

Java期末实验程序设计 项目介绍

实验题目：运用Java swing、socket网络编程、JDBC等实现
Java简易聊天软件

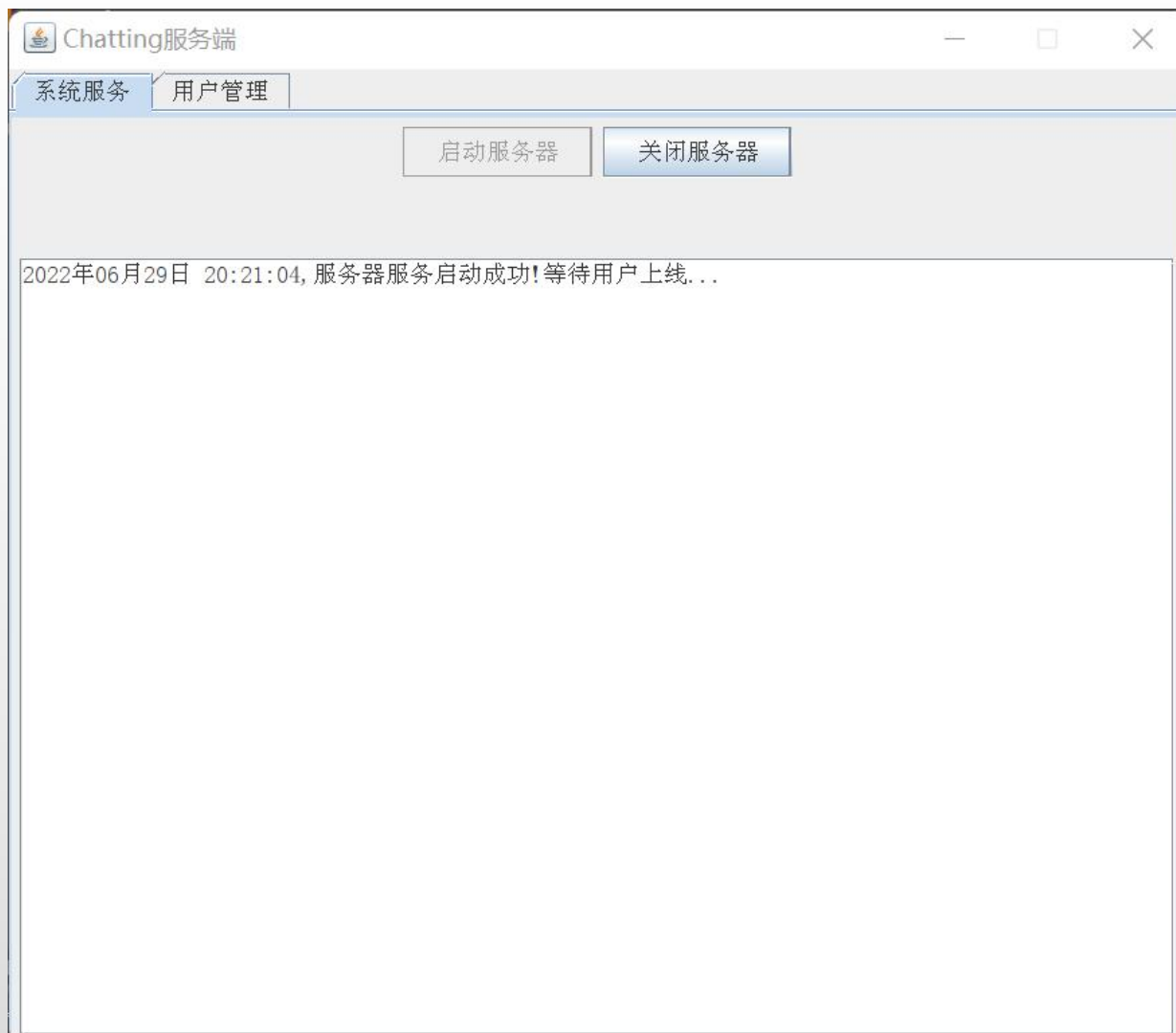
软件学院 软件工程一班 胡旖雯 2011277

程序功能

- 客户端：
 - 1.注册账号
 - 2.登录账号
 - 3.搜索对方账号添加好友
 - 4.在聊天窗口与对方实时聊天
- 服务端：
 - 管理用户账号、查看操作日志

演示运行方法

- 1. 启动服务器



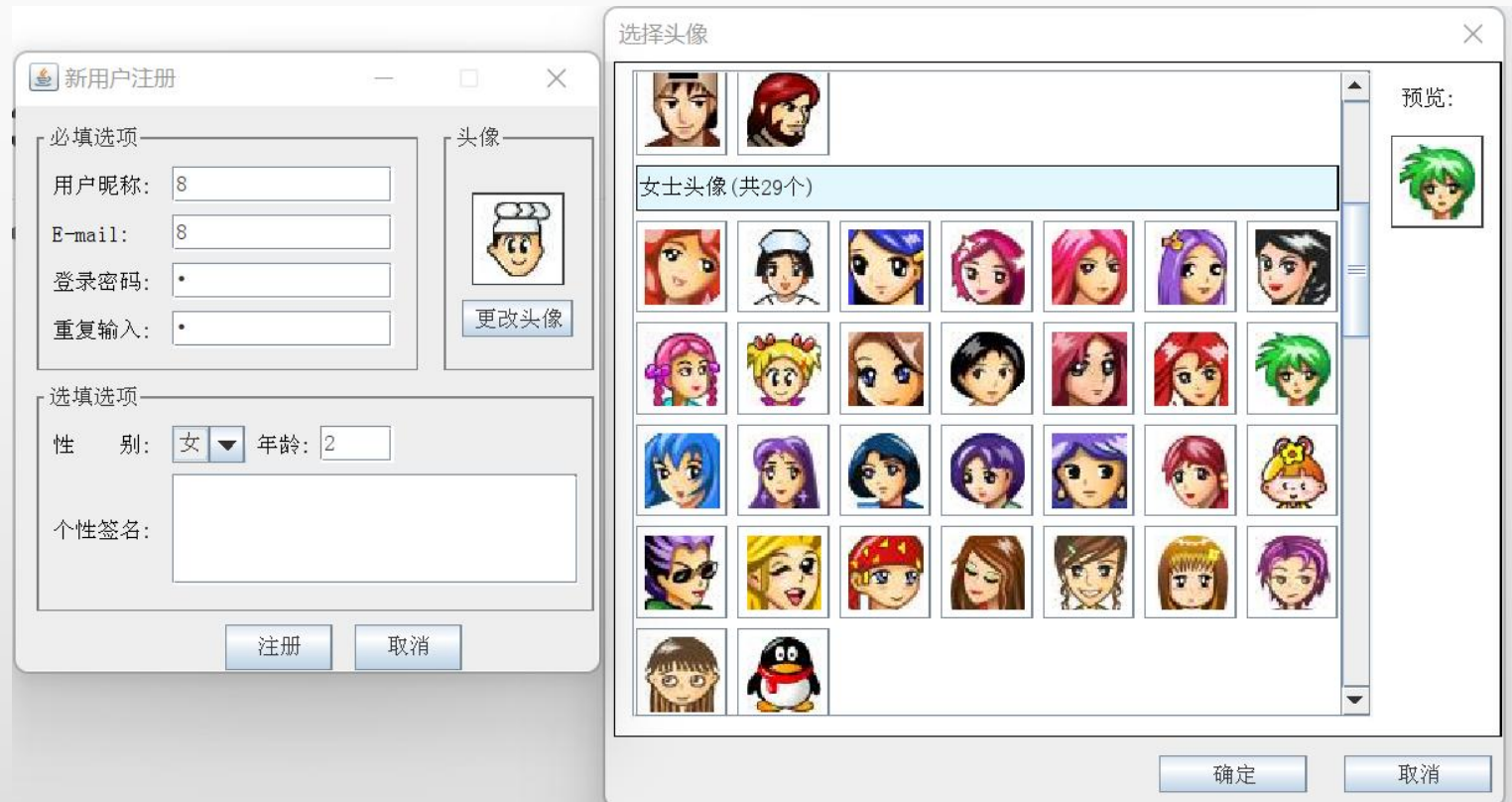
2.启动客户端

●登录



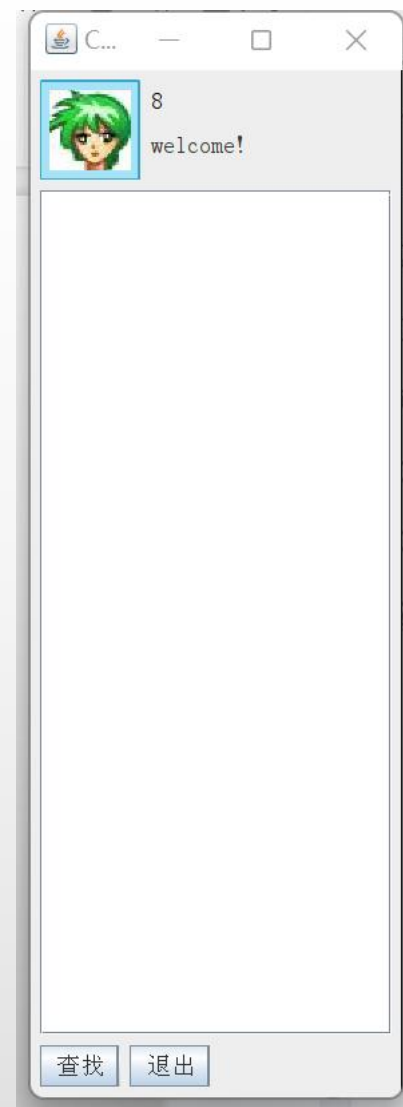
A login window titled "登录" (Login). It contains two input fields: "账号" (Account) and "密码" (Password). Below the fields are two buttons: "申请账号" (Apply for account) and "登录" (Login).

注册

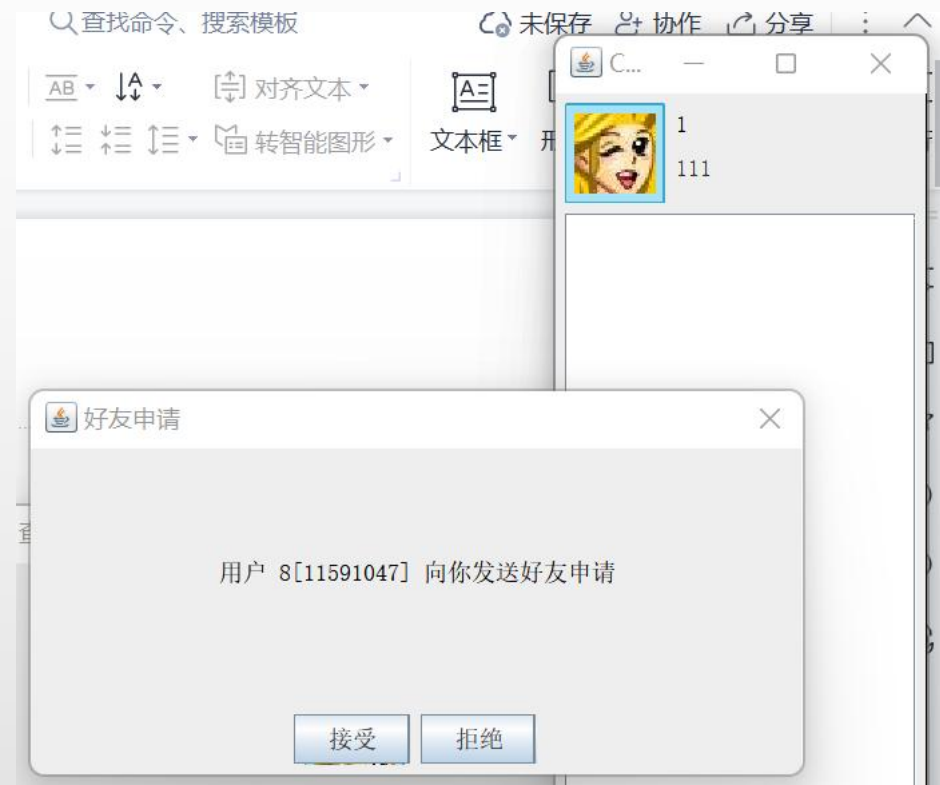
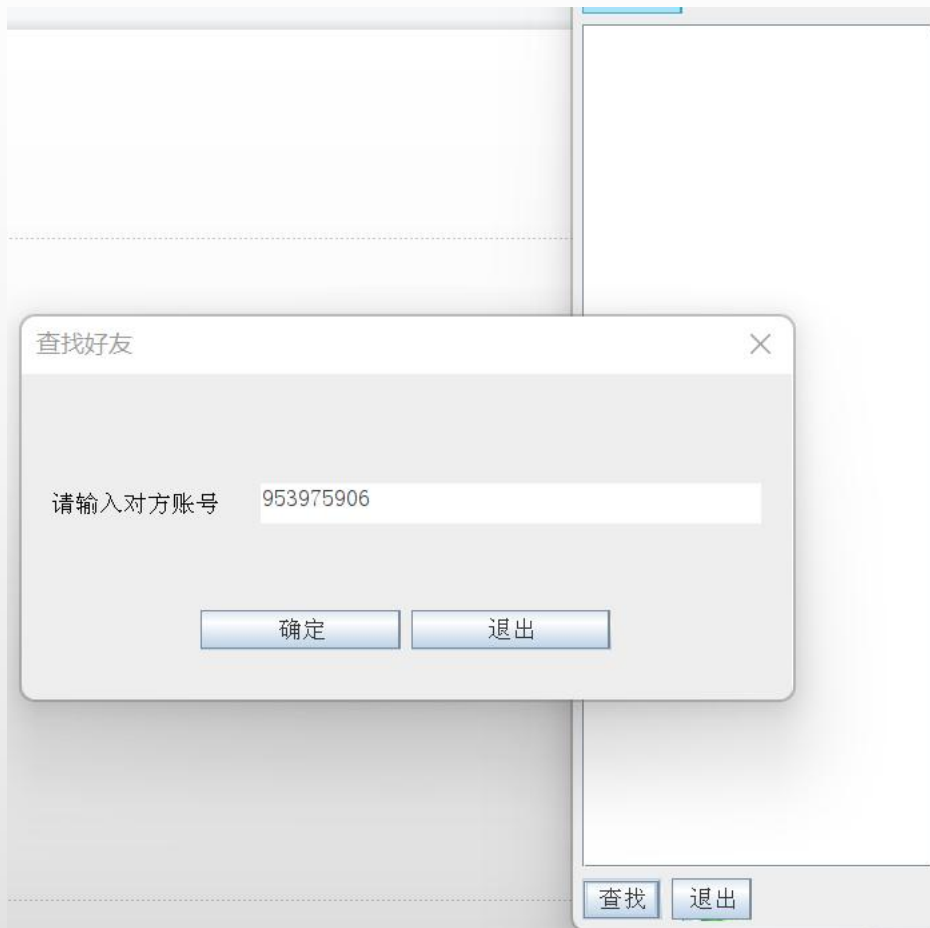


Two windows are shown side-by-side. The left window is titled "新用户注册" (New User Registration). It has two sections: "必填选项" (Required options) and "选填选项" (Optional options). The "必填选项" section includes fields for "用户昵称" (User nickname), "E-mail", "登录密码" (Login password), and "重复输入" (Repeat input). The "选填选项" section includes a "性别" (Gender) dropdown menu (set to "女"), an "年龄" (Age) input field (set to "2"), and a "个性签名" (Personal signature) text area. There is a "头像" (Avatar) section with a default avatar and a "更改头像" (Change avatar) button. The right window is titled "选择头像" (Select avatar). It displays a grid of 29 female avatars, with a "预览" (Preview) section on the right showing the selected avatar. The grid is labeled "女士头像 (共29个)" (Female avatars (29 total)). At the bottom of the right window are "确定" (Confirm) and "取消" (Cancel) buttons.

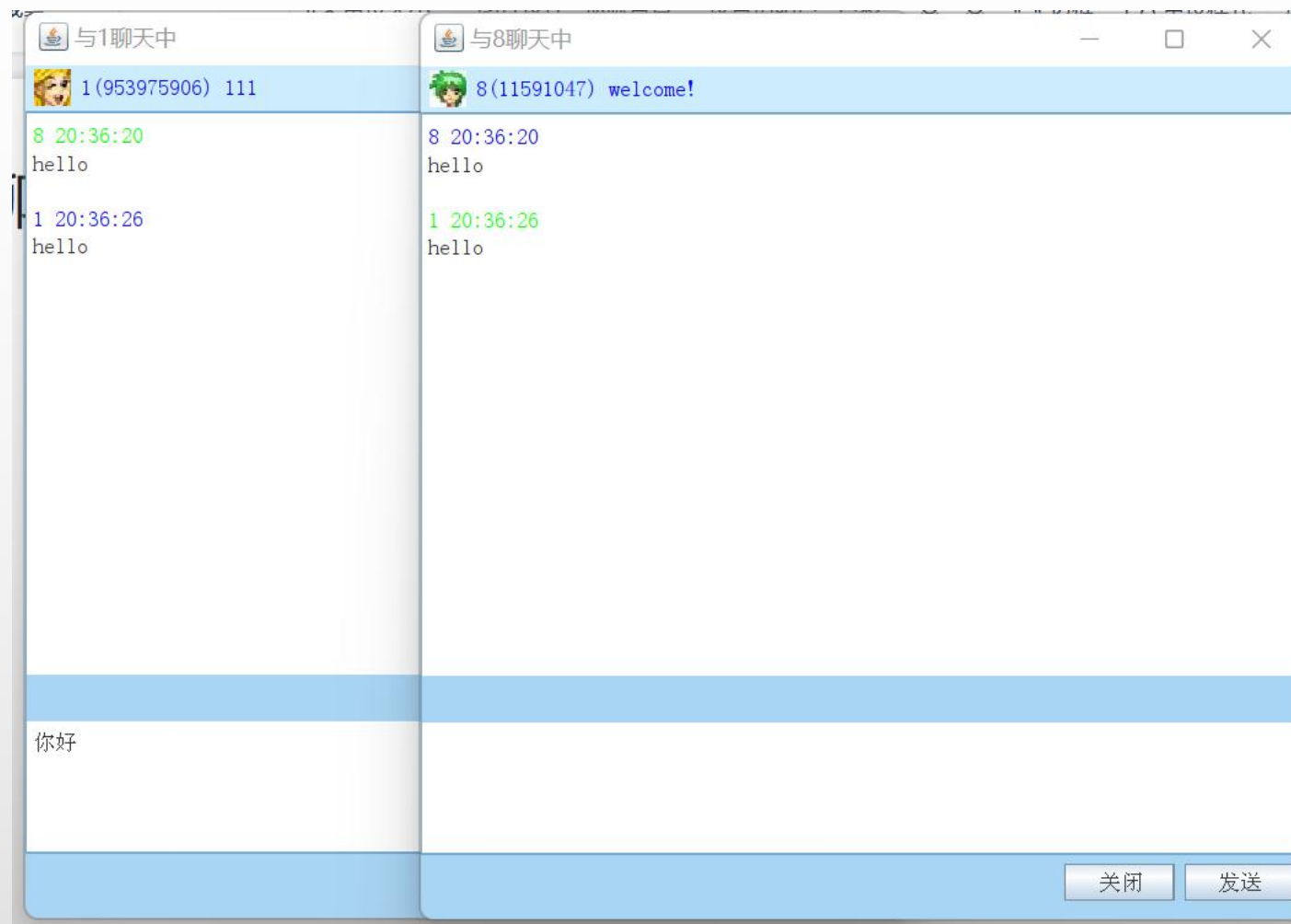
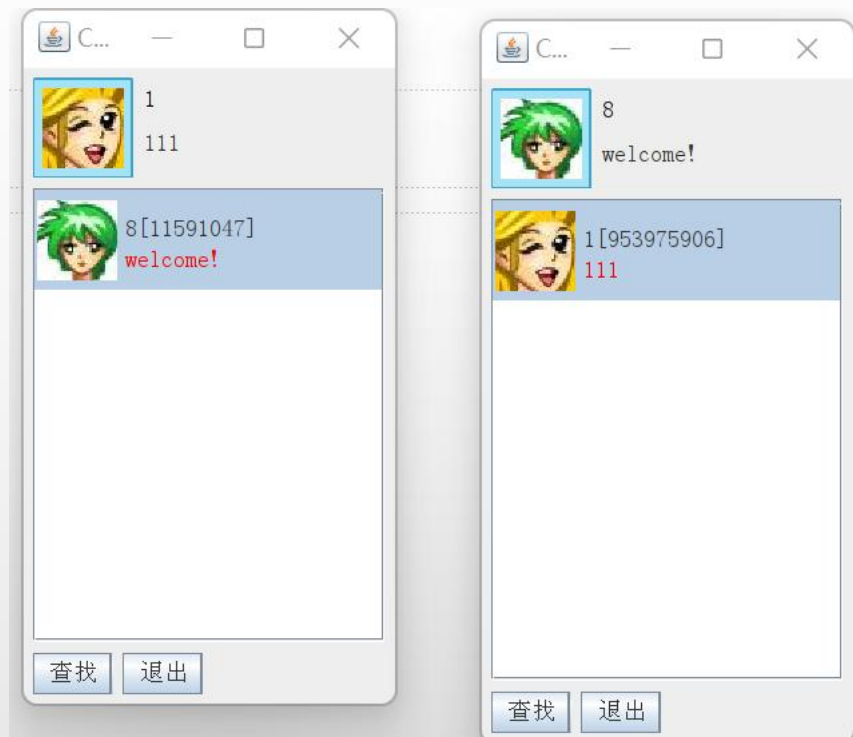
3.进入主界面



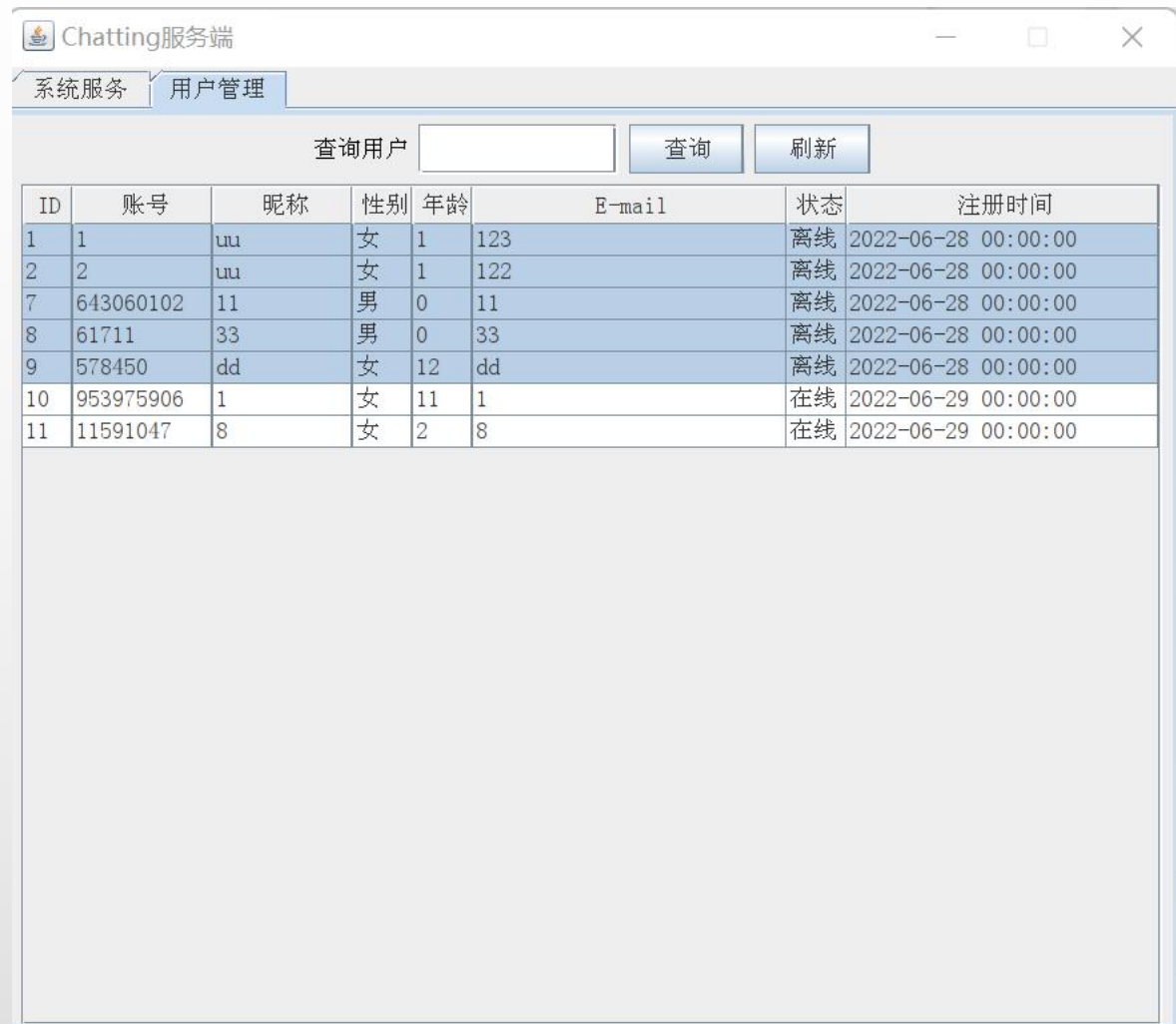
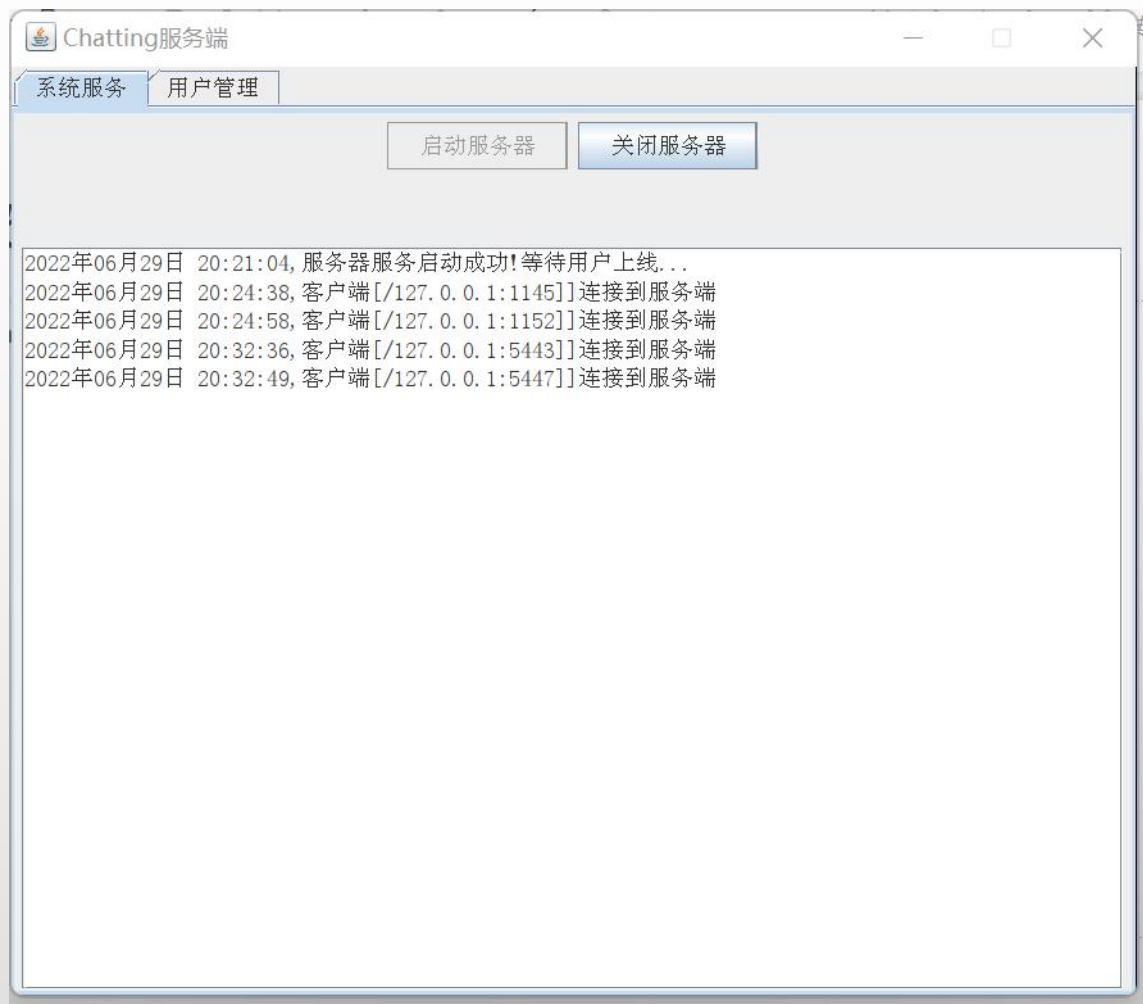
4. 查找好友，加好友



5.聊天



6. 服务器日志记录， 用户管理



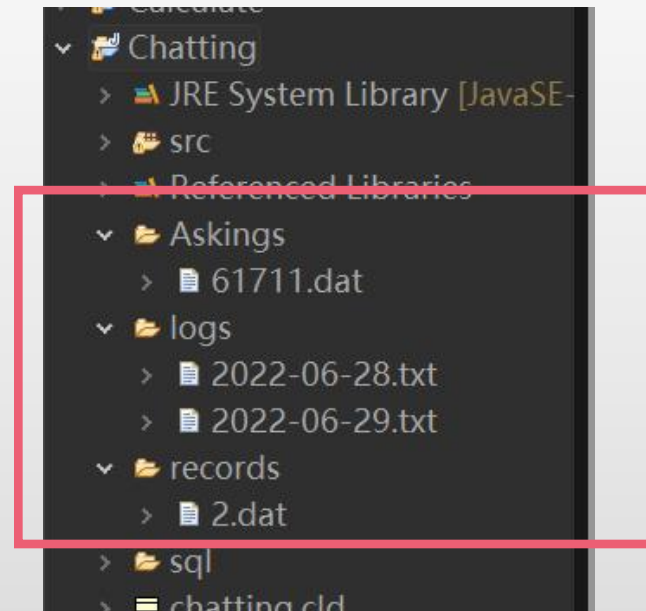
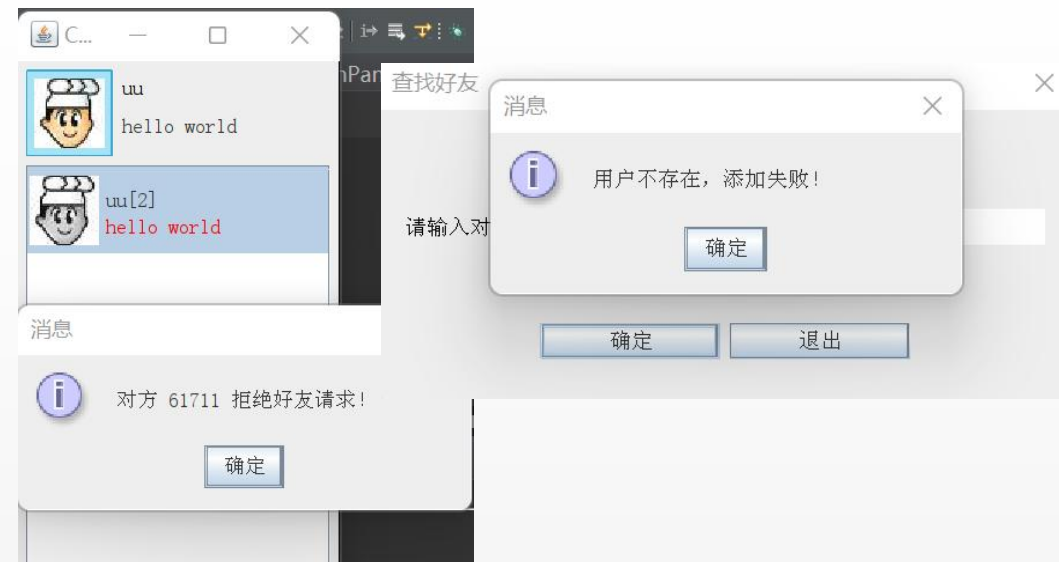
特殊情况处理

- 1.登陆界面，登录不成功；
- 2.加好友不成功；（查无此人或对方拒绝）
- 3.给对方发消息或好友申请，对方不在线，数据暂存进文件，当对方登录时弹出界面。

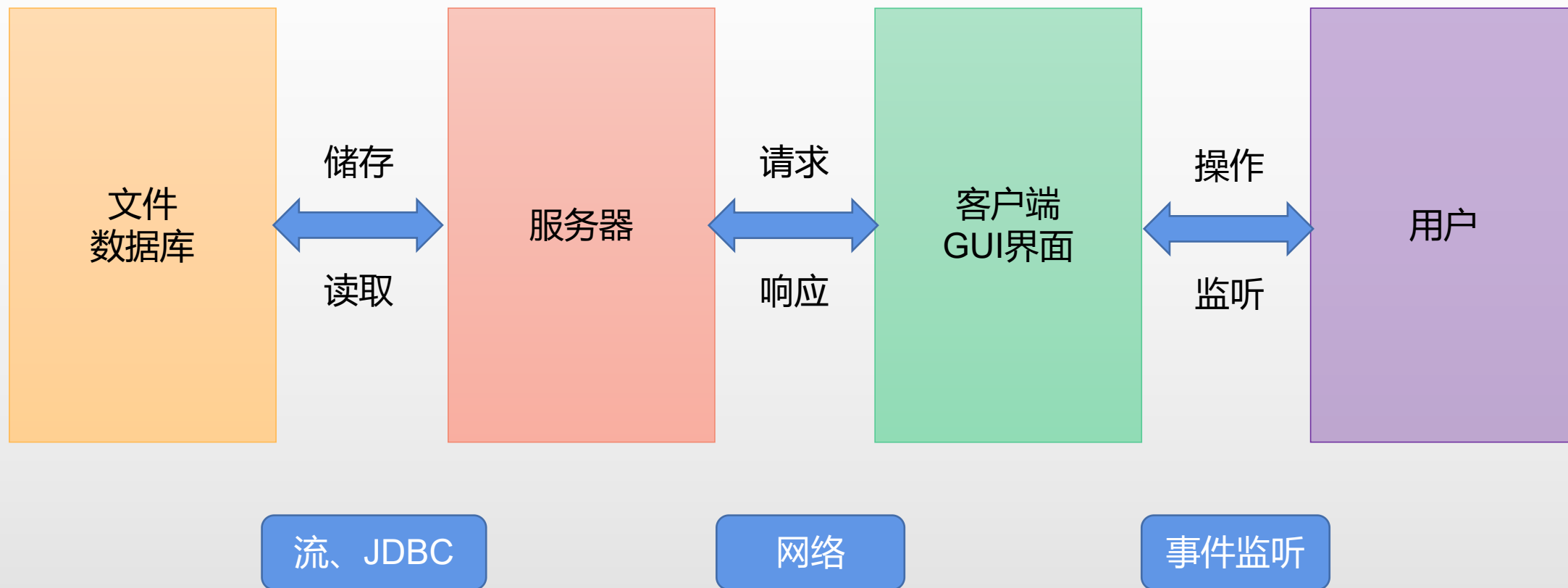


特殊情况处理

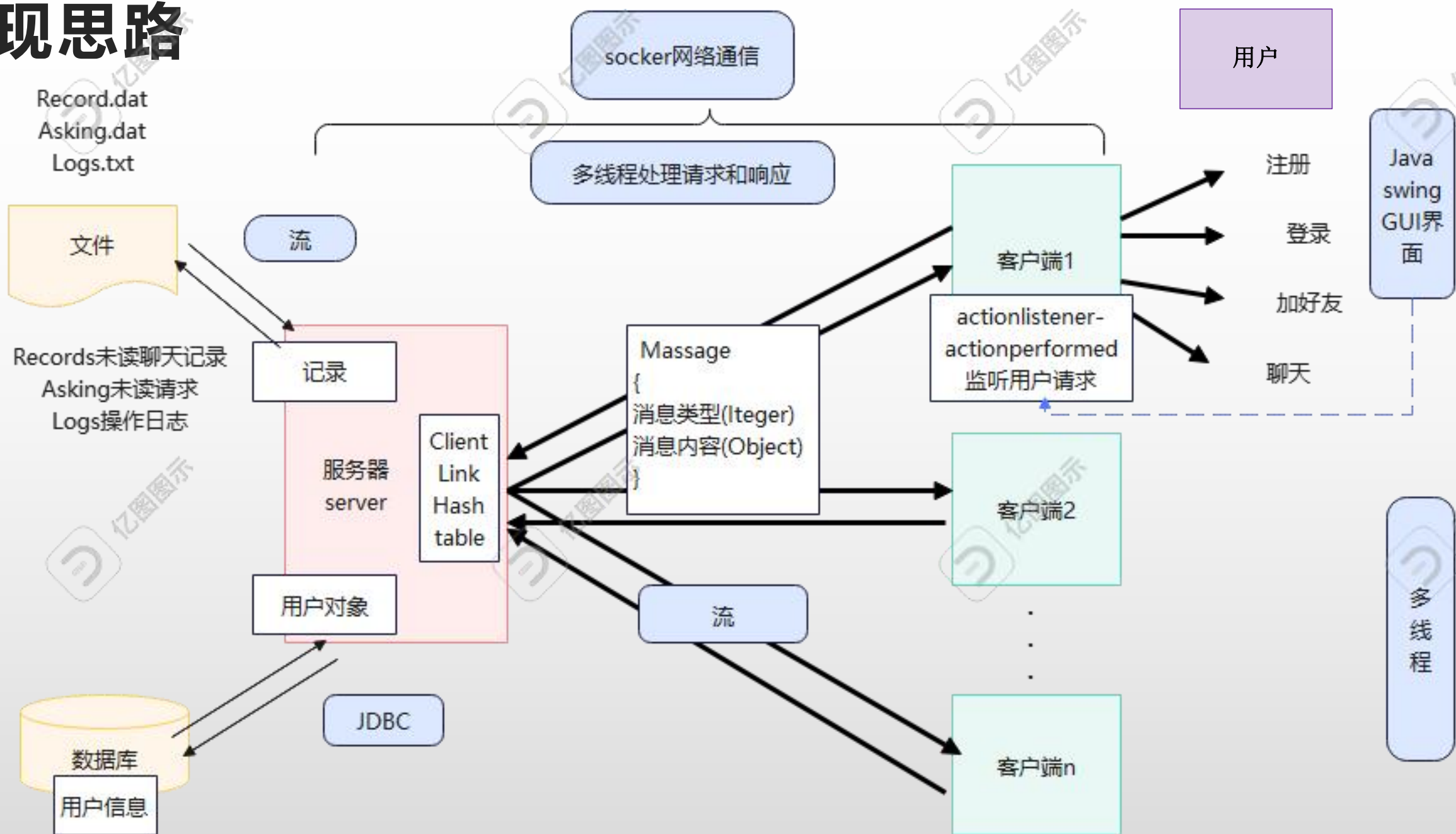
- 1. 登陆界面，登录不成功；
- 2. 加好友不成功；（查无此人或对方拒绝）
- 3. 给对方发消息或好友申请，对方不在线，数据暂存进文件，当对方登录时弹出界面。



实现思路



实现思路

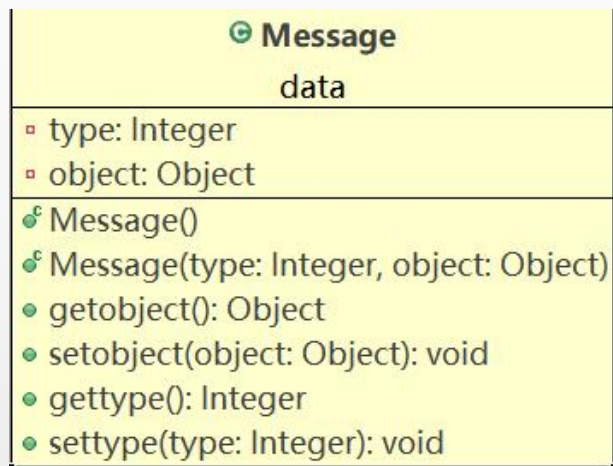


简述

- 以服务器为中心，前端使用Socket 网络编程、多线程、对象序列化流机制实现与客户端的交互，客户端用Java swing与用户交互、后端多线程处理客户端请求，用JDBC，对象序列化、文件输入输出流储存用户信息和操作日志。

技术重难点：客户端、服务器之间用socket进行信息交流

信息媒介：message类



服务器内部连接类

客户端主面板类

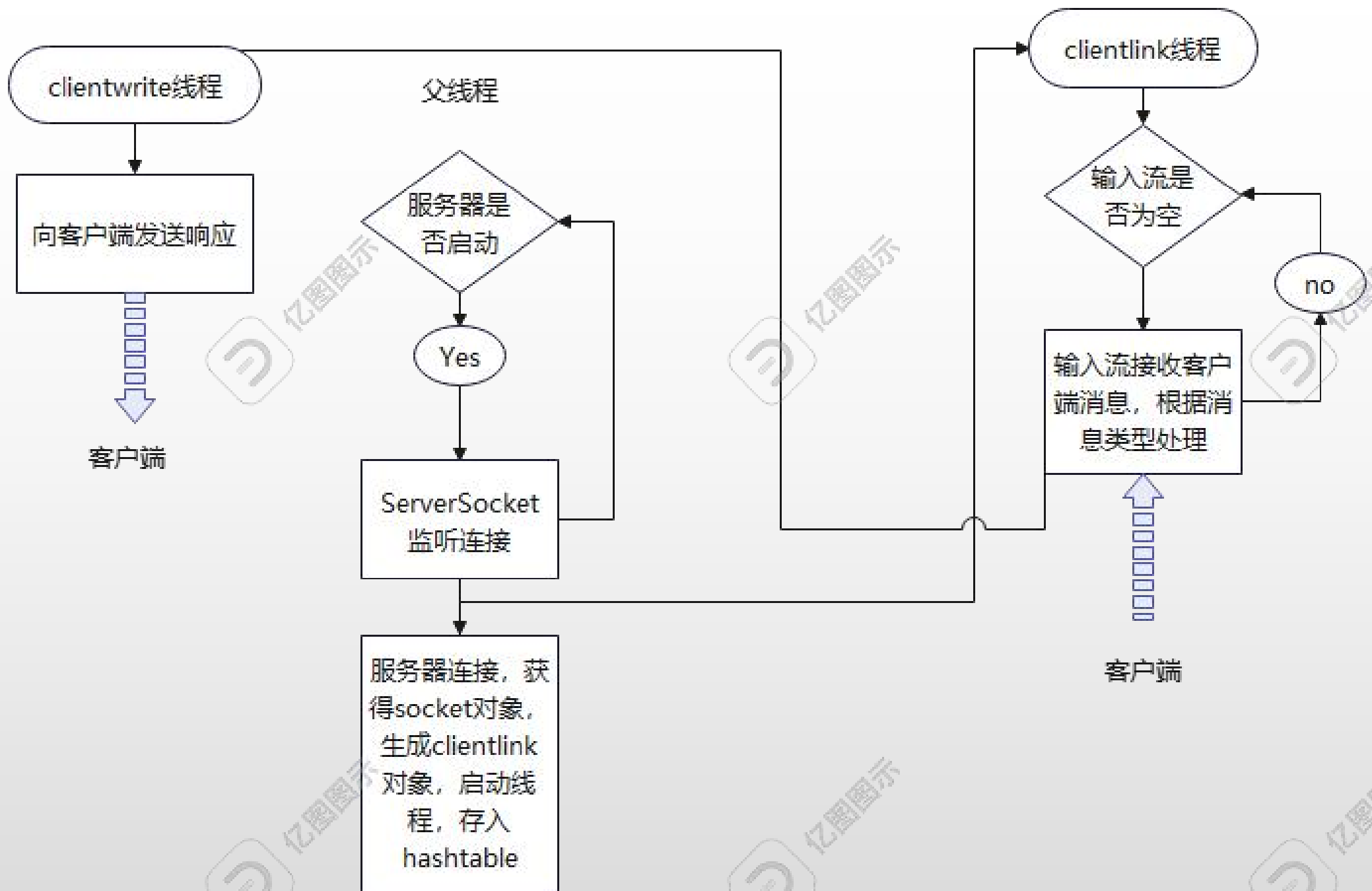


Message编号对应类型

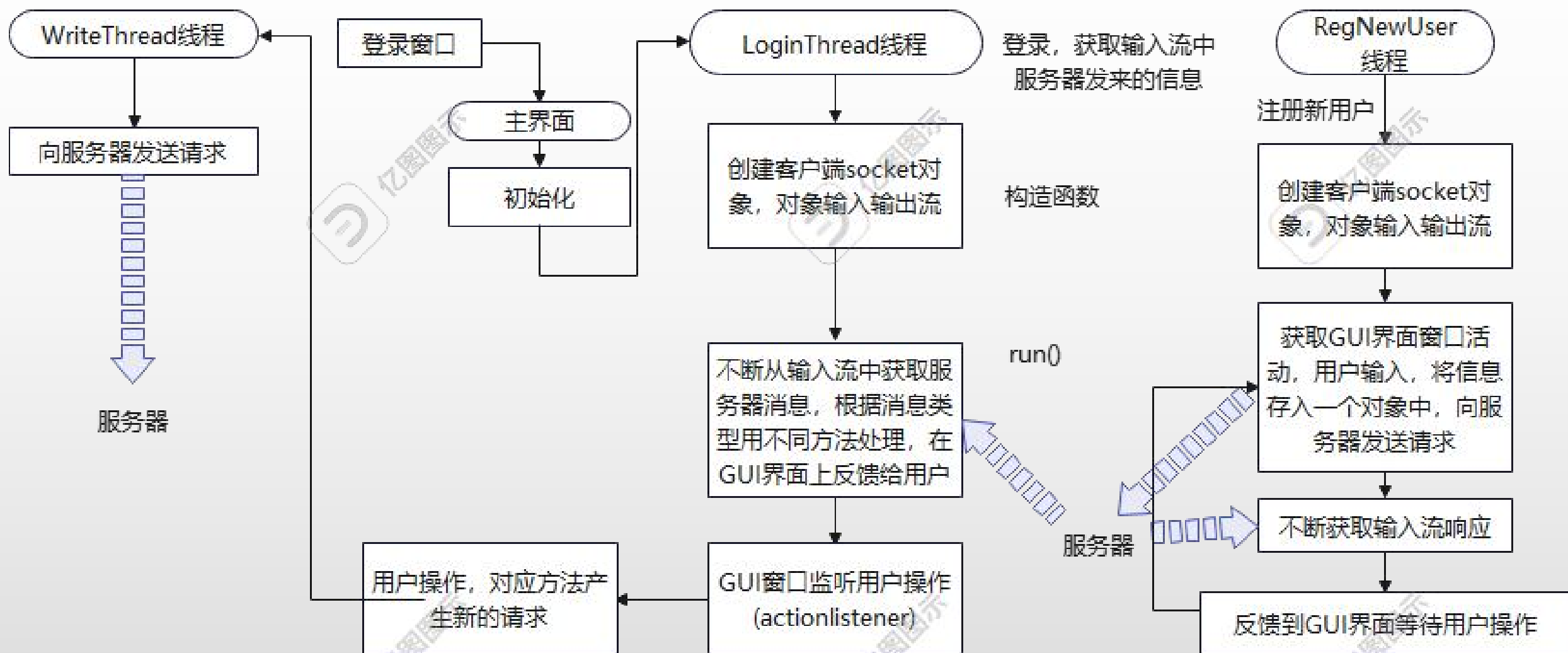
Type值	Object类型	接收方	消息
10	RegUser	服务器	客户端发送注册信息到服务端
11	RegUser	客户端	服务端回复注册成功到客户端
12	null	客户端	服务端回复注册失败到客户端
20	LoginUser	服务器	客户端发送登陆信息到服务端
21	Vector<FriendUser>	客户端	登陆成功
22	String	客户端	登录失败
23	String	客户端	服务端发送账号在别处登陆
24	FriendUser	服务器	客户端发送退出到服务端
25	FriendUser	客户端	服务端发送好友上线功能
30	Record	服务器	客户端发送消息到服务端
31	Record	客户端	服务端根据消息发送到客户端
40	Integer	服务器	客户端发送好友申请
41	Asking	客户端	服务器向对应客户端发送好友申请
45	Vector<Integer>	服务器	客户端发送互相添加好友请求
46	String	客户端	向客户端发送用户不存在添加失败
47	FriendUser	服务器	客户端拒绝添加好友，添加好友失败
49	FriendUser	客户端	服务器向客户端发送添加好友成功
90	null	客户端	服务端发送下线功能到客户端

- 1.服务器、客户端用socket对象输入输出流互相发送、接收Message。
- 2.在服务器、客户端的线程中，输入流接收到message后，根据message传递的消息类型分情况调用不同方法完成操作，并用输出流将响应或请求反馈给对方。

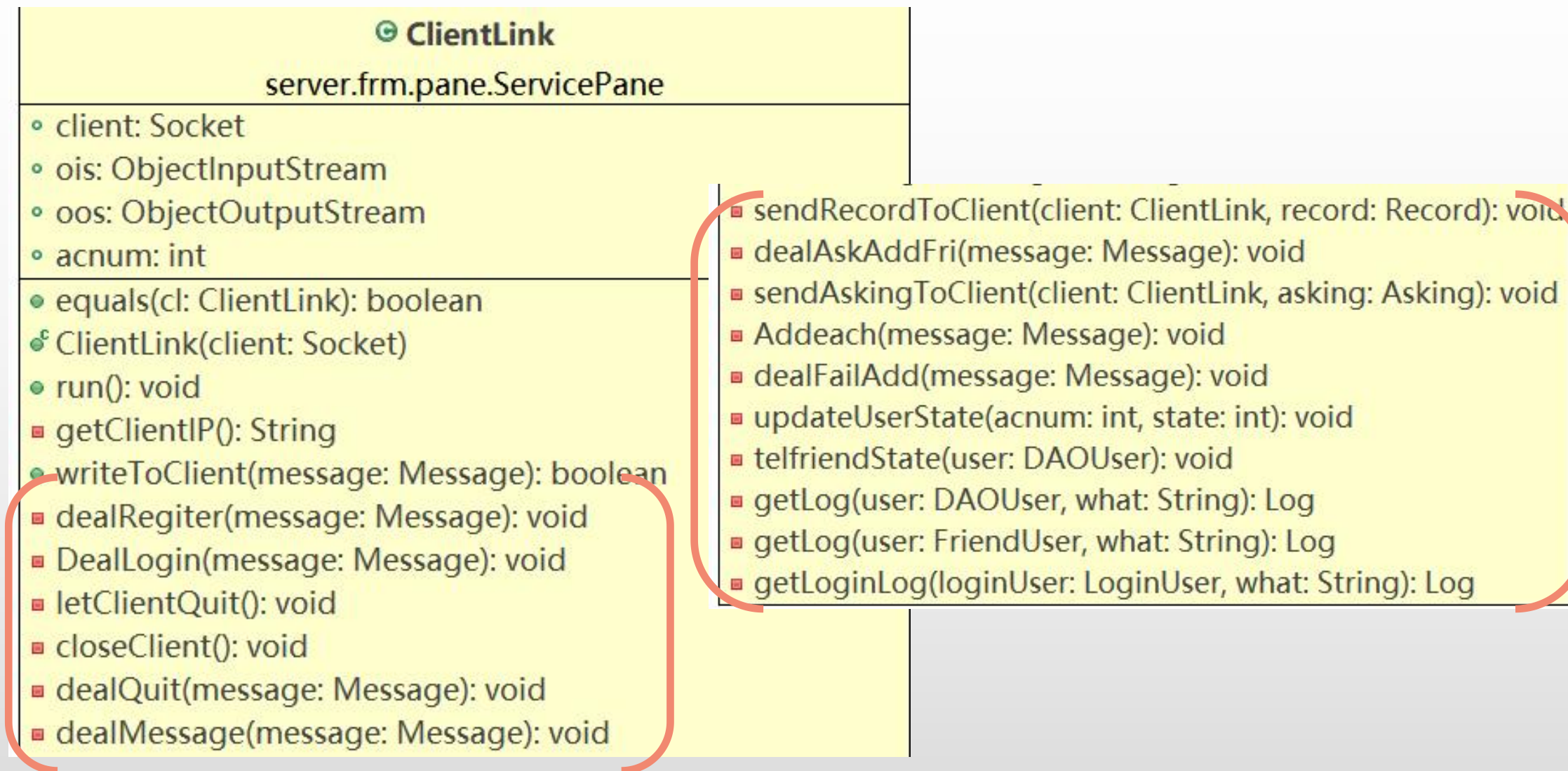
服务器线程



客户端线程



举例：服务器处理消息的线程类



run()方法中处理消息举例

```
switch (type) {  
    case 20://客户端登录  
    {  
        DealLogin(message);  
        break;}  
    case 24://客户端退出  
        dealQuit(message);  
        break;  
    case 30://发聊天消息  
        dealMessage(message);  
        break;  
    case 40://好友请求  
        dealAskAddFri(message);  
        break;  
    case 45://互相添加好友  
        Addeach(message);  
        break;  
    case 47://添加好友失败  
        dealFailAdd(message);  
        break;  
}
```


处理方法举例

```
//10: 处理注册信息
private void dealRegiter(Message message) throws FileNotFoundException {
    if(message.getobject() instanceof DAOUser) {
        DAOUser user=(DAOUser)message.getobject();
        JQCreator creator = new JQCreator();
        int id = creator.createID();           step1.取出
        int num = creator.createJQ();         object进行处理
        creator.saveIDJQ(id, num);
        user.setId(id);
        user.setAcnum(num);
        user.setRegisterTime(new java.sql.Date(System.currentTimeMillis()));
        UserDAObyMySQL udbm=new UserDAObyMySQL();
        boolean b=udbm.add(user);
    }
}
```

step2.综合对象，按不同
条件发送不同消息

```
Message regresultMessage=new Message();
if(b) {
    regresultMessage.settype(11);
    RegUser regUser=new RegUser();
    regUser.setAcnum(user.getAcnum());
    regUser.setNickname(user.getNickname());
    regUser.setPassword(user.getPassword());
    regresultMessage.setobject(regUser);
    writelog(getLog(user,"新用户注册成功!"));
}
else{
    regresultMessage.settype(12);
    regresultMessage.setobject(null);
    writelog(getLog(user,"用户注册失败!"));
}
writeToClient(regresultMessage);
}
else {
```

谢谢！