中南大学

《计算机网络》

课程设计报告

题 目 **基于C/S模式的通讯录备份和查询软件的设计与实现**

学生姓名 **陶婉莹**

指导教师 **刘嫔**

学 院 **信息科学与工程学院**

专业班级 **计算机科学与技术1505**

完成时间 **2017年10月10日**

**目录**

[第一章 需求分析 4](#_Toc26621)

[1.1服务器端功能要求 4](#_Toc21253)

[1.2客户端功能要求 4](#_Toc27846)

[1.3本程序可实现功能 4](#_Toc23079)

[第二章 总体设计 5](#_Toc16518)

[2.1总体结构 5](#_Toc9878)

[2.2 模块关系 5](#_Toc31442)

[2.3 总体流程图 6](#_Toc17178)

[第三章 详细设计 7](#_Toc32376)

[3.1类的描述 7](#_Toc2189)

[3.1.1 Login类 7](#_Toc12024)

[3.1.2 LoginListener类 7](#_Toc16935)

[3.1.3 Operation类 7](#_Toc14326)

[3.1.4 SeverFrame类 7](#_Toc16481)

[3.1.5 Server类 7](#_Toc16840)

[3.1.6 DES类 7](#_Toc9487)

[3.2重要类的定义伪代码及调用关系 7](#_Toc24185)

[3.2.1 LoginListener类 7](#_Toc31596)

[3.2.2Operation类 8](#_Toc12338)

[3.2.3 Server类 16](#_Toc31088)

[3.2.4 DES类 19](#_Toc1971)

[3.2.5 类的调用关系 21](#_Toc31966)

[第四章 调试分析 22](#_Toc25522)

[第五章 运行结果 23](#_Toc10665)

[5.1用户登录 23](#_Toc16414)

[5.2查询功能 23](#_Toc31907)

[5.3备份功能 25](#_Toc23704)

[5.4更新功能 26](#_Toc31898)

[5.5帮助功能 27](#_Toc7077)

[5.6增加功能 27](#_Toc17731)

[5.7删除功能 28](#_Toc17067)

[5.8修改功能 29](#_Toc16032)

[第六章 课程设计总结 30](#_Toc26886)

[参考文献 31](#_Toc4657)

[附录 31](#_Toc21519)

## 第一章 需求分析

本设计要求完成一个基于C/S模式的通讯录备份软件。采用C/S架构。应该具有易用、美观的图形界面。

### 1.1服务器端功能要求

（1）能够验证客户身份，接收客户端的备份通讯录的请求，能够实时备份和更新客户的通讯录。

（2）加密存储每个用户的通讯录

### 1.2客户端功能要求

（1）能登陆连接到服务器，回应：连接成功/失败。

（2）能备份本机通讯录。

（3）能实时更新本机通讯录。

（4）能查询本机通讯录。

### 1.3本程序可实现功能

客户端：

（1）能登陆连接到服务器，回应：连接成功/失败。

（2）能将本机通讯录备份到数据库。

（3）能实时从数据库获取最新内容更新本机通讯录。

（4）能查询本机通讯录。

（5）能增加、删除并且修改本机通讯录。

服务端：

（1）能够验证客户身份，接收客户端的备份通讯录的请求，能够实时备份和更新客户的通讯录。

（2）加、解密用户存储的通讯录

## 第二章 总体设计

### 2.1总体结构

客户端首先建立一个本地文件来存储本地的通讯录数据，通过本地文件对通讯录内容进行查询操作，在数据库中进行通讯录的增添、删除以及修改操作。同时备份功能可将本地文件中的通讯录内容上传至数据库中，更新功能则是将经过增添、删除或修改后的数据库中通讯录内容更新到本地文件中。

服务器负责验证客户端的登录账号和密码，若一致则与MySQL进行连接并回应客户端登陆成功，否则回应登陆失败。若登录成功，则服务端可响应客户端的备份请求，将本地文件中的通讯录内容经过DES加密后备份至数据库；还可响应客户端的更新要求，将数据库中通讯录的内容经过DES解密后更新至本地文件中。

### 2.2 模块关系

### 

通讯录

**服务端**

备份（加密）、更新（解密）

**客户端**

本地查询，连接数据库增、删、改

图2.2 通讯录系统模块关系图

### 

### 2.3 总体流程图

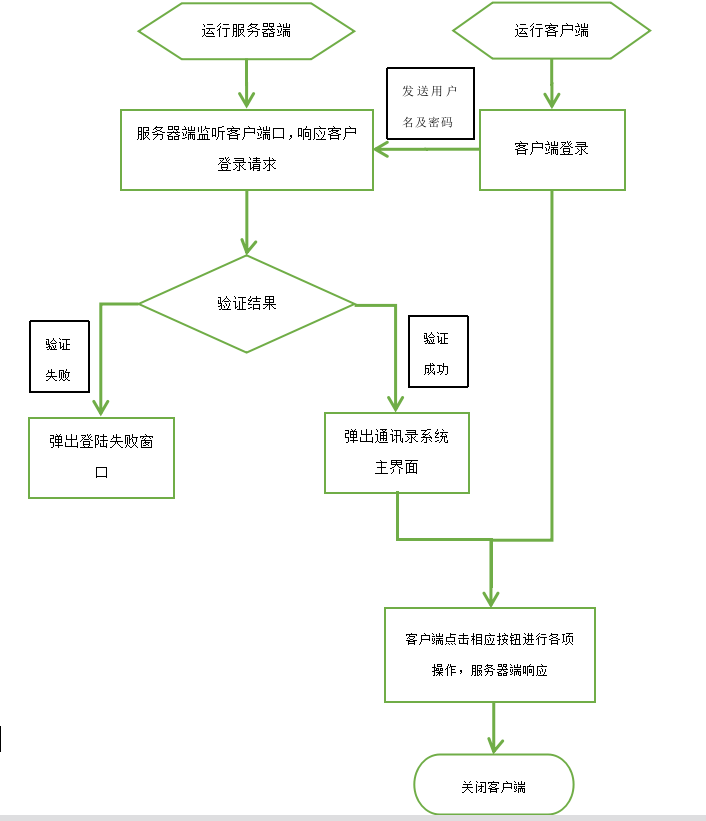


图2.3 通讯录系统流程图

## 第三章 详细设计

### 3.1类的描述

Client包中包含三个类：Login类，LoginListener类以及Operation类。

### 3.1.1 Login类

实现登录界面并实例化LoginListener 类给登录按钮添加ActionListner 监听方法。

### 3.1.2 LoginListener类

实现LoginListener类，继承ActionListener接口并重写抽象函数ActionPerformed。客户首先输入用户名和密码，然后单击登录按钮通过Java Socket连接到服务器，并且将账号和密码传给服务器，等待服务器响应后的验证结果。若验证通过则弹出通讯录系统主界面，否则弹出窗口显示用户名或密码错误、登陆失败。

### 3.1.3 Operation类

实现操作主界面，继承ActionListener接口并重写抽象函数ActionPerfomed。在构造方法中实现对各操作按钮的监听，用户点击相应界面上的按钮，通过传socket给服务器端并接收响应结果即可进行相应操作。

Server包中包含三个类：ServerFrame类，Sever类以及DESPlus类。

### 3.1.4 SeverFrame类

实现服务端界面，实时显示客户端进行的相应操作。

### 3.1.5 Server类

主要实现服务端的功能，包括验证用户名以及密码，连接MySQL，响应更新以及备份操作等等。

### 3.1.6 DES类

实现对通讯录内容的加密和解密。进行备份操作时加密通讯录内容后备份至数据库，进行更新操作时解密数据库内容更新至本地文件。

### 3.2重要类的定义伪代码及调用关系

### 3.2.1 LoginListener类

**public** **class** LoginListener **implements** ActionListener{

**public** LoginListener(JFrame frame,JTextField text,JPasswordField pw){

**this**.frame = frame;

**this**.text = text;

**this**.pw = pw;

}

@SuppressWarnings("deprecation")

**public** **void** actionPerformed(ActionEvent e){

**try**{

//发送密码和用户名到客户端

String user = text.getText();

String pass = pw.~~getText~~();

Socket s = **new** Socket("127.0.0.1",8800);

OutputStream os = s.getOutputStream();

OutputStreamWriter osw = **new** OutputStreamWriter(os);

PrintWriter pw = **new** PrintWriter(osw ,**true**);

pw.println(user+"%"+pass);

//接收服务器发回来的确认信息

InputStream is = s.getInputStream();

InputStreamReader isr = **new** InputStreamReader(is);

BufferedReader br = **new** BufferedReader(isr);

String answer = br.readLine();

//显示登录成功界面或密码错误界面

**if**(answer.equals("ok")){

Operation o = **new** Operation();

o.SetSocket(s);

frame.dispose();//关闭登录界面

}

**else**{

JTextField text = **new** JTextField(" 用户名或密码错误");

JFrame frame = **new** JFrame();

frame.setTitle("错误");

frame.setLocation(550,300);

frame.setSize(200,100);

frame.setDefaultCloseOperation(2);

frame.add(text);

frame.setVisible(**true**);

}

}**catch**(Exception e1){}

}

}

### 3.2.2Operation类

**public** **class** Operation {

// 获取已经连接好的Socket值

**public** Socket s;

**public** **void** SetSocket(Socket s) {

**this**.s = s;

}

// 客户端程序入口

**public** **static** **void** main(String args[]) {

Login login = **new** Login();

}

**public** Operation() **throws** Exception {

// 实例化窗口

javax.swing.JFrame frame = **new** javax.swing.JFrame();

// 实例化欢迎面板

JPanel Welcome = **new** JPanel();

// 实例化查询显示面板

JPanel CXPanel = **new** JPanel();

// 实例化添加显示面板

JPanel TJPanel = **new** JPanel();

// 实例化删除显示面板

JPanel SCPanel = **new** JPanel();

// 实例化修改显示面板

JPanel XGPanel = **new** JPanel();

// 实例化备份面板

JPanel BFPanel = **new** JPanel();

// 实例化更新面板

JPanel GXPanel = **new** JPanel();

// 实例化帮助面板

JPanel BZPanel = **new** JPanel();

// 实例化查询界面组件并添加

// 实例化添加面板组件并添加

// 实例化删除面板组件并添加

// 实例化修改面板组件并添加

// 实例化备份面板组件备份文本框并添加

// 实例化更新面板组件更新文本框并添加

// 实例化帮助面板组件更新文本框并添加

// 设置查询按钮的监听

itemcxdh.addActionListener(**new** ActionListener() {

QR1.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**try** {

// 在本地文件中查询通讯录信息

File f = **new** File("D:\\通讯录.txt");

FileReader fr = **new** FileReader(f);

BufferedReader br = **new** BufferedReader(fr);

FileReader fr2 = **new** FileReader(f);

BufferedReader br2 = **new** BufferedReader(fr2);

String str = InputText1.getText();

**if** (str.equals("")) {

CXText.append("\n" + "全部的联系人信息是：" + "\n");

**while** (br2.ready()) {

String string = br.readLine();

String str1 = string.split(" ")[0];

String str2 = string.split(" ")[1];

String str3 = string.split(" ")[2];

String str4 = string.split(" ")[3];

CXText.append("姓名: " + str1 + "\n" + "电话: " + str2 + "\n" + "地址: " + str3 + "\n" + "QQ: "

+ str4 + "\n\n");

}

br2.close();

} **else** {

**while** (br.ready()) {

String string = br.readLine();

String str1 = string.split(" ")[0];

String str2 = string.split(" ")[1];

String str3 = string.split(" ")[2];

String str4 = string.split(" ")[3];

**if** (InputText1.getText().equals(str1) || InputText1.getText().equals(str2)) {

CXText.append("\n" + "查询到的联系人信息是：" + "\n");

CXText.append("姓名: " + str1 + "\n" + "电话: " + str2 + "\n" + "地址: " + str3 + "\n"

+ "QQ: " + str4 + "\n\n");

**break**;

}

**if** (!br.ready())

CXText.append("\n" + "没有所查的联系人信息，请确认信息是否输入正确" + "\n");

}

br.close();

}

// 向服务器端传送查询的消息

OutputStream os = s.getOutputStream();

OutputStreamWriter osw = **new** OutputStreamWriter(os);

PrintWriter pw = **new** PrintWriter(osw, **true**);

pw.println("查询" + " " + InputText1.getText() + "的联系方式");

} **catch** (Exception e1) {

}

}

});

// 设置增添按钮的监听

QR2.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

TJText.setText("\n" + "请在下方的文本框输入想要添加的联系人信息" + "\n" + "然后单击添加按钮" + "\n");

**try** {

// 在数据库中添加新的联系人

Class.*forName*("com.mysql.jdbc.Driver");

Connection cn = DriverManager.*getConnection*(

"jdbc:mysql://127.0.0.1:3306/addressbook?characterEncoding=utf8&useSSL=true", "root",

"twy97620");

System.***out***.println("Connecting to database...");

System.***out***.println("Creating statement...");

PreparedStatement ps = cn.prepareStatement("insert into people values(?,?,?,?)");

// 将文本信息加密

ps.setString(1, des.encrypt(nametext.getText().toString()));

System.***out***.println(des.encrypt(nametext.getText().toString()));

ps.setString(2, des.encrypt(teltext.getText().toString()));

ps.setString(3, des.encrypt(addtext.getText().toString()));

ps.setString(4, des.encrypt(qqtext.getText().toString()));

**int** rows = ps.executeUpdate();

System.***out***.println("Rows impacted : " + rows);

ps.close();

cn.close();

nullarea.setText("已完成添加！" + "\n");

// 向服务器端传送添加的指令

OutputStream os = s.getOutputStream();

OutputStreamWriter osw = **new** OutputStreamWriter(os);

PrintWriter pw = **new** PrintWriter(osw, **true**);

pw.println("添加" + " " + nametext.getText());

} **catch** (SQLException se) {

// Handle errors for JDBC

se.printStackTrace();

} **catch** (Exception e1) {

// Handle errors for Class.forName

e1.printStackTrace();

InputText3.setText("");

}

}

});

// 实例化删除按钮的监听

QR3.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

SCText.setText("请在下方文本框中输入想要删除的联系人姓名" + "\n");

**try** {

// 在数据库中按照姓名删除联系人

Class.*forName*("com.mysql.jdbc.Driver");

Connection cn = DriverManager.*getConnection*(

"jdbc:mysql://127.0.0.1:3306/addressbook?characterEncoding=utf8&useSSL=true", "root",

"twy97620");

System.***out***.println("Connecting to database...");

System.***out***.println("Creating statement...");

PreparedStatement ps = cn.prepareStatement("delete from people where pname=?");

ps.setString(1, des.encrypt(InputText3.getText().toString()));

**int** rows = ps.executeUpdate();

System.***out***.println("Rows impacted : " + rows);

ps.close();

cn.close();

**if** (rows == 0) {

SCWCText.setText("没有该联系人" + "\n" + "请确认联系人姓名是否正确！" + "\n");

} **else** {

SCWCText.setText("删除成功！" + "\n");

OutputStream os = s.getOutputStream();

OutputStreamWriter osw = **new** OutputStreamWriter(os);

PrintWriter pw = **new** PrintWriter(osw, **true**);

pw.println("删除" + " " + InputText3.getText() + "的信息");

}

} **catch** (SQLException se) {

// Handle errors for JDBC

se.printStackTrace();

} **catch** (Exception e2) {

// Handle errors for Class.forName

e2.printStackTrace();

}

}

});

// 实例化修改按钮的监听

QRxg.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**try** {

// 在数据库中修改联系人

Class.*forName*("com.mysql.jdbc.Driver");

Connection cn = DriverManager.*getConnection*(

"jdbc:mysql://127.0.0.1:3306/addressbook?characterEncoding=utf8&useSSL=true", "root",

"twy97620");

System.***out***.println("Connecting to database...");

System.***out***.println("Creating statement...");

PreparedStatement ps = cn.prepareStatement("delete from people where pname=?");

ps.setString(1, des.encrypt(Inputxgname.getText().toString()));

**int** rows = ps.executeUpdate();

System.***out***.println("Rows impacted : " + rows);

**if** (rows != 0) {

PreparedStatement ps1 = cn.prepareStatement("insert into people values(?,?,?,?)");

ps1.setString(1, des.encrypt(Inputxgname.getText().toString()));

ps1.setString(2, des.encrypt(xgteltext.getText().toString()));

ps1.setString(3, des.encrypt(xgaddtext.getText().toString()));

ps1.setString(4, des.encrypt(xgqqtext.getText().toString()));

**int** rows1 = ps1.executeUpdate();

System.***out***.println("Rows1 impacted : " + rows1);

**if** (rows1 != 0) {

System.***out***.println("成功！");

xgwc.setText("修改成功！" + "\n");

} **else** {

xgwc.setText("修改失败！" + "\n");

}

ps1.close();

} **else** {

xgwc.setText("该联系人不存在，修改失败！" + "\n");

}

ps.close();

cn.close();

// 向服务器端传送添加的指令

OutputStream os = s.getOutputStream();

OutputStreamWriter osw = **new** OutputStreamWriter(os);

PrintWriter pw = **new** PrintWriter(osw, **true**);

pw.println("添加" + " " + nametext.getText());

} **catch** (SQLException se) {

// Handle errors for JDBC

se.printStackTrace();

} **catch** (Exception e1) {

// Handle errors for Class.forName

e1.printStackTrace();

InputText3.setText("");

}

}

});

// 设置备份按钮的监听

itembf.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

frame.remove(Welcome);

frame.remove(CXPanel);

frame.remove(XGPanel);

frame.remove(TJPanel);

frame.remove(SCPanel);

frame.remove(GXPanel);

frame.remove(BZPanel);

frame.add(BFPanel);

frame.revalidate();

frame.repaint();

BFText.setText("请稍等，正在备份！" + "\n");

**try** {

// 向服务器发起备份的请求

OutputStream os = s.getOutputStream();

OutputStreamWriter osw = **new** OutputStreamWriter(os);

PrintWriter pw = **new** PrintWriter(osw, **true**);//true表示自动刷新缓冲区

pw.println("备份");

File f = **new** File("D:\\通讯录.txt");

FileReader fr = **new** FileReader(f);

BufferedReader br1 = **new** BufferedReader(fr);

**while** (br1.ready()) {

String string = br1.readLine();

pw.println(string);

}

br1.close();

pw.println("end");

// 接收服务器端传来备份的结果

InputStream is = s.getInputStream();

InputStreamReader isr = **new** InputStreamReader(is);

BufferedReader br = **new** BufferedReader(isr);

BFText.append(br.readLine() + "\n");

} **catch** (Exception e1) {

}

}

});

// 设置更新按钮的监听

itemgx.addActionListener(**new** ActionListener() {

**public** **void** actionPerformed(ActionEvent e) {

**try** {

// 向服务器发起更新的请求

OutputStream os = s.getOutputStream();

OutputStreamWriter osw = **new** OutputStreamWriter(os);

PrintWriter pw = **new** PrintWriter(osw, **true**);

pw.println("更新");

// 接收服务器传来更新的结果

InputStream is = s.getInputStream();

InputStreamReader isr = **new** InputStreamReader(is);

BufferedReader br = **new** BufferedReader(isr);

File f = **new** File("D:\\通讯录.txt");

FileWriter fw = **new** FileWriter(f);

PrintWriter pw1 = **new** PrintWriter(fw);

String string = br.readLine();

**while** (!string.equals("更新成功!")) {

pw1.append(string + "\r\n");

string = br.readLine();

}

pw1.close();

GXText.append("更新成功!" + "\n");

} **catch** (Exception e1) {

}

}

});

}// Operation()结束

}

### 3.2.3 Server类

**public** **class** Server {

**public** **static** **void** main(String[] args) {

ServerFrame sf = **new** ServerFrame();

// 服务器在8800端口监听

@SuppressWarnings("resource")

ServerSocket ss = **new** ServerSocket(8800);

System.***out***.println("服务器正在8800端口监听……");

Socket s = ss.accept();

// 接收用户名和密码

InputStream is = s.getInputStream();

InputStreamReader isr = **new** InputStreamReader(is);

BufferedReader br = **new** BufferedReader(isr);

String uandp = br.readLine();

String u = uandp.split("%")[0];

String p = uandp.split("%")[1];

// 将用户名密码的验证信息传送到客户端

OutputStream os = s.getOutputStream();

OutputStreamWriter osw = **new** OutputStreamWriter(os);

PrintWriter pw = **new** PrintWriter(osw, **true**);

**if** (u.equals("twycwt") && p.equals("twy97620")) {

pw.println("ok");

sf.text.append("客户登录成功" + "\n");

**while** (**true**) {

String message = br.readLine();

DES des = **new** DES();

// 响应客户端备份命令

**if** (message.equals("备份")) {

sf.text.append("备份联系人信息" + "\n");

**try** {

Class.*forName*("com.mysql.jdbc.Driver");

Connection cn = DriverManager.*getConnection*(

"jdbc:mysql://127.0.0.1:3306/addressbook?characterEncoding=utf8&useSSL=true",

"root", "twy97620");

System.***out***.println("Connecting to database...");

System.***out***.println("Creating statement...");

Statement st = cn.createStatement();

**int** rows = st.executeUpdate("delete from people where pname is not null");

System.***out***.println("Rows impacted : " + rows);

String string = br.readLine();

**int** count = 0;

**while** (!string.equals("end")) {

String pname = string.split(" ")[0];

String telephone = string.split(" ")[1];

String address = string.split(" ")[2];

String qq = string.split(" ")[3];

PreparedStatement ps = cn.prepareStatement(

"insert into people(pname,telephone,address,qq) values(?,?,?,?)");

//将本地记录加密后备份至数据库

ps.setString(1, des.encrypt(pname));

ps.setString(2, des.encrypt(telephone));

ps.setString(3, des.encrypt(address));

ps.setString(4, des.encrypt(qq));

ps.executeUpdate();

count++;

System.***out***.println("正在备份..." + count);

string = br.readLine();

}

st.close();

cn.close();

pw.println("备份成功!");

} **catch** (SQLException se) {

// Handle errors for JDBC

se.printStackTrace();

pw.println("数据库操作失败!");

} **catch** (Exception e) {

// Handle errors for Class.forName

e.printStackTrace();

pw.println("备份失败!");

}

}

// 相应客户端更新的命令

**if** (message.equals("更新")) {

sf.text.append("更新联系人信息" + "\n");

**try** {

Class.*forName*("com.mysql.jdbc.Driver");

Connection cn = DriverManager.*getConnection*(

"jdbc:mysql://127.0.0.1:3306/addressbook?characterEncoding=utf8&useSSL=true",

"root", "twy97620");

Statement st = cn.createStatement();

System.***out***.println("Connecting to database...");

System.***out***.println("Creating statement...");

ResultSet rs = st.executeQuery("select pname,telephone,address,qq from people");

rs.last();

**int** rows = rs.getRow();

System.***out***.println("Rows : " + rows);

rs.beforeFirst();

**int** count = 0;

**while** (rs.next()) {

String string = **new** String();

System.***out***.println(string);

//将数据库中的记录解密后更新至本地文件

string = des.decrypt(rs.getString(1)) + " " + des.decrypt(rs.getString(2)) + " " + des.decrypt(rs.getString(3)) + " "

+ des.decrypt(rs.getString(4));

pw.println(string);

count++;

System.***out***.println("正在更新..." + count);

}

rs.close();

st.close();

cn.close();

pw.println("更新成功!");

} **catch** (SQLException se) {

// Handle errors for JDBC

se.printStackTrace();

pw.println("数据库操作失败！");

} **catch** (Exception e) {

// Handle errors for Class.forName

e.printStackTrace();

pw.println("更新失败!");

}

}

**if** (message.split(" ")[0].equals("查询")) {

sf.text.append(message + "\n");

}

**if** (message.split(" ")[0].equals("添加")) {

sf.text.append(message + "\n");

}

**if** (message.split(" ")[0].equals("删除")) {

sf.text.append(message + "\n");

}

}

} **else** {

// 发送错误信号到客户端

pw.println("error");

}

} **catch** (Exception e) {

}

}

}

### 3.2.4 DES类

**public** **class** DES {

**private** **static** String *strDefaultKey* = "national";

**private** Cipher encryptCipher = **null**;

**private** Cipher decryptCipher = **null**;

**public** **static** String byteArr2HexStr(**byte**[] arrB) **throws** Exception {

**int** iLen = arrB.length;

// 每个byte用两个字符才能表示，所以字符串的长度是数组长度的两倍

StringBuffer sb = **new** StringBuffer(iLen \* 2);

**for** (**int** i = 0; i < iLen; i++) {

**int** intTmp = arrB[i];

// 把负数转换为正数

**while** (intTmp < 0) {

intTmp = intTmp + 256;

}

// 小于0F的数需要在前面补0

**if** (intTmp < 16) {

sb.append("0");

}

sb.append(Integer.*toString*(intTmp, 16));

}

**return** sb.toString();

}

**public** **static** **byte**[] hexStr2ByteArr(String strIn) **throws** Exception {

**byte**[] arrB = strIn.getBytes();

**int** iLen = arrB.length;

// 两个字符表示一个字节，所以字节数组长度是字符串长度除以2

**byte**[] arrOut = **new** **byte**[iLen / 2];

**for** (**int** i = 0; i < iLen; i = i + 2) {

String strTmp = **new** String(arrB, i, 2);

arrOut[i / 2] = (**byte**) Integer.*parseInt*(strTmp, 16);

}

**return** arrOut;

}

**public** DES() **throws** Exception {

**this**(*strDefaultKey*);

}

**public** DES(String strKey) **throws** Exception {

Security.*addProvider*(**new** com.sun.crypto.provider.SunJCE());

Key key = getKey(strKey.getBytes());

encryptCipher = Cipher.*getInstance*("DES");

encryptCipher.init(Cipher.***ENCRYPT\_MODE***, key);

decryptCipher = Cipher.*getInstance*("DES");

decryptCipher.init(Cipher.***DECRYPT\_MODE***, key);

}

**public** **byte**[] encrypt(**byte**[] arrB) **throws** Exception {

**return** encryptCipher.doFinal(arrB);

}

**public** String encrypt(String strIn) **throws** Exception {

**return** *byteArr2HexStr*(encrypt(strIn.getBytes()));

}

**public** **byte**[] decrypt(**byte**[] arrB) **throws** Exception {

**return** decryptCipher.doFinal(arrB);

}

**public** String decrypt(String strIn) **throws** Exception {

**return** **new** String(decrypt(*hexStr2ByteArr*(strIn)));

}

**private** Key getKey(**byte**[] arrBTmp) **throws** Exception {

// 创建一个空的8位字节数组（默认值为0）

**byte**[] arrB = **new** **byte**[8];

// 将原始字节数组转换为8位

**for** (**int** i = 0; i < arrBTmp.length && i < arrB.length; i++) {

arrB[i] = arrBTmp[i];

}

// 生成密钥

Key key = **new** javax.crypto.spec.SecretKeySpec(arrB, "DES");

**return** key;

}

}

### 3.2.5 类的调用关系

客户端

STEP

01

Operation类

STEP

02

Login类

STEP

03

LoginListener类

服务器端

DES类

STEP

02

STEP

01

Server类

STEP

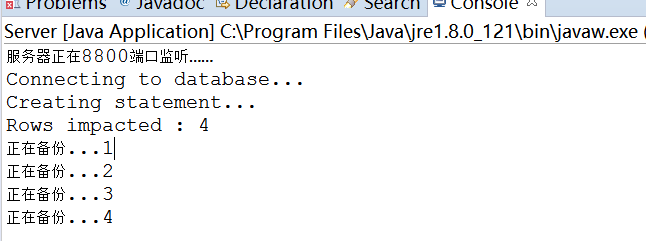
02

ServerFrame类

## 第四章 调试分析

在调试过程中经常出现sql语句语法出错的情况，尤其是使用preparedstatement的时候，后来每次运行前我都会先在mysql workbench中执行一遍该语句，看是否存在语法错误，如果执行失败就找出相应错误加以改正，这样逐渐地减少运行时出错的次数，同时每次执行一条sql语句后都在打印台上输出执行结果，以此判断该语句是否准确地实现了相应功能，以免运行后数据出错。

如图所示



这样可以清晰地看到程序运行情况，如果抛出异常则可以通过打印台输出的信息判断程序在哪一行运行失败，找出相应错误加以修改，并且可以通过与数据库中的实际数据作比较判断语句是否正确，完好地实现了相应功能。

这个程序最终实现了任务实验书中提出的所有要求，不过也存在一些可以改进的地方，比如界面还可以再优化完善，增添、删除修改功能可以更简洁，更贴近用户需求。

## 第五章 运行结果

### 5.1用户登录

运行operation类，显示登录界面，如图5.1所示

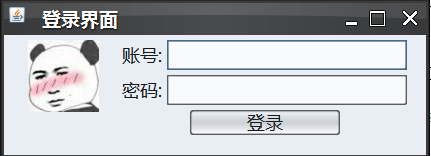


图5.1 通讯录备份软件客户端登录窗口

若用户名和密码错误，则弹出窗口提示登录失败，如图5.2所示

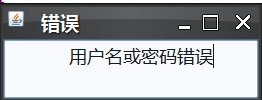


图5.2通讯录备份系统登录错误提示界面

若用户名和密码正确，则登录成功，出现操作主界面，如图5.3所示



图5.3 通讯录系统操作主界面

### 5.2查询功能

查询功能包括按名称查询和按电话查询，同时既可以查询所有联系人的信息，也可以查询某个联系人的信息。

点击查看菜单栏会出现按名称查看和按电话查看两个选项，点击一个进入查看界面，直接点击查询按钮即可查看所有联系人的信息，若输入相应联系人名称或电话即可查看该指定联系人的信息，如图5.4和5.5所示

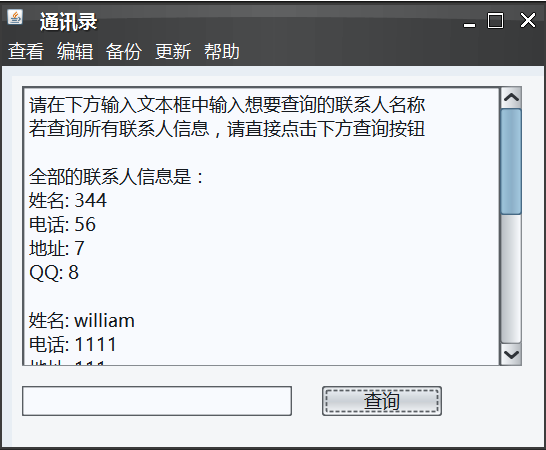


图5.4 通讯录系统查询所有联系人

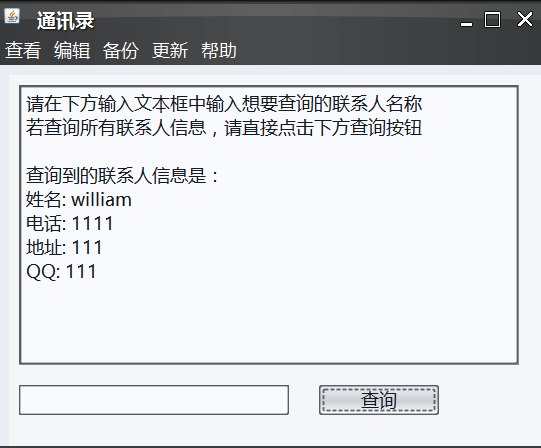


图5.5 通讯录系统查询条件查询联系人信息、

如果通讯录中不存在要查看的联系人，则显示相应提示，如图5.6所示

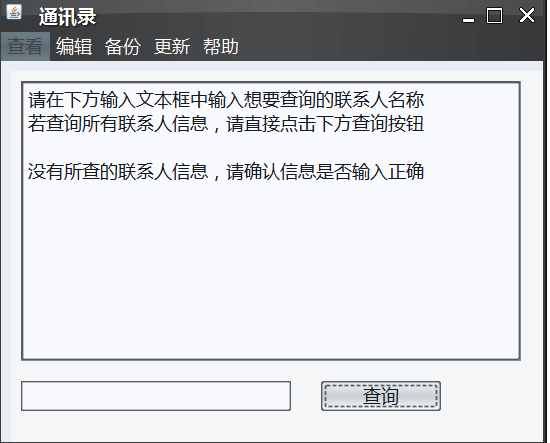


图5.6 通讯录系统查询联系人不存在

### 5.3备份功能

用户点击备份按钮后，客户端发送备份请求到服务端，服务端连接数据库后将本地文件中的通讯录内容经过DES算法加密后存储到数据库中，如图5.7所示



图5.7 通讯录系统备份界面

备份之前的数据库，如图5.8所示



图5.8 通讯录系统备份前的数据库

备份之后的数据库，如图5.9所示

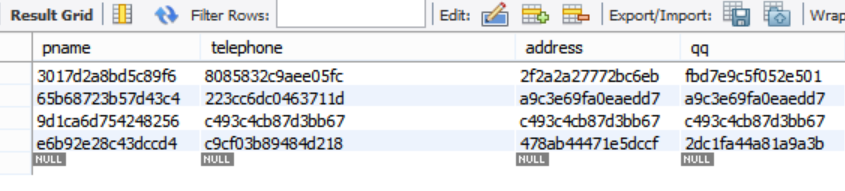


图5.9 通讯录系统备份成功后的数据库

### 5.4更新功能

经过增添、删除或修改操作后数据库中通讯录内容已经发生了改变，点击更新菜单栏即可将本地文件中的内容更新至最新的数据，如图5.10所示

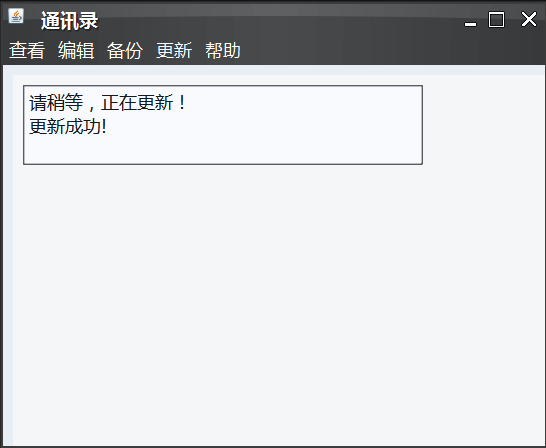


图5.10 通讯录系统更新联系人信息

更新之前的本地文件，如图5.11所示：

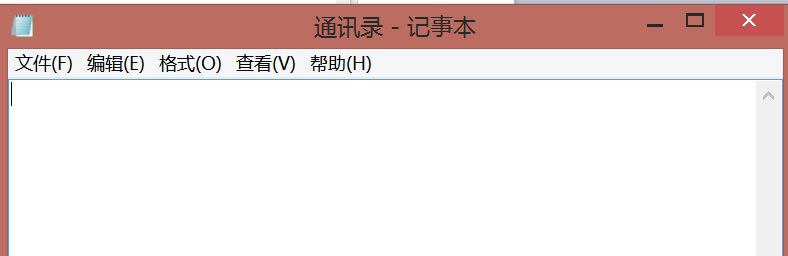


图5.11 通讯录系统更新前本地文件

更新之后的本地文件，如图5.12所示：

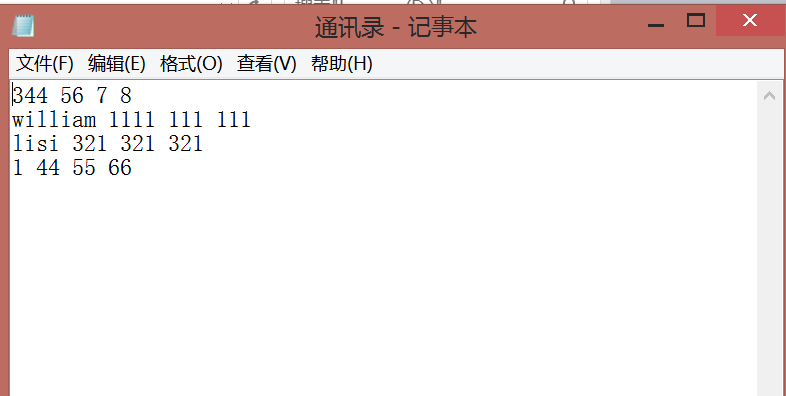


图5.12 通讯录系统更新成功后的本地文件

### 5.5帮助功能

点击帮助菜单栏，即可显示该通讯录系统的使用方法，方便用户操作，如图5.13所示。

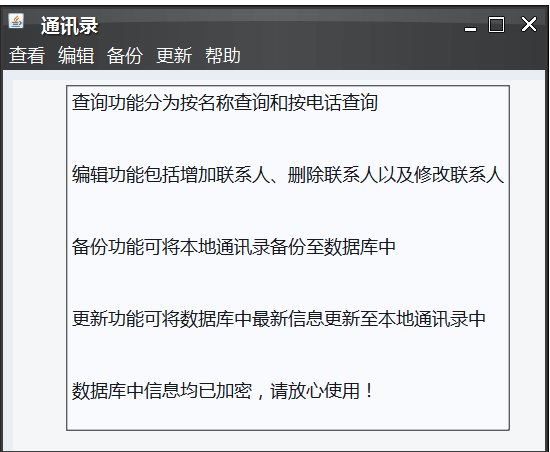


图 5.13 通讯录使用方法

### 5.6增加功能

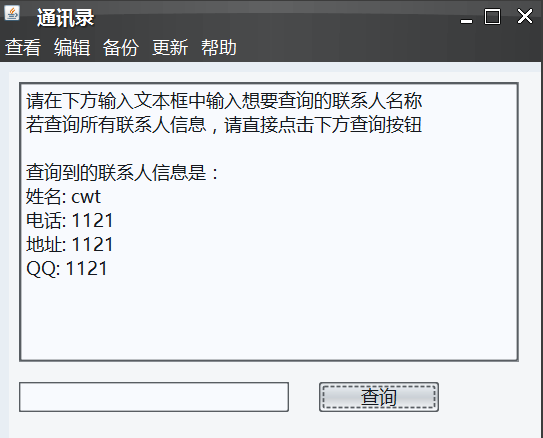
点击编辑菜单栏，选择增添选项，输入联系人的姓名，电话，qq和地址，点击添加按钮，完成添加，如图5.14所示。

### 

图 5.14 通讯录添加联系人

添加成功





### 5.7删除功能

点击编辑菜单栏，选择删除选项，输入要删除联系人的姓名，点击删除按钮即可删除该联系人，如图5.15所示。

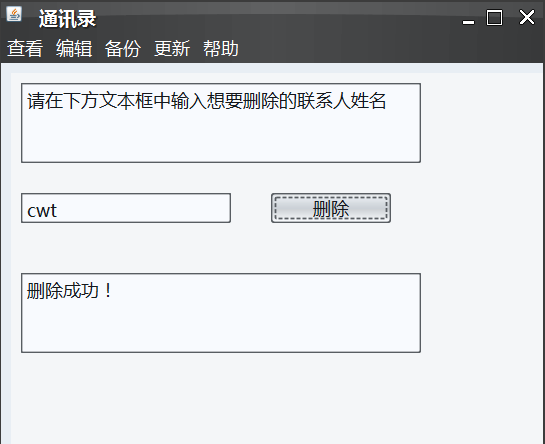


图 5.15 通讯录删除联系人

### 5.8修改功能

点击编辑菜单栏，选择修改选项，输入要修改的联系人名称，再输入修改后的电话、地址和qq，点击修改按钮即可，如图5.16所示

修改前



修改成功

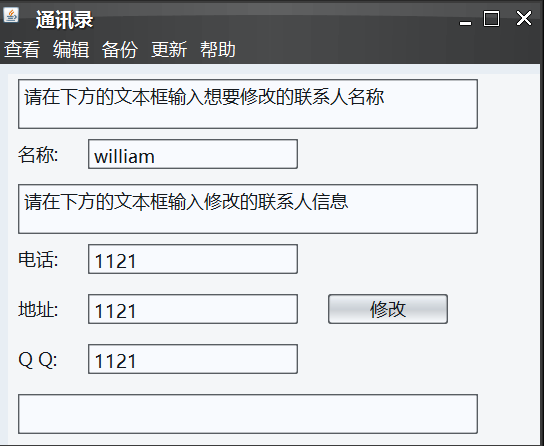
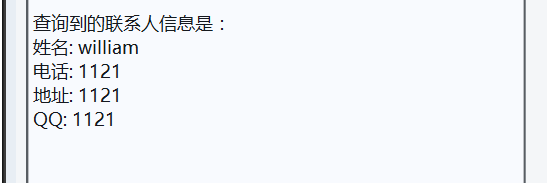


图 5.16 通讯录修改联系人

修改后



## 第六章 课程设计总结

经过两周的实验终于完成了这个课程设计，总体来说有一点难度，一方面是因为之前对套接字的使用不是很熟练，另一方面是因为这个课程设计里需要用到JDBC的知识，这个在以往的课程里也是没学过的，但好在前期靠自己翻阅相应资料、上网搜索并向老师和同学寻求帮助，很快掌握了本实验中所需的相应知识。

这次做的是通讯录系统，项目规模较以往做过的课设也更大，所以每完成一个模块都要运行测试功能是否完好。与此同时，图形界面的实现也是一个不小的问题，因为仅仅是上课时大致了解了一部分知识，实际中没有怎么运用到所以不太熟练。于是我选择先完成图形界面的部分，参考书上的例子，逐步修改完善我的登录界面、通讯录系统主界面以及服务器端界面，在最后优化的时候还导入了substance包，让整个界面的风格更加高级优美，最终效果我也比较满意。

实验要求中说明要对数据进行加密，因为之前选修过现代密码学课程有了一定的基础，所以实现起来没有困难，将DES算法封装在一个类中，加、解密时调用相应函数即可，十分方便。在调试过程中经常出现sql语句语法出错的情况，尤其是使用preparedstatement的时候，后来每次运行前我都会先在mysql workbench中执行一遍该语句，看是否存在语法错误，如果执行失败就找出相应错误加以改正，这样逐渐地减少运行时出错的次数，同时每次执行一条sql语句后都在打印台上输出执行结果，以此判断该语句是否准确地实现了相应功能，以免运行后数据出错。

通过这次课程设计，我对《计算机网络》这门课程有了更深的理解，同时也收获了很多以前没有学过的知识，提高了编程能力以及动手能力，自己发现问题、解决问题的能力也得到了提升。最后衷心感谢老师的辛勤付出与细心教导。相信在今后的课程设计中，我可以做得更好！

## 参考文献

[1]《Java语言程序设计》 郭克华、段桂华 清华大学出版社

## 附录

源程序代码：见文件夹中的“源程序代码”。