廣東工業大學

Java 考核

课程名称 Java 程序设计

题目名称 Java 版即时聊天程序

学生学院 计算机

专业班级 计算机 17(4)班

学　　号 3117004568

学生姓名 黄钰竣

指导教师 赵锐

2018　年 12 月　10 日

难度系数	
独立完成工作量占总工作量的比例	95%
程序功能完成情况	
报告内容与格式	
总评成绩	

目录

一、课程设计题目

1、课程题目名称：实现一个 Java 版即时聊天程序

2、设计要求（功能要求）：

1)用户登录及登录验证：用户能够使用固定帐号（帐号程序内置即可，无需完成额外的注册功能）登录系统，系统能对预定的帐号、密码进行验证;

2)聊天功能：能够在两台以上的机器上登录运行程序，能使用不同帐号完成在线的即时消息发送（聊天）;

3)文件传输;

4)好友管理：能够显示好友列表，并能够添加、修改、删除好友;

5)在线、离线状态显示：能够显示好友的在线状态或离线状态。

6)聊天记录管理：能够以文件或数据库形式将聊天记录进行存储，并能打开、显示、删除所存储的聊天记录。

补充：其它可选的还包括 Java 多人在线网络聊天室、B/S 结构的聊天系统等。

二、题目分析与设计

1、开发环境及配置:

开发环境（软件）：eclipse 的 photon 版本 64 位

相关配置：eclipse 所属公司提供的 windowbuilder 插件（用于界面设计）、

用于连接 MySQL 数据库的驱动程序。

2、设计要求（需求）分析

聊天系统的基础部分涉及程序间、电脑间的通讯，所以其实现必定涉及 Java 的网络编程部分，较为理想的知识模块选择是基于 TCP/IP 协议，使用 Socket 和 ServerSocket 实现客户端与客户端之间、客户端与服务器之间的通信。其中，聊天与文件传输功能的实现直接依赖于 socket 的输入输出流的读写，实现跨平台设备上的信息传递。

登陆验证的功能需求则需要有基本的用户信息持久化存储，需要提供一个官方公用的数据信息存储库，可采用开源轻量级数据库 MySQL 实现用户 id、昵称、登陆密码、聊天记录、好友信息等数据的存储功能。

最后，从程序的可拓展方面考虑，程序实现的通信不仅限于人为规定的少数客户端通信（2~5 个指定用户），需要引入多线程分批单独处理不同客户端发来的功能请求。由于程序实现的适用和涉及范围仅定位为局域网间，用户数量并不算太大，信息处理所占资源可以接受，所以可采用类似集中式网络星型结构的数据传输方式。由服务器端集中收发处理来自客户端的信息。（如图 1-1 所示结构）

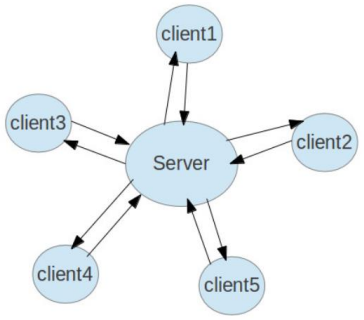


图 2-1

3、图形用户界面设计

因为服务器端主要负责自动处理客户端信息的逻辑处理（转发、数据库信息操作等），所以本程序重点设计的界面为客户端的登陆、收发信息以及聊天记录查看的界面。

1） 登陆界面

登陆界面主要涉及的元素有信息输入框（用户名和密码输入）和登陆确认按钮。客户端在发出登陆请求时往往先需要与服务器端建立通信链接，为了防止网络连接错误或者其他原因，需设计一个连接信息提示弹窗，用于提醒用户检查连接情况。最后，成功与服务器建立连接后需要经服务器核对用户登陆信息无误后，进行聊天界面跳转，如果出现用户信息不匹配（即用户名或登陆密码输入错误），则需有弹窗提示重新输入。

2） 聊天界面（主界面）

聊天界面涉及用户的主要基本功能选择——收发文件或聊天语句、聊天记录查询、好友状态信息显示、好友操作（增删好友），因此需合理使用界面大小，合理考虑各个功能组件或模块的设计。本程序参考现在流行通用的聊天软件——QQ / Tim 的界面设计方式，基于其优秀简洁的设计模板，根据程序需实现的基本功能做了适当修改。

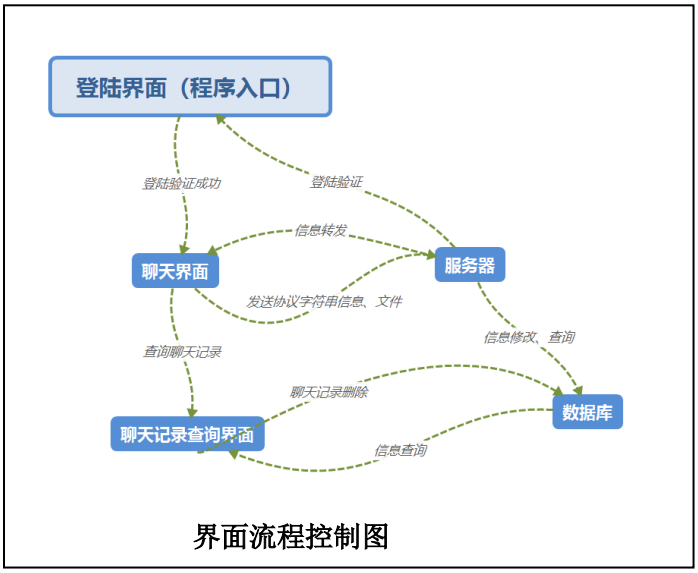
a、好友面板。于屏幕左侧放置好友列表，用于显示所有好友的信息以及其在线情况。当用户点击好友列表时，可选中用于确定私聊好友的对象。

b、收发信息操作。右侧放置文本编辑框和信息显示框，分别用于发送聊天信息和显示聊天内容（聊天内容用不同颜色字体标识不同发送方）。

c、用户个人信息显示以及正在聊天好友信息显示。窗口顶部标题显示有用户登陆的用户名，即时在同一台电脑上登陆不同账户也便于区分不同用户。聊天信息显示框上方设置了一个用于提示当前对话对象用户名的文本框，能即时根据好友面板选中的好友更新信息，提示

用户当前聊天的对象。

d、其他功能显示面板。在信息显示框和文本编辑框间加入功能选择栏，包含文件选择发送按钮和聊天记录查询管理按钮。



界面流程控制图

4、数据（存储）结构

本程序的数据（存储）结构，可根据是否可持续存储分为内存上的数据结构和数据库的数据存储结构。

1） 内存存储结构

由于本聊天程序是通过 socket 的形式实现通信，且在通信结构上采用了集中式的通信方式，所以对于服务器端上通过用户验证的 socket（也就是用户已经登陆成功的 socket）就需要特地采用非普通数据类型存储。结合程序聊天对象信息的实际存储内容和基本行为，设计的数据结构需要有以下特点

a、存储对象不能重复或相同（因同一用户不能同时登陆）

b、 能够快速查找并通过多种方式获取（便于服务器转发信息）

基于以上的要求，发现数据结构兼具哈希表（键—值对，快速查询）和集合（具有独异性，无重复元素）的特点，于是我通过重写实现继承自 HashMap 的 ClientsMap 类，实现了使用用户名作为查询关键字，连接对应用户的 socket 作为值的值不重复容器。

2） 可持续化存储结构（数据库表结构）

这一部分主要涉及数据库的信息存储方式。本程序采用了“用户信息为主，好友信息为次”的建表方式，即使用一张表存储所有用户信息（包括 id、用户名、密码），每个用户名直接映射为另一张存储该用户所有好友信息及其之间的聊天记录的方式。

（users）用户信息记录表

userName	passWord
Huang	123
Li	123
Liu	123

（*userName*）联系人表

friendName	history
Li	nullHuang 2018/11/11 4:28:12你好Li 2018/11/
Liu	

5、实现过程设计

这一部分将分为具体实现类和总体设计思路两部分来介绍。

具体实现类将一一列出项目中涉及的所有类名，选择其中最关键的方法进行设计分析。

总体实现思路将介绍程序的主要架构，结合类于类之间的关系，讲解程序执行过程中的不同操作的预期运行结果。

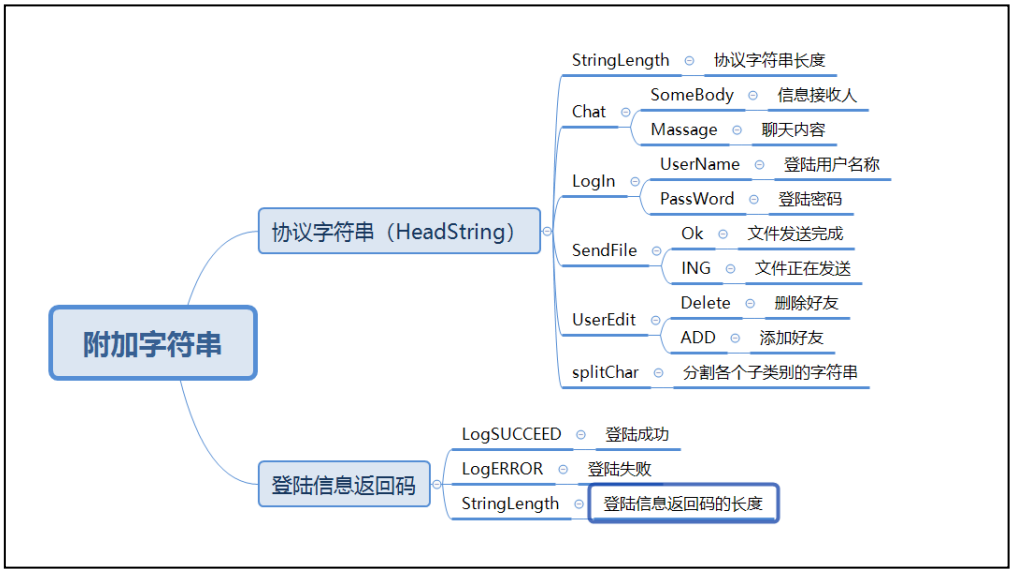
1） 具体实现类

A、客户端与服务器端共用类或接口：

ReturnCode——该接口中定义了用于服务器接收客户端登陆信息，并验证后的返回码。其包含参数有 StringLength（返回字符串长度）、LogERROR（登陆信息与数据库官方信息不匹配）、LogSUCCEED（登陆信息匹配，登陆成功）。

HeadString——该接口提供了定义的协议字符串，包括登陆（LogIn）、用户信息编辑（UserEdit）、聊天信息（Chat）、类别分割字符（splitChar）以及文件发送（SendFile）五大类的开头字符串，分别对应客户端的不同操作，便于服务器辨识并给予相应的逻辑处理。

具体的有关协议字符串的具体使用介绍可参考图 5-1。（拼接方法为有用信息首位添加）



B、客户端类介绍：

Login——登陆界面，程序入口。该类继承了 JFrame，主要实现了绘制登陆界面的窗体，获取用户名输入框和密码输入框的字符串，点击登陆按钮，建立 socket 连接服务器后，将登陆信息（也就是用户名和密码）与协议字符串拼接并发送到服务器端。等待，接收服务器的返回码（ReturnCode），由此判断是否连接成功。如果连接成功，则再次读取服务器返回的用户自己的好友信息，创建 Controler 类实例，创建聊天主界面窗口，并传入参数 Controler。

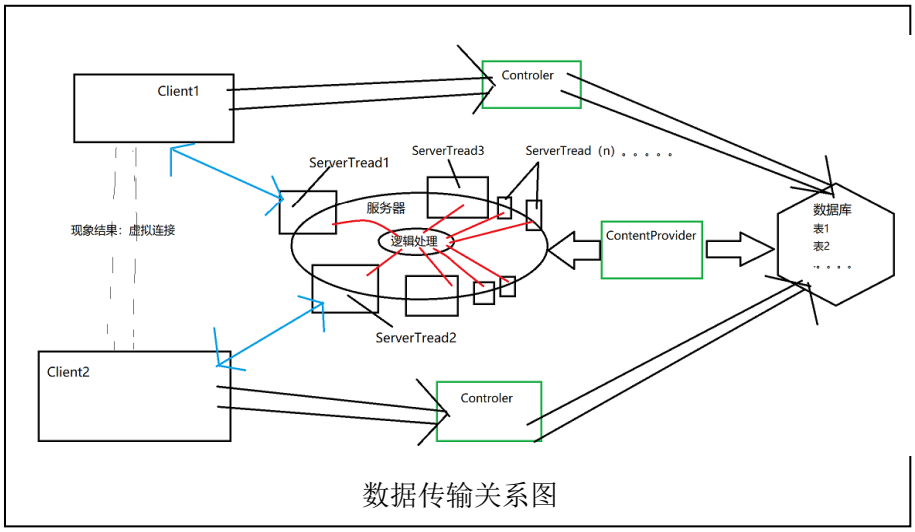
MainFrame——主界面。由登陆界面的 Controler 初始化窗口信息，绘制好友列表、窗体用户名、文本显示框等 UI 组件。（好友列表和窗体用户名可根据不同不同用户而绘制）设置了不同组件的按键响应事件。**点击好友列表中的一栏**，则改变当前对话用户的提示信息，显示与当前用户的对话信息（点击不同用户能显示与不同对象的聊天对话信息）。**点击“发送”按钮**，则将文本编辑栏的文字读入，并根据当前的对话对象，与协议字符串拼接成信息字符串发送给服务器。**点击“发送文件”按钮**，弹出文件选择器弹框，选择完毕后，获取文件绝对路径，创建 SendFileThread 线程并传入路径参数，开始传输文件。**点击“聊天记录”**后，将根据选中的当前对话好友，访问数据库，从中获取聊天记录信息并传入 HistoryWindow 对象，显示在历史记录弹窗中。**设置窗体完毕响应**，当窗口关闭，即程序退出时自动保存用户与好友之间的聊天记录并更新数据库中的聊天记录（history）一栏的信息。

SendFileThread——实现了 Runnable 接口，用于开辟发送文件的线程。该类的主要工作流程为：接收到要发送文件的地址后，创建字节流，先向服务器发送一条含接收对象的和发送文件模式的字符串信息，然后将文件数据写入字节流，文件传输完毕。（可能还要加入发送状态信息，用于判断文件是否已发送完毕）

HistoryWindow——继承自 JFrame，绘画聊天历史纪录显示窗体，以及设置删除历史纪录按钮响应。打开窗体后，通过 Controler 的方法会访问数据库，从中获取与当前对话对象的聊天记录。点击“删除聊天记录”，会调用 Controler 进行数据库的历史记录清除操作。

Controler——连接各个界面的桥梁，同时包含访问数据库的相关少数方法。可从数据库中获取聊天历史记录，保存聊天记录。

ClientThread——实现了 Runnable 接口，负责接收服务器信息，并将相关信息显示在聊天界面上。该类主要提供了对服务器两种不同信息类别的响应方法。接受文件，获取接受文件提示信息后，将文件保存在指定的默认路径中，并将文件接受信息显示在聊天内容显示面板上。接受聊天信息，根据聊天信息，以不同颜色为标题（发送用户名+发送时间），显示在聊天信息面板上。



数据传输关系图

C、服务器端类介绍

MyServer——服务器程序启动入口。创建 ServerSocket 对象后，主线程用于监听来自客户端的连接（死循环调用 accept 方法，造成线程阻塞），接收到连接后，直接创建 ServerThread 线程，并传入接收到的与新连接到的客户端的 Socket,开启新线程处理该客户的操作。

ServerThread——实现了 Runnable 接口，包含线程的运行方法。该类通过循环，不断读入来自客户端发送的字符串，针对字符串的协议字符串类型有四中不同处理方法。**若协议字符串为 LogIn（用户发出登陆请求）**，则从字符串中解析出用户名和密码，调用 ContentProvider 实例的方法访问数据库，从中获取该用户的个人信息并进行比对，如果比对成功则向该客户端发送登陆成功的 ReturnCode 以及用户的好友信息，不匹配则发送登陆失败信息。**接受信息的协议字符串为 Chat（聊天功能）**，处理方法将拆解字符串，从中解析出有用的数据信息（信息接收方、聊天信息本体），根据信息接受方，从 MyServer 实例的 ClientsMap 容器中，获取已经确认登陆成功（即在线上）的好友 Socket，转发接受到的聊天信息主体，实现聊天的实时通信。**若协议字符串为发送文件**，则首先获取字符串中的接收用户，获取接受用户 Socket（要求已登陆），打开分别对应发送用户的字节输入流和接受用户的输出流，向接受用户发送接受文件信息后，开始转发文件。**若协议字符串为好友编辑**，则调用 ContentProvider 的相关方法进行数据库操作。

ClientsMap——自己实现的继承自 HashMap，兼具集合值得互异性的容器，用于存储已经与服务器建立连接且验证成功的 Socket 实例。包含新增的通过值获取键值的方法。

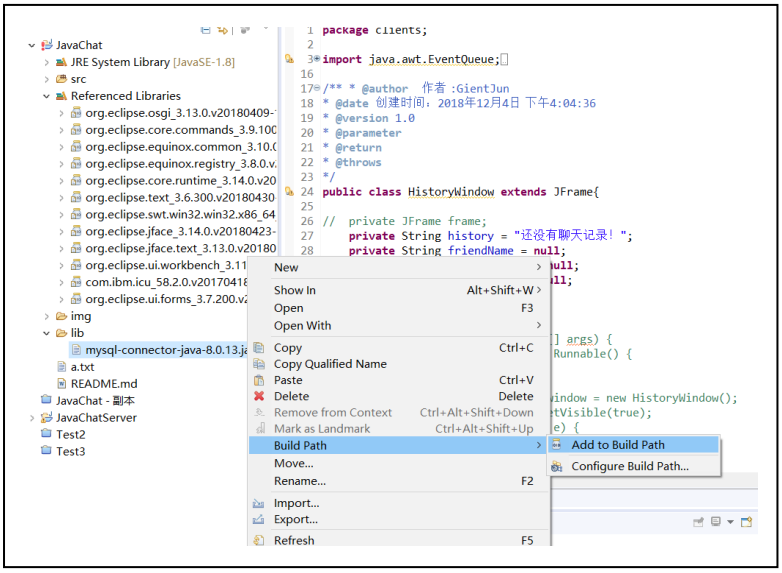
ContentProvider——用于实现与数据库沟通的工具类。构造方法即建立于数据库的连接访问，包括访问数据库、操作数据库的各种方法。

三、测试分析

1、拓展包及相关插件部署

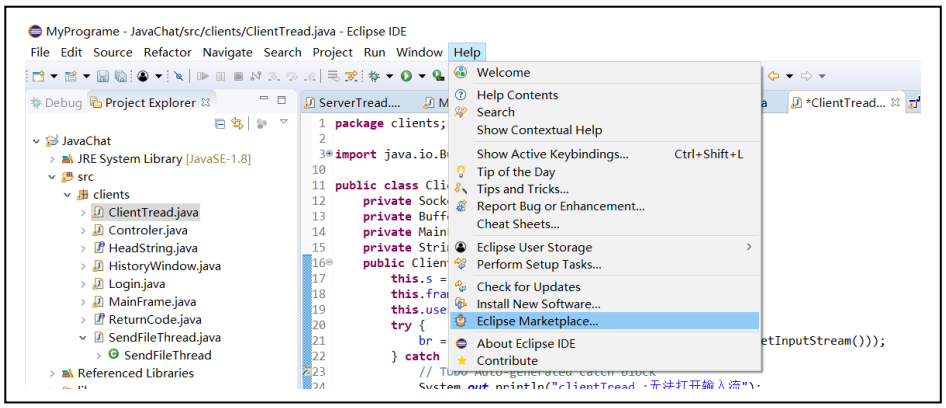
本程序的开发基本环境为 jdk1.8，以 eclipse 为开发平台。在拓展包方面使用了连接 MySQL 数据库的 mysql-connector-java-8.0.13.jar 包。其部署过程如下图所示：

在 eclipse 上的文件目录上新建一个 lib 文件夹，将 mysql-connector-java-8.0.13.jar 包复制进去，刷新以下 eclipse 的文件目录显示（右键-）refresh），选中 jar 包右键选择进行导入。

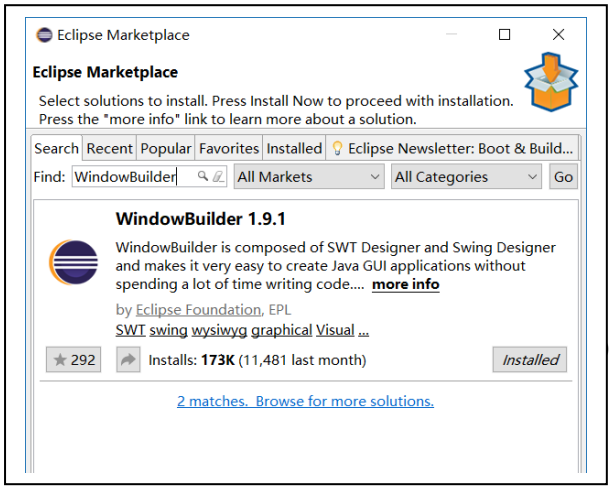


另外在编辑图形交互界面上，使用了从 eclipse 官网下载的 windowBuilder 插件，其部署过程如下：

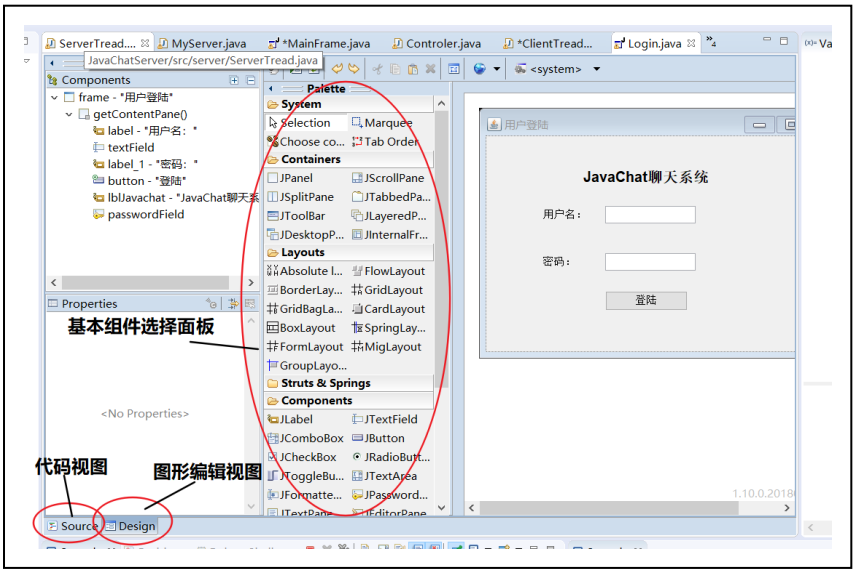
在 eclipse 的菜单栏选择 Help（帮助）->Eclipse MarketPlace（eclipse 市场）



在搜索栏搜索 WindowBuilder 即可找到相应的插件，点击下载即可自动安装



安装完成后，创建新的文件时选择 new->others->application window/JFrame 即创建出可视化编辑的界面。



2、程序数据构建

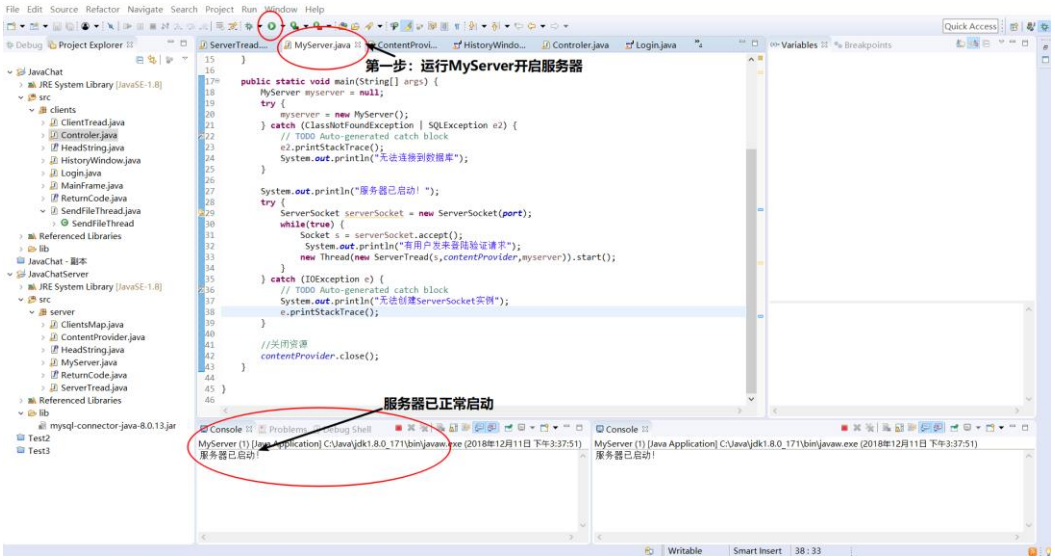
程序需要初始化用户信息，将用户信息存入 JavaChat 数据库中，将用户信息暂且分成用户名（userName）和用户密码（passWord）。初始化的用户名和密码信息有：Huang——123、Li——123、Liu——123。

其次，还需要初始化这三位用户的联系人表，创建三个联系人表名为：Huang、Li、Liu，各自好友（friendName）分别有{Li，Liu}、{Huang}、{Huang}，聊天记录一栏均为 null。

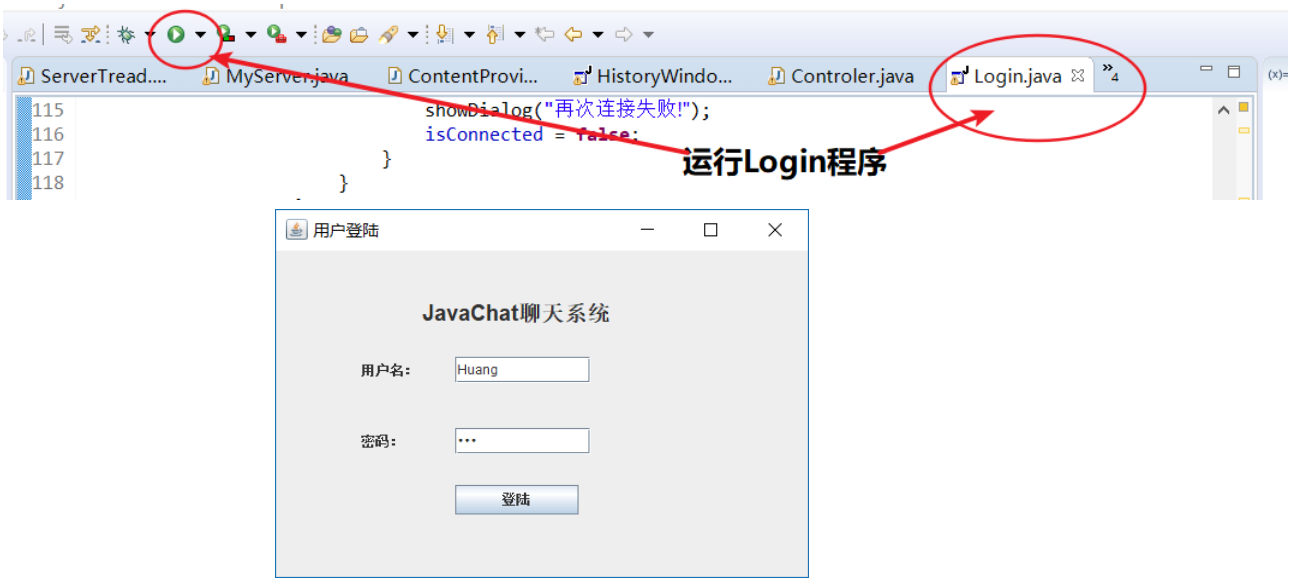
3、程序实际运行测试

服务器的正常启动需要首先启动电脑中的 mysql 服务，需以管理员权限打开 cmd 命令行窗口，开启服务（例如我的 mysql 服务为 mysql80，则需在命令行输入 net start mysql80 键入回车开启）。

1)首先开启服务器



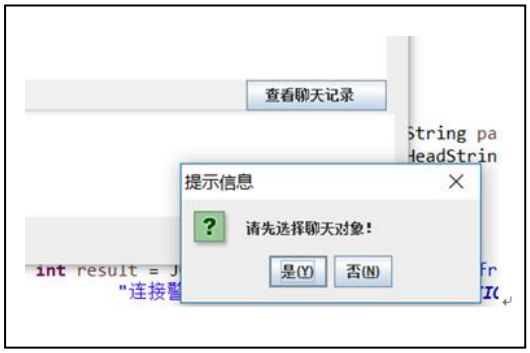
2） 打开服务器端，即可显示出登陆界面



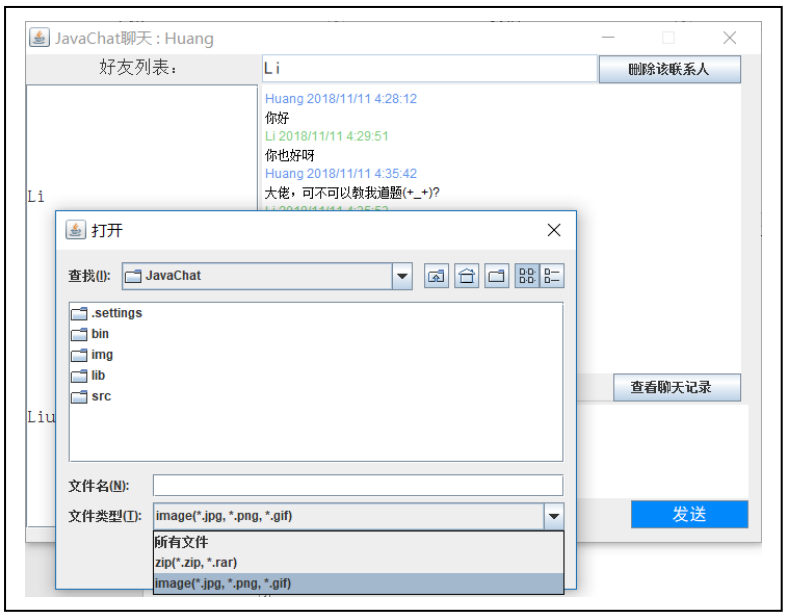
3） 键入用户名和密码，点击登陆，进行登陆。如果出现错误会有弹窗提示，无错误则直接跳转至聊天主界面。

4） 聊天界面弹出，先选中好友列表中要聊天的好友，就能开始聊天了，窗体标题显示用户登陆名字，聊天记录显示框上方显示当前对话好友名字。

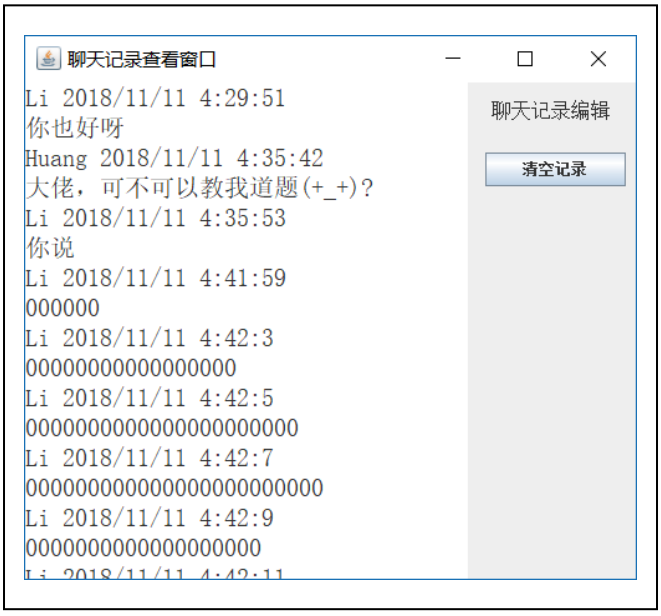
（好友不在线，仅会在服务器的命令行会有提示）

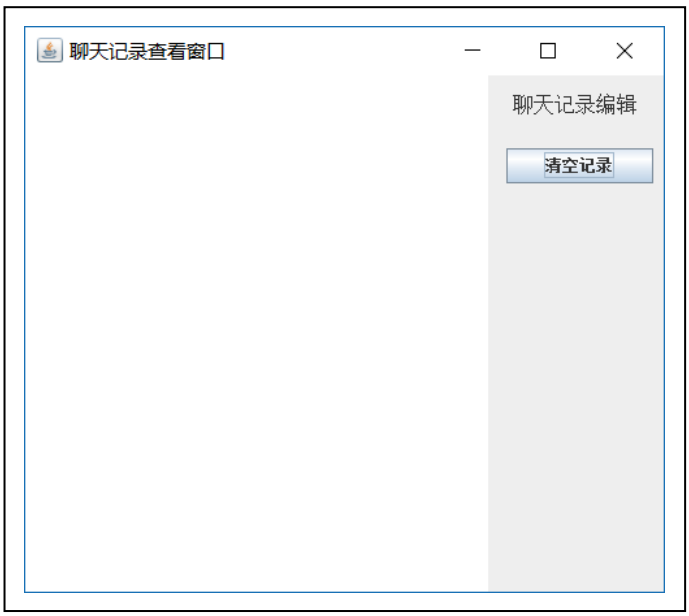


5） 发送文件，弹出文件选择框。文件的发送过程会在服务器命令行有显示进度



6） 查看聊天记录，将显示最近聊天的内容。清空聊天记录。





以上介绍及运行截图即程序的整体运行结果（无差错），但实际上程序仍然存在一些 bug 因时间原因，未能解决。以下将已经发现的一一列出，并尝试分析其原因：

A、好友列表选中颜色无变化，与代码实现过程中添加了颜色变化向违背。

B、聊天信息显示框部分会因聊天内容行数过多而无法显示。因在设置组件的时候未考虑道这个因素，仅使用了不可滑动查看的 JTextPane。后期将 JTextPane 加入 JScrollPane 中实现聊天信息的可滑动显示。同样的问题也出现在聊天记录查询界面 HistoryWindow 上。

C、关闭聊天记录查询界面时，会导致客户端程序的意外退出。应该时窗口关闭时间的响应出现问题。

D、文件发送功能不可用。选择好文件后，发送文件的客户端命令行有“文件发送成功”的提示，但是文件发送进程卡在了服务器转发文件上。原因应该是在于，使用字节流读取时是通过判断读取的内容是否到文件尾部来实现，但是转发过程中无真正意义上的文件尾部，故服务器端将在 readLine（）方法上出现线程阻塞。（原本这部分代码独立于本程序经过测试成功，但是是在双方的 socket 的输入输出流均关闭的情况下）后来在 debug 的时候，曾加入了间断传入状态信息的代码，但是为能实现文件的真正传输。

E、本程序未能实现远距离、使用真正意义上的网络进行两台或以上的计算机间的通信。因本程序未涉及查找准确计算机网络位置进行通信的算法实现，后期改进可改用 websocket 进行通信。

四、附录:源代码

开发环境：eclipse 的 photon 版本 64 位

数据库版本：mysql-8.0.13-winx64

1、客户端端程序（JavaChat 包）

ClientTread.java 、Controler.java 、HeadString.java 、History.java 、Login.java 、MainFrame.java 、ReturnCode.java 、SendFileThread.java

2、服务器端程序（JavaChatServer 包）

ClientsMap.java 、ContentProvider.java 、MyServer.java 、ReturnCode.java 、ServerThread.java

3、共有拓展包：

Mysql-connector-java-8.0.13.jar

源代码：

客户端：（JavaChat 包）

（ClientTread.java）

```
package clients;

import java.io.BufferedReader;
import java.io.DataInputStream;
import java.io.File;
import java.io.FileOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.Socket;
```

```
public class ClientTread implements Runnable{
    private Socket s = null;
    private BufferedReader br = null;
    private MainFrame frame;
    private String userName;
    public ClientTread(Socket s,MainFrame frame) {
        this.s = s;
        this.frame = frame;
        this.userName = frame.userName;
        try {
            br = new BufferedReader(new InputStreamReader(s.getInputStream()));
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("clientTread :无法打开输入流");
            e.printStackTrace();
        }
    }
    @Override
    public void run() {
        // TODO Auto-generated method stub
        String line = null;
        try {
            while((line = br.readLine()) != null) {
                if(line.startsWith(HeadString.SendFile)) {
                    String fromName = line.substring(HeadString.Stringlength,line.length());
                    byte[] bytes = new byte[1024];
                     int length = 0;
                     DataInputStream dis = null;
                     File file = null;
                     FileOutputStream fos = null;
```

```
                    //打开输入流
                    try {
                        dis = new DataInputStream(s.getInputStream());
                    } catch (IOException e) {
                        System.out.println("无法打开输入输出流！");
                        e.printStackTrace();
                    }
                    try {
                        //打开文件，准备接收数据
                        File directory = new File("E:\\JavaChat");//保存路径
                        String fileName = dis.readUTF();
                        if(!directory.exists()) {
                            directory.mkdirs();
                        }
                        file = new File(directory.getAbsolutePath() + File.separatorChar + fileName);

                        //打开文件输入流
                        fos = new FileOutputStream(file);
                        //开始接收文件
                        System.out.println("正在接收文件：");
                        frame.showMassage(frame.toUserName, userName," 正 在 接 收 文 件。。。。", false);
                        while((length= dis.read(bytes,0,bytes.length))!=-1){
                            fos.write(bytes,0,length);
                            fos.flush();
                        }
                        System.out.println("来自 client1 的文件接收成功！");
                        frame.showMassage(frame.toUserName,userName, "文件接收成功！\n文件保存在："+file.getAbsolutePath(), false);
```

```
                    } catch (IOException e) {
                        System.out.println("文件名读取错误或无法新建文件");
                        e.printStackTrace();
                    }
                }else {
                    frame.showMassage(frame.toUserName, userName,line, false);
                }
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("clientTread :已断开连接！");
        }
    }
}
```

（Controler.java）

```
package clients;

import java.awt.EventQueue;
import java.net.Socket;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

/** * @author  作者 :GientJun
* @date 创建时间：2018 年 12 月 2 日 下午 8:06:37
* @version 1.0
```

```
* @parameter
* @return
* @throws
*/
public class Controler {
    public final String url = "jdbc:mysql://localhost:3306/JavaChat?serverTimezone=GMT%2B8";

    public String userName = null;
    public Socket socket = null;
    public String friends = null;
    private Statement statement = null;

    public Controler(String userName,Socket socket,String friends) throws ClassNotFoundException, SQLException {
        this.userName = userName;
        this.socket = socket;
        statement = getStatement();
        this.friends = friends;
    }
    private  Statement getStatement() throws ClassNotFoundException, SQLException {
        // TODO Auto-generated method stub
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection(url, "root", "3117004568");
        return con.createStatement();
    }

    //保存聊天记录
    public void saveHistory(String friendName,String history) throws SQLException {
        String pass = null;
        pass = this.getHistory(friendName);
```

```
        String upDate = "update "+userName+" set history="+"'"+pass+history+"'"+" where friendName="+"'"+friendName+"'";
        statement.execute(upDate);
        System.out.println("聊天记录保存成功！");
        System.out.println("聊天记录保存成功！");
    }

    //查询聊天记录,返回 userName 和 friendName 的聊天记录
    public String getHistory(String friendName) throws SQLException {
        String query = "select * from "+userName;
        String temp = "";
        ResultSet result = statement.executeQuery(query);
        while(result.next()) {
            if(result.getString("friendName").equals(friendName)) {
                temp = temp + result.getString("history");
                break;
            }
        }
        return temp;
    }

    //删除聊天记录
        public void deleteHistory(String friendName) throws SQLException {
            String upDate = "update "+userName+" set history=null where friendName="+"'"+friendName+"'";
            statement.execute(upDate);
            System.out.println("聊天记录删除成功！");
        }
}
```

（HeadString.java）

```
package clients;
/** * @author  作者 :GientJun
* @date 创建时间：2018年12月6日 下午10:35:29
* @version 1.0
* @parameter
* @return
* @throws
*/
public interface HeadString {
    int Stringlength = 4;

    String LogIn = "%IN%";
    String UserName = "%NN%";
    String PassWord = "%PW%";

    String UserEdit = "%UE%";
    String Delete = "%DE%";
    String ADD = "%AD%";
    String History = "%HI%";

    String Chat = "%CH%";
    String SomeBody = "%SB%";
    String Massage = "%MS%";

    String splitChar = "%SC%";

    String SendFile = "%SF%";
    String OK = "%OK%";
    String ING = "%IG%";
}
```

（History.java）

```
package clients;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JTextArea;
import javax.swing.JLabel;
```

```
import java.awt.Font;
import javax.swing.JButton;
import javax.swing.SwingConstants;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.sql.SQLException;

/** * @author  作者 :GientJun
* @date 创建时间：2018 年 12 月 4 日 下午 4:04:36
* @version 1.0
* @parameter
* @return
* @throws
*/
public class HistoryWindow extends JFrame{

//    private JFrame frame;
      private String history = "还没有聊天记录！";
      private String friendName = null;
      private Controler controler = null;
      private JTextArea textArea = null;
//    /**
//     * Launch the application.
//     */
//    public static void main(String[] args) {
//          EventQueue.invokeLater(new Runnable() {
//                public void run() {
//                      try {
```

```
//                          HistoryWindow window = new HistoryWindow();
//                          window.frame.setVisible(true);
//                      } catch (Exception e) {
//                          e.printStackTrace();
//                      }
//                  }
//          });
//  }

    /**
     * Create the application.
     */
    public HistoryWindow(String history,String friendName,Controler controler) {
        if(!history.equals(null) && !history.equals("")) {
            this.history = history;
        }
        this.controler = controler;
        this.friendName = friendName;
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
//      frame = new JFrame();
        this.getContentPane().setFont(new Font("Adobe Garamond Pro Bold", Font.PLAIN, 17));
        this.setBounds(100, 100, 509, 444);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setVisible(true);
```

```
this.setTitle("聊天记录查看窗口");
this.getContentPane().setLayout(null);

textArea = new JTextArea();
textArea.setBounds(0, 0, 356, 397);
textArea.setFont(new Font("宋体", Font.PLAIN, 20));
this.getContentPane().add(textArea);
textArea.setText(history);

JLabel label = new JLabel("\u804A\u5929\u8BB0\u5F55\u7F16\u8F91");
label.setHorizontalAlignment(SwingConstants.CENTER);
label.setFont(new Font("宋体", Font.PLAIN, 16));
label.setBounds(370, 13, 107, 18);
this.getContentPane().add(label);

JButton button = new JButton("\u6E05\u7A7A\u8BB0\u5F55");
button.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        try {
            controler.deleteHistory(friendName);
            textArea.setText("");
        } catch (SQLException e1) {
            // TODO Auto-generated catch block
            System.out.println("清空聊天记录错误！");
            e1.printStackTrace();
        }
    }
});
button.setBounds(370, 56, 113, 27);
```

```
        this.getContentPane().add(button);

        this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {

              }
            public void windowClosed(WindowEvent e) {

              }
            });
    }
}
```

（Login.java）

```
package clients;

import java.awt.EventQueue;

import javax.swing.JFrame;
import javax.swing.JTextField;
import javax.swing.JEditorPane;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JButton;
import javax.swing.JPasswordField;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
```

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.net.InetSocketAddress;
import java.net.Socket;
import java.net.UnknownHostException;
import java.sql.SQLException;

/** * @author  作者 :GientJun
* @date 创建时间：2018 年 12 月 1 日 下午 11:12:17
* @version 1.0
* @parameter
* @return
* @throws
*/
public class Login {

    JFrame frame;
    private JTextField textField;
    private JPasswordField passwordField;
    private static final int port = 6666;
```

```
private static final String ip = "localhost";
private static String headString = HeadString.LogIn;
private boolean isConnected = false;

/**
 * Launch the application.
 */
public static void main(String[] args) {
    EventQueue.invokeLater(new Runnable(){
        public void run(){
        try{
            Login window = new Login();
            window.frame.setVisible(true);
        }catch(Exception e){
            e.printStackTrace();
        }
    }
});
}

/**
 * Create the application.
 */
```

```
public Login() {
    initialize();
}

/**
 * Initialize the contents of the frame.
 */
private void initialize() {
    frame = new JFrame();
    frame.setTitle("用户登陆");
    frame.setBounds(100, 100, 505, 345);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.getContentPane().setLayout(null);

    JLabel label = new JLabel("\u7528\u6237\u540D\uFF1A");
    label.setBounds(78, 100, 72, 18);
    frame.getContentPane().add(label);

    textField = new JTextField();
    textField.setBounds(164, 97, 124, 24);
    frame.getContentPane().add(textField);
    textField.setColumns(1);
```

```
JLabel label_1 = new JLabel("\u5BC6\u7801\uFF1A");
label_1.setBounds(78, 165, 55, 18);
frame.getContentPane().add(label_1);

JButton button = new JButton("\u767B\u9646");
button.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        String userName = textField.getText();
        String passWord = new String(passwordField.getPassword());
        Socket s = null;
        if(userName == null || passWord == null||userName.equals("")||passWord.equals("")) {
            showDialog("请正确输入用户名和密码");
        }else {
            try {
                s = new Socket();
                s.connect(new InetSocketAddress(ip,port),1000);
                isConnected = true;
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                System.out.println("无法与服务器的通信！ ");
                //e1.printStackTrace();
```

```
                //如果警告弹窗中选择 ok 重新连接，如果选择 cancle 则取消连接
                if(showDialog("无法与服务器建立连接，请确认你的连接情况后按确认！")) {
                    try {
                        s = new Socket();
                        s.connect(new InetSocketAddress(ip,port),1000);
                        isConnected = true;
                    } catch (IOException e2) {
                        // TODO Auto-generated catch block
                        //e2.printStackTrace();
                        System.out.println("再次连接失败");
                        showDialog("再次连接失败!");
                        isConnected = false;
                    }
                }
            }
            if(isConnected == true) {
                try {
                    String a = null;
                    if((a=sendLogin(s,userName,passWord)) != null) {
                        Controler controler = new Controler(userName,s,a);
                        frame.setVisible(false);
```

```
                    MainFrame mainFrame = new
MainFrame(controler);
                    }else {
                        showDialog("用户名或密码错误！");
                    }
                } catch (IOException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                } catch (ClassNotFoundException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    e1.printStackTrace();
                }
            }
        }

    }
});
button.setBounds(164, 214, 113, 27);
frame.getContentPane().add(button);
```

```
        JLabel lblJavachat = new JLabel("JavaChat\u804A\u5929\u7CFB\u7EDF");
        lblJavachat.setBounds(134, 48, 184, 18);
        lblJavachat.setFont(new java.awt.Font("Dialog", 1, 20));
        frame.getContentPane().add(lblJavachat);

        passwordField = new JPasswordField();
        passwordField.setBounds(164, 162, 124, 24);
        frame.getContentPane().add(passwordField);

        frame.setVisible(true);
    }

    //如果登陆成功，返回 true，否则返回 false
    public String sendLogin(Socket socket,String userName,String passWord) throws IOException {
        String temp = null;
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
        BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));

    bw.write(HeadString.LogIn+wrapData(userName,passWord)+HeadString.L
```

```
ogIn+"\n");
            bw.flush();
            String line = br.readLine();
            if(line.startsWith(ReturnCode.LogERROR)) {
                System.out.println("登陆密码或用户名错误");
                return null;
            }
            else {
                System.out.println("登陆成功！");
                return br.readLine();
            }

        }

        private String wrapData(String userName,String passWord) {
            return
HeadString.UserName+userName+HeadString.UserName+HeadString.splitCha
r+HeadString.PassWord+passWord+HeadString.PassWord;
        }

        //如果选择 ok 返回 true，选择 cancel 返回 false
        private boolean showDialog(String warning) {
            int result = JOptionPane.showConfirmDialog(frame,warning,
```

```
                "连接警告",JOptionPane.OK_CANCEL_OPTION);
        if(result == JOptionPane.OK_OPTION)   return true;
        return false;
    }
}
```

（MainFrame.java）

```
package clients;

import java.awt.BorderLayout;
import java.awt.Color;
import java.awt.Component;
import java.awt.Dimension;
import java.awt.EventQueue;
import java.awt.FlowLayout;
import java.net.Socket;
import java.sql.SQLException;
import java.util.Calendar;
import java.util.HashMap;

import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;
import javax.swing.BorderFactory;
```

```
import javax.swing.JButton;
import javax.swing.JFileChooser;
import javax.swing.JScrollPane;
import javax.swing.JToolBar;
import javax.swing.ScrollPaneConstants;
import javax.swing.filechooser.FileNameExtensionFilter;
import javax.swing.text.BadLocationException;
import javax.swing.text.SimpleAttributeSet;
import javax.swing.text.StyleConstants;
import javax.swing.text.StyledDocument;

import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.io.BufferedWriter;
import java.io.File;
import java.io.IOException;
import java.io.OutputStreamWriter;

import javax.swing.JTextPane;
import javax.swing.SwingConstants;
import javax.swing.JLabel;
```

```
import javax.swing.JOptionPane;

import java.awt.Font;
import java.awt.GridLayout;
import java.awt.Panel;
import java.awt.CardLayout;
import javax.swing.JTextArea;

/** * @author  作者 :GientJun
* @date 创建时间：2018 年 12 月 2 日 下午 9:13:34
* @version 1.0
* @parameter
* @return
* @throws
*/
public class MainFrame extends JFrame{

//  private JFrame frame;
    protected String userName;
    protected String toUserName = null;
    private String[] friends;
    private Socket socket;
    private JTextField textField_2;
```

```
    private BufferedWriter bw= null;
    private Controler controler = null;

    private JTextArea textArea;
    private JScrollPane scrollPane=null;
    private JPanel jContentPane = null;
    private JLabel priviousLabel = null;
    private HashMap<String,JTextPane> textPaneMap = new HashMap<>();
    private JTextPane priviousTextPane = null;
    public static JButton button_1;
    private JButton button    = null;

    private Thread listenTread = null;

    private HashMap<String,String> history;

    private static final int LoginMode = 666;
    private static final int ChatMode = 555;
    private static final int EditMode = 444;


//    /**
//     * Launch the application.
```

```
//     */
//    public static void main(String[] args) {
//        EventQueue.invokeLater(new Runnable() {
//            public void run() {
//                try {
//                    MainFrame window = new MainFrame();
//                    window.frame.setVisible(true);
//                } catch (Exception e) {
//                    e.printStackTrace();
//                }
//            }
//        });
//    }

    /**
     * Create the application.
     */
    public MainFrame(Controler controler) {
        this.controler = controler;
        this.userName = controler.userName;
        this.socket = controler.socket;
        this.friends = controler.friends.split(",");
        history = new HashMap<>();
```

```
        for(String name:friends) {
            history.put(name, "");
        }

        try {
            bw = new BufferedWriter(new
OutputStreamWriter(socket.getOutputStream()));
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("MainFrame:无法打开输入流");
            e.printStackTrace();
        }
        initialize();
    }

    /**
     * Initialize the contents of the frame.
     */
    private void initialize() {
//        frame = new JFrame();
        this.setBounds(100, 100, 713, 504);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.getContentPane().setLayout(null);
```

```
this.setTitle("JavaChat 聊天 ："+userName);
this.setBackground(new Color(65,105,225));
this.setResizable(false);

button = new JButton("\u53D1\u9001");
button.setForeground(Color.WHITE);
button.setFont(new Font("黑体", Font.PLAIN, 17));
button.setBackground(new Color(1, 136, 251));
button.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        if(toUserName != null) {
            String massage =textArea.getText();
            textArea.setText("");
            try {
                sendMassage(toUserName,massage,ChatMode);
            } catch (IOException e1) {
                // TODO Auto-generated catch block
                System.out.println("无法发送信息！ ");
            e1.printStackTrace();
            }
            showMassage(userName,toUserName,massage,true);
        }else {
```

```
                showDialog("请先选择聊天对象！ ");
            }
        }
});
button.setBounds(582, 429, 113, 27);
this.getContentPane().add(button);

JPanel panel = new JPanel();
panel.setVisible(true);
panel.setBackground(Color.white);
panel.setLayout(new GridLayout(0, 1, 0, 0));
for(int i=0 ;i < friends.length; i++) {
    JLabel label = new JLabel();
    label.setFont(new Font("SimSun", Font.PLAIN, 17));
    label.setHorizontalAlignment(SwingConstants.LEFT);
    label.setText(friends[i]);
    label.setVisible(true);

    JTextPane textPane = new JTextPane();
    textPane.setEditable(false);
    textPane.setBounds(227, 29, 461, 278);
    textPane.setVisible(false);
    textPaneMap.put(label.getText(), textPane);
```

```
label.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseClicked(MouseEvent e) {
        textField_2.setText(label.getText());
        if(priviousLabel != null) {
            priviousLabel.setBackground(null);
        }
        priviousLabel = label;

        label.setBackground(Color.BLACK);

        priviousTextPane.setVisible(false);
        priviousTextPane = textPaneMap.get(label.getText());
        priviousTextPane.setVisible(true);

        toUserName = label.getText();
        textArea.setText("");
    }
});

panel.add(label);
this.getContentPane().add(textPane);
```

```
        }

//      panel.add(new JLabel("haha"));
//      panel.add(new JLabel("haha"));
//      panel.add(new JLabel("haha"));
        panel.setPreferredSize(new Dimension(29, friends.length*75));
        scrollPane = new JScrollPane();
        scrollPane.setBounds(0, 29, 224, 427);
        scrollPane.setViewportView(panel);
        scrollPane.setVisible(true);
        scrollPane.setBackground(Color.WHITE);
        this.getContentPane().add(scrollPane);

        JToolBar toolBar = new JToolBar();
        toolBar.setBackground(new Color(240, 240, 240));
        toolBar.setBounds(227, 307, 338, 27);
        this.getContentPane().add(toolBar);

        button_1 = new JButton("\u53D1\u9001\u6587\u4EF6");
        button_1.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                String address = showFileOpenDialog(button_1);
```

```
                System.out.println(address);
                new Thread(new SendFileThread(socket,address,toUserName)).start();
            }
        });
        toolBar.add(button_1);

        textField_2 = new JTextField();
        textField_2.setEditable(false);
        textField_2.setBounds(227, 0, 323, 27);
        textField_2.setFont(new Font("黑体", Font.PLAIN, 17));
        textField_2.setBackground(Color.white);
        this.getContentPane().add(textField_2);
        textField_2.setColumns(10);

        priviousTextPane = new JTextPane();
        priviousTextPane.setEditable(false);
        priviousTextPane.setBounds(227, 29, 461, 278);
        priviousTextPane.setVisible(true);
        this.getContentPane().add(priviousTextPane);

        JButton button_2 = new JButton("\u67E5\u770B\u804A\u5929\u8BB0\u5F55");
```

```
        button_2.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
                String his = null;
                try {
                    his = controler.getHistory(toUserName);
                } catch (SQLException e1) {
                    // TODO Auto-generated catch block
                    System.out.println("无法从数据库获取聊天记录！");
                    e1.printStackTrace();
                }
                if( his == null||his.equals("null")) {
                    his = "";
                }
                new
HistoryWindow(history.get(toUserName)+his,toUserName,controler);
            }
        });
        button_2.setHorizontalAlignment(SwingConstants.LEFT);
        button_2.setBounds(565, 307, 123, 27);
        this.getContentPane().add(button_2);

        JLabel label = new JLabel("\u597D\u53CB\u5217\u8868\uFF1A");
```

```
        label.setBackground(null);
        label.setFont(new Font("宋体", Font.PLAIN, 17));
        label.setHorizontalAlignment(SwingConstants.CENTER);
        label.setBounds(0, 0, 224, 27);
        this.getContentPane().add(label);

        textArea = new JTextArea();
        textArea.setBounds(227, 337, 468, 90);
        this.getContentPane().add(textArea);

        JButton button_3 = new
JButton("\u5220\u9664\u8BE5\u8054\u7CFB\u4EBA");
        button_3.addMouseListener(new MouseAdapter() {
            @Override
            public void mouseClicked(MouseEvent e) {
            }
        });
        button_3.setBounds(551, 1, 137, 27);
        getContentPane().add(button_3);
        this.setVisible(true);
        listenTread = new Thread(new ClientTread(socket,this));
        listenTread.start();
```

```
this.addWindowListener(new WindowAdapter() {
        public void windowClosing(WindowEvent e) {
          for(String name:friends) {
           try {
               controler.saveHistory(name, history.get(name));
               System.out.println(history.get(name));
           } catch (SQLException e1) {
               // TODO Auto-generated catch block
               System.out.println("数据库保存聊天记录错误");
           }
          }
          }
        public void windowClosed(WindowEvent e) {
          for(String name:friends) {
           try {
               controler.saveHistory(name, history.get(name));
               System.out.println(history.get(name));
           } catch (SQLException e1) {
               // TODO Auto-generated catch block
               System.out.println("数据库保存聊天记录错误");
           }
          }
          }
```

```
            });
    }

    //选择要发送的文件
    private static String showFileOpenDialog(Component parent) {
        JFileChooser fileChooser = new JFileChooser();
        File rFile = null;
        //设置当前文件夹位置
        fileChooser.setCurrentDirectory(new File("."));
        //设置文件选择模式，这里为仅文件可选
        fileChooser.setFileSelectionMode(JFileChooser.FILES_ONLY);
        //不可多选文件，一次只能传一个文件
        fileChooser.setMultiSelectionEnabled(false);
        //添加可用的文件过滤器，第一个参数是描述，后面是需要过滤的文件扩展名
        fileChooser.addChoosableFileFilter(new FileNameExtensionFilter("zip(*.zip, *.rar)", "zip", "rar"));
        //设置默认文件过滤器
        fileChooser.setFileFilter(new FileNameExtensionFilter("image(*.jpg, *.png, *.gif)", "jpg", "png", "gif"));
        int result = fileChooser.showOpenDialog(button_1);
        if(result == JFileChooser.APPROVE_OPTION) {
            rFile = fileChooser.getSelectedFile();
```

```
            //File[] files = fileChooser.getSelectedFile();
        }
        return rFile.getAbsolutePath();
    }

    //将接收到的信息显示在聊天面板上
    public void showMassage(String fromUser,String toUser,String massage,boolean isUser) {

        if(isUser == true) {
            StyledDocument d = textPaneMap.get(toUser).getStyledDocument();
            SimpleAttributeSet attr = new SimpleAttributeSet();
            SimpleAttributeSet attr2 = new SimpleAttributeSet();

            StyleConstants.setForeground(attr, new Color(100,149,237));
            StyleConstants.setForeground(attr2, Color.black);
        try {
            d.insertString(d.getLength(),fromUser+" "+getTime()+"\n",attr);
            d.insertString(d.getLength(), massage+"\n", attr2);
            String temp = history.get(toUser)+fromUser+" "+getTime()+"\n"+massage+"\n";
            history.replace(toUser, temp);
        } catch (BadLocationException e) {
```

```
            // TODO Auto-generated catch block
            e.printStackTrace();
        }

        }else {
            StyledDocument d = textPaneMap.get(fromUser).getStyledDocument();
            SimpleAttributeSet attr = new SimpleAttributeSet();
            SimpleAttributeSet attr2 = new SimpleAttributeSet();

            StyleConstants.setForeground(attr, new Color(124,205,124));
            StyleConstants.setForeground(attr2, Color.black);

            try {
                d.insertString(d.getLength(),fromUser+" "+getTime()+"\n",attr);
                d.insertString(d.getLength(), massage+"\n", attr2);
                String temp2 = history.get(fromUser)+fromUser+" "+getTime()+"\n"+massage+"\n";
                history.replace(fromUser, temp2);
            } catch (BadLocationException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
```

```
        }
    }

    //获取接受到信息的时间
    public String getTime() {
        Calendar cal = Calendar.getInstance();
        String result = cal.get(Calendar.YEAR)+"/"+
cal.get(Calendar.MONTH)+"/"+cal.get(Calendar.DATE)+" "+

    cal.get(Calendar.HOUR)+":"+cal.get(Calendar.MINUTE)+":"+cal.get(Cale
ndar.SECOND);
        return result;
    }
    //发送指定的信息到服务器
    public void sendMassage(String toUserName,String massage,int mode)
throws IOException {
        System.out.println("发送的有用信息为："+toUserName+" "+massage);
        switch(mode) {
        case
ChatMode:bw.write(HeadString.Chat+HeadString.SomeBody+toUserName+He
adString.SomeBody+HeadString.splitChar+HeadString.Massage+massage+Hea
dString.Massage+HeadString.Chat+"\n");
                    bw.flush();
```

```
                break;
//          case LoginMode:bw.write(HeadString.LogIn
//                  +HeadString.UserName+toUserName+HeadString.UserName
//                  +HeadString.splitChar
//                  +HeadString.PassWord+massage+HeadString.PassWord
//                  +HeadString.LogIn
//                  +"\n");
//              break;
            case EditMode:
                break;
            default:System.out.println("发送模式选择错误！");
            }
        }
        //如果选择 ok 返回 true，选择 cancel 返回 false
        private void showDialog(String warning) {
                int result = JOptionPane.showConfirmDialog(button ,warning,
                        "提示信息",JOptionPane.OK_OPTION);
            }
}
```

（ReturnCode.java）

```
package clients;
/** * @author  作者 :GientJun
* @date 创建时间：2018年12月1日 下午9:17:24
* @version 1.0
* @parameter
```

```
* @return
* @throws
*/
public interface ReturnCode {
    int StringLength = 2;
    String LogERROR = "$$";
    String LogSUCCEED = "##";
}
```

（SendFileThread.java）

package clients;

import java.io.BufferedWriter;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.IOException;

import java.io.OutputStreamWriter;

import java.net.Socket;

/ * @author 作者 :GientJun**

*** @date 创建时间：2018 年 12 月 3 日 上午 9:56:35**

*** @version 1.0**

*** @parameter**

*** @return**

*** @throws**

***/**

public class SendFileThread implements Runnable{

```
private Socket socket = null;
private String fileAdress = null;
private String toUserName = null;
public SendFileThread(Socket socket,String fileAdress,String toUserName) {
    this.socket = socket;
    this.fileAdress = fileAdress;
    this.toUserName = toUserName;
}
@Override
public void run() {
    // TODO Auto-generated method stub
    //打开指定的文件
    File file = null;
     try {
          file = new File(fileAdress);
     } catch (NullPointerException e) {
          System.out.println("输入地址错误！ ");
          e.printStackTrace();
     }
     DataOutputStream dos = null;
     FileInputStream fis = null;
     BufferedWriter bw = null;
```

```
try {
    if (file.exists()) {
        bw = new BufferedWriter(new OutputStreamWriter(socket.getOutputStream()));
        dos = new DataOutputStream(socket.getOutputStream());
        fis = new FileInputStream(file);

bw.write(HeadString.SendFile+HeadString.SomeBody+toUserName+HeadString.SomeBody+HeadString.SendFile+"\n");
        bw.flush();
        dos.writeUTF(file.getName());
        dos.flush();

        //传输文件具体操作
        System.out.println("文件传输中！ ");
        byte[] bytes = new byte[1024];
        int length = 0;
        while ((length = fis.read(bytes, 0, bytes.length)) != -1) {
            dos.write(bytes, 0, length);
            dos.flush();
            bw.write(HeadString.ING+"\n");
            bw.flush();
```

```
                }
                bw.write(HeadString.OK+"\n");
                bw.flush();
                System.out.println("文件传输完成!");

            } else {
                System.out.println("无法打开文件或文件不存在！");
            }
        } catch (IOException e) {
            System.out.println("读或写文件出错！");
            e.printStackTrace();
        }
    }
}
```

服务器端程序（JavaChatServer 包）

（ClientsMap.java）

```
package server;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Set;

public class ClientsMap<K,V> extends HashMap<K,V> {
```

```
    //根据 value 值来删除 map 中的元素
    public void removeByValue(Object value) {
        for(Object key:keySet()) {
            if(get(key) == value) {
                remove(key);
                break;
            }
        }
    }
    //重写 map 的 put 方法，使得 map 中的元素没有重复
    public synchronized V put(K key, V value) {
        for(V val:valueSet()) {
            if(val.equals(value) && val.hashCode()==value.hashCode()) {
                throw  new  RuntimeException("ClientMap  实例中不允许含有重
复的 value 值！");
            }
        }
        return super.put(key, value);
    }
    //获取 value 的集合
    public Set<V> valueSet(){
        Set<V> result = new HashSet<>();
```

```
        for(K key : keySet()) {
            result.add(get(key));
        }
        return result;
    }
}
```

（ContentProvider.java）

```
package server;

import java.awt.List;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.Map;

/** * @author  作者 :GientJun
* @date 创建时间：2018 年 12 月 1 日 下午 8:23:41
* @version 1.0
* @parameter
```

```
* @return
* @throws
*/
/*
 * 用户表为：用户名(userName)  密码(passWord)
 * (users) Huang                123
 *              Li                      123
 *              Liu                     123
 *
 * 联系人表：          friendName  history
 * (Huang)      Li          null
 *                      Liu             null
 *
 * 联系人表：          friendName  history
 * (Li)         Huang       null
 *
 * 联系人表：          friendName  history
 * (Liu)        Huang       null
 * */
public class ContentProvider {
    public              final           String          url             =
"jdbc:mysql://localhost:3306/JavaChat?serverTimezone=GMT%2B8";
    private Statement statement = null;
```

```
    private ArrayList<String> userNames;

    public ContentProvider() throws ClassNotFoundException, SQLException {
        statement = getStatement(url);
        userNames = query("users","userName");
    }

    //仅限 ContentProvider 内部使用，数据库信息查找
    private ArrayList<String> query(String tableName,String column) throws
SQLException{
        ArrayList<String> temp = new ArrayList<>();
        String query = "select * from "+tableName;
        ResultSet result = statement.executeQuery(query);
        while(result.next()) {
            String data = result.getString(column);
            temp.add(data);
        }
        return temp;
    }

    private Statement getStatement(String url) throws ClassNotFoundException,
SQLException {
        // TODO Auto-generated method stub
```

```
        Class.forName("com.mysql.cj.jdbc.Driver");
        Connection con = DriverManager.getConnection(url, "root",
"3117004568");
        return con.createStatement();
    }

    //登陆接口方法，返回登陆信息验证结果
    public boolean checkIn(String userName,String passWord) {
        try {
            String query = "select * from users";
            ResultSet result = statement.executeQuery(query);
            while(result.next()) {
                String name = result.getString("userName");
                String pw = result.getString("passWord");
                if(name.equals(userName) && pw.equals(passWord)) {
                    return true;
                }
            }
        } catch (SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return false;
```

```
}
//获取所有好友的名字
public String getFriends(String userName) throws SQLException {
    String temp = "";
    ArrayList<String> list = query(userName,"friendName");
    for(String name:list) {
        temp = temp + ","+name;
    }
    return temp.substring(1,temp.length());
}



//关闭所有资源
public void close() {
    try {
        statement.close();
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}
```

```
    //单方面删除好友
    public void deleteFriends(String userName,String friendName) throws SQLException {
        String delete = "delete from "+userName+" where friendName="+friendName;
        statement.execute(delete);
    }

    //添加好友
    public void addFriends(String userName,String friendName) throws SQLException {
        String add = "insert into "+userName+" values("+"'"+friendName+"'"+","+"null)";
        statement.execute(add);
    }
}
```

（MyServer.java）

```
package server;
import java.io.IOException;
import java.net.*;
import java.sql.SQLException;
import java.util.ArrayList;
```

```
public class MyServer {
    private static final int port = 6666;

    public static ClientsMap<String,Socket> clientsmap = new ClientsMap<>();
    public static ContentProvider contentProvider ;

    public MyServer() throws ClassNotFoundException, SQLException {
        clientsmap = new ClientsMap<>();
        contentProvider = new ContentProvider();
    }

    public static void main(String[] args) {
        MyServer myserver = null;
        try {
            myserver = new MyServer();
        } catch (ClassNotFoundException | SQLException e2) {
            // TODO Auto-generated catch block
            e2.printStackTrace();
            System.out.println("无法连接到数据库");
        }

        System.out.println("服务器已启动！");
        try {
```

```
            ServerSocket serverSocket = new ServerSocket(port);
            while(true) {
                Socket s = serverSocket.accept();
                 System.out.println("有用户发来登陆验证请求");
                new                                                    Thread(new
ServerTread(s,contentProvider,myserver)).start();
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("无法创建 ServerSocket 实例");
            e.printStackTrace();
        }

        //关闭资源
        contentProvider.close();
    }
}
```

（ReturnCode.java）

```
package server;
/** * @author  作者 :GientJun
* @date 创建时间：2018年12月1日 下午9:17:24
* @version 1.0
* @parameter
* @return
* @throws
*/
public interface ReturnCode {
```

```
    int StringLength = 2;
    String LogERROR = "$$";
    String LogSUCCEED = "##";
}
```

（ServerThread.java）

package server;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.OutputStreamWriter;

import java.io.PrintStream;

import java.net.Socket;

import java.sql.SQLException;

import java.util.HashMap;

import java.util.Map;

public class ServerTread implements Runnable {

```
    private BufferedReader br = null;
    private BufferedWriter bw = null;
    private String userName = null;
    private Socket s = null;
    private Map<String,String> data = null;
    private ContentProvider cp = null;
    private MyServer myserver;

    public ServerTread(Socket s,ContentProvider cp,MyServer myserver) {
        this.s = s;
        this.cp = cp;
        this.myserver = myserver;
        try {
            br = new BufferedReader(new
InputStreamReader(s.getInputStream()));
            bw = new BufferedWriter(new
OutputStreamWriter(s.getOutputStream()));
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("无法打开输入流！ ");
            e.printStackTrace();
        }
```

```
    }

    //一直从客户端读取数据，解析成可用数据，发给指定用户
    @Override
    public void run() {
        // TODO Auto-generated method stub
        String line = null;
        while((line = readFromClient()) != null) {
            //聊天功能
             if(line.startsWith(HeadString.Chat)                    &&
line.endsWith(HeadString.Chat)) {
                //去掉协议字符串
                line = getContent(line);
                //获取字符串中的有用信息
                data = getData(line);
                System.out.println(data.get(HeadString.SomeBody));

sendMassage(data.get(HeadString.SomeBody),data.get(HeadString.Massage));
            //登陆验证,如果成功，传输好友列表
            }else          if(line.startsWith(HeadString.LogIn)          &&
line.endsWith(HeadString.LogIn)) {
                line = getContent(line);
                data = getData(line);
```

```
                System.out.println("开始验证
"+data.get(HeadString.UserName)+"的身份");
                //如果用户退出程序,终止线程

if(!checkLogIn(s,data.get(HeadString.UserName),data.get(HeadString.PassWor
d))) {
                    System.out.println("验证失败！");
                    continue;
                }else {
                    userName = data.get(HeadString.UserName);
                    myserver.clientsmap.put(userName, s);
                    System.out.println(userName+":验证成功！");
                    try {
                        bw.write(cp.getFriends(userName)+"\n");
                        bw.flush();
                    } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                    } catch (SQLException e) {
                        // TODO Auto-generated catch block
                        System.out.println("数据数据库读取出错！");
                        e.printStackTrace();
                    }
```

```
                }
            //发送文件功能
            }else if(line.startsWith(HeadString.SendFile) &&
line.endsWith(HeadString.SendFile)) {
                line = getContent(line);
                data = getData(line);
                String toUser = data.get(HeadString.SomeBody);
                Socket somebody = myserver.clientsmap.get(toUser);
                if(somebody == null) {
                    System.out.println(toUser+"：已下线！");
                    continue;
                }
                DataInputStream dis = null;
                DataOutputStream dos = null;
                BufferedWriter bw2 = null;
                System.out.println(userName+"----文件----->"+toUser);
               //打开输入、输出流
                    try {
                        dis = new DataInputStream(s.getInputStream());
                        dos = new
DataOutputStream(somebody.getOutputStream());
                        bw2 = new BufferedWriter(new
OutputStreamWriter(somebody.getOutputStream()));
```

```
} catch (IOException e) {
        System.out.println("无法打开输入输出流！");
        e.printStackTrace();
}

byte[] bytes = new byte[1024];
int length = 0;

//开始接收文件
System.out.println("正在转发文件：");
try {
  //发送文件转发提示信息
     bw2.write(HeadString.SendFile+userName+"\n");
     bw2.flush();
  while((length= dis.read(bytes,0,bytes.length))!=-1){
        System.out.print("。");
        dos.write(bytes, 0, length);
          dos.flush();
          String flag = br.readLine();
          System.out.println(flag);
          if(flag.equals(HeadString.OK))
             break;
  }
```

```
                } catch (IOException e1) {
                    // TODO Auto-generated catch block
                    System.out.println("转发过程错误！");
                    e1.printStackTrace();
                }
                  System.out.println("来自"+userName+"的文件转发到"+toUser+"成功！");
            }else if(line.startsWith(HeadString.UserEdit) && line.endsWith(HeadString.UserEdit)){
                //客户端发送好友修改信息，操作数据库
                line = getContent(line);
                System.out.println(userName+"开始操作好友列表");
                try {
                    if(line.startsWith(HeadString.Delete)) {
                        String friendName = line.substring(HeadString.Stringlength, line.length());
                        cp.deleteFriends(userName, friendName);
                        System.out.println(userName+"删除了好友:"+friendName);
                    }else if(line.startsWith(HeadString.ADD)) {
                        String friendName = line.substring(HeadString.Stringlength, line.length());
                        cp.addFriends(userName, friendName);
```

```
                    System.out.println(userName+"  添  加  了  好
友:"+friendName);
                }
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                System.out.println("数据数据库操作出错！");
                e.printStackTrace();
            }
        }
    }
    }

    //发送信息到指定对象
    private void sendMassage(String toName,String massage) {
        Socket somebody = myserver.clientsmap.get(toName);
        if(somebody == null) {
            System.out.println(toName+"：下线了!！");
            return ;
        }
        try {
            BufferedWriter tobw = new BufferedWriter(new
OutputStreamWriter(somebody.getOutputStream()));
            tobw.write(massage+"\n");
```

```
            tobw.flush();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println(toName+"：连接错误!！");
            e.printStackTrace();
        }
        System.out.println(userName+"------>"+toName+"： "+massage);
    }

    //获取用户发送的信息（未处理）
    private String readFromClient() {
        try {
            return (String)br.readLine();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("用户:"+userName+"退出程序！");
            //发生异常则说明断开了连接，移除 socket
            if(myserver.clientsmap.containsValue(s)) {
                myserver.clientsmap.removeByValue(s);
            }
        }
        return null;
    }
```

```
    public String getContent(String line) {
        return
line.substring(HeadString.Stringlength,line.length()-HeadString.Stringlength);
    }

    //由得到的直接字符串转化为数据
    public Map<String,String> getData(String line) {
        String[] strings = line.split(HeadString.splitChar);
        Map<String,String> map = new HashMap<>();
        for(String part :strings) {
            String temp = part.substring(0,HeadString.Stringlength);
            if(temp.equals(HeadString.Massage)) {
                map.put(HeadString.Massage, getContent(part));
            }else if(temp.equals(HeadString.PassWord)){
                map.put(HeadString.PassWord,getContent(part));
            }else if(temp.equals(HeadString.SomeBody)) {
                map.put(HeadString.SomeBody, getContent(part));
            }else if(temp.equals(HeadString.UserName)) {
                map.put(HeadString.UserName, getContent(part));
            }
            System.out.println(temp+" 数据入图 "+getContent(part));
        }
```

```
    return map;
}

//登陆用户名与密码配对检验，若不配对则返回错误码
public boolean checkLogIn(Socket s,String userName,String passWord) {
    if(!myserver.contentProvider.checkIn(userName, passWord)) {
        try {
            bw.write(ReturnCode.LogERROR+"\n");
            bw.flush();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            //发生异常则说明用户断开了连接，终止线程
            System.out.println(userName+"登陆失败!!!");
            e.printStackTrace();
        }
        return false;
    }else {
        try {
            bw.write(ReturnCode.LogSUCCEED+"\n");
            bw.flush();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            //发生异常则说明用户断开了连接，终止线程
```

```
                System.out.println(userName+"登陆失败!!!");
                e.printStackTrace();
            }
            return true;
        }

    }

    private void SayHello() {
        try {
            bw.write("已建立与服务器的连接！\n");
        } catch (IOException e) {
            // TODO Auto-generated catch block
            System.out.println("ServerTread:无法写入输出流！ ");
            e.printStackTrace();
        }
    }
}
```