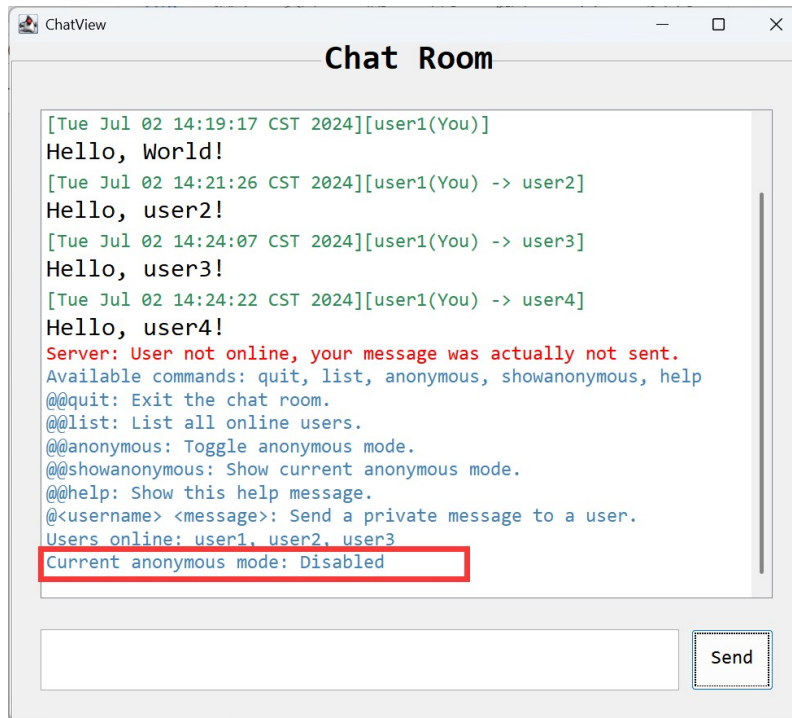
- 显示在线用户

发送 @@list 命令即可查看目前所有的在线用户。



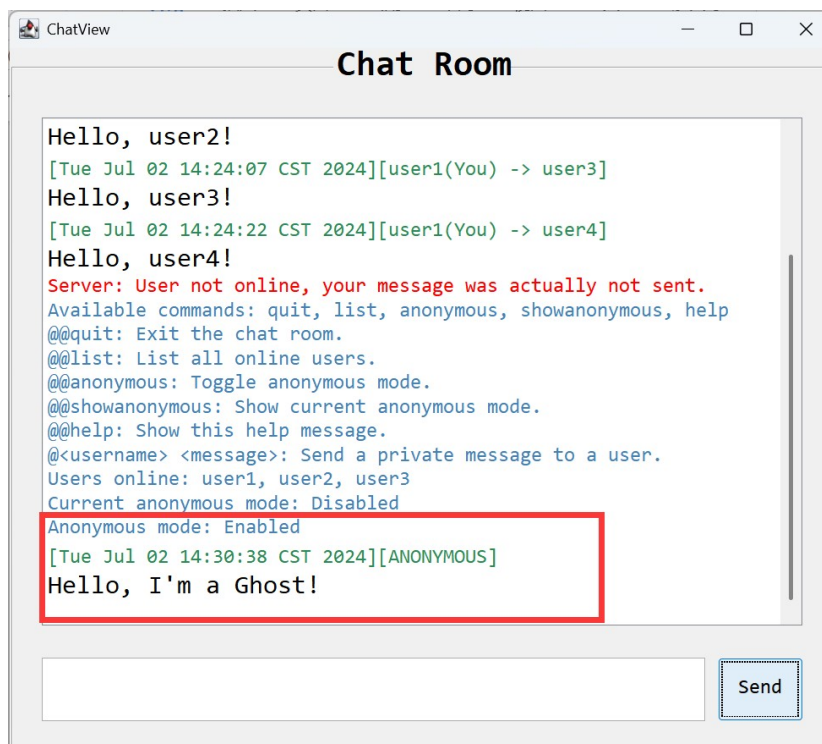
- 匿名功能

发送 @@showanonymous 命令即可查看当前是否处于匿名状态。

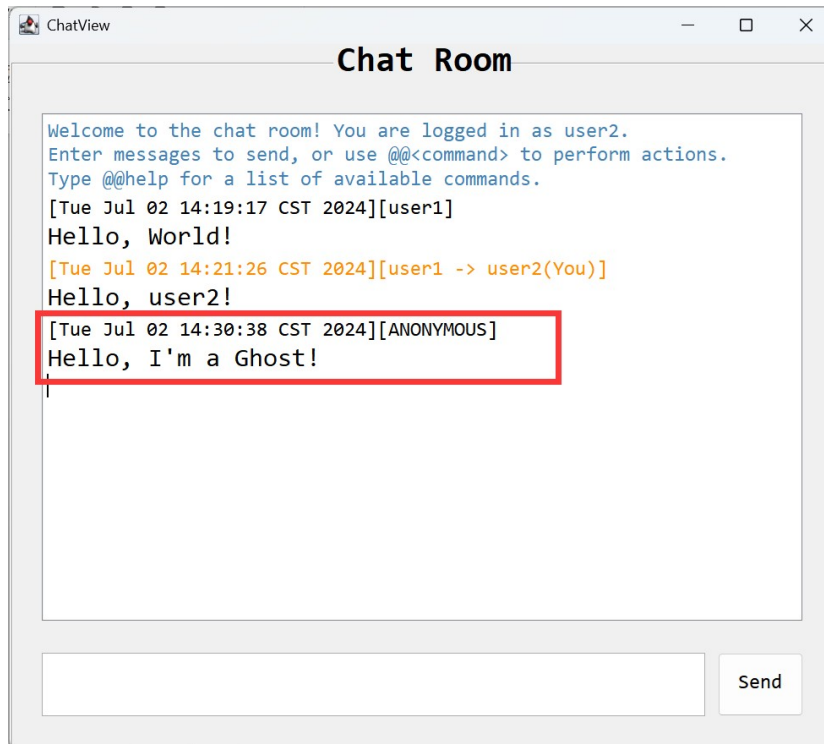


发送 @@anonymous 命令即可切换匿名状态。在匿名状态下，发送人的用户名将以 ANONYMOUS 显示。

发送人视角：

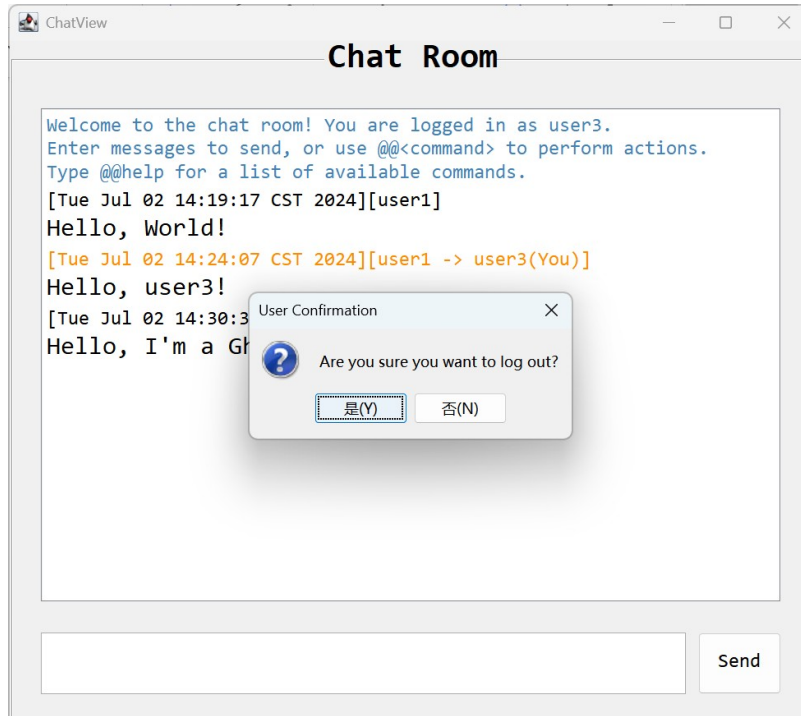


他人视角：



- 退出聊天室

可以通过发送 `@@quit` 命令退出聊天室，也可以通过直接关闭窗口来退出聊天室，客户端会自动与服务端断开连接。



三、未来改进的部分

1. 将 MessageManager 接入到服务器端程序中，做到历史聊天记录的显示。
2. 进一步优化客户端的 UI 界面，添加命令对应的按钮。
3. 将项目程序打包为可移动运行的 exe 文件。

四、作业经验和心得

在这次聊天室项目的开发过程中，我学到了很多宝贵的知识和技能，也遇到了不少挑战。整个项目从设计到实现，再到调试和优化，每一步都让我对软件开发有了更深的理解和认识。

设计思路

在项目开始时，我首先进行了详细的设计思路规划。通过遵循 C/S（客户端/服务器）设计模式，我能够更好地组织和管理项目代码。C/S 模式的应用不仅提升了项目的结构化程度，还为后续的功能扩展和维护提供了便利。在项目源码目录中，合理的包结构设计，如 `common` 包、`server` 包、`client` 包等，使得代码逻辑清晰，职责分明。

代码实现

在代码实现过程中，我特别注重代码的复用性和可维护性。例如，`common` 包中定义的 `Command` 类和 `Message` 类，既减少了代码冗余，又保证了服务器端和客户端之间的通信一致性。在 `CommunicationManager` 类的实现中，通过封装 `Socket` 传输的对象输入输出流，简化了对象的传输过程，同时捕获并处理了潜在的错误。

服务器端设计

服务器端的设计尤为重要。在服务器端，我实现了用户管理、信息处理、客户端连接处理等核心功能。其中，`ClientHandler` 类作为服务器端与客户端通信的关键类，负责接受、识别、处理和转发信息及指令。通过 `UserManager` 类，我能够方便地管理用户数据，并实现用户登录鉴权、注册服务等功能。`ServerLogger` 类则有效地记录了服务器端的运行日志，帮助我在调试和优化过程中及时发现和解决问题。

客户端设计

客户端的设计同样不容忽视。通过实现 `ClientConfig`、`ClientConnection`、`LoginHandler`、`MessageHandler` 等类，我完成了客户端的配置管理、服务器连接、登录注册处理和消息处理等功能。在 UI 设计方面，通过 `Java Swing` 实现了登录界面和聊天室界面，并结合 `MessageHandler` 类，实现了信息的收发和显示。

功能测试与调试

在完成代码实现后，我进行了全面的功能测试与调试。通过 JUnit 框架对关键类和方法进行了单元测试，确保代码的正确性和稳定性。同时，通过手动测试验证了客户端与服务器端的交互功能，确保聊天室功能正常可靠。

总结

总的来说，这次聊天室项目的开发让我受益匪浅。通过实践，我不仅巩固了所学知识，还培养了独立解决问题的能力。未来，我将继续努力，提升自己的编程技能和项目管理能力，迎接更多的挑战和机遇。

这次作业心得的撰写，也让我对整个 Java 面向对象的思想有了更加全面和深入的认识。希望在今后的学习和工作中，我能将这些宝贵的经验和教训应用到更多的实际项目中，持续提升自己的专业素养和实践能力。