

# 面向对象程序设计实践 (Java) 课程报告

## C/S 架构的简易聊天室

司睿洋

项目地址: [www.github.com/RuiyangSi/BUPT-JavaOOP-Chatroom](https://www.github.com/RuiyangSi/BUPT-JavaOOP-Chatroom)

### 摘要

本报告详细阐述了一个基于 C/S 架构的 Java 聊天室系统的设计与实现过程。项目采用面向对象编程思想，运用 `Socket` 网络编程技术实现客户端与服务器间的实时通信，通过 `Swing` 框架构建友好的图形用户界面，支持多用户并发聊天、私聊消息、匿名模式等核心功能。系统架构采用模块化设计，分为核心模块 (`core`)、服务器模块 (`server`)、客户端模块 (`client`) 和消息处理模块 (`messaging`)，确保了代码的可维护性和可扩展性。项目成功验证了面向对象设计原则在实际开发中的应用，展现了封装、继承、多态等核心概念的实践价值，为进一步的功能扩展和性能优化奠定了坚实基础。

# 目录

<b>1 设计思路</b>	<b>3</b>
1.1 需求分析	3
1.2 系统设计	3
1.3 类的关系	7
1.4 核心类功能设计	8
1.4.1 核心模块 (core)	8
1.4.2 服务器模块 (server)	10
1.4.3 消息处理模块 (messaging)	10
1.4.4 客户端模块 (client)	11
<b>2 使用说明</b>	<b>12</b>
2.1 服务器端使用	12
2.2 客户端使用	13
<b>3 未来工作</b>	<b>15</b>
3.1 从明文到加密存储	15
3.2 支持多模态信息发送	15
<b>4 总结与反思</b>	<b>17</b>
<b>A 效果展示</b>	<b>18</b>

# 1 设计思路

本项目旨在设计并实现一个基于图形用户界面的客户端-服务器（C/S）架构聊天室系统。

项目采用自顶向下的设计理念，遵循面向对象软件工程的标准流程。首先在第 1.1 节中进行全面的需求分析，明确系统的功能边界和技术约束；接着在第 1.2 节中构建整体的系统架构，确立模块划分和技术选型；然后在第 1.3 节中深入分析类之间的关系设计，体现面向对象的核心思想——通过继承、组合、接口等机制建立“谁和谁有关系”的概念框架；最后在第 1.4 节中详细阐述各类的具体功能实现，深入“谁做什么”的实现细节。这种从抽象到具体、从整体到局部的设计思路，既符合 UML 建模的标准流程——先有类图展示关系结构，再有详细的类规约描述行为，也突出了面向对象设计的本质特征，确保系统架构的合理性和代码实现的可维护性。

## 1.1 需求分析

本系统采用客户端/服务器（C/S）架构进行设计。在该架构下，服务器端与客户端作为两个独立的模块进行开发，它们之间通过 Socket 协议进行网络通信。

服务器端的核心功能包括用户数据加载、命令处理、端口监听、用户登录验证、聊天通信、匿名模式以及日志记录，各功能的详细说明如表 1 所示。

客户端需要实现与服务器的连接建立、用户登录认证、消息显示、用户输入处理以及系统命令执行等核心功能，具体功能规格如表 2 所示。

基于上述需求分析，可以构建系统的用例图来描述各功能模块间的关系，如图 1 所示。

## 1.2 系统设计

基于图 1 所代表的系统用例图，系统采用分层模块化设计，遵循单一职责和分离关注点原则。整体架构分为四个核心模块：`core` 模块提供 `Message`、`User`、`Logger` 等基础数据结构和系统服务；`messaging` 模块封装 `MessageParser` 和 `MessageBroadcaster`，实现统一的消息解析和分发机制；`client` 模块包含 `ChatClient` 主控制器、`ChatFrame` 和 `LoginFrame` 界面组件、`MessageStyleRenderer` 渲染器和 `NetworkManager` 网络管理器，

表 1: 服务端功能需求

功能名称	解释
用户加载功能	服务器启动时从本地文件读取用户的用户名和密码
命令功能	服务器可接受命令 list (查看在线用户), listall (查看所有用户) 和 quit (关闭)
端口侦听功能	服务器启动后侦听某个指定端口, 等待连接
用户登录验证功能	对客户端提供的登录信息进行验证, 并提供反馈
聊天功能	支持至少 5 个客户端同时聊天
匿名功能	支持匿名聊天
日志功能	用文件记录用户的登录和退出事件信息

表 2: 客户端功能需求

功能名称	解释
连接服务器功能	客户端能够连接到服务器
登录功能	将登录信息提交服务器, 并展示服务器的反馈信息
消息展示功能	聊天室有提示符, 且能显示所有用户的非私聊信息和自己的私聊信息
用户输入消息功能	用户可以输入三类字符串: 系统命令 (@@ 开头)、私聊消息 (@+ 用户名开头) 和非私聊消息 (其他)
命令功能	可接受命令 list (查看当前在线用户), quit (退出), showanonymous (查看当前是否匿名) 和 anonymous (切换是否匿名)

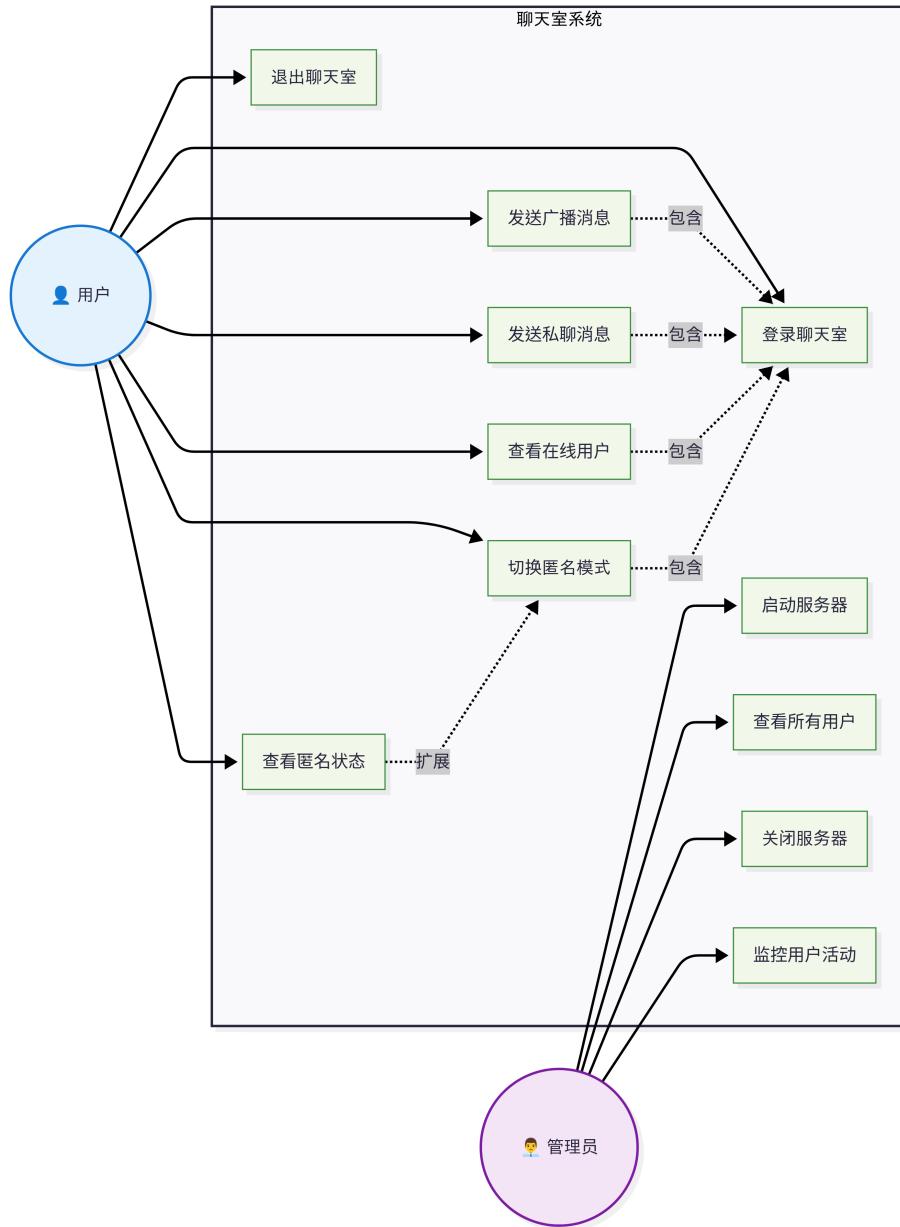


图 1: 系统用例图：本图描述了“聊天室系统”的功能需求模型。系统定义了“用户”和“管理员”两类角色。用户可以执行登录、退出、发送广播/私聊消息、查看在线用户以及切换匿名模式等操作。管理员则负责系统的维护与管理，包括启动/关闭服务器、查看所有用户和监控用户活动。图中还显示了部分功能（如发送消息）需要以“登录聊天室”为前提条件（包含关系）。

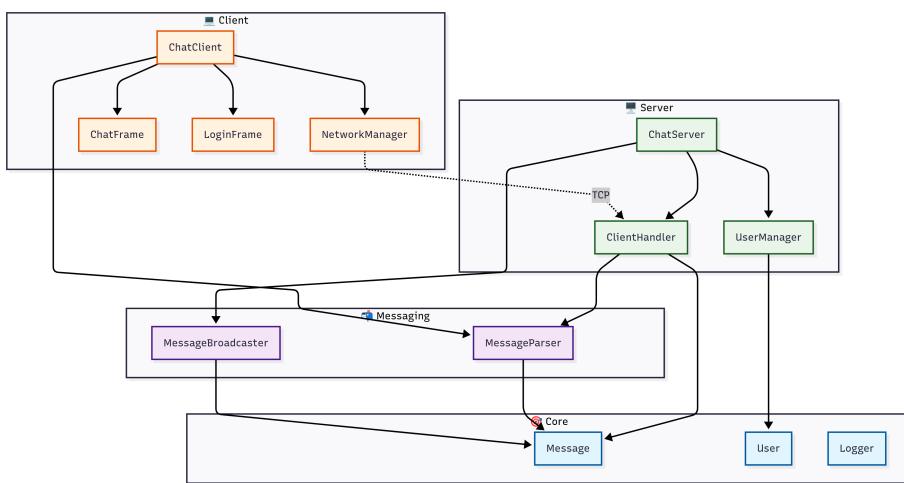


图 2: 系统架构图：该图展示了聊天室系统的分层架构设计，采用 C/S 模式通过 TCP Socket 通信。系统遵循分离关注点原则，将功能划分为客户端 (client)、服务器 (server)、消息处理 (messaging) 和核心基础设施 (core) 四个模块。客户端通过 NetworkManager 与服务器建立连接，服务器端使用 ClientHandler 处理并发连接并通过 MessageBroadcaster 实现消息分发，MessageParser 提供统一的消息解析服务，core 模块定义系统的基础数据结构。这种模块化设计实现了松耦合的系统架构，提高了代码的可维护性和可扩展性。

负责用户交互和网络通信；server 模块由 ChatServer、ClientHandler 和 UserManager 组成，处理并发连接、用户管理和消息路由。

系统设计的关键特性包括多线程并发处理、事件驱动架构和状态管理。ClientHandler 采用多线程处理并发用户连接，每个客户端连接对应一个独立的处理线程，确保系统可扩展性；MessageBroadcaster 实现消息的精确路由和广播，支持私聊和群聊功能；UserManager 提供完整的用户状态管理和持久化机制，维护用户在线状态和匿名状态。客户端与服务器通过 NetworkManager 建立 TCP 连接，使用 MessageParser 解析用户输入，通过 MessageStyleRenderer 进行消息的格式化显示。系统架构如图 2 所示，清晰展现了各模块间的依赖关系和数据流向。

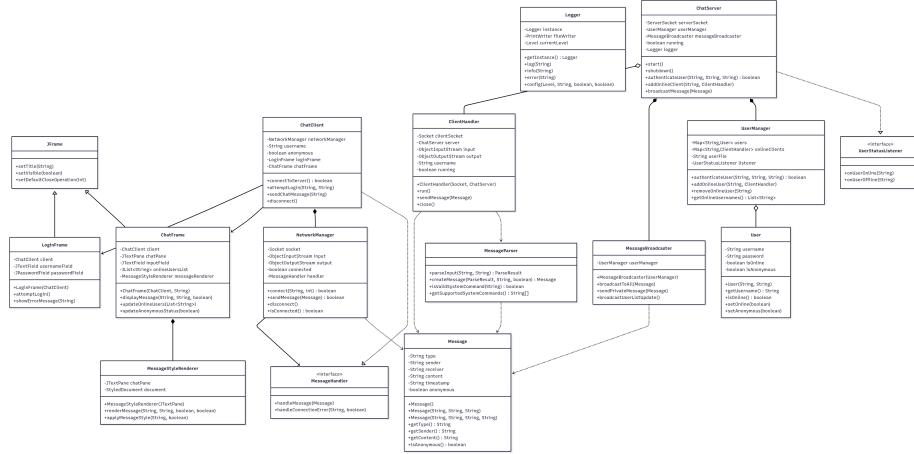


图 3：系统类图：该图采用标准 UML 表示法详细展现了聊天室系统的面向对象设计。图中清晰展示了各类的完整结构，包括私有属性（-）、公有方法（+）以及类间的标准关系。继承关系通过空心三角箭头表示（**JFrame** 为 **ChatFrame** 和 **LoginFrame** 的父类），组合关系用实心菱形标识强拥有关系，聚合关系用空心菱形表示弱拥有关系，依赖关系通过虚线箭头展现临时使用关系。整体设计遵循封装、继承、多态的面向对象原则，体现了高内聚低耦合的架构思想，确保了系统的可维护性和可扩展性。

### 1.3 类的关系

在确定了系统的整体架构后，我们进一步细化了各模块内部的类设计和实现细节。从用例图定义的功能需求，到架构图展现的模块结构，再到

类图描述的具体实现，形成了完整的设计层次。系统类图如图 3 所示，采用标准 UML 表示法展现了面向对象设计的核心要素。在继承设计方面，`ChatFrame` 和 `LoginFrame` 继承自 `JFrame`，复用了 Swing 框架的窗口功能，体现了代码重用的设计思想。在组合关系设计中，`ChatServer` 强拥有 `UserManager` 和 `MessageBroadcaster`，确保了服务器核心功能的内聚性；`ChatClient` 组合 `NetworkManager`，实现了网络通信的封装。接口设计方面，`MessageHandler` 和 `UserStatusListener` 接口的使用实现了观察者模式，提高了系统的可扩展性和解耦程度。整体设计严格遵循封装、继承、多态的面向对象原则，实现了高内聚低耦合的架构目标。

为了更清晰地展示系统各组件间的动态交互过程，图 4 展示了私聊消息处理的完整时序图，详细描述了从客户端输入到消息送达的全过程。

## 1.4 核心类功能设计

系统的核心功能通过精心设计的类层次结构实现，各类职责明确，协作有序。以下按模块详细介绍主要类的功能设计与实现特点。

### 1.4.1 核心模块 (core)

**User 类** 作为用户实体的数据封装类，`User` 类采用简洁的 POJO 设计模式，封装了用户的基本属性：用户名 (`username`) 作为唯一标识、密码 (`password`) 用于身份验证、在线状态 (`online`) 和 IP 地址 (`ipAddress`) 用于状态管理。该类重写了 `equals()` 和 `hashCode()` 方法，确保基于用户名的对象比较和集合操作的正确性。类的设计遵循了不可变对象的部分原则，通过标准的 `getter/setter` 方法提供属性访问控制。

**Message 类** 作为系统通信的数据载体，`Message` 类实现了 `Serializable` 接口，支持对象的网络序列化传输。该类通过静态常量定义了完整的消息类型体系，包括登录认证 (`LOGIN`、`LOGIN_SUCCESS`、`LOGIN_FAILED`)、聊天通信 (`CHAT`、`PRIVATE_CHAT`)、系统交互 (`SYSTEM_COMMAND`、`SYSTEM_RESPONSE`) 和用户管理 (`USER_LIST`、`ANONYMOUS_TOGGLE`) 等类型。类设计支持广播消息、私聊消息和系统消息三种构造模式，并自动维护时间戳信息，为系统的消息审计和日志记录提供支持。

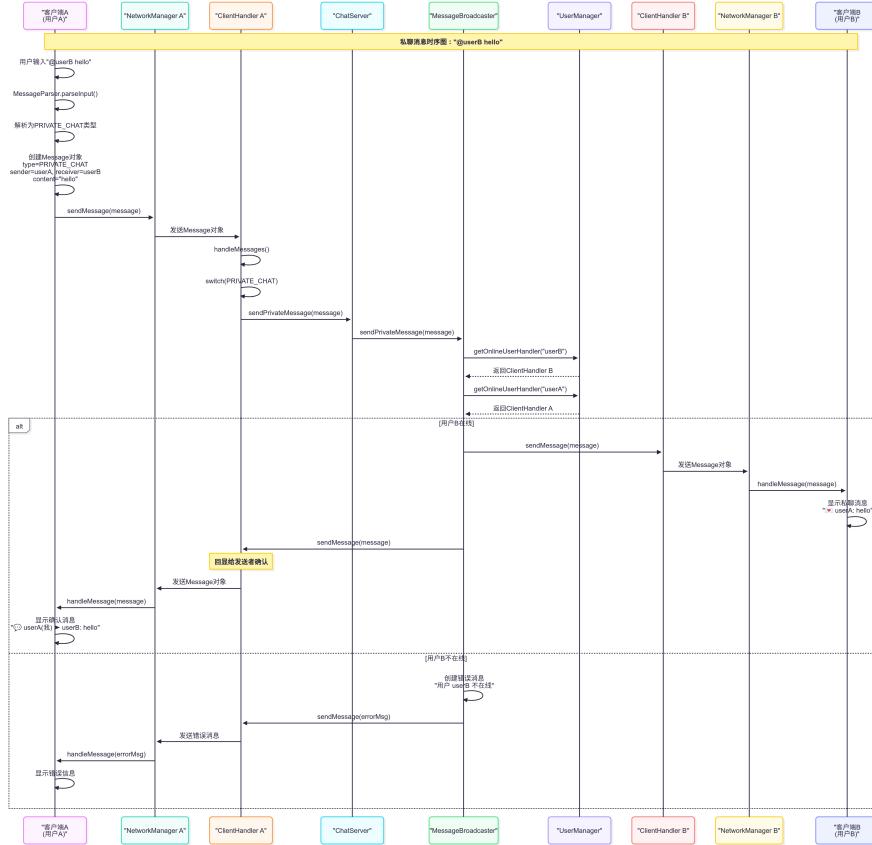


图 4: 私聊消息时序图: 该图展示了用户 A 向用户 B 发送一条私聊消息的完整交互过程。消息从客户端发出, 经由服务器的 ClientHandler 接收, 通过 MessageBroadcaster 进行路由, 最终转发给目标用户的 ClientHandler 并推送至客户端, 清晰地体现了系统各组件间的动态协作关系。

**Logger 类** 采用单例模式设计的日志管理器，提供统一的日志记录服务。该类支持多级别日志输出 (DEBUG、INFO、WARN、ERROR)，同时支持控制台和文件双重输出模式。日志器具备线程安全特性，使用同步机制确保多线程环境下日志记录的一致性和完整性。

#### 1.4.2 服务器模块 (server)

**ChatServer 类** 作为服务器的主控制器，`ChatServer` 类负责系统的核心协调功能。该类实现了 `UserStatusListener` 接口，监听用户状态变化并触发相应的业务逻辑。类的主要职责包括：端口监听和连接接受、用户认证和连接管理、消息广播和私聊转发、服务器命令处理 (`list`, `listall`, `quit`) 以及系统优雅关闭。类采用组合模式集成 `UserManager` 和 `MessageBroadcaster`，实现了功能的模块化分离。

**UserManager 类** 用户管理器类封装了所有用户相关的操作逻辑，包括用户数据的加载与持久化、用户认证机制、在线状态维护和用户查询服务。该类使用 `ConcurrentHashMap` 确保线程安全的并发访问，支持用户的动态添加和状态更新。类设计中引入了 `UserStatusListener` 回调接口，实现了用户状态变化的事件通知机制，为系统的响应式设计提供了基础。

**ClientHandler 类** 客户端连接处理器采用一连接一线程的处理模型，每个 `ClientHandler` 实例负责管理单个客户端的完整生命周期。该类实现了 `Runnable` 接口，支持多线程并发处理。类的核心功能包括：消息接收和解析、业务逻辑分发、系统命令处理、匿名状态管理以及连接异常处理。通过与 `MessageParser` 的协作，实现了用户输入的智能解析和路由。

#### 1.4.3 消息处理模块 (messaging)

**MessageBroadcaster 类** 消息广播器专门负责服务器端的消息分发逻辑，实现了消息的精确路由和可靠传递。该类提供了多种广播模式：全员广播 (`broadcastToAll`)、私聊消息 (`sendPrivateMessage`)、系统通知 (`broadcastSystemNotification`) 和用户列表更新 (`broadcastUserListUpdate`)。类设计中引入了 `BroadcastResult` 结果封装，提供了操作结果的详细反馈，包括成功状态、接收者数量和错误信息，增强了系统的可观测性。

**MessageParser 类** 消息解析器实现了用户输入的智能分类和处理，支持三种输入模式：系统命令（以 `@@` 开头）、私聊消息（以 `@` 用户名开头）和普通聊天消息。该类采用正则表达式和字符串解析技术，实现了用户输入的准确识别和参数提取，为系统的命令处理和消息路由提供了统一的入口。

#### 1.4.4 客户端模块 (client)

**ChatClient 类** 客户端主控制器实现了客户端的核心业务逻辑，作为 GUI 组件和网络通信的协调中心。该类实现了 `MessageHandler` 接口，负责处理服务器推送的各类消息。类的设计采用了事件驱动模式，通过消息类型分发实现不同业务场景的处理。主要功能包括：登录流程管理、聊天界面控制、消息显示和发送、系统命令处理以及用户状态管理。

**NetworkManager 类** 网络管理器专门负责客户端的网络通信功能，封装了 TCP 连接的建立、维护和释放过程。该类采用独立线程处理消息接收，使用 `SwingUtilities.invokeLater()` 确保 UI 更新在 EDT 线程中执行，保证了线程安全。类设计提供了连接状态监控、消息发送确认和异常处理机制，实现了网络通信的可靠性保障。

## 2 使用说明

本项目支持跨平台运行，第 2.1 节提供了服务器端使用说明，第 2.2 节提供了客户端使用说明。附录 A 展示了系统各项功能的完整演示截图。

### 2.1 服务器端使用

在终端窗口中启动服务器 (效果如图 5 所示):

```
1 # macOS/Linux
2 ./run_server.sh
3
4 # Windows
5 java -cp build server.ChatServer
```

Listing 1: 启动服务器

服务器启动后会监听 8080 端口，支持以下控制台命令：

- list - 查看当前在线用户
- listall - 查看所有注册用户
- quit - 关闭服务器



The screenshot shows a terminal window on a Mac OS X system. The user has run the command `./run_server.sh`. The server starts up and displays its configuration and supported features:

- 启动聊天室服务器...
- 准备启动服务器...
- 监听端口: 8080
- 支持的功能:
  - 多用户聊天
  - 私聊消息 (@用户名 消息)
  - 匿名模式 (@anonymous)
  - 用户列表 (@@list)
- 服务器命令:
  - list: 查看在线用户
  - listall: 查看所有用户
  - quit: 关闭服务器

在启动过程中，日志显示了成功加载账户数和服务器启动信息。最后，命令行提示符显示为 `server>`。

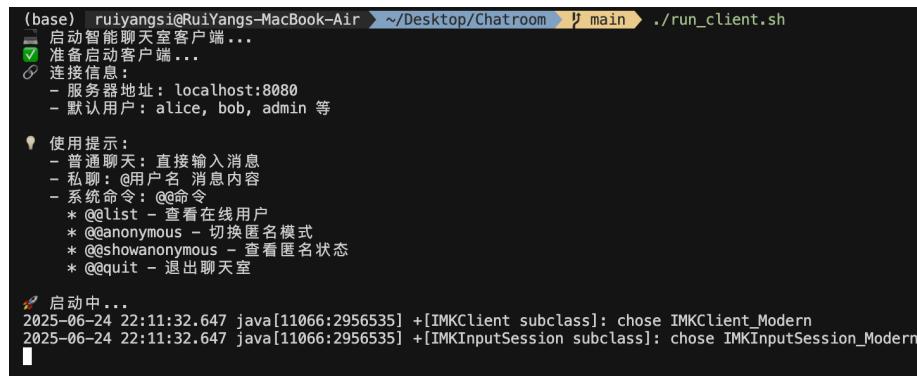
图 5：服务器运行，可以在这里输入 `list`,`listall`,`quit` 来执行系统命令。同时可以发现有日志功能记录了启动服务器（所有日志记录同时写入 `server.log` 的日志中）

## 2.2 客户端使用

在终端窗口中启动客户端 (效果如图 6 所示):

```
1 # macOS/Linux  
2 ./run_client.sh  
3  
4 # Windows  
5 java -cp build client.ChatClient
```

Listing 2: 启动客户端



```
(base) ruiyangsi@RuiYangs-MacBook-Air ~/Desktop/Chatroom ➜ main ➜ ./run_client.sh  
启动智能聊天室客户端...  
准备启动客户端...  
连接信息：  
- 服务器地址：localhost:8080  
- 默认用户：alice, bob, admin 等  
使用提示：  
- 普通聊天：直接输入消息  
- 私聊：@用户名 消息内容  
- 系统命令：@@命令  
* @@list - 查看在线用户  
* @@anonymous - 切换匿名模式  
* @@showanonymous - 查看匿名状态  
* @@quit - 退出聊天室  
启动中...  
2025-06-24 22:11:32.647 java[11066:2956535] +[IMKClient subclass]: chose IMKClient_Modern  
2025-06-24 22:11:32.647 java[11066:2956535] +[IMKInputSession subclass]: chose IMKInputSession_Modern
```

图 6: 客户端运行

**登录界面** 客户端启动后会显示登录界面 (如图 7a 所示), 需要输入:

- **用户名**: 想要登陆的用户名
- **密码**: 对应的用户密码
- **服务器地址**: 默认为 127.0.0.1, 可修改为远程服务器 IP
- **端口号**: 默认为 8080, 可根据实际情况修改

**聊天功能** 登录成功后, 支持以下聊天方式 (如图 7b 所示):

- **公开聊天**: 直接输入消息, 所有在线用户可见
- **私聊消息**: 输入格式为 @ 用户名消息内容
- **系统命令**: 以 @@ 开头的特殊命令

- @@list - 查看在线用户列表
- @@anonymous - 切换匿名模式
- @@showanonymous - 查看当前匿名状态
- @@quit - 退出聊天室



图 7: 用户界面

## 3 未来工作

本项目成功构建了一个功能完备的 C/S 架构聊天室系统，但作为一个模拟真实应用的系统，仍有许多值得深入和扩展的方向。未来的工作将主要围绕提升系统的安全性与健壮性（第 3.1 节）以及丰富用户体验（第 3.2 节）两个核心维度展开。

### 3.1 从明文到加密存储

当前项目为了简化实现，直接将用户密码以明文形式存储在 `users.txt` 文件中，这在任何生产环境应用中都是严重的安全漏洞。一旦数据文件泄露，所有用户的账户信息将立即暴露，可能引发连锁的安全问题。

未来的首要改进目标是实现密码的哈希加盐（Salted Hashing）存储机制。具体实现路径如下：当用户注册时，系统不再直接存储原始密码，而是为每个用户生成唯一的随机字符串（即“盐值”），然后将盐值与用户密码拼接，通过单向高强度哈希算法（如 `bcrypt`、`PBKDF2` 或 `Argon2`）计算出哈希值。最终，存储系统中只保存用户名、哈希值和对应的盐值。当用户登录时，系统根据用户名检索哈希值和盐值，将用户输入的密码与盐值进行相同的哈希计算，比对计算结果与存储的哈希值是否一致。通过这种方式，即使数据被攻破，攻击者也无法直接获得用户的原始密码，从而显著提升系统的安全性。

### 3.2 支持多模态信息发送

目前的聊天室仅支持纯文本消息的传输，这限制了用户交流的丰富性和表现力。在现代通信应用中，图片、文件、表情符号乃至语音片段等多模态信息的分享是不可或缺的基础功能。

为了实现这一目标，需要对现有系统进行全面的升级。首先，通信协议需要重新设计，`Message` 类需要进行扩展，增加 `messageType` 字段（如 `TEXT`、`IMAGE`、`FILE`）以及用于承载非文本数据的 `byte[] binaryData` 属性。对于图片或文件这类二进制数据，可以通过 `Base64` 编码后嵌入消息对象，或采用更高效的文件传输方案——客户端先将文件上传至服务器的特定存储位置，服务器再将文件的访问链接或标识符转发给接收方。

相应地，客户端的图形用户界面（GUI）也需要重大改造。需要在 `ChatFrame` 中增加文件选择器（`JFileChooser`）、图片预览组件、文件下载进度条等交

互元素，并能够根据收到的消息类型，选择直接渲染图片（使用 `ImageIcon`）还是显示可点击的文件下载链接。此外，`MessageStyleRenderer` 需要支持多种消息类型的渲染，确保用户界面的一致性和美观性。

这一改进将使聊天室从一个简单的文本工具演进为功能更完整、更贴近现代用户习惯的通信平台。

## 4 总结与反思

本项目成功实现了一个功能完备的 C/S 架构聊天室系统，从需求分析到系统设计，从类关系建模到功能实现，完整地体验了面向对象软件开发的全过程。通过 `Socket` 网络编程实现了客户端与服务器的可靠通信，通过多线程技术支持了并发用户访问，通过 `Swing` 框架构建了友好的图形用户界面，通过模块化设计确保了代码的可维护性和可扩展性。

在开发过程中，深刻理解了面向对象设计的核心理念：封装确保了数据安全和接口简洁，继承实现了代码复用和功能扩展，多态提供了灵活的消息处理机制。同时也认识到软件工程实践的重要性：良好的架构设计是系统成功的基石，充分的需求分析是功能实现的前提，规范的文档编写是项目维护的保障。

虽然当前系统在安全性和功能丰富度方面仍有提升空间，但这正是软件开发的魅力所在——技术的演进永无止境，优秀的架构为未来的扩展奠定了坚实基础。通过这次实践，不仅掌握了 Java 面向对象编程的核心技术，更重要的是培养了系统性思考和工程化实现的能力，这将成为未来软件开发道路上的宝贵财富。

## A 效果展示

本节通过截图演示展现系统的完整功能实现效果，表 3 列出了各项功能演示的详细说明和对应图片引用。

表 3: 功能演示对照表

功能模块	演示内容	对应图片
多用户登录	四个用户同时启动客户端并输入登录信息	图 8
聊天界面	用户登录成功后进入完整的聊天界面布局	图 9
服务器管理	使用 <code>list</code> 和 <code>listall</code> 命令查询用户信息	图 10
广播消息	<code>alice</code> 发送广播消息，所有用户都能接收	图 11
私聊功能	<code>bob</code> 向 <code>alice</code> 发送私聊消息，仅两人可见	图 12
匿名模式切换	<code>frank</code> 切换为匿名模式，界面显示相应提示	图 13
匿名状态查询	不同用户查询各自的匿名状态，系统返回准确信息	图 14
匿名消息发送	匿名状态下发送的消息在所有界面均显示为“匿名用户”	图 15
用户管理	部分用户退出后，查询在线用户列表显示准确状态	图 16
系统关闭	服务器关闭后客户端显示连接断开信息	图 17
日志记录	系统自动记录各类操作的详细日志信息	图 18



图 8: 多用户同时登录

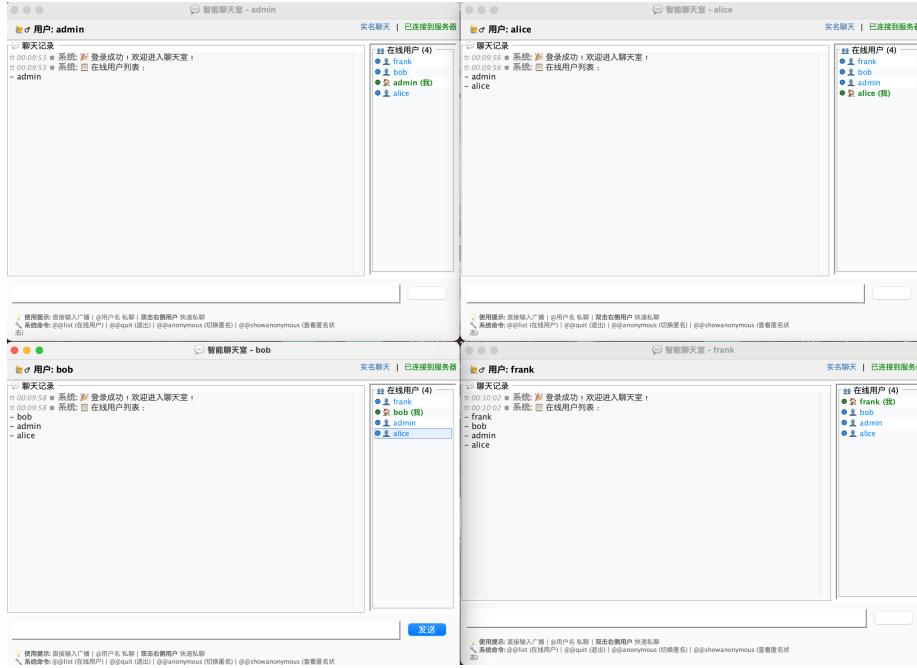


图 9: 多用户登录后进入聊天界面

```
server> list
在线用户 (4):
- frank
- bob
- admin
- alice
server> listall
所有用户 (12):
- frank (在线)
- iris (离线)
- admin (在线)
- alice (在线)
- jack (离线)
- henry (离线)
- bob (在线)
- eve (离线)
- kate (离线)
- grace (离线)
- diana (离线)
- charlie (离线)
server> ■
```

图 10: 服务器端 list, listall 命令查询

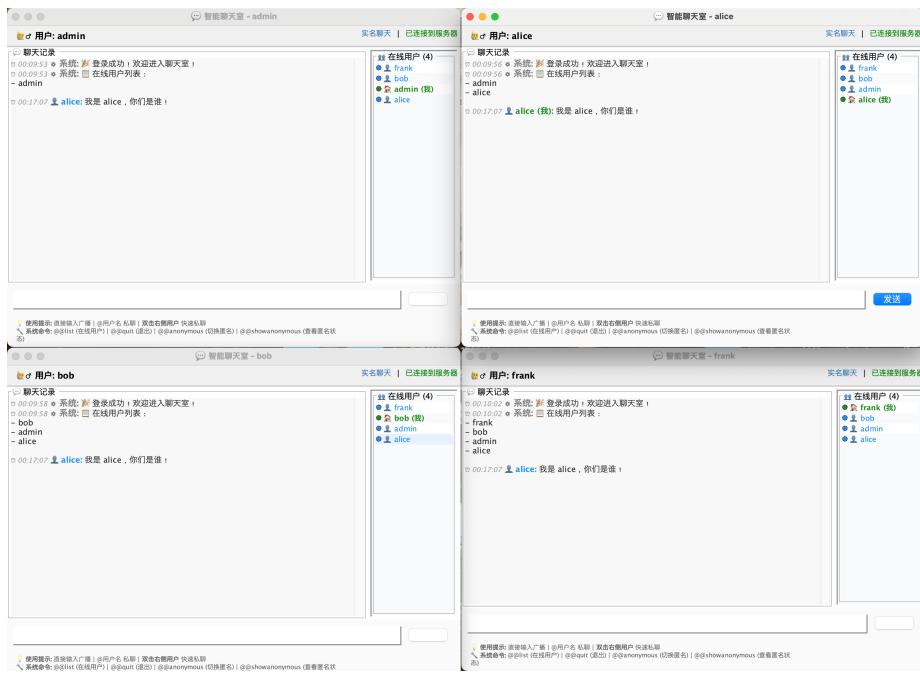


图 11: 广播功能: 在 `alice` 聊天框写入 `我是 alice, 你们是谁?`的效果, 所有人都能看到消息

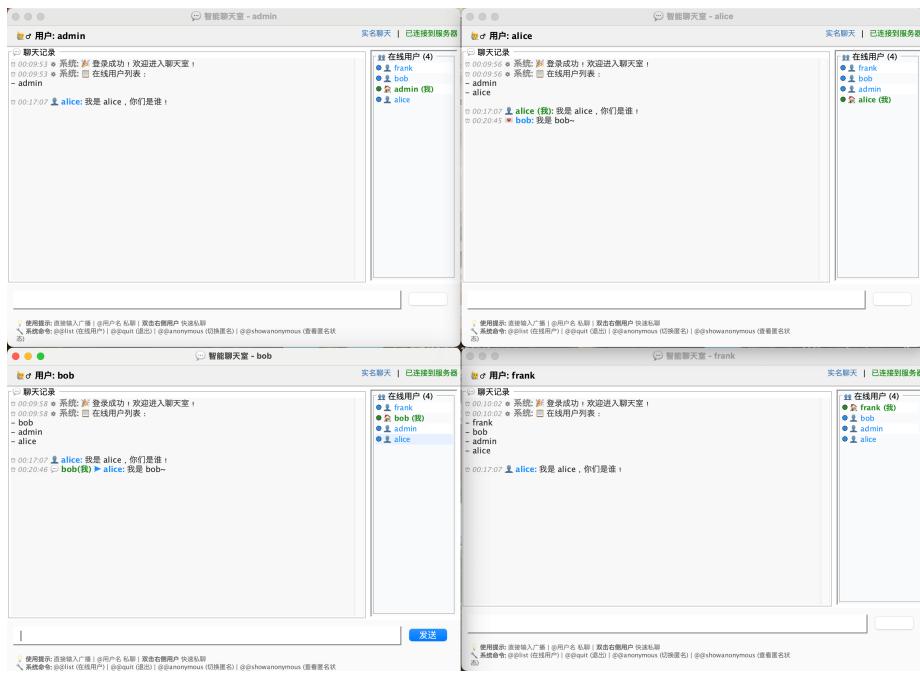


图 12: 私聊功能：在 bob 聊天框写入 @alice 我是 bob 后的效果，只有 alice 和 bob 能看到消息

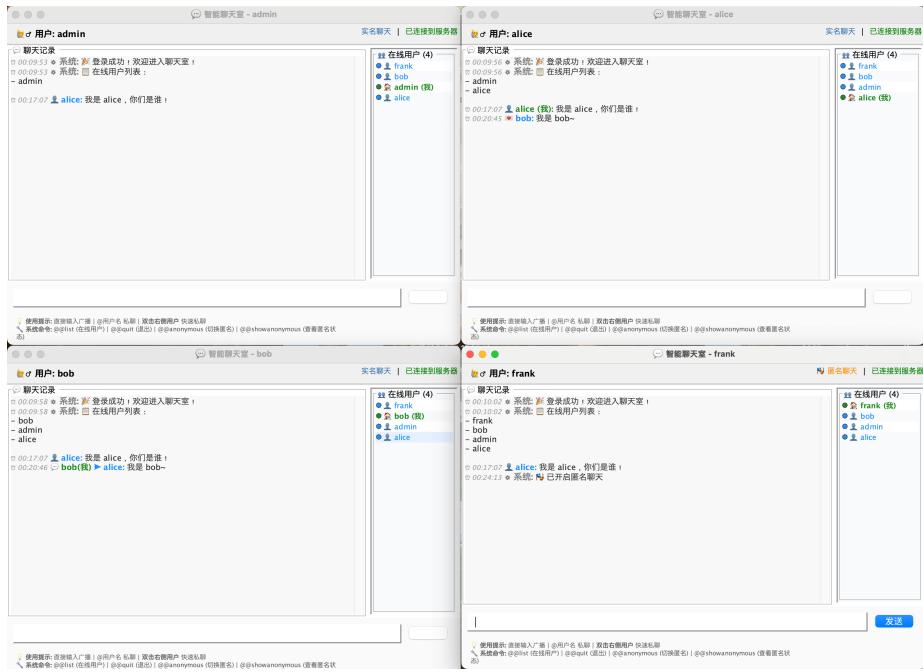


图 13: 切换匿名功能: 在 `frank` 聊天框写入 `@@anonymous` 后的效果, 可以发现在 `frank` 屏内变成匿名聊天, 其他同学都还是实名聊天

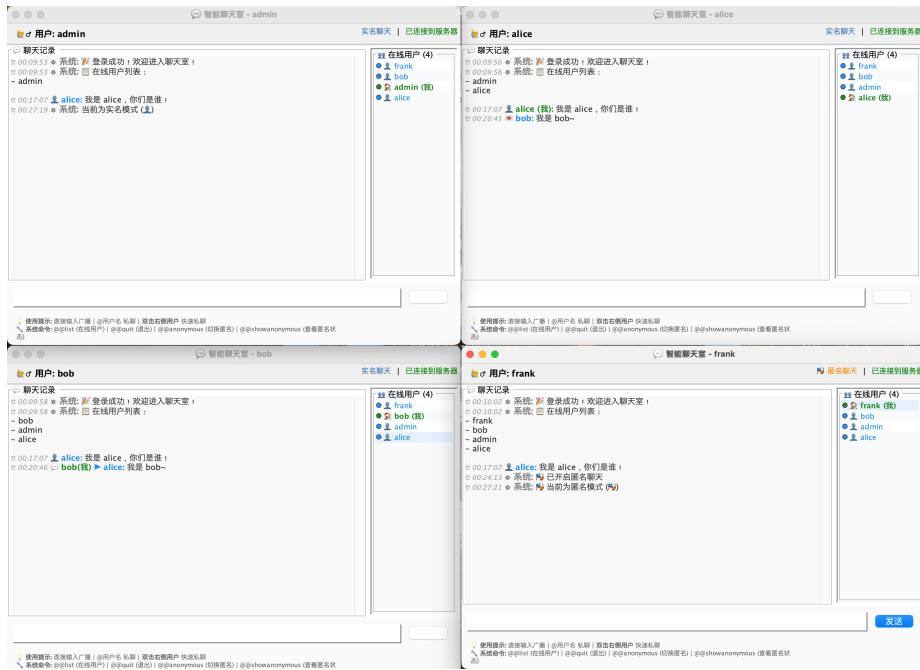


图 14: 查询匿名功能: 在 admin, frank 聊天框写入 @@showanonymous 后的效果, admin 显示实名而 frank 显示匿名

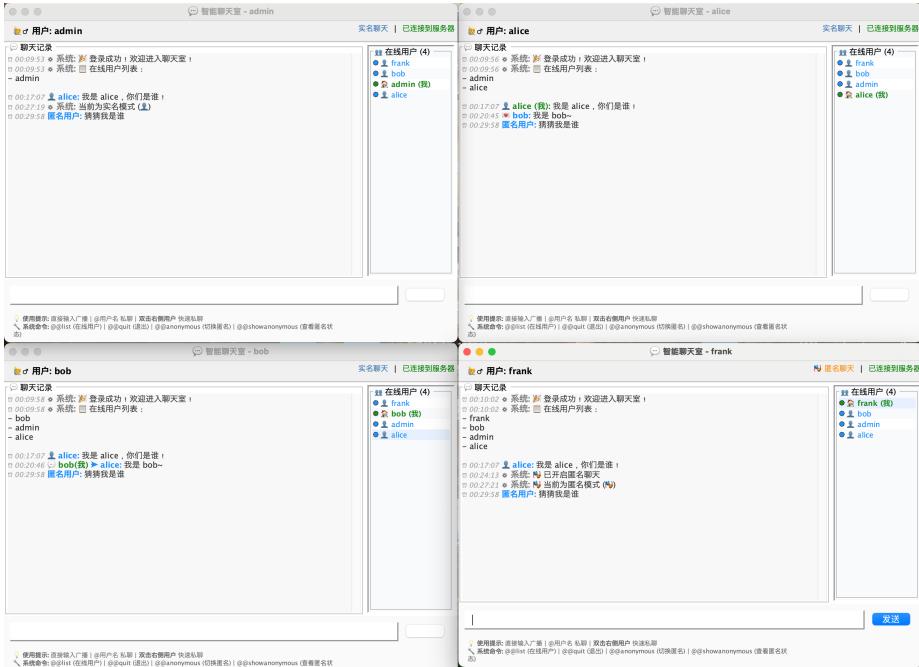


图 15: 测试匿名功能：在 `frank` 聊天框写入 `猜猜我是谁`的效果，所有屏幕上显示的都为匿名用户发送的消息。

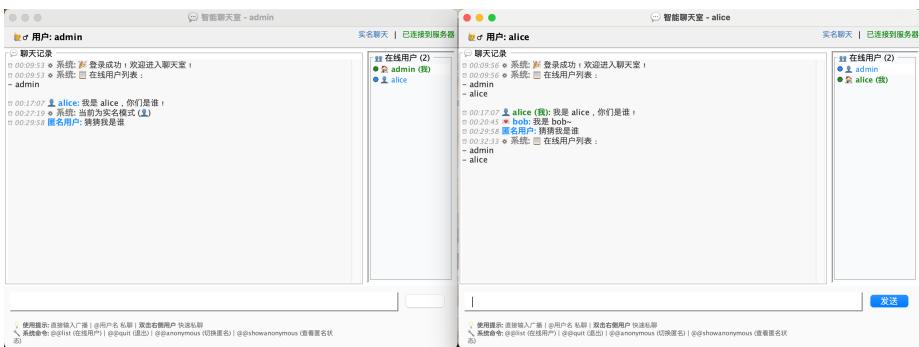


图 16: 展示在线用户功能：在 `bob`, `frank` 聊天框写入 `@@quit` 后二人退出聊天，并在 `alice` 聊天框写入 `@@list` 后的效果

```

server> quit
[2025-06-25 00:35:23.623] [INFO] [Thread-0] 服务器关闭中...
[2025-06-25 00:35:23.625] [INFO] [Thread-0] 用户下线: admin (IP: 127.0.0.1)
[2025-06-25 00:35:23.626] [INFO] [Thread-0] 用户离线: admin (当前在线: 1 人)
[2025-06-25 00:35:23.627] [INFO] [Thread-0] 广播消息: 系统 -> alice (发送给 1/1 个用户)
[2025-06-25 00:35:23.627] [INFO] [Thread-0] 广播用户列表更新: 1 用户在线, 发送给 1 个客户端
[2025-06-25 00:35:23.634] [INFO] [Thread-0] 用户下线: alice (IP: 127.0.0.1)
[2025-06-25 00:35:23.635] [INFO] [Thread-0] 用户离线: alice (当前在线: 0 人)
[2025-06-25 00:35:23.635] [WARN] [Thread-0] 没有在线用户, 无法广播消息
[2025-06-25 00:35:23.635] [INFO] [Thread-0] 广播用户列表更新: 0 用户在线, 发送给 0 个客户端
[2025-06-25 00:35:23.637] [INFO] [Thread-0] 用户管理器关闭中...
[2025-06-25 00:35:23.639] [INFO] [Thread-0] 用户管理器已关闭
[2025-06-25 00:35:23.639] [INFO] [Thread-0] 服务器已关闭

```

图 17: 服务器关闭功能: 在服务器写入 `exit` 后关闭服务器以及 `admin`, `alice` 的客户端进程。

```

[2025-06-25 00:09:53.339] [INFO] [Thread-1] 广播消息: 系统 -> admin (发送给 1/1 个用户)
[2025-06-25 00:09:53.341] [INFO] [Thread-1] 广播用户列表更新: 1 用户在线, 发送给 1 个客户端
[2025-06-25 00:09:56.436] [INFO] [main] 新客户连接: 127.0.0.1
[2025-06-25 00:09:56.436] [INFO] [Thread-1] 用户登录成功: admin (IP: 127.0.0.1)
[2025-06-25 00:09:56.456] [INFO] [Thread-2] 用户登录成功: alice (IP: 127.0.0.1)
[2025-06-25 00:09:56.456] [INFO] [Thread-2] 用户上线: alice (当前在线: 2 人)
[2025-06-25 00:09:56.457] [INFO] [Thread-2] 广播消息: 系统 -> admin,alice (发送给 2/2 个用户)
[2025-06-25 00:09:56.457] [INFO] [Thread-2] 广播用户列表更新: 2 用户在线, 发送给 2 个客户端
[2025-06-25 00:09:56.457] [INFO] [main] 新客户连接: 127.0.0.1
[2025-06-25 00:09:56.188] [INFO] [main] 新客户连接: 127.0.0.1
[2025-06-25 00:09:56.206] [INFO] [Thread-3] 用户登录成功: bob (IP: 127.0.0.1)
[2025-06-25 00:09:56.207] [INFO] [Thread-3] 用户上线: bob (当前在线: 3 人)
[2025-06-25 00:09:56.207] [INFO] [Thread-3] 广播消息: 系统 -> bob,admin,alice (发送给 3/3 个用户)
[2025-06-25 00:09:56.208] [INFO] [Thread-3] 广播用户列表更新: 3 用户在线, 发送给 3 个客户端
[2025-06-25 00:10:01.598] [INFO] [main] 新客户连接: 127.0.0.1
[2025-06-25 00:10:01.616] [INFO] [Thread-4] 用户登录成功: frank (IP: 127.0.0.1)
[2025-06-25 00:10:01.616] [INFO] [Thread-4] 用户上线: frank (当前在线: 4 人)
[2025-06-25 00:10:01.617] [INFO] [Thread-4] 广播消息: 系统 -> frank,bob,admin,alice (发送给 4/4 个用户)
[2025-06-25 00:10:01.617] [INFO] [Thread-4] 广播用户列表更新: 4 用户在线, 发送给 4 个客户端
[2025-06-25 00:17:07.938] [INFO] [Thread-2] 广播消息: alice -> 我是 alice,你们是谁? (发送给 4/4 个用户)
[2025-06-25 00:20:45.978] [INFO] [Thread-3] 私聊消息: bob -> alice: 我是 bob-
[2025-06-25 00:29:58.699] [INFO] [Thread-4] 广播消息: frank -> frank 我是谁 (发送给 4/4 个用户)
[2025-06-25 00:32:16.116] [INFO] [Thread-3] 用户下线: bob (IP: 127.0.0.1)
[2025-06-25 00:32:16.112] [INFO] [Thread-3] 用户离线: bob (当前在线: 3 人)
[2025-06-25 00:32:16.113] [INFO] [Thread-3] 广播消息: 系统 -> frank,admin,alice (发送给 3/3 个用户)
[2025-06-25 00:32:16.113] [INFO] [Thread-3] 广播用户列表更新: 3 用户在线, 发送给 3 个客户端
[2025-06-25 00:32:23.630] [INFO] [Thread-4] 用户下线: frank (IP: 127.0.0.1)
[2025-06-25 00:32:23.631] [INFO] [Thread-4] 用户离线: frank (当前在线: 2 人)
[2025-06-25 00:32:23.633] [INFO] [Thread-4] 广播消息: 系统 -> admin,alice (发送给 2/2 个用户)
[2025-06-25 00:32:23.633] [INFO] [Thread-4] 广播用户列表更新: 2 用户在线, 发送给 2 个客户端
[2025-06-25 00:35:23.623] [INFO] [Thread-0] 服务器关闭中...
[2025-06-25 00:35:23.625] [INFO] [Thread-0] 用户下线: admin (IP: 127.0.0.1)
[2025-06-25 00:35:23.626] [INFO] [Thread-0] 用户离线: admin (当前在线: 1 人)
[2025-06-25 00:35:23.627] [INFO] [Thread-0] 广播消息: 系统 -> alice (发送给 1/1 个用户)
[2025-06-25 00:35:23.627] [INFO] [Thread-0] 广播用户列表更新: 1 用户在线, 发送给 1 个客户端
[2025-06-25 00:35:23.634] [INFO] [Thread-0] 用户下线: alice (IP: 127.0.0.1)
[2025-06-25 00:35:23.635] [INFO] [Thread-0] 用户离线: alice (当前在线: 0 人)
[2025-06-25 00:35:23.635] [WARN] [Thread-0] 没有在线用户, 无法广播消息
[2025-06-25 00:35:23.635] [INFO] [Thread-0] 广播用户列表更新: 0 用户在线, 发送给 0 个客户端
[2025-06-25 00:35:23.637] [INFO] [Thread-0] 用户管理器关闭中...
[2025-06-25 00:35:23.639] [INFO] [Thread-0] 用户管理器已关闭
[2025-06-25 00:35:23.639] [INFO] [Thread-0] 服务器已关闭

```

图 18: 日志功能: 展示了部分上述过程的日志记录