课程： Java 应用技术

实验： 期末大程—远程信息管理系统

姓名： 魏一鸣

学号： 3150100565

日期： 2019 年 12 月 30 日

指导老师： 楼学庆

目录

一、 程序目的

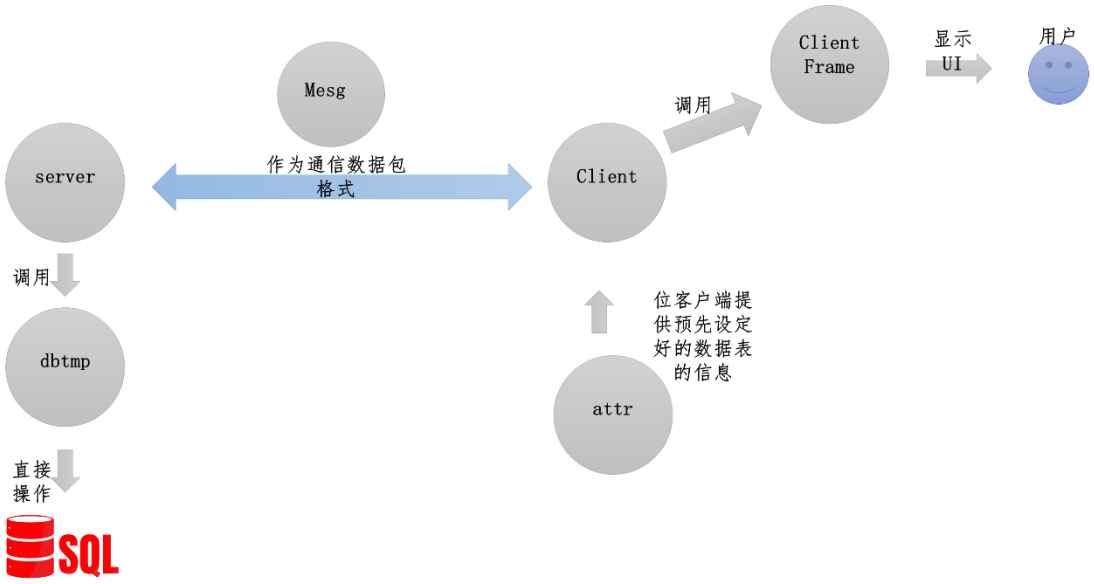
设计一个支持远程操作的学生信息管理系统。使用户能够通过网络远程控制在服务器的数据库上的学生信息管理系统。

具体要求如下：

1. 通过 Java 的 Socket 编程实现客户机与服务器之间的 TCP 连接和通信；
2. 将数据库操作结果显示在 Java 的 UI 界面上；
3. 实现多线程编程；
4. 支持多用户访问

二、 程序实现原理与算法说明

程序共有 6 个 Java 类。其中具体类名称和类之间的关系可由下图表示。



服务器端为 server，服务器通过与客户端建立 TCP 连接，通过每个连接生成一个线程，令线程与各自对应的客户端通信。

客户端为 Client 类，客户端起到了用户和服务器之间的桥接的作用。Client 本身定义的函数不多。其实现 UI 界面主要依靠调用 UI 类 ClientFrame，其实现数据传递和数据预处理主要依靠消息传递格式类 Mesg 类。

Mesg 类有三个成员变量，分别代表数据包传递方向 SorC、数据库操作 Cmd 和命令或信息内容 inst 可变 String 数组。SorC 为 0 是表示客户端发往服务器，为 1 时表示服务器发往客户端，当 SorC 为 -1 时，表示用户输入命令出错。Cmd 一共有 5 个命令，命令号为 0~4，分别表示显示全表、增删改查。inst 可变数组存储服务器发往客户端的信息或者客户端发往服务器的操作命令，因数据包传递方向和命令号不同而不同。

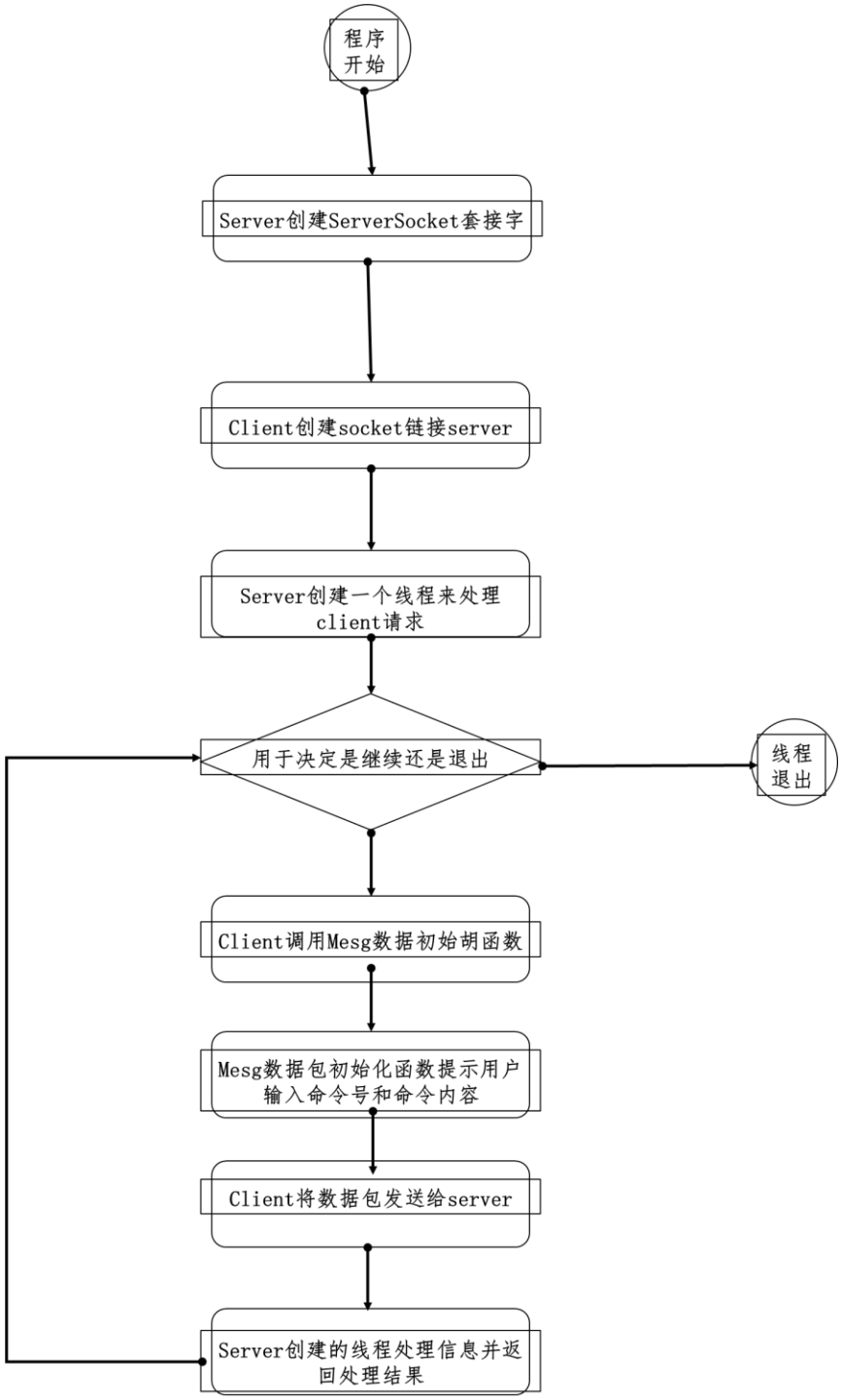
dbtmp 类直接操作数据库，本身定义了数据库的链接功能和各种操作功能的函数。在整个程序中为了接口定义统一，这里将 dbtmp 类的各个函数的参数和返回值都定义为 ArrayList<String>可变数组类。

ClientFrame 类定义了 UI 界面的各个参数及按钮的动作。UI 界面可以显示服务器发往客户端的所有数据。UI 界面实现了界面清空和退出按钮的功能。但是由于时间有限，没有完全实现数据输入功能，所有本程序的数据输入依赖于命令行。

attr 类定义了学生信息管理系统的各个属性。由于存储学生信息管理系统的数据库不在客户端中，所以客户端并不知道数据库的属性数量和属性名，attr 是预先定义好的、存储属性名和属性数量的类，用于辅助客户端提示用户命令输入。

三、 程序具体流程图和主要函数调用关系

1. 程序具体流程图



2. 主要函数调用关系

Client 对 Mesg 函数的调用

```
Sending.ClientToServer(cframe);
```

```
//客户端发送给服务器的数据包初始化
    public void ClientToServer(ClientFrame cf) {
        this.SorC = 0;
        //================
        Scanner input1 = new Scanner(System.in);
        System.out.println("请选择操作类型\n"
                + "0--显示全表\t"
                + "1--插入一行\t"
                + "2--删除一行\t"
                + "3--更新数据\t"
                + "4--查询\t"
                + "其他--退出");
        cf.jtaChat.append(
                "请选择操作类型\n"
                        + "0--显示全表\t"
                        + "1--插入一行\t"
                        + "2--删除一行\t"
                        + "3--更新数据\t"
                        + "4--查询\t"
                        + "其他--退出\n"
                );
        int choice = input1.nextInt();
        ArrayList<String> instruct = new ArrayList<String>();
        attr attrtmp = new attr();
        switch (choice) {
        case 0://显示全表
            instruct.clear();
            break;
        case 1://插入一行
            instruct.clear();
            for(int i = 0; i < attrtmp.getattr().size(); ++i) {
                System.out.println("Input " + attrtmp.getattr().get(i));
                cf.jtaChat.append("Input " + attrtmp.getattr().get(i) + "\n");
                instruct.add(input1.next());
            }
            break;

        case 2://删除数据
            instruct.clear();
            System.out.println("输入你想删除行的行号(以实际标号为准)");
            cf.jtaChat.append("输入你想删除行的行号(以实际标号为准)\n");
            instruct.add(input1.next());
            break;

        case 3://修改数据
            instruct.clear();
            System.out.println("输入你想修改行的行号(以实际标号为准)");
            cf.jtaChat.append("输入你想修改行的行号(以实际标号为准)");
            instruct.add(input1.next());
            System.out.println("输入你想修改的列号(从 1 号开始编号)");
            cf.jtaChat.append("输入你想修改的列号(从 1 号开始编号)\n");
            instruct.add(input1.next());
            System.out.println("输入你想修改后的内容");
            cf.jtaChat.append("输入你想修改后的内容\n");
            instruct.add(input1.next());
            break;

        case 4://查询数据
            instruct.clear();
            System.out.println("输入需要查询的第二列的信息");
            cf.jtaChat.append("输入需要查询的第二列的信息\n");
```

```
            System.out.println("Input the " + attrtmp.getattr().get(1) + " to search for the tuppl
e:");
            cf.jtaChat.append("Input the " + attrtmp.getattr().get(1) + " to search for the tupple
:"+"\n");
            instruct.add(input1.next());
            break;
        default:
            instruct.clear();
            this.SorC = -1;
            break;
        }
        //================
        this.Cmd = choice;
        this.inst = new ArrayList<String>(instruct);
    }
```

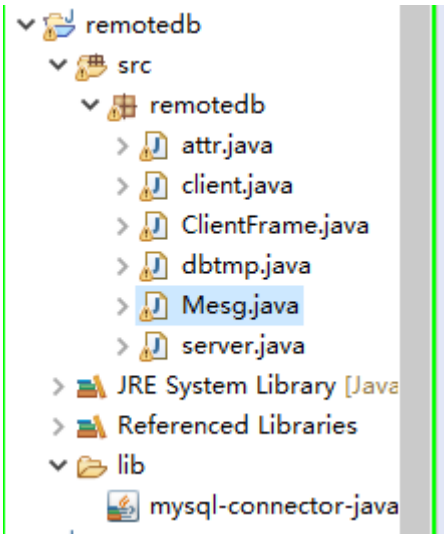
server 对 dbtmp 函数的调用

```
        //接受信息并处理
        ArrayList<String> returnClient = new ArrayList<String>();
        dbtmp tmdbcon = new dbtmp();

        if(instCliMesg.getSorC() == 0) {
            returnClient = tmdbcon.function(instCliMesg.getCmd(), instCliMesg.getinst());
            returnMesg.ServerToClient(returnClient);

            os.writeObject(returnMesg);
            os.flush();
        }
```

dbtmp 自身的函数组织

```
    ArrayList<String> function(int cmd, ArrayList<String> instruction){
        ArrayList<String> outcome = new ArrayList<String>();
        switch(cmd) {
        case 0://显示全表
            outcome = disp();
            break;
        case 1://插入数据
            outcome = insert(instruction);
            break;
        case 2://删除数据
            outcome = delete(instruction);
            break;
        case 3://修改数据
            outcome = update(instruction);
            break;
        case 4://查询数据
            outcome = select(instruction);
            break;
        default:
            outcome.add("error command");
            break;
        }
        return outcome;
    }
```

四、 文件列表

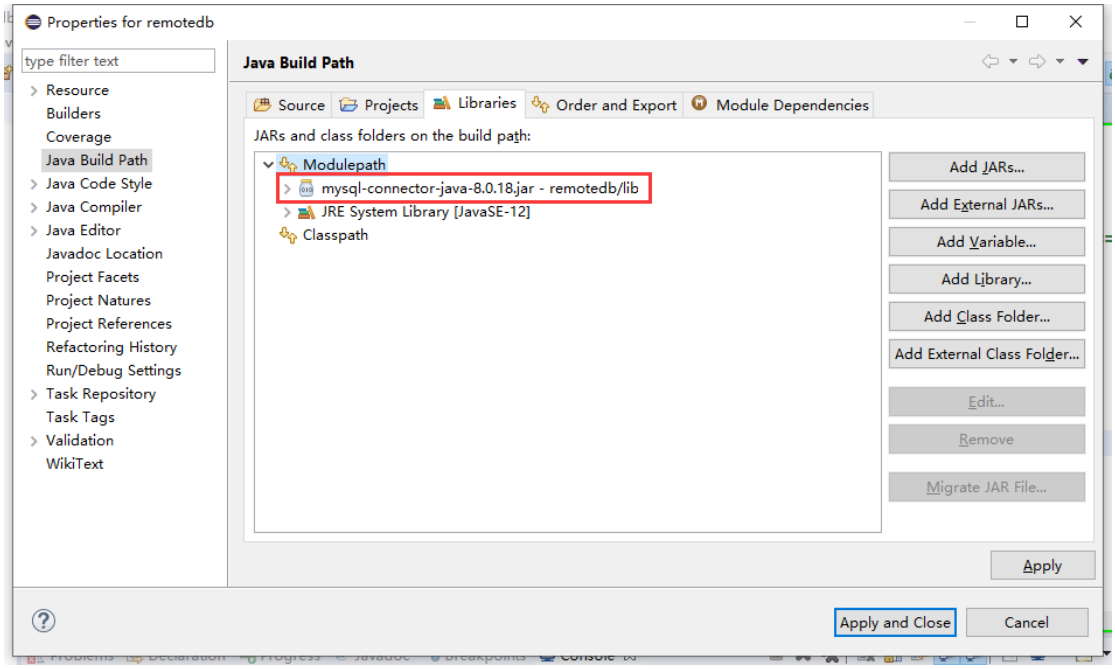


五、 项目所用到的模式

本程序需要用到数据库和网络，所以客户端和服务器必须处于网络连接状态，且服务器上必须装有 mysql 数据库，在程序运行时，保持 mysql 数据库处于开启状态。由于不同数据库的账号密码不同，在迁移程序是，需要及时调整数据库的账号和密码。

1. 程序预先配置

JDBC 的操作需要电脑端安装 mysql 软件，并且需要提供 Java connector 接口。此外，在程序中需要导入 mysql-connector-java-8.0.18.jar 的程序包。并且需要在工程文件的 Java Build Path 中导入该程序包



六、 难点、要点、得意点

1. TCP 连接

服务器端首先创建一个 ServerSocket 套接字，通过 ServerSocket 的 accept 函数与客户端的 Socket 建立 TCP 连接。

建立连接后，通过 Socket 的 getInputStream 和 getOutStream 流类函数建立数据连接，以 Mesg 为数据包格式进行通信。

TCP 连接部分的主要代码如下：

服务器：

```
    // 监听8000端口
    ServerSocket server=new ServerSocket(8000);
    while(true)
    {
        //接收客户端的连接
        Socket socket=server.accept();
        //调用客户端的数据处理方法
        invoke(socket);
    }
}
```

客户端：

```
Socket socket=null;
try {
    socket=new Socket("localhost",8000);
}
catch (Exception e) {
    e.printStackTrace();
}

ObjectOutputStream os=null;
ObjectInputStream is=null;
```

2. 服务器多线程的创建

服务器主进程循环检查 ServerSocket 端口有无新的客户端连接，没当检查到新的客户端连接，就调用服务器类定义的 invoke 函数处理该客户端的请求。invoke 通过开启一个线程，与客户端通信。

主要代码如下：

```
            Socket socket=server.accept();
            //调用客户端的数据处理方法
            invoke(socket);
        }
    }
    private static void invoke(final Socket socket) throws IOException
    {
        //开启一个新线程
        new Thread(new Runnable()
        {
            public void run()
            {
                //创建输入流对象
                ObjectInputStream is=null;
                //创建输出流对象
                ObjectOutputStream os=null;
                try
                {
                    is=new ObjectInputStream(socket.getInputStream());
                    os=new ObjectOutputStream(socket.getOutputStream());
                    //读取一个对象
                    Object obj = is.readObject();
```

3. 数据库和操作

数据库连接和操作的功能函数都在 dbtmp 中定义。

dbtmp.java 分为四个部分。第一个是成员变量定义区，该部分因数据表选择不同而不同。第二部分是成员变量赋值区，该部分根据数据表列属性信息初始化部分成员变量。第三部分是操作函数定义区，总共定义了显示函数 disp()、插入函数 insert()、删除函数 delete()、修改函数 update 和查询函数 select()。最后一部分是主函数区，主函数区在 testdb 调用时并没有使用到，它时用于单独测试 dbtmp.java 的。

4. Mesg 的数据格式定义及初始化函数

Mesg 除了具有数据包格式定义的功能外，还具有数据初始化的功能函数，尤其是在客户端发往服务器时，需要根据命令号决定不同的内容命令。这个时候 Mesg 类的 ClientToServer 函数就起到的提示和分类作用。事实上，Mesg 是具有数据包格式定义和数据预处理两个功能的类。

Mesg 作为服务器和客户端之间通信的数据包格式的具体定义如

下。

```
//该类定义数据包格式
public class Mesg implements java.io.Serializable{
    private int SorC;//发送方是服务器还是客户端
    private int Cmd;//客户端发送的命令号
    private ArrayList<String> inst;//客户端发送的命令内容，或者，服务器发送的回复内容

    //客户端发送给服务器的数据包初始化
    public void ClientToServer(ClientFrame cf) {
        this.SorC = 0;

     //若干步骤
     //================

        this.Cmd = choice;
        this.inst = new ArrayList<String>(instruct);
    }

    //服务器发送个客户端的数据包初始化
    public void ServerToClient(ArrayList<String> con) {
        this.SorC = 1;
        this.Cmd = -1;
        this.inst = con;
    }

    public ArrayList<String> getinst(){
        return this.inst;
    }

    public void Print(ClientFrame cf) {//用于客户端打印数据包内容
        for(int i = 0; i < inst.size(); ++i) {
            System.out.println(inst.get(i));
            cf.jtaChat.append(inst.get(i) + "\n");
        }
        return;
    }

    public int getSorC() {
        return this.SorC;
    }

    public int getCmd() {
        return this.Cmd;
    }
}
```

5. ClientFrame 的 UI 设计

相对来说，本程序的 UI 设计较为简单，主要出于时间不足的原因。但是对于 UI 界面的操作定义十分完备。ClientFrame 完整的定义了用户界面的大小、位置、字体和格式，以及说明 UI 界面不能由右上角的关闭键关闭，这是因为突然的关闭将造成客户端和服务器的议程，给系统带来不稳定因素。

ClientFrame 类中定义了客户端的唯一合法出口，及“退出”按钮。

此外客户端实现了文字显示和清屏功能。

客户端 UI 界面效果如下图。

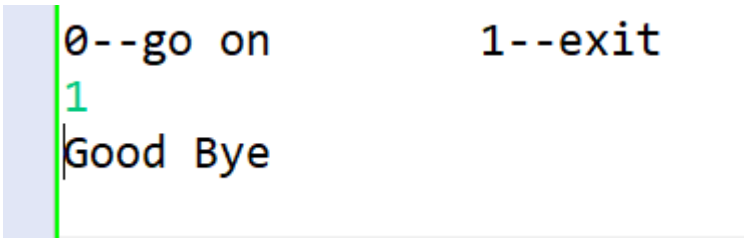
七、 程序使用说明

1. 整体命令

程序将循环提示用户是否要继续使用客户端并给出了这两个方向的操作命令。

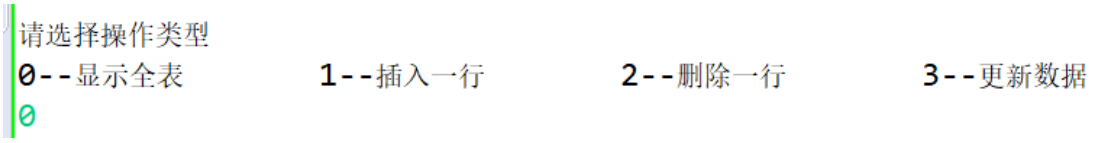
2. 数据库操作命令

如果选择 1。则客户单运行终止，但不退出，需要用户按下退出按钮后客户端才完全退出。



如果不退出，则进入数据库操作界面。客户端会提示用户输入响应的命令号来操作数据库，随后客户端会根据 Mesg 的数据初始化要求提示用户输入响应的命令内容。

选择 0 号操作命令，显示全表。

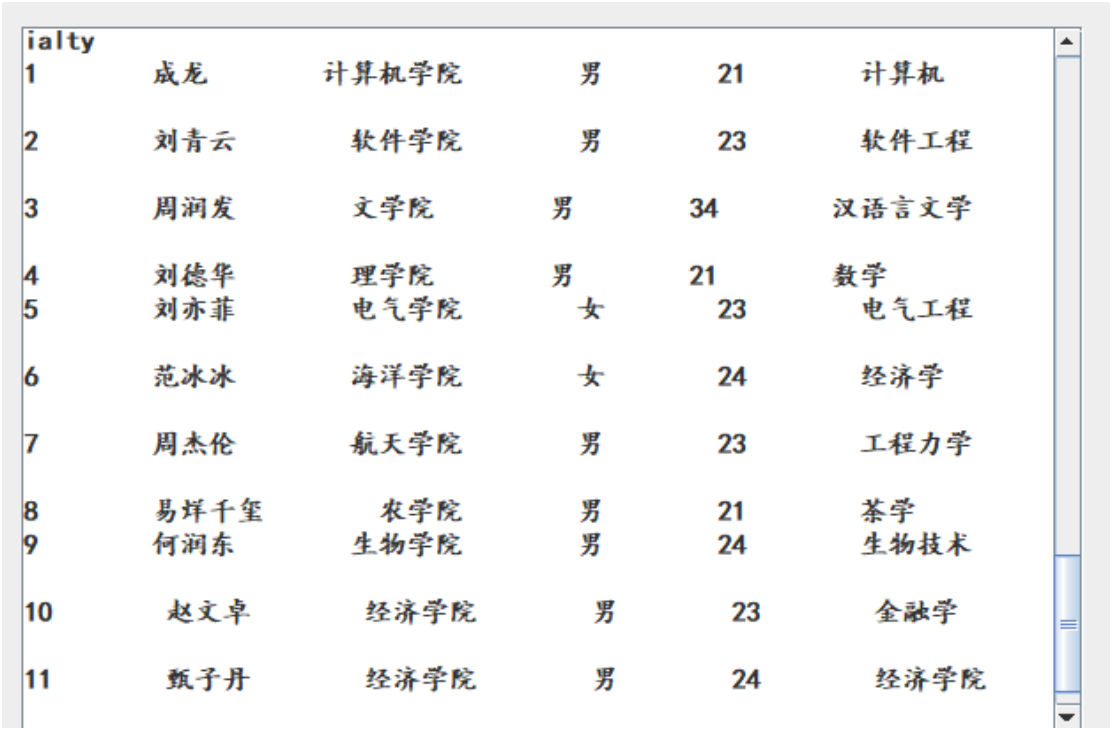


1	成龙	计算机学院	男	21	计算机
2	刘青云	软件学院	男	23	软件工程
3	周润发	文学院	男	34	汉语言文学
4	刘德华	理学院	男	21	数学
5	刘亦菲	电气学院	女	23	电气工程
6	范冰冰	海洋学院	女	24	经济学
7	周杰伦	航天学院	男	23	工程力学
8	易烊千玺	农学院	男	21	茶学
9	何润东	生物学院	男	24	生物技术
10	赵文卓	经济学院	男	23	金融学

选择 1 号操作命令插入一行数据。

```
请选择操作类型
0--显示全表        1--插入一行        2--删除一行        3--更新数据        4--查询    其他--退出
1
Input stuid
11
Input name
甄子丹
Input department
经济学院
Input sender
男
Input age
24
Input specialty
经济学院
Insert Successfully
```

插入操作后显示全表

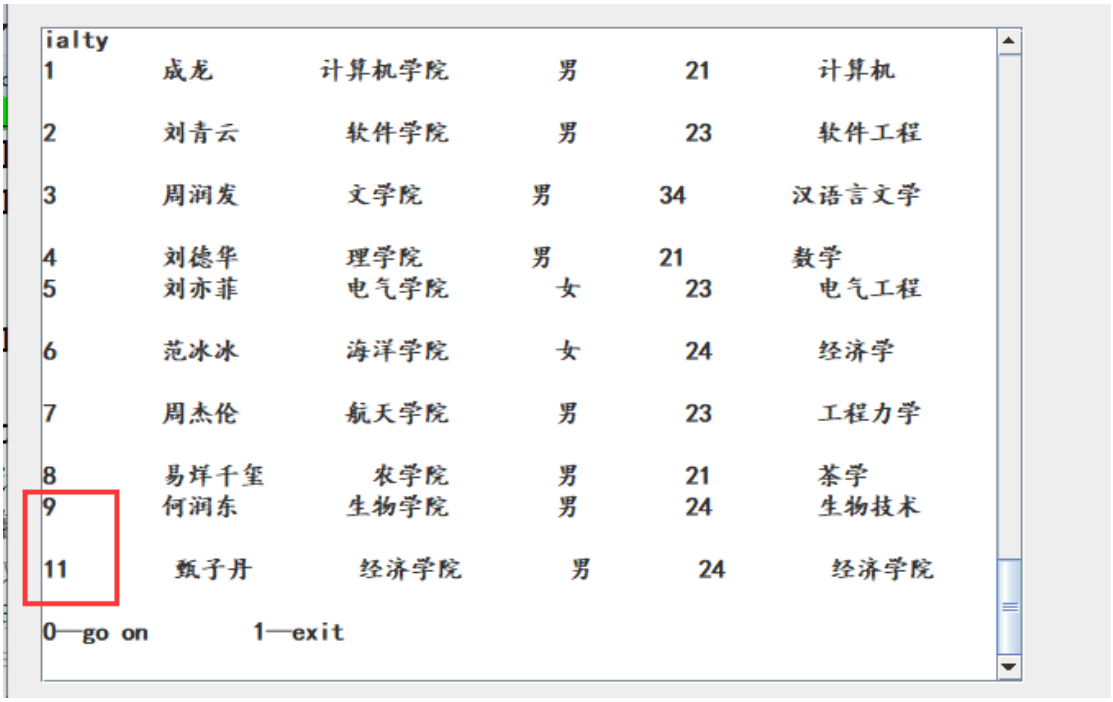


ialty					
1	成龙	计算机学院	男	21	计算机
2	刘青云	软件学院	男	23	软件工程
3	周润发	文学院	男	34	汉语言文学
4	刘德华	理学院	男	21	数学
5	刘亦菲	电气学院	女	23	电气工程
6	范冰冰	海洋学院	女	24	经济学
7	周杰伦	航天学院	男	23	工程力学
8	易烊千玺	农学院	男	21	茶学
9	何润东	生物学院	男	24	生物技术
10	赵文卓	经济学院	男	23	金融学
11	甄子丹	经济学院	男	24	经济学院

选择 2 号命令删除一行数据。

```
请选择操作类型
0--显示全表        1--插入一行        2--删除一行        3--更新数据        4--查询    其他--退出
2
输入你想删除行的行号(以实际标号为准)
10
delete Successfully
0--go on        1--exit
```

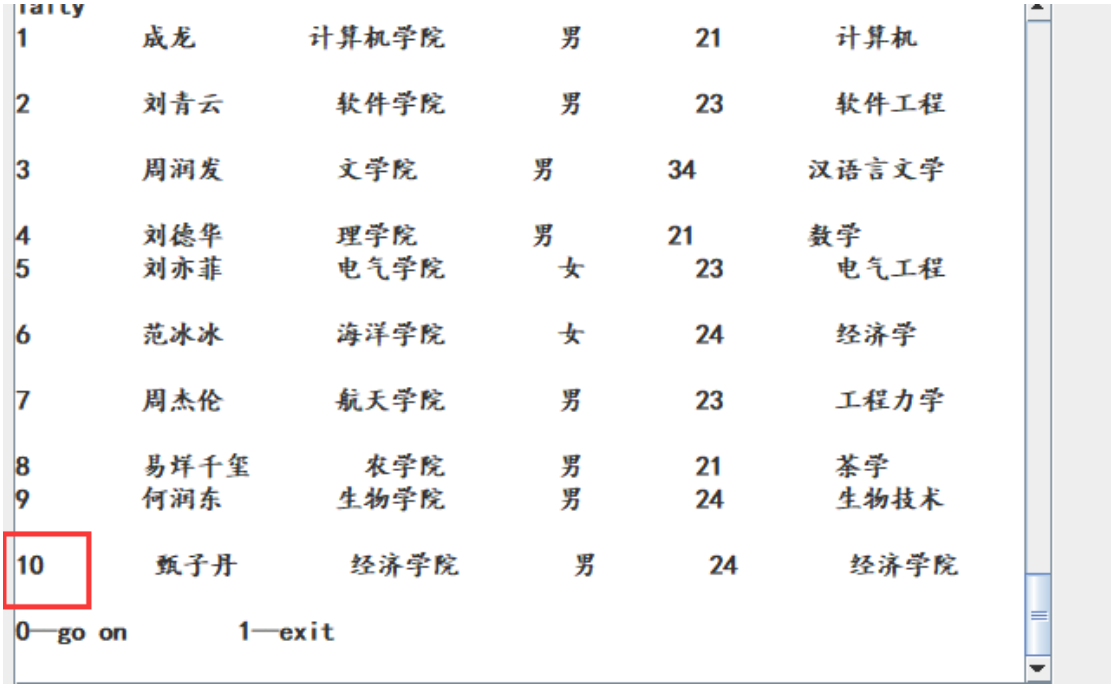
删除操作后显示全表。



选择 3 号命令修改一条数据

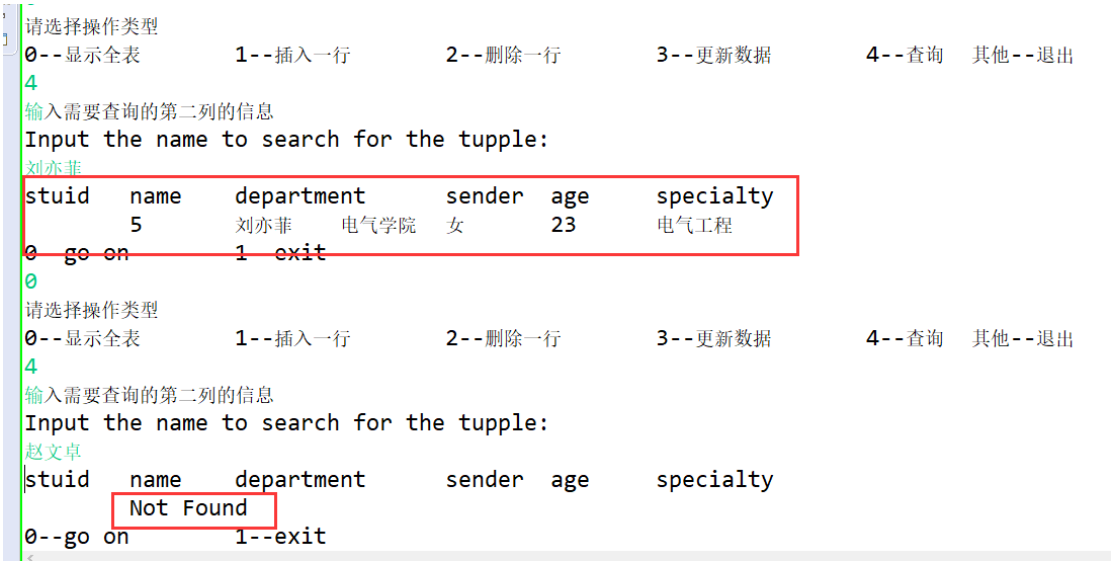
```
请选择操作类型
0--显示全表        1--插入一行        2--删除一行        3--更新数据        4--查询    其他--退出
3
输入你想修改行的行号(以实际标号为准)
11
输入你想修改的列号(从1号开始编号)
1
输入你想修改后的内容
10
update Successfully
0--go on        1--exit
```

修改后显示全表



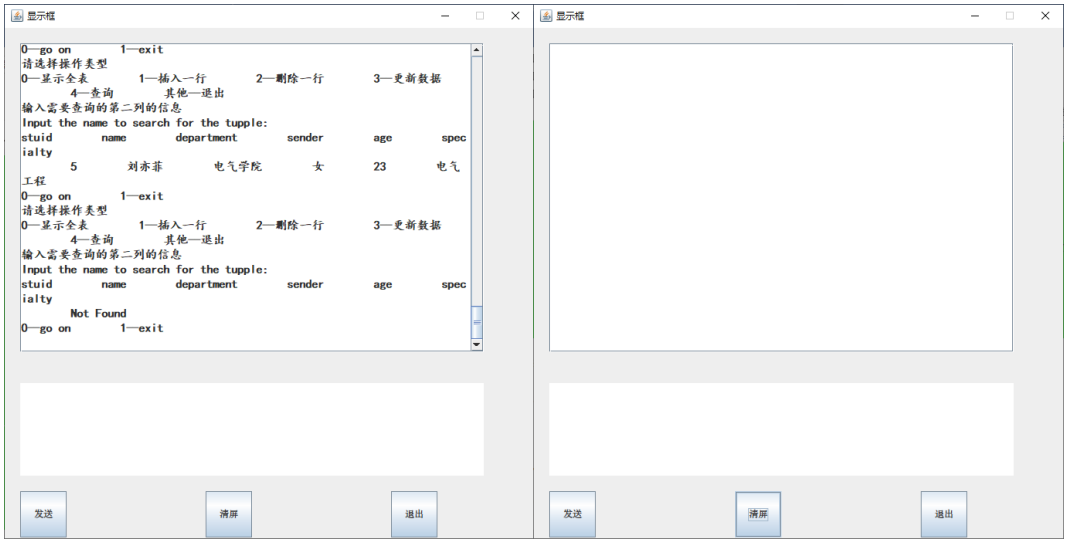
选择 4 号命令查询

程序会根据数据库中是否存在需要查找的数据给出响应的提示。



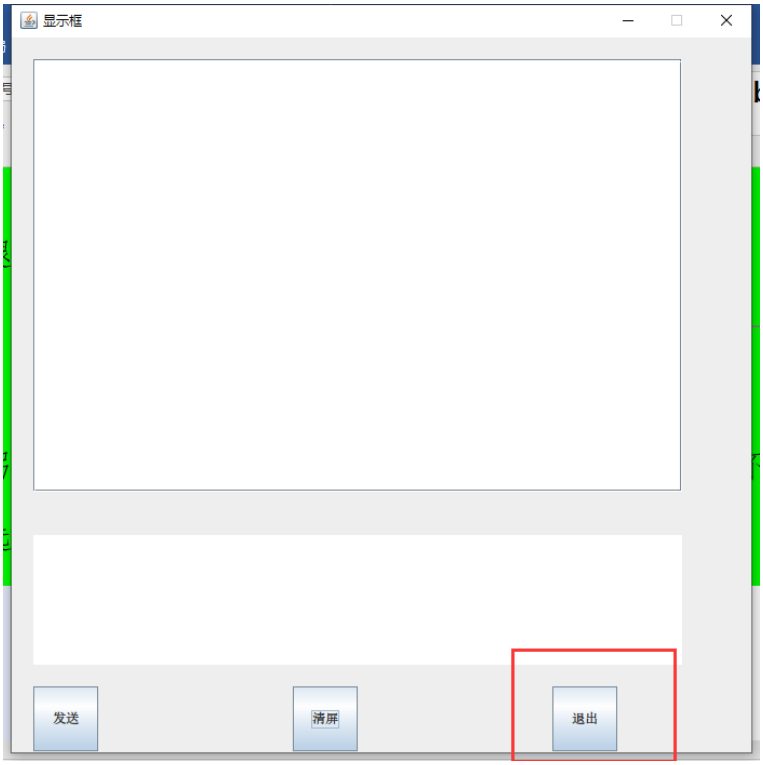
3. UI 按钮

清屏按钮



退出按钮

按下退出按钮，客户端进程结束。UI 界面销毁。



八、 结论展望

1. 结论

对于信息系统的管理归根到底是对数据库的操作。之前的实验做过相关的对于数据库的操作，所以这次大程在数据库操作方面会稍微熟悉一些。

本程序对于多线程的调用采用了 runnable 的接口。通过实现 Runnable 接口创建线程时，首先需要编写一个实现 Runnable 接口的类，然后实例化该类的对象，这样就创建了 Runnable 对象。接下来使用相应的构造方法创建 Thread 实例。最后使用该实例调用 Thread 类的 start()方法启动线程，如下图所示。

2. 展望

1) jsp 链接数据库的实现

其实在程序设计之初，我就有用 jsp 连接数据库的想法。因为 jsp 天生就面向网络和 Java，并且在 UI 显示具有非常大的灵活性。自己也看了相关的一些教程和教材，但是在实际操作中遇到了一个暂时没有逾越的障碍——jsp 无法连接数据库。这个动作的实现非常容易，与 Java 类本身连接数据库几乎一致，但我总是链接失败，我猜测可能是我 Tomcat 环境配置有误，才造成了这个错误，同样由于时间有限我没能深究，以后要仔细找出这个 bug。

2) UI 界面的交互

UI 界面本来就是用来人机交互的，但是在信息传递方面，本程

序只实现了机器对人的信息传递，没有实现人对机器的信息操作，即没有实现UI 界面的发送消息按钮功能。事实上，从客户端的UI 界面发送信息给服务器时我可以实现的，但是，在客户端获取命令的操作过程中同样需要UI 界面的输入，这个输入设计按钮定义方法和其他类之间的通信，由于时间有限，没有找到相关解决方法，只能等日后完善了。

3) 图片的插入

如果能是 jsp 或者UI 界面使用图片，无疑会使程序锦上添花。日后有时间要琢磨如何给 UI 界面插入图片。