网络应用开发实训项目报告

目录

[第一章 引言 3](#_Toc12116194)

[第二章 概要设计 3](#_Toc12116195)

[1 使用平台及技术 3](#_Toc12116196)

[⑴ Java的优点 3](#_Toc12116197)

[⑵ MySQL介绍 4](#_Toc12116198)

[2环境的搭建过程 4](#_Toc12116199)

[⑴ Eclipse的安装 4](#_Toc12116200)

[⑵ MySQL的安装 4](#_Toc12116201)

[⑶ JDBC驱动包的下载与引入 4](#_Toc12116202)

[3 功能和设计方法 5](#_Toc12116203)

[⑴ 注册功能 5](#_Toc12116204)

[⑵ 登录功能 5](#_Toc12116205)

[⑶ 忘记密码 5](#_Toc12116206)

[⑷ 单独聊天（私聊） 5](#_Toc12116207)

[⑸ 多人聊天（群聊） 6](#_Toc12116208)

[⑹ 添加好友 6](#_Toc12116209)

[⑺ 修改密码 6](#_Toc12116210)

[第三章 详细设计 6](#_Toc12116211)

[1 流程图介绍 6](#_Toc12116212)

[⑴ 系统总体结构设计 6](#_Toc12116213)

[⑵ 系统主体处理流程 7](#_Toc12116214)

[⑶ 系统流程逻辑 7](#_Toc12116215)

[3 数据库表展示 8](#_Toc12116216)

[⑴ 数据库构成截图 8](#_Toc12116217)

[⑵ 用户表（user）的结构及存放的数据 8](#_Toc12116218)

[⑶ 某一用户好友(该账号为1)表结果及数据 9](#_Toc12116219)

[4 主要算法、类的描述 10](#_Toc12116220)

[⑴ 客户端中主要类、算法的描述 10](#_Toc12116221)

[⑵ 服务器端主要类、算法的描述 10](#_Toc12116222)

[5 主要的代码段 10](#_Toc12116223)

[⑴ 客户端的主要代码段（略去包名） 10](#_Toc12116224)

[⑵ 服务器端主要代码段（略去包名） 23](#_Toc12116225)

[第四章 测试和演示 41](#_Toc12116226)

[1 测试内容及预测结果 41](#_Toc12116227)

[2 测试结果与截图 41](#_Toc12116228)

[(1) 测试点一 41](#_Toc12116229)

[(2) 测试点二 41](#_Toc12116230)

[(3) 测试点三 41](#_Toc12116231)

[(4) 测试点四 41](#_Toc12116232)

[(5) 测试点五 41](#_Toc12116233)

[(6) 测试点六 41](#_Toc12116234)

[(7) 测试点七 41](#_Toc12116235)

[(8) 测试点八 41](#_Toc12116236)

[第五章 总结 42](#_Toc12116237)

# 第一章 引言

本系统是基于java开发的聊天软件。有用户注册、用户登陆、修改密码、忘记密码、添加好友、用户聊天、群聊等功能。如果服务器还没有启动，则客户端是不可以登陆、注册、忘记密码。

# 第二章 概要设计

## 1 使用平台及技术

此项目运行在Windows 10上，使用Intellij IDEA作为IDE，用MySQL作为数据库。以Java为主要设计语言。

⑴ Java的优点

①简单性

Java实现了自动的功能收集，简化了内存管理的工作，这使程序设计更简便，同时减少了出错的可能。

②面向对象

Java提供了简单的类机制和动态的架构模型。对象中封装了它的状态变量和方法。很好地实现了模块化和信息隐藏；而类则提供了一类对象的原型，通过继承和重载机制，子类可以使用或重新定义父类或超类所提供的方法，从而既实现了代码的复用，又提供了一种动态的解决方案。

Java同样支持继承特性，java的类可以从其他类中继承行为，但java只支持类的单重继承，即每个类只能从一个类中继承。

Java支持界面，界面允许程序员定义方法但不立即实现，一个类可以实现多个界面，利用界面可以得到多重继承的许多优点而又没有多重继承的问题。

③多线程

多线程使应用程序可以同时进行不同的操作，处理不同的事件。在多线程机制中，不同的线程处理不同的任务，他们之间互不干涉，不会由于一处等待影响其他部分，这样容易实现网络上的实时交互操作。多线程保证了较高的执行效率。

④安全性

用于网络、分布环境下的java必须要防止病毒的入侵，java不支持指针，一切对内存的访问都必须通过对象的实例变量来实现，这样就防止了程序员使用欺骗手段访问对象的私有成员，同时也避免了指针操作中容易产生的错误。

⑵ MySQL介绍

MySQL是一个关系型数据库管理系统，由瑞典MySQL AB 公司开发，目前属于 Oracle 旗下产品。MySQL 是最流行的关系型数据库管理系统之一，在 WEB 应用方面，MySQL是最好的 RDBMS (Relational Database Management System，关系数据库管理系统) 应用软件。

MySQL是一种关系数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。

MySQL所使用的 SQL 语言是用于访问数据库的最常用标准化语言。MySQL 软件采用了双授权政策，分为社区版和商业版，由于其体积小、速度快、总体拥有成本低，尤其是开放源码这一特点，一般中小型网站的开发都选择 MySQL 作为网站数据库。

## 2环境的搭建过程

⑴ Eclipse的安装

在<https://www.jetbrains.com/idea/>中选择合适的版本下载安装到本地。

⑵ MySQL的安装

在<https://www.mysql.com/downloads/>中选择Microsoft Windows版本下载到本地，然后安装（按默认安装即可），设置密码MySQL Root Password并记住。

⑶ JDBC驱动包的下载与引入

由于Eclipse不自带JDBC驱动，然而JDBC驱动是Java连接数据库必要的包，因此按以下步骤下载引入包。

①JDBC的下载

在浏览器输入<https://dev.mysql.com/downloads/connector/j/>，选择zip文件下载到本地，解压到合适的位置。

②JDBC包的导入与项目导入

* 在MySql中建立数据库vchatroom，运行vchatroom.sql对数据库进行初始化。接着在VirtualChatRoomServer的工程文件里，把src/DAO/Dao.java里面的数据库配置给修改成自己的mysql账号密码。
* 进入VirtualChatRoom/src/frame/loginFrame.java，点击菜单栏的Run，选择Edit Configurations，在VM options选项后加入-Dfile.encoding=GB18030，防止乱码。
* 在主界面中，点击右边侧栏的 Database ，在点击 + ，再Data Source 选择数据库
* 填入 Database 数据库名，在输入 User 和 Password ，点击 Test Connection
* 在intelij IDEA 中，点击File-Project Structure，出现界面的左侧点击Modules，在Dependencies标签界面下，点击+号新建“JARs or directories...”，到刚才下载的位置选中mysql-connector-java-8.0.15.jar，点确定。

## 3 功能和设计方法

总体而言，在客户端：当用户登录后，生成唯一的socket， 存放在ManageClientThread实体类中，在整个客户端就一个Client类和一个socket。有一个窗口控制器——ChatUIList，用来记录用户和好友聊天框是否打开，当收到消息后，首先在ManageFriendList中查询是否有好友的窗口，如果没有则新建聊天框弹出并显示消息，如果存在与好友的窗口则将消息追加到原聊天框并重新着重显示该窗口。在客户端还拥有一个“命令控制中心”——ClientConServerThread类，在该类中判断并处理来自服务器中的命令（消息），如果是“nessage\_comm\_mes”那么客户端收到是来自好友的消息，类似有“message\_frameShake”, “message\_newOnline\_friend”，“,essage\_regist\_permit”等命令。在服务端：有多个Thread，用ManageClientThread管理连接成功的用户名及其Thread对应的Socket。同样在服务端也有一个“命令控制中心”——ServerConClientThread类，它负责处理来自客户端的命令（消息），判断命令的类型，并正确处理他们，给出处理结果和判断是否处理成功，将处理后的命令转发给正确的用户。

⑴ 注册功能

设计一个注册UI（RegistFrame）类，在打开程序后，模仿QQ在左下方有一个注册按钮，点击注册按钮后弹出注册页面，用户填完必填信息后由客户端将命令发送给服务端（如果服务器在线），服务器收到“message\_regist\_request”命令后，连接数据库判断，如果注册成功则返回注册成功消息并弹窗提示，如果失败则弹窗提示注册失败。

⑵ 登录功能

打开客户端后，类似QQ有登录按钮，当用户填完用户账号和用户密码并提交后，客户端将登录请求发送给给服务端判断（如果服务端在线），如果密码正确则用户登录成功，显示朋友列表（ManageFriendList），否则提示密码错误或服务器异常。

⑶ 忘记密码

该功能尚未实现，需要修改密码请联系管理员。

⑷ 单独聊天（私聊）

用户登录成功后，双击好友后，首先判断用户与该好友是否有聊天框存在，如果不存在则创建新的聊天框（ChatFrame）并在ManageChat中登记，如果存在则将改聊天框突出显示。用户可以再聊天页面（ChatFrame）发送消息，如果好友在线收到消息则无提示（类似Linux，没有消息就是好消息）。好友收到消息时，在ManageChat中查询是否有与该好友的聊天窗口，如果没有则新建窗口显示并在ManageChat中注册，如果存在则直接将消息追加到聊天窗口上。

⑸ 多人聊天（群聊）

这里的多人聊天功能是单独地创建了一批账号，它们的作用就是群聊专用账号，当其余的用户添加该账号就相当于加入了该群。不同于私聊的地方就在于，当服务器拿到一个用户发给该群聊账号的消息后，把消息分发给所有加了这个账号的人，并且在该聊天窗口中回显该消息以达成群聊效果。

⑹ 添加好友

在好友列表页面左下角有添加好友按钮，点击该按钮后弹出添加好友框（AddNewFriend），在添加好友框中输入欲添加的好友id(即账号)便可向服务器发送好友请求。目前默认自动同意加为好友，成功之后好友会直接在好友列表刷新，自动出现在好友列表底部(目前没有加入置顶上线好友，好友排序等功能)。

⑺ 修改密码

该功能尚未添加，需要修改密码可持账号以及旧密码联系管理员。

# 第三章 详细设计

## 1 流程图介绍

⑴ 系统总体结构设计

基于java实现聊天室

客户端

服务器端

聊天室

接收信息

清屏

聊天记录

发送信息

⑵ 系统主体处理流程

发送请求

服务端

客户端

返回应答

⑶ 系统流程逻辑

开始

**­**

运行服务端代码

运行客户端代码

登录成功

发送信息

接收信息

清屏

显示聊天内容

是否退出

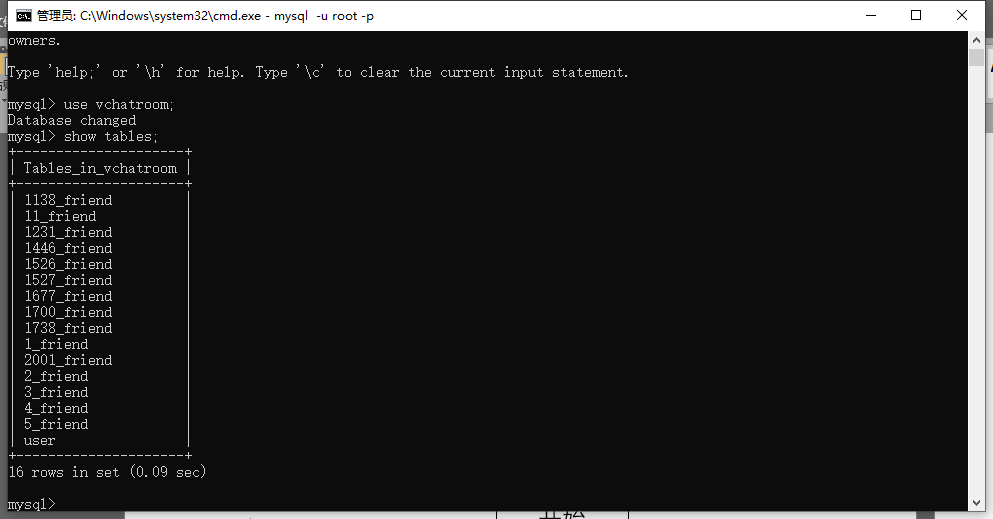
结束

## 3 数据库表展示

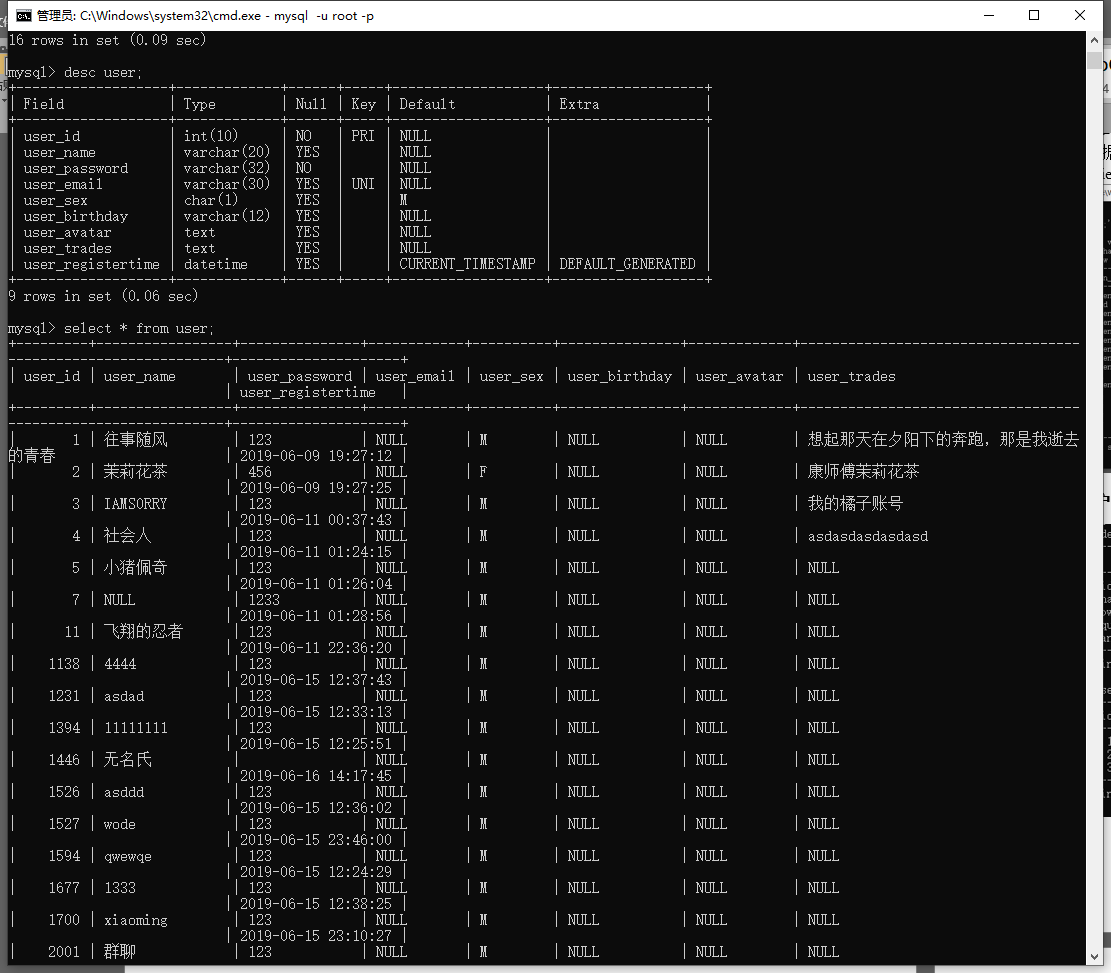
数据库名为vchatroom， 此数据库中若干个表，一个用户表（user），用来存储用户的信息，如用户名，用户密码，头像，个性签名等；每一个用户有一个好友表，好友表里存着用户的好友名。

⑴ 数据库构成截图

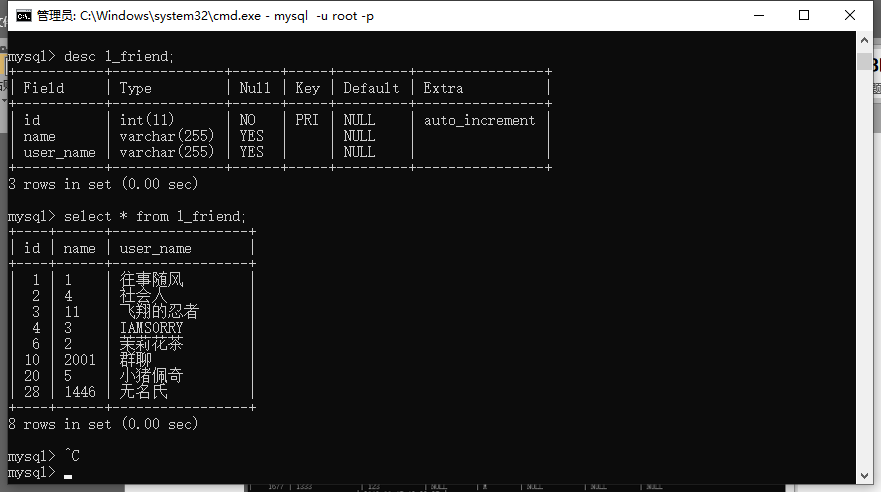
此时数据库中一共有如下几个表，最下面的user表就是用户表，其余的账号下划线加friend的表都是每个账号对应的好友列表。



⑵ 用户表（user）的结构及存放的数据



⑶ 某一用户好友(该账号为1)表结果及数据



## 4 主要算法、类的描述

⑴ 客户端中主要类、算法的描述

① UI类

AddNewFriend、ChatFrame、LoginFrame、MainFrame、RegistFrame、RetraceAccount分别为添加好友，聊天界面，登录界面，主界面（好友列表界面），注册界面，找回密码界面（开发中）。这一部分主要是显示处理和逻辑处理。

② 消息处理、通信类

ManageChat类主要记录客户端打开的聊天页面，处理与好友的消息弹窗。ManageClientConServerThread类中有socket负责和服务器通信。ClientConServerThread类是客户端的消息处理中心，处理来自服务器的各种消息并做出相应。

⑵ 服务器端主要类、算法的描述

① UI类

服务器端只有两个UI页面——MyserverFrame和MyRegisterFrame，因为服务器没有太多消息要显示，所以分别一个启动窗口即可。

② 消息处理、通信类

model类中有Myserver，负责与登录的客户端建立通信，每个建立的通信都通过ManageClientThread存储中，供服务器查询哪些用户上线。ServerConClientThread类是服务器端的控制中心，负责处理来自用户端的消息，并转发给正确的用户，有时还会对数据库进行操作。

③ 数据库处理相关

DAO类负责和数据库建立连接，UserDao类负责处理具体的和数据库交互的内容，如查询用户账号和密码是否匹配、修改密码、注册用户、忘记密码、添加好友等操作。

## 5 主要的代码段

⑴ 客户端的主要代码段（略去包名）

① 登录界面（LoginFrame）

|  |
| --- |
| package frame;  import bean.Message;  import bean.MessageType;  import bean.User;  import frame.FrameTools.DialogFrame;  import frame.FrameTools.FrameAlignCenter;  import frame.FrameTools.JTextFieldHintListener;  import javafx.scene.layout.Background;  import model.ClientConServer;  import model.ClientUser;  import sun.management.jdp.JdpJmxPacket;  import tools.MD5;  import tools.ManageClientConServerThread;  import tools.ManageFirendList;  import javax.imageio.ImageIO;  import javax.swing.\*;  import java.awt.\*;  import java.awt.event.ActionEvent;  import java.awt.event.ActionListener;  import java.awt.event.FocusEvent;  import java.awt.event.FocusListener;  import java.io.File;  import java.io.ObjectOutputStream;  import java.security.MessageDigest;  public class LoginFrame extends JFrame implements ActionListener {  public JLabel first, header;  public JPanel second;  public JPanel jp\_userId;  public JPanel jp\_password;  public JPanel jp\_but;  private ImageIcon bgImage = null;  private JLabel userId = null;  private JLabel password = null;  private JTextField tfUid;  private JTextField tfPwd;  private JPasswordField tpfPwd;  private JButton login, regist, reTrace;  private JCheckBox rememberPwd, autoLogin;  private JLabel remLab,autoLab;  private ObjectOutputStream oos;  public static void main(String[] args) {  LoginFrame loginFrame = new LoginFrame();  loginFrame.init();  }  public void init() {  this.setLayout(new GridLayout(2, 1));  this.setTitle("用户登录");  this.setSize(450, 320);  this.setAlwaysOnTop(true);  Toolkit toolkit = Toolkit.getDefaultToolkit();  //设置图标  Image iconImage = toolkit.createImage("src/frame/image/qq\_logo.png");  this.setIconImage(iconImage);  //设置窗体居中  /\* int width = Toolkit.getDefaultToolkit().getScreenSize().width;  int height = Toolkit.getDefaultToolkit().getScreenSize().height;  int windowWidth = this.getWidth();  int windowHeight = this.getHeight();  this.setBounds((width - windowWidth) / 2, (height - windowHeight) / 2, windowWidth, windowHeight);\*/  new FrameAlignCenter(this);  //添加上背景  first = new JLabel();  first.setOpaque(true);  first.setBackground(new Color(30, 144, 255));  header = new JLabel();  header.setBounds(14, 5, 82, 83);  String headPortraitPostion = "src/frame/image/tempheadportrait.jpg";  // Todo sometings get head portrait position  Image headPic = (new ImageIcon(headPortraitPostion)).getImage().getScaledInstance(82, 83, Image.SCALE\_DEFAULT);  header.setIcon(new ImageIcon(headPic));  //添加下背景  second = new JPanel();  second.setLayout(null);  second.setBackground(Color.white);  this.getContentPane().add(first, "North");  this.getContentPane().add(second, "South");  //给下背景添加元素  userId = new JLabel("账号");  userId.setBounds(new Rectangle(0, 0, 50, 30));  userId.setFont(new Font("微软雅黑", Font.BOLD, 14));  password = new JLabel("密码");  password.setBounds(new Rectangle(0, 0, 50, 30));  password.setFont(new Font("微软雅黑", Font.BOLD, 14));  tfUid = new JTextField();  tfUid.addFocusListener(new JTextFieldHintListener(tfUid, "请输入账号"));  tfUid.setBounds(new Rectangle(40, 0, 160, 30));  tpfPwd = new JPasswordField(10);  tpfPwd.addFocusListener(new FocusListener() {  @Override  public void focusGained(FocusEvent e) {  }  @Override  public void focusLost(FocusEvent e) {  jp\_password.add(tfPwd);  tfPwd.setText("密码");  tfPwd.setForeground(Color.gray);  jp\_password.remove(tpfPwd);  jp\_password.updateUI();  }  });  tfPwd = new JTextField("密码", 10);  tfPwd.setBounds(new Rectangle(40, 0, 160, 30));  tfPwd.setForeground(Color.gray);  tfPwd.addFocusListener(new FocusListener() {  @Override  public void focusGained(FocusEvent e) {  jp\_password.add(tpfPwd);  tpfPwd.requestFocus();  tpfPwd.setBounds(new Rectangle(40, 0, 160, 30));  jp\_password.remove(tfPwd);  jp\_password.updateUI();  }  @Override  public void focusLost(FocusEvent e) {  }  });  login = new JButton("登录");  login.setFocusPainted(false);  login.requestFocus();  login.addActionListener(this);  login.setBackground(new Color(30, 144, 255));  regist = new JButton();  regist.setIcon(new ImageIcon("src/frame/loginUi/zhuce.png"));  regist.setRolloverIcon(new ImageIcon("src/frame/loginUi/zhuce\_hover.png"));  regist.setPressedIcon(new ImageIcon("src/frame/loginUi/zhuce\_press.png"));  regist.setOpaque(false);  regist.setContentAreaFilled(false);  regist.setBorder(null);  regist.addActionListener(this);  second.add(regist);  regist.setBounds(new Rectangle(325, 10, 90, 30));  regist.setFont(new Font("微软雅黑", Font.PLAIN, 10));  reTrace = new JButton();  reTrace.setIcon(new ImageIcon("src/frame/loginUi/mima.png"));  reTrace.setRolloverIcon(new ImageIcon("src/frame/loginUi/mima\_hover.png"));  reTrace.setPressedIcon(new ImageIcon("src/frame/loginUi/mima\_press.png"));  reTrace.setOpaque(false);  reTrace.setBorder(null);  reTrace.setContentAreaFilled(false);  second.add(reTrace);  reTrace.setBounds(new Rectangle(325, 50, 90, 30));  reTrace.setFont(new Font("微软雅黑", Font.PLAIN, 10));  rememberPwd = new JCheckBox();  rememberPwd.setMargin(new Insets(0,0,0,0));  rememberPwd.setBounds(new Rectangle(108, 88, 17, 17));  rememberPwd.setIcon(new ImageIcon("src/frame/image/checkbox\_normal.png"));  rememberPwd.setRolloverIcon(new ImageIcon("src/frame/image/checkbox\_normal.png"));  rememberPwd.setPressedIcon(new ImageIcon("src/frame/image/checkbox\_press.png"));  rememberPwd.setSelectedIcon(new ImageIcon("src/frame/image/checkbox\_tick\_normal1.png"));  remLab=new JLabel("记住密码");  remLab.setForeground(Color.gray);  remLab.setBounds(new Rectangle(130, 80, 90, 30));  second.add(rememberPwd);  second.add(remLab);  autoLogin = new JCheckBox();  autoLogin.setMargin(new Insets(0,0,0,0));  autoLogin.setIcon(new ImageIcon("src/frame/image/checkbox\_normal.png"));  autoLogin.setRolloverIcon(new ImageIcon("src/frame/image/checkbox\_normal.png"));  autoLogin.setPressedIcon(new ImageIcon("src/frame/image/checkbox\_press.png"));  autoLogin.setSelectedIcon(new ImageIcon("src/frame/image/checkbox\_tick\_normal1.png"));  autoLogin.setBounds(new Rectangle(253, 88, 17, 17));  autoLab=new JLabel("自动登录");  autoLab.setForeground(Color.gray);  autoLab.setBounds(new Rectangle(275, 80, 90, 30));  second.add(autoLogin);  second.add(autoLab);  jp\_userId = new JPanel();  jp\_userId.setOpaque(false);  jp\_userId.setLayout(null);  jp\_password = new JPanel();  jp\_password.setOpaque(false);  jp\_password.setLayout(null);  jp\_but = new JPanel();  jp\_but.setLayout(new GridLayout());  jp\_but.setOpaque(true);  jp\_userId.add(userId);  jp\_userId.add(tfUid);  jp\_password.add(password);  jp\_password.add(tfPwd);  //jp\_password.add(tpfPwd);  jp\_but.add(login);  jp\_userId.setBounds(110, 10, 200, 60);  jp\_password.setBounds(110, 50, 200, 60);  jp\_but.setBounds(110, 110, 200, 30);  second.add(jp\_userId);  second.add(jp\_password);  second.add(jp\_but);  second.add(header);  this.setDefaultCloseOperation(WindowConstants.EXIT\_ON\_CLOSE);  this.setVisible(true);  this.requestFocusInWindow();  }  public void loginAction() {  ClientUser clientUser = new ClientUser();  User u = new User();  String psw=new String(tpfPwd.getPassword());  u.setUserId(tfUid.getText().trim());  //对密码进行MD5加密  u.setUserPwd(MD5.converMD5(psw));  if (clientUser.checkUser(u)) {  try {  //先创建好友列表  MainFrame mainFrame = new MainFrame(u.getUserId());  ManageFirendList.addMainFrame(u.getUserId(), mainFrame);  //发送一个要求返回在线好友的包  oos = new ObjectOutputStream(ManageClientConServerThread.getClientConServerThread(u.getUserId()).getS().getOutputStream());  Message m = new Message();  m.setMessageType(MessageType.message\_get\_onLineFriend);  //标记请求发送方ID  m.setSender(u.getUserId());  System.out.println(m.getSender()+"请求刷新好友列表");  oos.writeObject(m);  } catch (Exception ex) {  ex.printStackTrace();  }  this.dispose();  } else {  new DialogFrame("警告","用户名或密码错误，请重试",300,100);  }  }  @Override  public void actionPerformed(ActionEvent e) {  if (e.getSource() == login) {  loginAction();  }else if(e.getSource()==regist){  new RegistFrame();  }  }  } |

② 消息处理核心——ClientConServerThread类

|  |
| --- |
| /\*  客户端和服务器端保持通信的线程  \*/  package tools;  import bean.Message;  import bean.MessageType;  import frame.ChatFrame;  import frame.FrameTools.ShakeFrame;  import frame.MainFrame;  import model.ServerReceiveFile;  import javax.swing.\*;  import javax.swing.text.SimpleAttributeSet;  import javax.swing.text.StyleConstants;  import java.awt.\*;  import java.io.IOException;  import java.io.ObjectInputStream;  import java.net.Socket;  import java.text.SimpleDateFormat;  import java.util.ArrayList;  public class ClientConServerThread extends Thread {  private Socket s;  public int friendNum;  public ArrayList<String> friendList;  public ArrayList<String> friendNameList;  public Socket getS() {  return s;  }  public ClientConServerThread(Socket s) {  this.s = s;  }  public int getFriendNum() {  return friendNum;  }  public ArrayList<String> getFriendList() {  return friendList;  }  public void run() {  while (true) {  //等待读取从服务器端发来的消息  try {  if (s.getInputStream().available() != 0) {  ObjectInputStream ois = new ObjectInputStream(s.getInputStream());  Message m = (Message) ois.readObject();  if (m.getMessageType().equals(MessageType.message\_comm\_mes)) {  //普通信息包，返回聊天内容  System.out.println("读取到:" + m.getSender() + "发送" + m.getContent() + "给" + m.getReciever());  //把从服务器获得的消息，显示到目标客户端的聊天界面  if(ManageChat.hasChat(m.getReciever()+" "+m.getSender())){  ChatFrame chatFrame = ManageChat.getChat(m.getReciever() + " " + m.getSender());  //显示  chatFrame.showMessage(m);}  else{  ChatFrame chatFrame=new ChatFrame(m.getReciever(),m.getSender());  ManageChat.addChat(m.getReciever()+" "+m.getSender(),chatFrame);  chatFrame.showMessage(m);  }  } else if (m.getMessageType().equals(MessageType.message\_comm\_mes\_group)){  System.out.println("读取到:"+m.getSender()+"发送"+m.getContent()+"给"+m.getReciever());  for(int i=0;i<m.getOnlineList().length;i++){  ChatFrame chatFrame=ManageChat.getChat(m.getOnlineList()[i]+" "+m.getReciever());  System.out.println("发送人："+m.getSender()+";发送信息："+m.getContent());  if(chatFrame!=null)  chatFrame.showMessage(m);  }  }else if (m.getMessageType().equals(MessageType.message\_ret\_onLineFriend)) {  //返回好友列表信息包  //收到好友信息包之后将好友的id和与之对应的昵称放入hashMap中，调用ManageFriendList  String reciever = m.getReciever();  String content = m.getContent();  String user\_name = m.getName();  friendNum = m.getFriendNum();  System.out.println("好友人数为："+friendNum);  System.out.println("ID: ");  for(String temp:m.getFriendList()){  System.out.print(temp+"\t");  }  System.out.println("\n昵称:");  for(String temp:m.getFriendNameList()){  System.out.print(temp+" ");  }  System.out.println();  System.out.println("我是" + reciever + "昵称: " + user\_name + " " + content + "上线了");  MainFrame mainFrame = ManageFirendList.getMainFrame(reciever);  if (mainFrame != null) {  mainFrame.updateFirendList(m);  }  } else if (m.getMessageType().equals(MessageType.message\_newOnline\_friend)) {  //返回好友列表信息包  //收到好友信息包之后将好友的id和与之对应的昵称放入hashMap中，调用ManageFriendList  String reciever = m.getReciever();  String content = m.getContent();  String user\_name = m.getName();  System.out.println("我是" + reciever + "昵称: " + user\_name + " " + content + "上线了");  MainFrame mainFrame = ManageFirendList.getMainFrame(reciever);  if (mainFrame != null) {  mainFrame.updateNewOnlineFriend(m);  }  } else if (m.getMessageType().equals(MessageType.message\_ret\_logout)) {  //返回下线好友信息包  MainFrame mainFrame = ManageFirendList.getMainFrame(m.getReciever());  if (mainFrame != null) {  mainFrame.updateFriendOffLine(m);  }  } else if(m.getMessageType().equals(MessageType.message\_frameShake)){  if(ManageChat.hasChat(m.getReciever()+" "+m.getSender())){  ChatFrame chatFrame=ManageChat.getChat(m.getReciever()+" "+m.getSender());  new ShakeFrame(chatFrame);  SimpleAttributeSet simpleAttributeSet=new SimpleAttributeSet();  StyleConstants.setForeground(simpleAttributeSet,Color.gray);  chatFrame.stringInsertContent("\n"+ManageFirendList.getFriendName(m.getSender())+"向您发送了窗口抖动",simpleAttributeSet);  }else{  ChatFrame chatFrame=new ChatFrame(m.getReciever(),m.getSender());  ManageChat.addChat(m.getReciever()+" "+m.getSender(),chatFrame);  new ShakeFrame(chatFrame);  SimpleAttributeSet simpleAttributeSet=new SimpleAttributeSet();  StyleConstants.setForeground(simpleAttributeSet,Color.gray);  chatFrame.stringInsertContent("\n"+ManageFirendList.getFriendName(m.getSender())+"向您发送了窗口抖动",simpleAttributeSet);  }  }else if(m.getMessageType().equals(MessageType.message\_request\_friend\_info)){  System.out.println("收到了来自服务器发来的好友"+m.getReciever()+"个性签名:"+m.getDescription());  ChatFrame chatFrame=ManageChat.getChat(m.getSender()+" "+m.getReciever());  chatFrame.showDescription(m);  }else if(m.getMessageType().equals(MessageType.message\_file\_transfer)){  ServerReceiveFile serverReceiveFile=new ServerReceiveFile();  serverReceiveFile.load();  ChatFrame chatFrame=ManageChat.getChat(m.getReciever()+" "+m.getSender());  chatFrame.showFileMes(m);  }  }  } catch (Exception e) {  e.printStackTrace();  break;  }  }  }  } |

③ ClientConServer 类——负责通信

|  |
| --- |
| package model;  import bean.Message;  import bean.MessageType;  import bean.User;  import frame.FrameTools.DialogFrame;  import tools.ClientConServerThread;  import tools.ManageClientConServerThread;  import java.io.ObjectInputStream;  import java.io.ObjectOutputStream;  import java.net.InetAddress;  import java.net.Socket;  public class ClientConServer {  public Socket s;  private ClientConServerThread ccst;  public boolean sendLoginInfoToServer(Object o){  boolean b=false;  try {  s=new Socket(InetAddress.getLocalHost(),9999);  ObjectOutputStream oos=new ObjectOutputStream(s.getOutputStream());  oos.writeObject(o);  ObjectInputStream ois=new ObjectInputStream(s.getInputStream());  Message ms=(Message)ois.readObject();  if(ms.getMessageType().equals(MessageType.message\_success)){  System.out.println("登录成功");  //创建一个该客户端和服务器端保持通信的线程  ccst=new ClientConServerThread(s);  ccst.start();  ManageClientConServerThread.addClientConServerThread(((User)o).getUserId(),ccst);  b=true;  }else if(ms.getMessageType().equals(MessageType.message\_login\_fail)){  b=false;  }  }catch (Exception e){  e.printStackTrace();  }finally {  }  return b;  }  Socket sb;  public boolean senRegistInfoToServer(Object o){  boolean a=false;  try {  sb=new Socket("127.0.0.1",9998);  ObjectOutputStream oos=new ObjectOutputStream(sb.getOutputStream());  oos.writeObject(o);  ObjectInputStream ois=new ObjectInputStream(sb.getInputStream());  Message m=(Message)ois.readObject();  System.out.println(m.getMessageType());  if(m.getMessageType().equals(MessageType.message\_regist\_permit)){  System.out.println("注册成功");  //创建一个该客户端和服务器端保持通信的线程  new DialogFrame("通知", "恭喜你，注册成功,您的账号是:"+m.getContent()+"请妥善保管", 300, 100);  return true;  }else if(m.getMessageType().equals(MessageType.message\_regist\_deny)){  return false;  }  }catch (Exception e){  e.printStackTrace();  }finally {  }  return a;  }  } |

⑵ 服务器端主要代码段（略去包名）

① Message类——用于命令传递

|  |
| --- |
| package bean;  import java.io.File;  import java.io.Serializable;  import java.util.ArrayList;  public class Message implements Serializable {  private static final long serialVersionUID = 8141984962750547675L;  private String messageType;  private String sender;  private String reciever;  private String content;  private String time;  private String name;  private String description;  private String filePath;  private String fileName;  public void setFileName(String fileName) {  this.fileName = fileName;  }  public String getFileName() {  return fileName;  }  public void setFilePath(String filePath) {  this.filePath = filePath;  }  public String getFilePath() {  return filePath;  }  public void setDescription(String description) {  this.description = description;  }  public String getDescription() {  return description;  }  private String onlineList[];  public String[] getOnlineList() {  return onlineList;  }  public void setOnlineList(String[] onlineList) {  this.onlineList = onlineList;  }  public void setName(String name) {  this.name = name;  }  public String getName() {  return name;  }  private ArrayList<String> friendList;  private ArrayList<String> friendNameList;  public void setFriendNameList(ArrayList<String> friendNameList) {  this.friendNameList = friendNameList;  }  public ArrayList<String> getFriendNameList() {  return friendNameList;  }  private int friendNum;  public void setFriendNum(int friendNum) {  this.friendNum = friendNum;  }  public int getFriendNum() {  return friendNum;  }  public void setFriendList(ArrayList<String> friendList) {  this.friendList = friendList;  }  public ArrayList<String> getFriendList() {  return friendList;  }  public void setSender(String sender) {  this.sender = sender;  }  public void setReciever(String reciever) {  this.reciever = reciever;  }  public void setContent(String content) {  this.content = content;  }  public void setTime(String time) {  this.time = time;  }  public String getSender() {  return sender;  }  public String getReciever() {  return reciever;  }  public String getContent() {  return content;  }  public String getTime() {  return time;  }  public String getMessageType() {  return messageType;  }  public void setMessageType(String messageType) {  this.messageType = messageType;  }  } |

② ServerConClientThread类——服务器端的消息控制中心

|  |
| --- |
| /\*  服务器和某个客户端的通信线程  \*/  package model;  import bean.Message;  import bean.MessageType;  import dao.DAO;  import dao.ReturnAllOnlineFriendInfo;  import dao.SelectQuery;  import java.io.ObjectInputStream;  import java.io.ObjectOutputStream;  import java.net.Socket;  import java.sql.\*;  import java.util.ArrayList;  import java.util.HashMap;  import java.util.Iterator;  public class ServerConClientThread extends Thread {  Socket s;  public Socket getS() {  return s;  }  private String offLineUser;  private Message m;  public ServerConClientThread(Socket s) {  //把服务器与该客户端的连接赋给s  this.s = s;  }  public void notifyFriendRequestReciever(String whoNewIs){  Message m=new Message();  m.setSender(whoNewIs);  Message m2=new Message();  m2.setContent(ManageClientThread.getAllOnlineUserId());  ReturnAllOnlineFriendInfo raofi=new ReturnAllOnlineFriendInfo(m);  m2.setFriendNum(raofi.getFriendNum());  m2.setFriendNameList(raofi.getFriendNameList());  m2.setFriendList(raofi.getFriendlist());  m2.setReciever(m.getSender());  m2.setMessageType(MessageType.message\_ret\_onLineFriend);  try {  ObjectOutputStream oos = new ObjectOutputStream(ManageClientThread.getClientThread(whoNewIs).s.getOutputStream());  oos.writeObject(m2);  }catch (Exception e){  e.printStackTrace();  }  }  //让该线程通知其他用户  public void notifyOtherOnlieUsers(String whoAmI) {  //得到所有的线程  HashMap hm = ManageClientThread.hm;  Iterator it = hm.keySet().iterator();  while (it.hasNext()) {  String res = ManageClientThread.getAllOnlineUserId();  String onlineFriend[] = res.split(" ");  Message m = new Message();  m.setContent(whoAmI);  m.setMessageType(MessageType.message\_newOnline\_friend);  m.setFriendNum(Integer.MAX\_VALUE);  m.setFriendList(null);  m.setOnlineList(onlineFriend);  //取出在线用户id  String onLineUserId = it.next().toString();  System.out.println("新上线人:" + whoAmI);  try {  ObjectOutputStream oos = new ObjectOutputStream(ManageClientThread.getClientThread(onLineUserId).s.getOutputStream());  m.setReciever(onLineUserId);  System.out.println("接收人:" + onLineUserId);  oos.writeObject(m);  } catch (Exception e) {  e.printStackTrace();  }  }  }  public void notifyOtherOnlieUsersOff(String whoAmI) {  //得到所有的线程  HashMap hm = ManageClientThread.hm;  Iterator it = hm.keySet().iterator();  while (it.hasNext()) {  //取出在线用户id  String onLineUserId = it.next().toString();  System.out.println("当前在线用户为:" + onLineUserId);  try {  ObjectOutputStream oos = new ObjectOutputStream(ManageClientThread.getClientThread(onLineUserId).s.getOutputStream());  Message m3 = new Message();  m3.setContent(onLineUserId);  m3.setSender(whoAmI);  m3.setMessageType(MessageType.message\_ret\_logout);  m3.setReciever(onLineUserId);  oos.writeObject(m3);  System.out.println("通知" + whoAmI + "下线");  } catch (Exception e) {  e.printStackTrace();  }  }  }  public void run() {  while (true) {  //该线程可以接受客户端的信息  try {  //读取客户端发送的消息  if (s.getInputStream().available() != 0) {//使用inputStream.available判断是否还有字节流可读，如果没有就不执行下面的语句。  ObjectInputStream ois = new ObjectInputStream(s.getInputStream());  m = (Message) ois.readObject();  System.out.println(m.getSender() + "给服务器发送了一条消息");  //对客户端的消息进行类型分类判断进行处理  if (m.getMessageType().equals(MessageType.message\_comm\_mes)) {  //转发  //取得接收人的通信线程  System.out.println(m.getSender() + "在聊天");  ServerConClientThread sc = ManageClientThread.getClientThread(m.getReciever());  ObjectOutputStream oos = new ObjectOutputStream(sc.s.getOutputStream());  oos.writeObject(m);  } else if (m.getMessageType().equals(MessageType.message\_comm\_mes\_group)) {  //先查找到群中所有人，方便通知  String sql = "select name from " + m.getReciever() + "\_friend ";  Connection conn = null;  conn = DAO.getConn();  ArrayList<String> groupMember = new ArrayList<>();  try {  PreparedStatement ps = conn.prepareStatement(sql);  ResultSet rs = ps.executeQuery();  while (rs.next()) {  groupMember.add(rs.getString("name"));  }  } catch (Exception eg) {  eg.printStackTrace();  }  System.out.println("该群成员是：");  for (String temp : groupMember) {  System.out.print(temp + " ");  }  //通知在线群成员  String res = ManageClientThread.getAllOnlineUserId();  String onlineFriend[] = res.split(" ");  m.setOnlineList(onlineFriend);  for (int i = 0; i < groupMember.size(); i++) {  for (int j = 0; j < onlineFriend.length; j++) {  if (onlineFriend[j].equals(groupMember.get(i))) {  //将消息推送给在线的群成员  System.out.print("当前在线人" + onlineFriend[j]);  ObjectOutputStream oos = new ObjectOutputStream(ManageClientThread.getClientThread(onlineFriend[j]).s.getOutputStream());  oos.writeObject(m);  }  System.out.println();  }  }  System.out.println("发送消息的内容是:" + m.getContent());  } else if (m.getMessageType().equals(MessageType.message\_get\_onLineFriend)) {  System.out.println(m.getSender() + "请求查看好友在线情况");  //返回给客户端，目前在线的好友  String res = ManageClientThread.getAllOnlineUserId();  System.out.println("目前在线人为：" + res);  Message m2 = new Message();  m2.setMessageType(MessageType.message\_ret\_onLineFriend);  m2.setContent(res);  m2.setReciever(m.getSender());  Connection conn = null;  conn = DAO.getConn();  //返回给客户端，客户端的昵称  String sql1 = "select user\_name from user where user\_id=" + m.getSender();  SelectQuery selectQuery = new SelectQuery();  System.out.println("请求发送人:" + selectQuery.querySql(sql1));  m2.setName(selectQuery.querySql(sql1));  //返回给客户端用户的签名  PreparedStatement ps1 = null;  ResultSet rs1 = null;  try {  ps1 = conn.prepareStatement("select user\_trades from user where user\_id=" + m.getSender());  rs1 = ps1.executeQuery();  while (rs1.next()) {  m2.setDescription(rs1.getString("user\_trades"));  }  } catch (Exception ed) {  ed.printStackTrace();  }  //返回给客户端好友的昵称  //返回给客户端目前好友列表中的好友id  ReturnAllOnlineFriendInfo ra=new ReturnAllOnlineFriendInfo(m);  m2.setFriendList(ra.getFriendlist());//好友id列表  m2.setFriendNameList(ra.getFriendNameList());//好友昵称列表  m2.setFriendNum(ra.getFriendNum());  ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());  oos.writeObject(m2);  } else if (m.getMessageType().equals(MessageType.message\_request\_addFriend)) {  System.out.println(m.getSender() + "请求加" + m.getContent() + "好友");  //将这他想加的好友加到双方的好友列表中  //检查数据库中是否有该账号  //首先检索数据库中该账号和好友账号是否建立了好友table,有则直接使用，无则创建  Connection conn = null;  conn = DAO.getConn();  Message m2 = new Message();  PreparedStatement ps = null;  PreparedStatement ps2 = null;  String sql = "insert into " + m.getSender() + "\_friend (name,user\_name) values(?,?)";  String sql2 = "insert into " + m.getContent() + "\_friend (name,user\_name) values (?,?)";  String ids, idr;  SelectQuery selectQuery = new SelectQuery();  ids = selectQuery.querySql("select user\_name from user where user\_id=" + m.getSender());  System.out.println("发送人id" + ids);  idr = selectQuery.querySql("select user\_name from user where user\_id=" + m.getContent());  System.out.println("接收人id:" + idr);  try {  ps = conn.prepareStatement(sql);  ps2 = conn.prepareStatement(sql2);  /\* ps3 = conn.prepareStatement(sql3);  ps4 = conn.prepareStatement(sql4);\*/  ps.setString(1, m.getContent());  ps.setString(2, idr);  ps.executeUpdate();  ps2.setString(1, m.getSender());  ps2.setString(2, ids);  ps2.executeUpdate();  /\* ps3.setString(1, ids);  ps3.executeUpdate();  ps4.setString(1,idr);  ps4.executeUpdate();\*/  } catch (Exception ep) {  ep.printStackTrace();  } finally {  try {  if (ps != null) {  ps.close();  }  if (ps2 != null) {  ps2.close();  }  } catch (Exception e1) {  e1.printStackTrace();  }  }  m2.setContent(ManageClientThread.getAllOnlineUserId());  ReturnAllOnlineFriendInfo raofi=new ReturnAllOnlineFriendInfo(m);  m2.setFriendNum(raofi.getFriendNum());  m2.setFriendNameList(raofi.getFriendNameList());  m2.setFriendList(raofi.getFriendlist());  m2.setReciever(m.getSender());  m2.setMessageType(MessageType.message\_ret\_onLineFriend);  ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());  oos.writeObject(m2);  notifyFriendRequestReciever(m.getContent());  } else if (m.getMessageType().equals(MessageType.message\_request\_logout)) {  System.out.println(m.getSender() + "请求下线");  ManageClientThread.hm.remove(m.getSender());  System.out.println("删除" + m.getSender() + "的hashMap值,剩余在线人" + ManageClientThread.getAllOnlineUserId());  System.out.println(m.getSender() + "已下线");  notifyOtherOnlieUsersOff(m.getSender());  System.out.println("关闭服务器与" + m.getSender() + "的连接");  } else if (m.getMessageType().equals(MessageType.message\_get\_logout)) {  System.out.println(m.getSender() + "请求查看目前好友状况");  String res = ManageClientThread.getAllOnlineUserId();  System.out.println("目前在线人为：" + res);  Message m2 = new Message();  m2.setMessageType(MessageType.message\_ret\_logout);  m2.setContent(res);  m2.setReciever(m.getSender());  ObjectOutputStream oos = new ObjectOutputStream(s.getOutputStream());  oos.writeObject(m2);  } else if (m.getMessageType().equals(MessageType.message\_frameShake)) {  System.out.println(m.getSender() + "给" + m.getReciever() + "发送抖动窗口");  try {  Message m2 = new Message();  m2.setMessageType(MessageType.message\_frameShake);  m2.setReciever(m.getReciever());  m2.setSender(m.getSender());  ObjectOutputStream oos = new ObjectOutputStream(ManageClientThread.getClientThread(m.getReciever()).s.getOutputStream());  oos.writeObject(m2);  } catch (Exception es) {  es.printStackTrace();  }  } else if (m.getMessageType().equals(MessageType.message\_alter\_name)) {  System.out.println(m.getSender() + "想给网名改成" + m.getContent());  //对数据库中user表对应的数据进行修改  Connection conn = null;  conn = DAO.getConn();  try {  PreparedStatement ps = conn.prepareStatement("update user set user\_name=? where user\_id=" + m.getSender());  ps.setString(1, m.getContent());  ps.executeUpdate();  System.out.println("修改成功");  } catch (Exception e) {  e.printStackTrace();  }  } else if (m.getMessageType().equals(MessageType.message\_alter\_des)) {  System.out.println(m.getSender() + "想给签名改成" + m.getContent());  //对数据库中user表对应的数据进行修改  Connection conn = null;  conn = DAO.getConn();  try {  PreparedStatement ps = conn.prepareStatement("update user set user\_trades=? where user\_id=" + m.getSender());  ps.setString(1, m.getContent());  ps.executeUpdate();  System.out.println("修改成功");  } catch (Exception e) {  e.printStackTrace();  }  } else if (m.getMessageType().equals(MessageType.message\_request\_friend\_info)) {  System.out.println(m.getSender() + "想获取好友" + m.getReciever() + "的个人信息");  String des = "";  Connection conn = null;  conn = DAO.getConn();  try {  PreparedStatement ps = conn.prepareStatement("select user\_trades from user where user\_id=" + m.getReciever());  ResultSet rs = ps.executeQuery();  while (rs.next()) {  des = rs.getString("user\_trades");  }  } catch (Exception e) {  }  m.setDescription(des);  m.setMessageType(MessageType.message\_request\_friend\_info);  ObjectOutputStream oos = new ObjectOutputStream(ManageClientThread.getClientThread(m.getSender()).s.getOutputStream());  oos.writeObject(m);  }else if(m.getMessageType().equals(MessageType.message\_file\_transfer)){  try {  ObjectOutputStream oos=new ObjectOutputStream(ManageClientThread.getClientThread(m.getReciever()).getS().getOutputStream());  m.setMessageType(MessageType.message\_file\_transfer);  oos.writeObject(m);  ServerReceiveFile serverReceiveFile=new ServerReceiveFile();  serverReceiveFile.load();  ClientTranserFile clientTranserFile=new ClientTranserFile();  clientTranserFile.sendFileToServer("C:\\Users\\Administrator.DESKTOP-NECSBS8\\Desktop\\newF\\"+m.getFileName());  }catch (Exception et){  et.printStackTrace();  }  }  }  } catch (Exception e) {  //ManageClientThread.hm.remove(m.getSender());  //System.out.println("删除了key为" + m.getSender() + "的hashMap值,剩余在线人" + ManageClientThread.getAllOnlineUserId());  e.printStackTrace();  break;  }  }  }  } |

③ DAO类——连接数据库

|  |
| --- |
| package dao;  import javax.swing.\*;  import java.sql.Connection;  import java.sql.DriverManager;  public class DAO {  private static DAO dao = new DAO(); // 声明DAO类的静态实例  /\*\*  \* 利用静态模块加载数据库驱动  \*/  static {  try {  Class.forName("com.mysql.cj.jdbc.Driver"); // 加载数据库驱动  } catch (ClassNotFoundException e) {  JOptionPane.showMessageDialog(null, "数据库驱动加载失败，请将驱动包配置到构建路径中。\n"  + e.getMessage());  e.printStackTrace();  }  }  /\*\*  \* 获得数据库连接的方法  \*  \* @return Connection  \*/  public static Connection getConn() {  try {  Connection conn = null; // 定义数据库连接  String url = "jdbc:mysql://127.0.0.1:3306/vchatroom?useSSL=false&serverTimezone=Hongkong&characterEncoding=utf-8&autoReconnect=true"; // 数据库db\_Express的URL  String username = "root"; // 数据库的用户名  String password = "1234"; // 数据库密码  conn = DriverManager.getConnection(url, username, password); // 建立连接  return conn; // 返回连接  } catch (Exception e) {  JOptionPane.showMessageDialog(  null,  "数据库连接失败。\n请检查是否安装了SP4补丁，\n以及数据库用户名和密码是否正确。"  + e.getMessage());  e.printStackTrace();  return null;  }  }  public static void main(String[] args) {  System.out.println(getConn());  }  } |

④ UserDao类——对数据库进行增删查改

|  |
| --- |
| package dao;  import bean.Message;  import bean.User;  import model.CheckTableExists;  import model.MD5;  import javax.swing.\*;  import java.io.DataOutputStream;  import java.sql.Connection;  import java.sql.PreparedStatement;  import java.sql.ResultSet;  import java.util.Random;  public class UserDao {  public static boolean userRegist(User user,String name) {  Connection conn = null;  try {  String username = user.getUserId();  String userid=name;  System.out.println("自动生成随机账号"+userid);  String password = user.getUserPwd();  conn = DAO.getConn();  PreparedStatement ps = conn.prepareStatement("insert into user (user\_id,user\_name,user\_password) values (?,?,?)");  ps.setString(1,userid);  ps.setString(2, username);  ps.setString(3, password);  PreparedStatement ps1=conn.prepareStatement("select count(\*) from user where user\_id=?");  ps1.setString(1,userid);  ResultSet rs=ps1.executeQuery();  if(rs.next()&&rs.getRow()>0){  int flag = ps.executeUpdate();  if(flag>0){  System.out.println("注册成功");  //顺便给该账号添加一个好友列表  Message mu=new Message();  mu.setSender(userid);  System.out.println("userid+"+userid);  mu.setName(username);  System.out.println("username+"+username);  new CheckTableExists(mu,conn);  PreparedStatement ps2=conn.prepareStatement("insert into "+userid+"\_friend (name,user\_name) value (?,?)");  ps2.setString(1,userid);  ps2.setString(2,username);  ps2.executeUpdate();  System.out.println("好友列表数据库初始化完毕");  return true;  }else{  System.out.println("注册失败");  return false;  }  }else{  System.out.println("用户名已存在");  return false;  }  } catch (Exception e) {  e.printStackTrace();  } finally {  if (conn != null) {  try {  conn.close();  } catch (Exception e) {  e.printStackTrace();  }  }  }  return false;  }  public static boolean userLogin(User user) {  Connection conn = null;  try {  System.out.println("正在验证账号密码合法性");  String userid = user.getUserId();  String password = user.getUserPwd();  String userName="";  conn = DAO.getConn();  PreparedStatement ps = conn.prepareStatement("select user\_password from user where user\_id=?");  PreparedStatement ps1=conn.prepareStatement("select user\_name from user where user\_id=?");  ps.setNString(1, userid);  ResultSet rs = ps.executeQuery();  ps1.setString(1,userid);  ResultSet rs1=ps1.executeQuery();  System.out.println(rs1);  while(rs1.next()){  for(int i=1;i<=rs1.getMetaData().getColumnCount();i++){  userName=userName+rs1.getString(i);  }  }  System.out.println("用户名是"+userName);  if (rs.next() && rs.getRow() > 0) {  String pwd = rs.getString(1);  System.out.println("正确的密码是" + pwd);  System.out.println("输入的密码是" + password);  //MD5解密  if (pwd.equals(MD5.converMD5(password))) {  System.out.println("账号密码正确");  return true;  } else {  System.out.println("密码错误");  return false;  }  } else {  System.out.println("用户不存在");  return false;  }  } catch (Exception e) {  System.out.println("数据库异常");  e.printStackTrace();  JOptionPane.showMessageDialog(null, "数据库异常");  return false;  } finally {  if (conn != null) {  try {  conn.close();  } catch (Exception e) {  e.printStackTrace();  }  }  }  }  } |

# 第四章 测试和演示

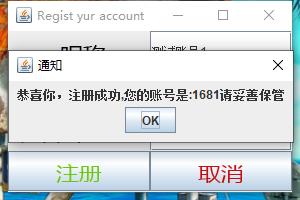
## 1 测试内容及预测结果

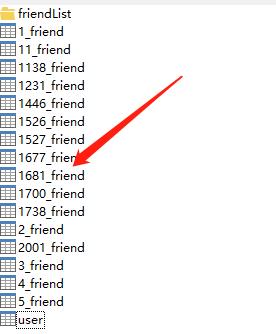
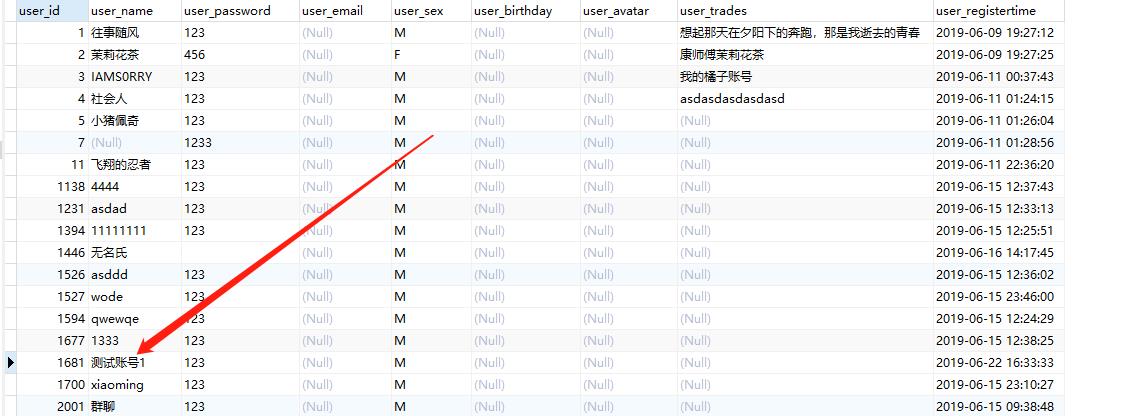
|  |  |  |  |
| --- | --- | --- | --- |
| 测试号 | 测试名称 | 测试操作 | 预期结果 |
| 1 | 注册用户 | 注册用户名 测试账号1  注册密码 123 | 数据库user中出现该用户信息 |
| 2 | 登录 | 登录该账号 | 登录成功显示好友列表 |
| 3 | 修改个性签名和网名 | 将个性签名修改为：个性签名修改测试，网名改为 测试账号1.0 | 修改成功 |
| 4 | 添加好友 | 注册账号1.0添加往事随风(1) | 双方好友列表都得到了新好友 |
| 5 | 用户聊天 | 测试账号1.0与 往事随风聊天 | 双方收到消息 |
| 6 | 群聊 | 往事随风  社会人  测试账号1.0 | 没互相加好友显示陌生人 |
| 7 | 窗口抖动 | 往事随风发送抖动窗口给测试账号1.0 | 均接收到抖动窗口 |
| 8 | 发文件 | 往事随风发给测试账号1.0 | 发送接受成功 |

## 2 测试结果与截图

1. 测试点一

测试结果与预期结果一致，注册成功！截图如下：

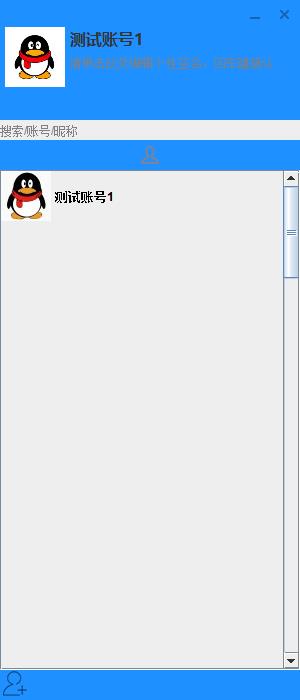




1. 测试点二

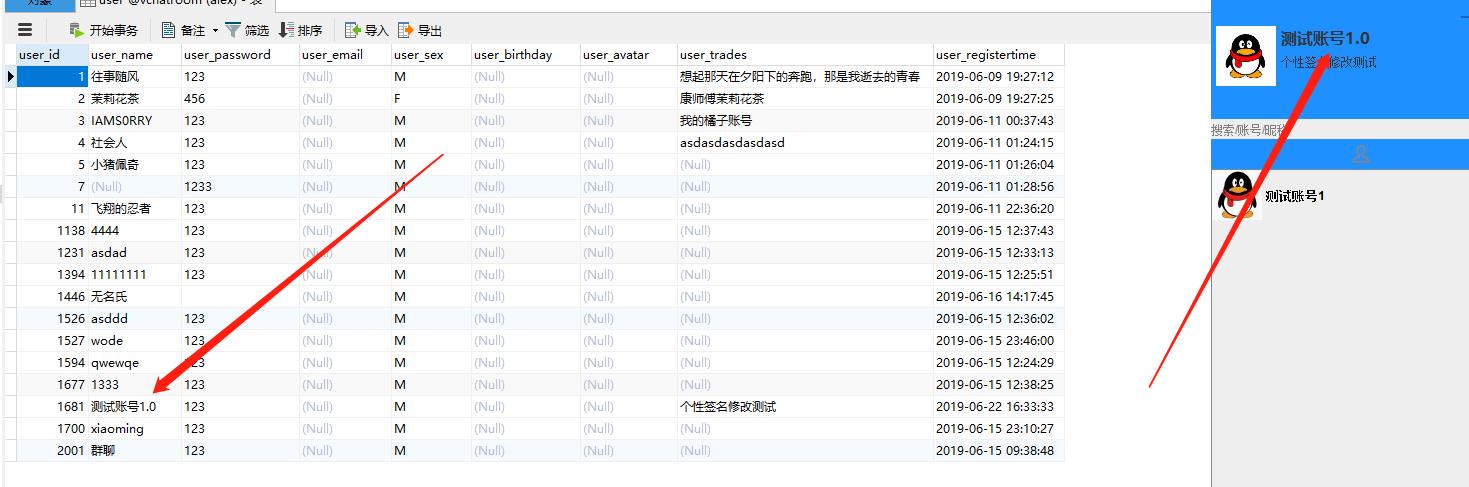
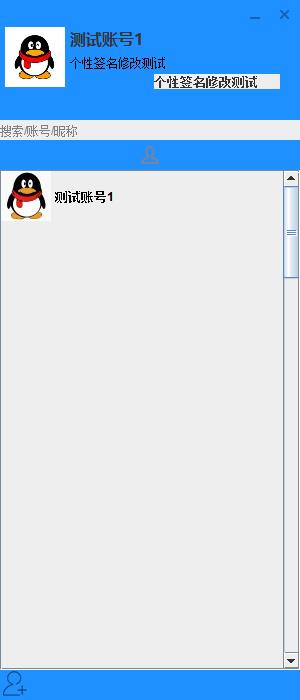
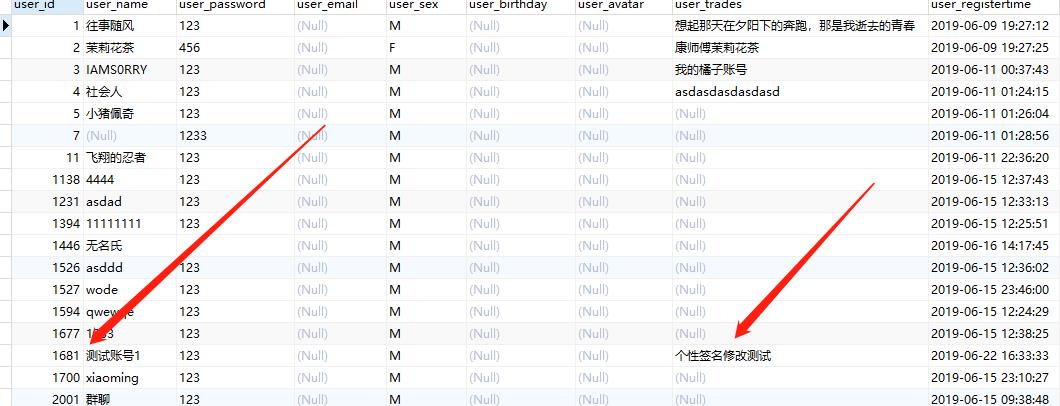
测试结果与预期结果一致，登录成功！截图如下：

输入上一步获得的账号1681和密码123点登录



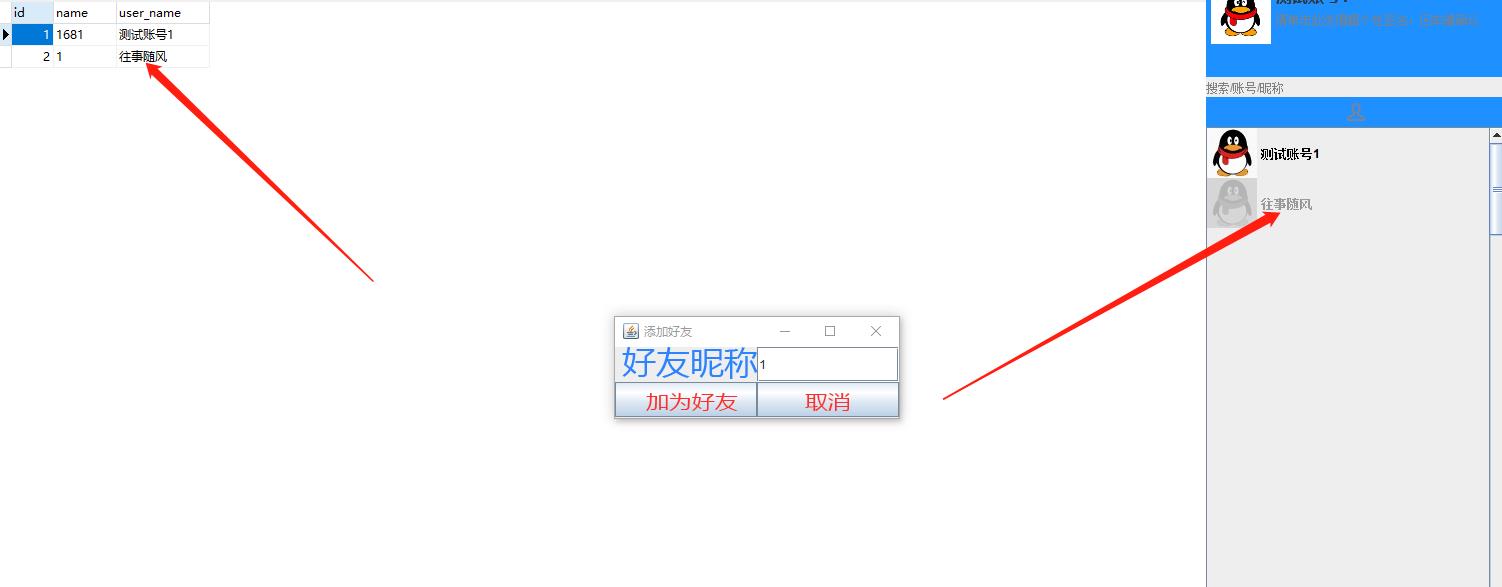
1. 测试点三

测试结果与预期结果一致，修改密码成功！截图如下：



1. 测试点四

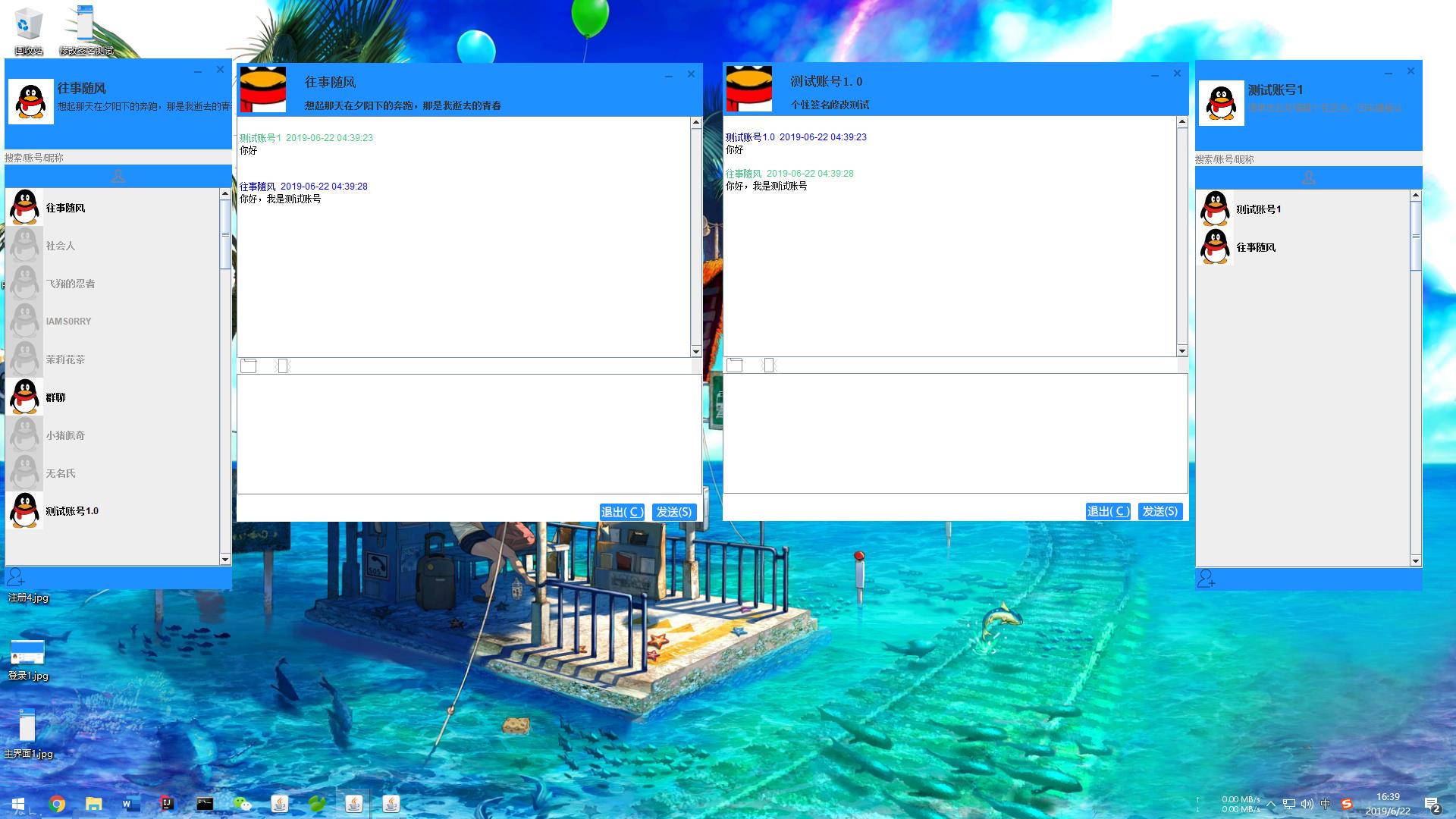
测试结果与预期结果一致，加好友成功！截图如下：





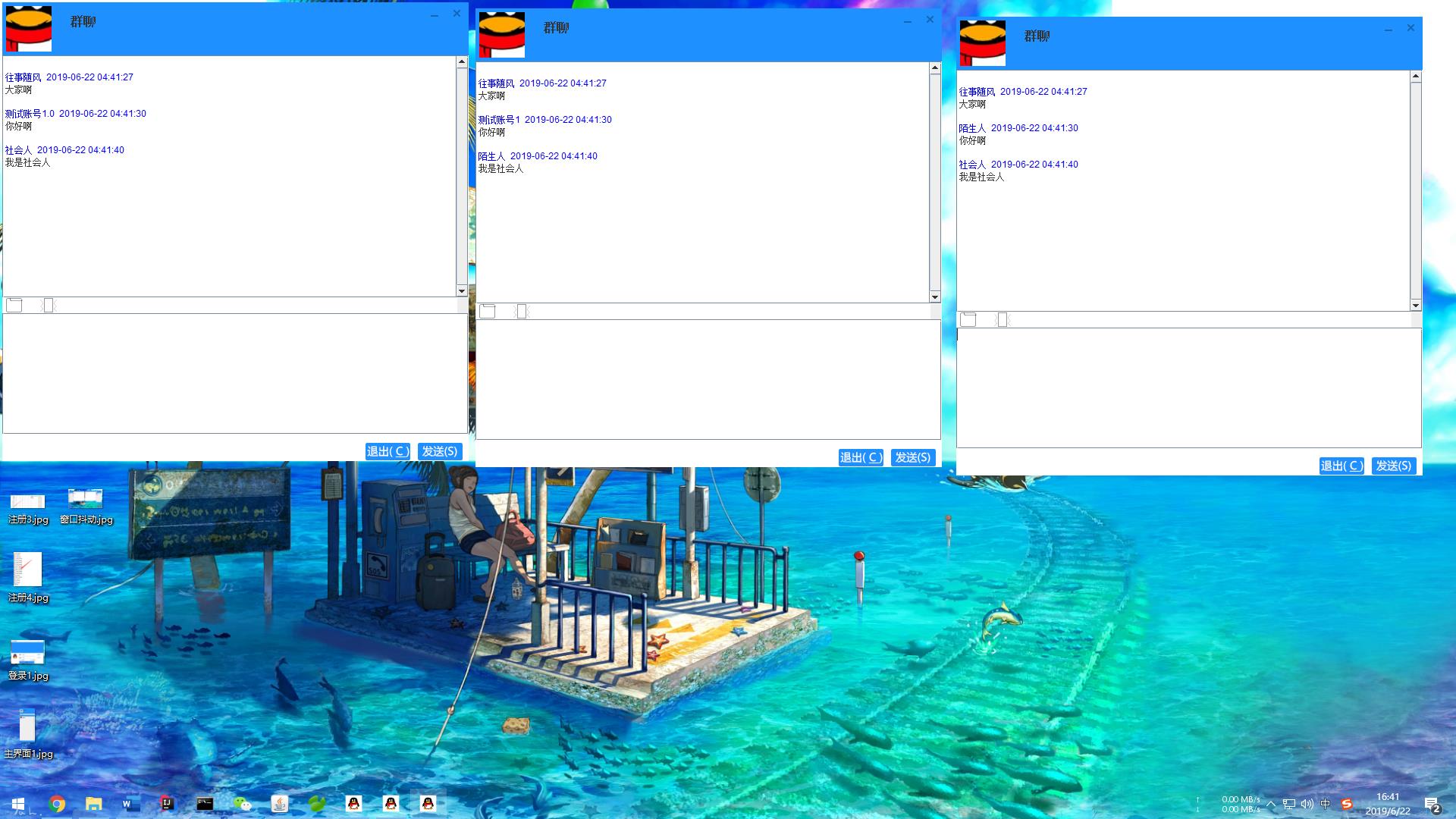
1. 测试点五

测试结果与预期结果一致，私聊成功！截图如下：



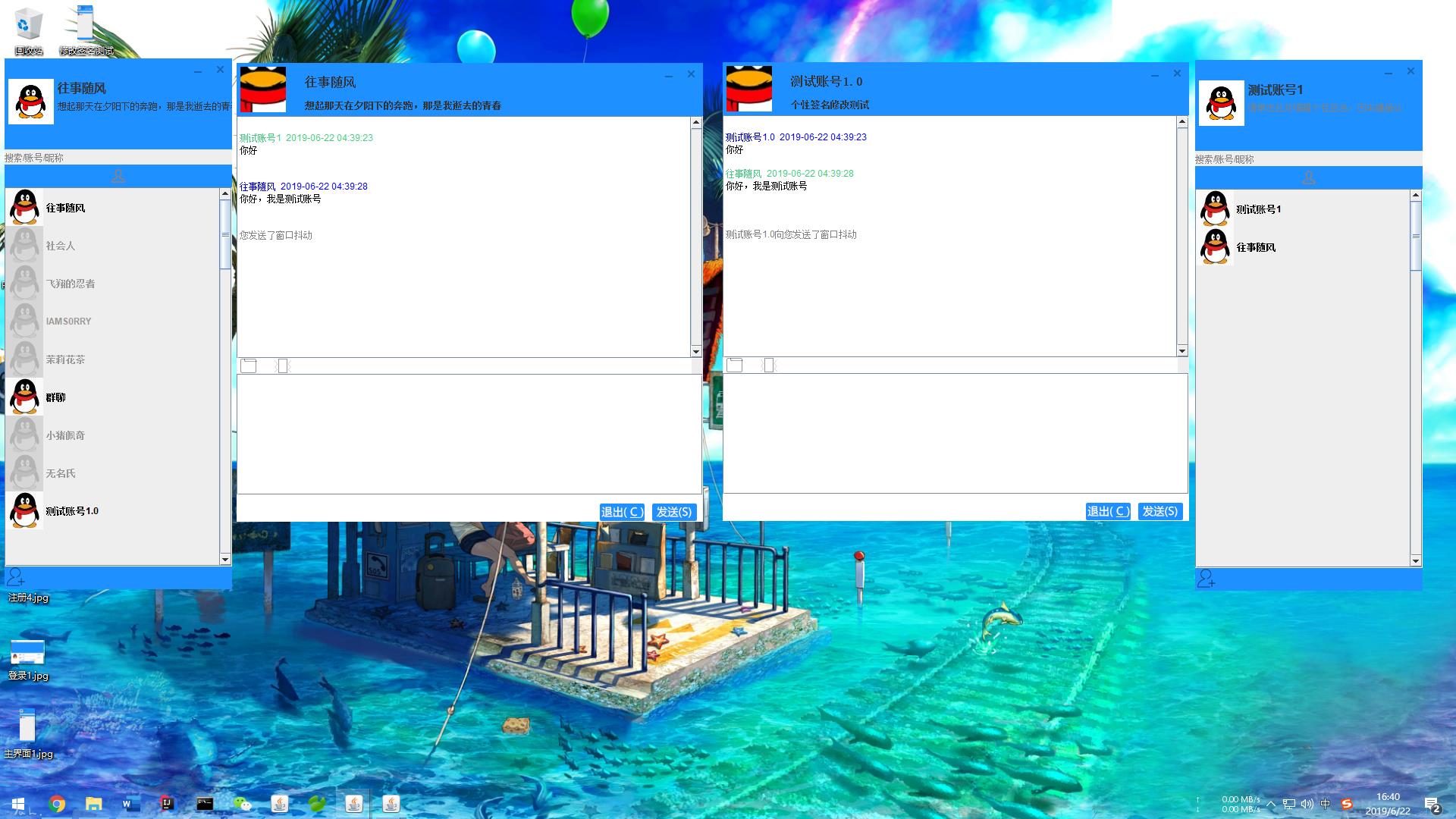
1. 测试点六

测试结果与预期结果一致，群聊成功！截图如下：



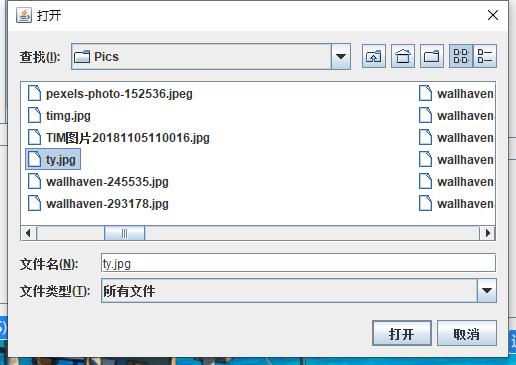
1. 测试点七

测试结果与预期结果一致，抖动成功！截图如下：



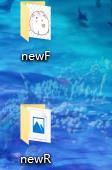
1. 测试点八

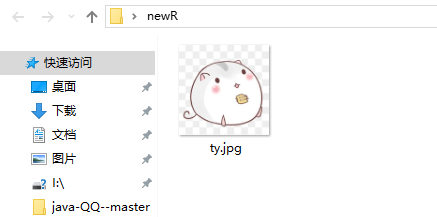
测试结果与预期结果一致，文件传输成功！截图如下：





newF是服务器存放文件的目录 newR是接受文件的默认目录





# 第五章 总结

这次设计经过不懈的努力，程序基本满足聊天的需要。完成后的程序实现了用户注册、用户登录、忘记密码、写消息、发消息、回复、群聊、添加好友、修改密码等功能。给用户较好的体验。完成了登录页面、注册页面、好友列表、忘记密码、修改密码等页面框架，使得程序较美观。由于一个人完成这个项目，时间比较紧张，一些功能如文件传输还没有得到很好的优化。

大概用了一个月的时间，每天在写bug改bug中度过，中间学到了很多东西也发现之前的学习有很多不足的地方，知道了基础的重要性。特别是通过这个项目，了解到多线程和网络传输等原理及应用。第一次做项目，很多东西都不知道，比如刚开始的需求分析没有做到位，以至于到后面想添加新的功能的时候发现很难添加，或者是添加上新功能会导致老功能无法使用。

我的代码目前很臃肿，很多地方写的比较乱。对于经常使用的一段代码可以放到一个函数里或者类中，这样再使用就可以直接调用了，不仅节省时间，还让代码的可读性变强，方便以后的调试以及其他开发者修改。写一个功能之前应该先弄懂原理，画出大致的数据关系图，模块关系图，有了大致的思路和框架在进行编写程序会达到事半功倍的效果。

在调试的过程中，希望多出一些错误，这样可以将这些错误通过看书或者网上查阅资料解决，最怕的是没有报错，但是程序运行有问题。目前常用的查资料引擎是Google,会在stackoverflow、github、csdn等平台上查阅资料以及下载源码。

世上无难事只怕有心人，不管是多大的困难，只要硬着头皮去做，即使会消耗一部分时间，但是坚持到最后肯定做得出来，能力就是这样一点点培养的。厉害的人不是天生聪明，而是遇到问题多了，解决的多了，自然就熟练了。

最后，这个项目会在以后继续更新，作为自己练手项目。

已经放到github供大家下载<https://github.com/cutety/VirtualChatRoom>