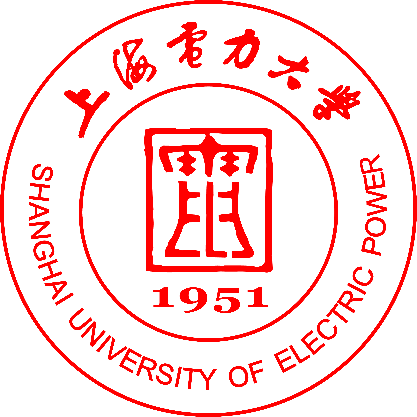
**上海电力大学**

**数据结构（JAVA）课程设计**



**题　　目:**  实验19

　　　校园导游咨询

**姓 名：** 洪勇

**学 号：** 20181389

**院　　系：**　　 计算机科学与技术学院

**专业年级：**　 本科二年级

**2020年　 1 月　 13 日**

[一、 设计题目 2](#_Toc29900432)

[二、 需求分析 3](#_Toc29900433)

[1） 运行环境（软、硬件环境） 3](#_Toc29900434)

[2）输入的形式和输入值的范围 4](#_Toc29900435)

[3）输出的形式描述 4](#_Toc29900436)

[4）功能描述 4](#_Toc29900437)

[5）测试数据 4](#_Toc29900438)

[二、 概要设计 5](#_Toc29900439)

[1） 抽象数据类型定义描述 5](#_Toc29900440)

[2）功能模块设计（如主程序模块设计） 14](#_Toc29900441)

[3）模块层次调用关系图 15](#_Toc29900442)

[四、详细设计 15](#_Toc29900443)

[最短路径算法：shortestPath类实现 15](#_Toc29900444)

[五、调试分析 18](#_Toc29900445)

[Hash映射 18](#_Toc29900446)

[鼠标点击偏移修正 19](#_Toc29900447)

[数据库调用中文时出现乱码 20](#_Toc29900448)

[最短路径算法 20](#_Toc29900449)

[六、测试结果 21](#_Toc29900450)

[登陆界面 21](#_Toc29900451)

[导游界面 23](#_Toc29900452)

[修改信息界面 24](#_Toc29900453)

[七 、附录：程序设计源代码 28](#_Toc29900454)

[welcomWindow.class: 28](#_Toc29900455)

[SwingConsole.class: 34](#_Toc29900456)

[mainWin.class: 34](#_Toc29900457)

[administerWin.class: 40](#_Toc29900458)

[allPointData.class: 44](#_Toc29900459)

[connectSQL.class: 45](#_Toc29900460)

[ensureViewPointData.class: 54](#_Toc29900461)

[pointInformation.class: 55](#_Toc29900462)

[pointRowData.class: 56](#_Toc29900463)

[returnViewPointMessage.class: 57](#_Toc29900464)

[txtMessage.class: 58](#_Toc29900465)

[viewMessage.class: 59](#_Toc29900466)

[shortestPath.class: 60](#_Toc29900467)

## 设计题目

【问题描述】

设计一个校园导游程序，为来访的客人提供各种信息查询服务。

【基本要求】

(1)设计你所在学校的校园平面图，所含景点不少于10个。以图中顶点表示校内各景点，存放景点名称、代号、简介等信息；以边表示路径，存放路径长度等相关信息。

(2)为来访客人提供图中任意景点相关信息的查询。

(3)为来访客人提供图中任意景点的问路查询，即查询任意两个景点之间的一个最短的简单路径。

【测试数据】

由读者根据实际情况指定。

【实现提示】

一般情况下，校园的道路是双向通行的，可设校园平面图是一个无向网。顶点和边均含有相关信息。

【选作内容】

(1)求校园图的关节点。

(2)提供图中任意景点问路查询，即求任意两个景点之间的所有路径。

(3)提供校园图中多个景点的最佳访问路线查询，即求途经这多个景点的最佳(短)路径。

(4)校园导游图的景点和道路的修改扩充功能。

(5)扩充道路信息，如道路类别(车道、人行道等)、沿途景色等级，以至可按客人所需分别查询人行路径或车行路径或观景路径等。

(6)扩充每个景点的邻接景点的方向等信息，使得路径查询结果能提供详尽的导向信息。

(7) 实现校园导游图的仿真界面。

## 二、 需求分析

### 运行环境（软、硬件环境）

1. 硬件：
   * + 1. Processor: Intel(R) Core(TM) i5-8300H CPU @ 2.30GHz (8 CPUs), ~2.3GHz
       2. Memory: 8192MB RAM
2. 软件：
   * + 1. Window10 家庭版
       2. Mysql 数据库
       3. IntelliJ IDEA Educational Edition 2019.3 x64

### 2）输入的形式和输入值的范围

由管理员输入景点信息。管理员直接操作数据库添加景点，更改景点具体信息可以从界面UI上操作。而使用用户能通过点击界面来获取景点信息，以及得到具体的最短路线。

### 3）输出的形式描述

用户通过点击，可以在对话框中可以得到输出信息，如景点名字，景点介绍和路线选择，在地图中获取相应的最短路径。

### 4）功能描述

用户通过本系统，查询本校的景点的具体信息，查询两个景点之间的最短路径，多个景点之间的最短路径以及将路径可视化。

### 5）测试数据

数据库操作：

insert into viewMessage values('1047','101','特来电充电站','这个地方可以充电');

insert into viewMessage values('919','150','上海电力大学临港校区留学生教师宿舍','这是给高贵的留学生住的宿舍');

insert into viewMessage values('884','348','上海电力大学临港校区第一食堂','这里是第一食堂，可以看我们吃饭');

……类似的其他数据

update viewMessage SET ind=1 where pointY='101';

update viewMessage SET ind=2 where pointY='150';

update viewMessage SET ind=3 where pointY='348';

……其他的类似数据

insert into graphTable values('1','11');

insert into graphTable values('1','3');

insert into graphTable values('11','12');

……其他的类似数据

## 概要设计

### 抽象数据类型定义描述

* + 1. 管理员界面设计

该类是为了显示管理员GUI的类。因为用户不能随便对景点进行更改，因此特别重写了一个管理员界面类，用于管理景点和管理可登入账户的信息。

public class administerWin extends JFrame {

private JPanel base;//主界面画布

private JPanel Main;//主界面画布

private JMenuBar tool;//菜单栏

private final int VIEW=0,NUMBER=1;//用于区分景点功能和账户功能

private final int ADDVIEW=0,CHANGVIEW=1,DELETEVIEW=2;//用于区分增加景点，修改景点信息，删除景点信息功能的子选项

private final int ADDNUMBER=3,CHANGELOGINPASSWORD=5,DELETENUMBER=4;//用于区分增加账户，修改密码，删除账户的子选项

private boolean flag=true;//图片开始接受点击的开关

private connectSQL data=new connectSQL();//链接数据库

private JMenu[] menus;//菜单选项类

private JLabel graph;//用来放地图

public administerWin(){//构造方法，将管理员界面建立起来，里面有几个子方法构成

init();//初始化所有组件

setLayout();//设定画布的布局

setItemsListener();//菜单功能的子选项添加监听器

setItems();//将菜单功能子选项加进菜单中

setMenuBar();//往窗体里添加菜单栏

setGraph();//给Label设置地图

setGraphListener();//给地图添加监听器

Main.add(base);//把画布加进主画布

add(BorderLayout.CENTER,Main);//把主画布放进窗体里

}

public void init();

public void setLayout() ;

public void setItemsListener();

public void setItems();

public void setMenuBar() ;

public void setGraph();

public void setGraphListener();

public static void main(String args[]);

public String getUser();//在输入对话框中获取用户名

public String getPassword();//在输入对话框中获取密码

}

* + 1. 导游主界面设计

这边是导游界面的主要设计。里面主要实现：

* + - 1. 将地图显示出来
      2. 点击景点时显示景点的具体信息
      3. 用户点击景点后要在景点显示一面旗子，表示已经被选中
      4. 当用户选择好点之后，将最短路径描绘出来
      5. 能够清除界面上的最短路径

下面时该类的抽象数据描述。

public class mainWin extends JFrame {

private Point Temp=new Point();

private boolean watchFlag =false;//景点展示功能的开关

private int clickCount=0, modern =0;//鼠标点击计数器和模式选择器

private final int OFF=0,RESET=0;//开关关闭和计数器置0

private final int DISPLAYMODERNON=0,DISPLAYMODERNOFF=1,DISPALYMODERN=0;//展示模式的状态

private final int GUIDE=1,TWOPOINTMODERN=2,GREATPOINTMODERN=3;//导游模式的启动标识

private final int CLEAR=2,CLEARMODERNON=4;//清空路线的启动标识

private Vector<Point> set=new Vector();//小黄旗出现的坐标容器

private Vector<Point> line=new Vector();//描绘路线的的坐标容器

private shortestPath path=new shortestPath();//最短路径算法类的初始化

private JPanel basic=new JPanel();//主画布初始化

private JPanel menu=new JPanel();//菜单画布初始化

private allPointData data=new allPointData();//所有数据的集合类

private ensureViewPointData temp=new ensureViewPointData();//确认是否点在了景点上的类

private JPanel Graph=new JPanel(){//重写JPanel里的画图方法，为了将路径可视化

private Vector<Point> a=new Vector();

public void setLocation(Point e);

@Override

public void paintComponents(Graphics g);

@Override

public void paintAll(Graphics g) ;

};

private JLabel map=new JLabel();//存放地图的Label

private JMenuBar menuBar=new JMenuBar();//菜单栏初始化

private JMenu Menu[];//菜单选项

public mainWin(){//建立导游主窗体的构造方法，由下面的子方法构成

setLayout();//设定布局方式

setMenuItem();//设定菜单功能子选项

addMenuItemListener();//为菜单功能子选项添加监听器

setMenuBar();//设定好菜单栏

setGraph();//放地图在Label上

setJMenuBar(menuBar);//将菜单栏放在窗体上

basic.add(Graph);//将地图放进画布

add(BorderLayout.CENTER,basic);//将画布放在窗体中

};

public void clear();//清理地图上的显示的路径，并会提示已清除

public void clean();//清理地图上的路径，不会提示已清除

public void cleanCounter();//在显示路径后定时一秒，询问用户是否清除路径

public void messageOfView(MouseEvent e);//展示用户所点击的景点的具体信息

public void clickInPoint(MouseEvent e);//判断用户是否点击在景点上

public void disPlaySimplyPath();//描绘两个景点的的最短路径

public void disPlayablePath();//描绘多个景点的最短路径

public void setMenuItem();

public void addMenuItemListener(){//菜单功能监听器

item[DISPLAYMODERNON].addActionListener(new ActionListener() {

@Override//景点展示模式打开功能监听器

public void actionPerformed(ActionEvent e) ;

});

item[DISPLAYMODERNOFF].addActionListener(new ActionListener() {

@Override//景点展示模式关闭功能监听器

public void actionPerformed(ActionEvent e) ;

});

item[TWOPOINTMODERN].addActionListener(new ActionListener() {

@Override//两个景点的最短路径展示的监听器

public void actionPerformed(ActionEvent e) ;

});

item[GREATPOINTMODERN].addActionListener(new ActionListener() {

@Override//多个景点的最短路径展示的监听器

public void actionPerformed(ActionEvent e) ;

});

item[CLEARMODERNON].addActionListener(new ActionListener() {

@Override//清除屏幕描绘的路径的功能的监听器

public void actionPerformed(ActionEvent e) ;

});

}

public void setMenuBar();

public void setLayout();

public void setGraph(){

map.setIcon(new ImageIcon(getClass().getResource("map.png")));

Graph.add(map);

Graph.addMouseListener(new MouseAdapter() {

@Override//给地图建立一个点击监听器，一旦发生点击，就做出相应操作

public void mouseClicked(MouseEvent e) ;

});

}

public static void main(String afgs[]);//窗体建立

}

public class SwingConsole {

public static void run(final JFrame f,final int width,final int height,String title){

SwingUtilities.invokeLater(new Runnable());//建立窗体的模板，将任务提交到线程进行

}

}

* + 1. 连接SQL数据库操作

该类的操作时连接数据库进行操作。因为链接数据库后，数据库对要调用的数据的处理比较方便友好，java调用也比较方便，可以很好地处理要用的数据，因此采用连接数据库处理数据。

而下面就是对需要导出的数据进行处理的方法。该类的方法可以很好的得到我期望得到的数据，便于处理。

public class connectSQL {

private int cap=0;//在图中的所有点数

private int viewCap=0;//在图中的景点数

private String driverName="com.mysql.jdbc.Driver";//加载驱动命令

private String url="jdbc:mysql://127.0.0.1:3306/viewPoint?characterEncoding=UTF-8";//连接到viewPoint数据库，并以UTF-8进行编码

private String user="root",password="123456";//数据库的用户名和密码

private String selectMessageCommand="select \* from pointMessage";//选择表命令

private String getSelectMessageTable="select \* from viewMessage";//选择表命令

private String getUserMessage="select \* from user where userName=";

private String getGraphTable="select \* from graphTable";

private String getRotutePointMessageTable="select \* from rotutePointMessage";

private String insertCommand="insert into pointMessage values";//插入数据命令

private String insertCommandIntoRotuteMessageTable="insert into rotutePointMessage values";//插入路径结点命令

private String insertUserMessage="insert into user values";

private String deleteUser="delete from user where userName=";//删除数据命令

private Connection con;

private Statement cmd;//建立SQL连接的模板

private Vector<pointRowData> sentMessage=new Vector();

private Vector<viewMessage> sentViewMessage=new Vector();//接受信息的容器

private int clickCounter=0;//点击计数器

private int viewPointNumber=0,allPointNumber=0;

public void initClickCounter();//初始化计数器

public void conSQL();//连接数据库的模板函数

public void builtCon();//连接数据库的模板函数

//Raw click point data

private Vector<pointRowData> getAllRowMessage();//获取所有景点点击数据

public Vector<pointRowData> loadData();//加载上述数据

//View point message , such as name of view , read in china

private Vector<viewMessage> getAllViewMessage();//获取所有景点的文本数据

public Vector<viewMessage> loadViewMessage();}//加载上述数据

//Register raw data in SQL

public int tableLimit();//表长的最大值

public void addViewPointPosition(Point click);//添加景点的点击位置

public void addRotePointPostion(Point click);//添加路径结点的位置

public void changeViewPointMessage(Point click);//更改景点的文本信息

//To get AllPointData part

private Vector<pointInformation> getViewPointData();//获取景点的位置数据

private Vector<pointInformation> getRotutePointData();//获取路径结点位置的数据

private Vector<pointInformation> combineData();//合并上面获得的位置数据

public Vector<pointInformation> returnAllPointData();//将数据抛出

//To get main algorithm of data

public double[][] getGraphData();//获取距离联通图数据

private int[][] getViewPointLinkData();//获取景点联通矩阵

private int[][] getRotutePointLinkData();//获取路径点联通矩阵

private int[][] combineToGraph(int[][] e, int[][] g);//将两个联通矩阵合并

private double[][] initDisArray();//初始化距离矩阵

private double[][] generateDisArray(double[][] dis,int[][] e);//将联通矩阵转化成距离矩阵

//Login Message

public int insureUser(String user,String password);//确定是否是已经录入的用户

public void addUser(String user,String password);//添加用户

public void changePassword(String user,String password);//更改密码

public void deleteUser(String user,String password);//删除用户

public int getViewPointNumber();//获取景点的数量

public int getAllPointNumber();//获取所有结点的数量

public connectSQL();

}

public class ensureViewPointData {

private Map<Point,Point> hashTableToEnsureViewPoint=new HashMap();//确认对应的景点的哈希表

public ensureViewPointData();//构造

public Point realViewPoint(Point clickPosition);//返回鼠标点击应该对应的景点位置

private Vector<pointRowData> getAllPointMessage();//获取收集到的所有的点击数据

private Vector<Point> getClickPoint(Vector<pointRowData> allData);//获取可能会点击到的数据

private Vector<Point> getViewPoint(Vector<pointRowData> allData);//获取应该唯一对应的点击到的景点

private void setConnect(Vector<Point> clickPoint,Vector<Point> viewPoint);//以可能会点击的位置作为键值，实际上的唯一对应为实际会到的点

private void builtHashTable();//建立这样的hashTable

}

* + 1. 坐标信息收集数据类型

该类型是为了接收收集之后处理得到的数据，方便后来的处理。

public class pointInformation {

private Point position;//位置

private int index;//序号

private int type;//类型

private final int view=1;//景点类型为1

private final int rotute=2;//路径结点类型为2

public pointInformation();//初始化

public void setPosition(Point position);//设置位置

public void setIndex(int index);//设置序号

public void setType(int type);//设置类型

public Point getPosition();//获取位置坐标

public int getIndex();//获取序号

public int getType();//获取类型

public boolean isRotute();//判断是否是路径

public boolean isView();//判断是否是景点

}

* + 1. 原始点击数据操作类

该类是为了存储用户点击之后可能出现的所有情况。为了后面处理成hash映射而准备的类。

public class pointRowData {

private int clickPointX,clickPointY;//可能会点到的位置

private int viewPointX,viewPointY;//点到的点对应的唯一正确的位置

public pointRowData();//构造

public void setClickPointX(int x);//设定点击的位置为x

public void setClickPointY(int y);//设定点击的位置为y

public void setViewPointX(int x);//设定与其对应的正确的位置

public void setViewPointY(int y);//设定与其对应的正确的位置

public int getClickPointX();//获取点击的位置的x坐标

public int getClickPointY();//获取点击的位置的y坐标

public int getViewPointX();//获取唯一对应的景点位置的x坐标

public int getViewPointY();//获取唯一对应的景点位置的y坐标

}

* + 1. 返回景点具体信息的操作类

该类是用hash实现复杂度为1的景点信息调用，提高景点信息的调用速率，提高数据调用效率的一个类。

public class returnViewPointMessage {

private Map<Point,txtMessage> hashTabelViewMessage=new HashMap();//建立景点与文本信息的hashTable

private Vector<viewMessage> getAllData();//获取所有点的数据

private Vector<Point> getPointMessage(Vector<viewMessage> allData);//获取点的数据

private Vector<txtMessage> getTxtMessage(Vector<viewMessage> allData);//获取点的文本信息

private void buildHashTable();//用上述数据建立哈希映射

public returnViewPointMessage();//构造方法

public txtMessage getViewPointMessage(Point click);//返回坐标点对应的文本信息

}

* + 1. 景点的具体信息存储类

该类主要存贮的是景点的名字和具体介绍。这是为了方便前面数据库处理专门写的一个类，让程序条理更清楚的，不会处理地太乱。

public class txtMessage {

private String name;//景点的名字

private String txt;//景点介绍

public void setName(String name);//设置景点的名字

public void setTxt(String txt);//设置景点的文本信息

public String getName();//获取景点名字

public String getTxt();//获取文本信息

}

* + 1. 包含坐标的所有景点信息类

这里面包含的是所有点的信息，数据库连接之后是直接利用这个数据类进行存储，然后再区分成各种子类，方便后面的数据处理。

public class viewMessage {

private int pointX;

private int pointY;

private String viewName;

private String txt;

public void setPointX(int x);//设置景点的x坐标

public void setPointY(int y);//设置景点的y坐标

public int getPointX();//获取x坐标

public int getPointY();//获取y坐标

public void setViewName(String name);//设置景点名字

public void setTxt(String txt);//设置景点信息

public String getViewName();//获取景点名字

public String getTxt();//获取景点的文本信息

}

* + 1. 最短路径算法操作类

这块是算法的核心。由于下面会仔细描述该算法的运作方法，因此不在这里过多赘述。

public class shortestPath {

private double e[][];//距离矩阵

private connectSQL dataCon=new connectSQL();//与数据库建立连接

private int cap=0,viewPointCap=0;//点数多少

private final double inf=0x3fffffff;//无限大初始化

private int[] pre;//pre数组指向前一个位置

private boolean[] vis;//记录历遍数组

private double[] dis;//记录距离的数组

private allPointData getPoint;//获取点的数据

private returnViewPointMessage getViewMessage=new returnViewPointMessage();

Vector<String> allMessage=new Vector();

public shortestPath();//构造方法

public void init();//初始化距离数组等一系列在dijkstra要用到的数据

public void Dijkstra(int index);//算一遍

public Vector<Point> shortPath(int start,int end);//获取一对景点的最短路径的数据

private Vector<String > returnViewName(Vector<Point> path);//将经过的点的名字收集起来

private Vector<Point> linkPath(int start,int end);//将经过的点全部收集起来

public Vector<Point> shortPathInGreatPoint(Vector<Integer> pointIndex);//多个景点的最短路径算法

public void toPrintMessage();//将路径的转法打印出来

}

### 2）功能模块设计（如主程序模块设计）

本次实现模块分三部分：GUI界面，数据库连接处理，算法生成数据

本次实验主要是对最短路径算法的实践，此模块是本次实验的核心，因此我在下面重点介绍该算法的操作类的抽象数据类型

在此之中，我实现了对两景点的最短路径求解，和途径多个景点的最短路径的求解，并能生成路线的文字报告。

最短路径算法生成模块

public shortestPath();//构造方法

public void init();//初始化距离数组等一系列在dijkstra要用到的数据

public void Dijkstra(int index);//传入起点，利用最短路径算法算一遍，获取dis数组和pre数据的信息

public Vector<Point> shortPath(int start,int end);//获取一对景点的最短路径的数据

private Vector<String > returnViewName(Vector<Point> path);//将经过的点的名字收集起来

private Vector<Point> linkPath(int start,int end);//将经过的点全部收集起来，作为后来需要描绘的点的坐标

public Vector<Point> shortPathInGreatPoint(Vector<Integer> pointIndex);//多个景点的最短路径算法，会生成一个顶点坐标的集合，给GUI的描绘功能提供数据

public void toPrintMessage();//将路径的转法打印出来，告诉用户应该从哪个地点到达哪个地点。

### 3）模块层次调用关系图

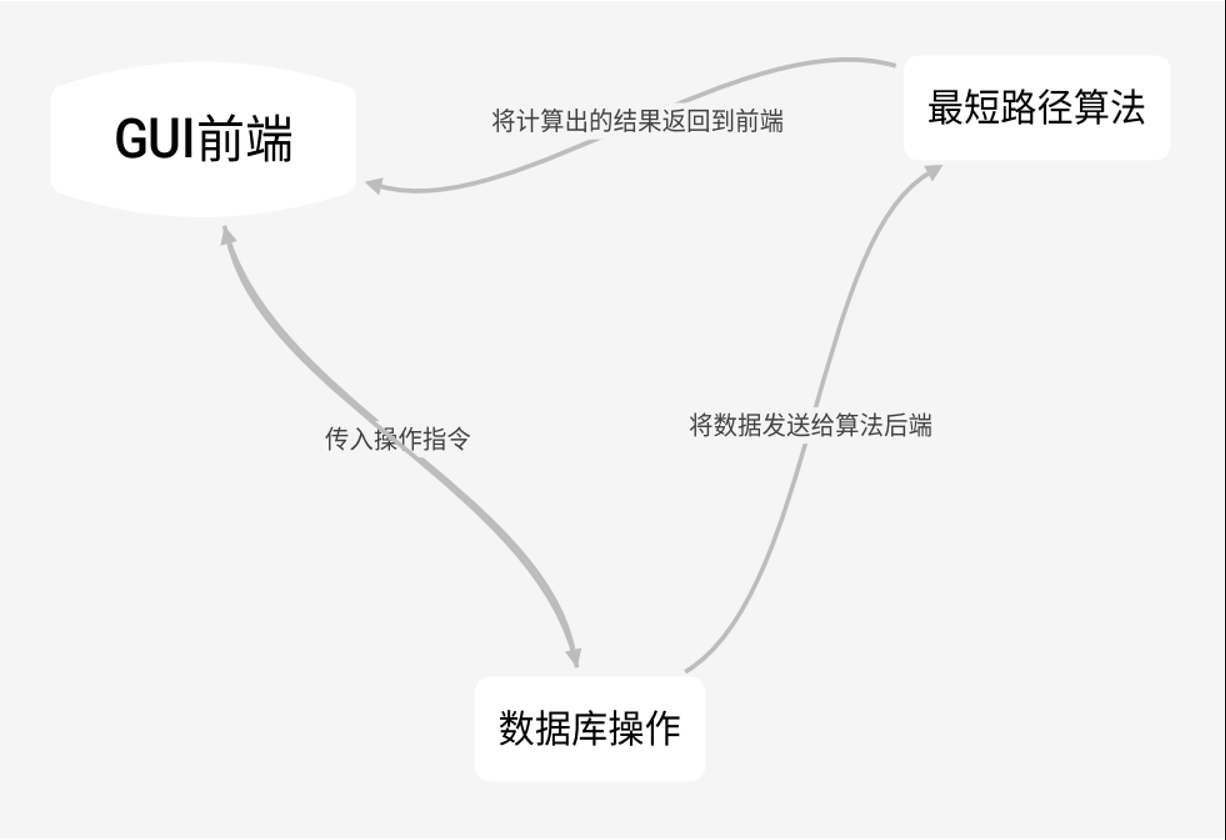


图1 模块层次调用关系图

## 四、详细设计

实现概要设计中定义的所有的类的定义及类中成员函数，并对主要的模块写出伪码算法。

由于此次实验我主要是对最短路径算法的实践，因此下面主要对最短路径算法做出主要的解释，详细阐述我的设计。

### 最短路径算法：shortestPath类实现

此类存放着必要的算法实践数据和操作方法，并能返回需要的描绘坐标。下面是对算法的主要方法进行伪码描述。

public shortestPath(){

getPoint=new allPointData();

dataCon.returnAllPointData();

e =dataCon.getGraphData();

cap=dataCon.getAllPointNumber()+1;

viewPointCap=dataCon.getViewPointNumber()+1;

pre=new int[cap];

vis=new boolean[cap];

dis=new double[cap];

}

public void init(){//初始化一会dijkstra算法要用的数组

for(历遍所有的点)

将pre数组的所有结点指向自己;

将dis数组全部初始为无限大;

将vis数组全部初始化为false，表示未历遍;

}

}

public void Dijkstra(传入起点坐标){

初始化数组;

for(历遍所有的点){

if(如果起点和历遍到的点有连接)

将有连接的点对应的dis数组中数据初始化为起点到该点的距离;

}

初始化dis数组中起点的到起点的距离为0;

for(历遍所有点){

u序号初始化为-1;

min初始化为无限大;

for(历遍所有点){

if(如果该点尚未被加入集合且该点的距离比记录的最短距离短){

更新最短距离;

u更新为该点的序号;

}

}

if(如果u没更新，依然是-1)

说明算法完成，算法终止;

否则，将u加入集合;

for(历遍所有点){

if(如果该点没有加入集合且与刚刚找到的起点有连接){

if(如果到v的距离大于到u的距离){

更新到v的距离为到u的距离加从u到v的距离;

Pre数组更新;

}

}

}

}

}

private Vector<Point> linkPath(起点序号,终点序号){

初始化path数组;

while (pre数组终点序号下指向的不是自己){

将该序号对应的坐标放进path数组;

终点更新为pre数组下指向的序号;

}

将终点坐标加进path数组;

将起点坐标加进path数组;

return path;

}

public Vector<Point> shortPath(起点序号，终点序号){

Dijkstra(起点);

获取最短路径的坐标数组;

将起点和终点和距离写进allMessage中;

return path;

}

private Vector<String > returnViewName(path数组){

初始化nameset数组;

for(从数组的最后开始历遍){

获取景点的具体信息;

将具体信息加入nameSet中;

}

return nameSet数组;

}

public Vector<Point> shortPathInGreatPoint(要经过的景点的序号pointIndex数组){

初始化path数组;

初始化visit数组;

index设置为pointIndex的第一个点的序号;

初始化distance;

for(历遍所有要去的景点){

初始化临时存point的数组;

start初始化为index;

将该点放进已历遍的集合;

Dijkstra(index);

初始化min=inf;

for(历遍所有要去的景点序号){

if(该点没有加入已历遍集合){

if(最短距离大于去该节点的距离){

更新最短距离;

Index更新为这个点的序号;

}

}

}

End设为index;

Temp接收要经过的结点集合;

Distance加上min;

for(临时数组从最后开始历遍){

将Point加入path数组;

}

将经过点景点的信息存进allMessage;

}

return path;

}

public void toPrintMessage(){

String allData=new String();

将路径数据存进字符串;

用对话框将字符串显示出来;

allMessage.clear();

}

## 五、调试分析

包括调试过程中遇到的问题及解决的方法、算法的时间空间复杂性分析、经验体会。

### Hash映射

由于我选定的是我们学校的地图，在对地图进行抽象成点和线进行描述的时候，发现数据很多，而且需要频繁调用。

在需要大量调用数据的类中，我采用了哈希表法，利用hashMap提高数据调用效率，使得各种数据的调用的复杂度变为O(1)，提高了数据的调用效率。

由于不止一处地方采用了hash，就只举实验中的一个例子

public class returnViewPointMessage {

private Map<Point,txtMessage> hashTabelViewMessage=new HashMap();

private Vector<viewMessage> getAllData(){

connectSQL temp=new connectSQL();

return temp.loadViewMessage();

}

private Vector<Point> getPointMessage(Vector<viewMessage> allData){

Vector<Point> temp=new Vector();

for(int i=0;i<allData.size();i++){

Point tempPoint=new Point(allData.elementAt(i).getPointX(),allData.elementAt(i).getPointY());

temp.add(tempPoint);

}

return temp;

}

private Vector<txtMessage> getTxtMessage(Vector<viewMessage> allData){

Vector<txtMessage> temp=new Vector();

for(int i=0;i<allData.size();i++){

txtMessage tempTxt=new txtMessage();

tempTxt.setName(allData.elementAt(i).getViewName());

tempTxt.setTxt(allData.elementAt(i).getTxt());

temp.add(tempTxt);

}

return temp;

}

private void buildHashTable(){

Vector<viewMessage> allData=getAllData();

Vector<Point> key=getPointMessage(allData);

Vector<txtMessage> values=getTxtMessage(allData);

for(int i=0;i<key.size();i++){

hashTabelViewMessage.put(key.elementAt(i),values.elementAt(i));

}

}

public returnViewPointMessage(){

buildHashTable();

}

public txtMessage getViewPointMessage(Point click){

return hashTabelViewMessage.get(click);

}

}

### 鼠标点击偏移修正

用户点击景点的时候，并不是每次都十分准确地点到景点的极其精准的X，Y坐标。用户点的不够准，我们需要修正这个差距。

我使用的方法是，收集一个景点后，以这个点为中心扩散一定范围，用这些点建立一个并查集，该范围的所有结点都指向景点，从而修正用户鼠标点击不准的问题。

public void addViewPointPosition(Point click){

conSQL();

builtCon();

for(int i=-5\*2;i<6\*2;i++)

for(int j=-5\*2;j<6\*2;j++){

String massage="('"+((int)click.getX()+i)+"','"+((int)click.getY()+j)+"','"+((int)click.getX())+"','"+((int)click.getY())+"')";

try{

cmd.executeUpdate(insertCommand+massage);

System.out.println("Successful!");

}catch (SQLException e){

System.out.println(e.toString());

}

}

}

### 数据库调用中文时出现乱码

原因是字符集。Mysql的字符集要设置成utf-8，在java里导入时url要加一句encode，把utf-8编译过来才能不显示乱码。

private String url="jdbc:mysql://127.0.0.1:3306/viewPoint?characterEncoding=UTF-8";

### 最短路径算法

本次实现我采用是dijkstra最短路径算法。编写代码途中发现有些地方出了错，调试了很久才调试出来。

此算法复杂度为。在本实验中由于实验数据并没有太大，80，90左右的点可以算的过来。

但算法依然可以优化，利用邻接表和二叉堆，能够将复杂度降为。

public void Dijkstra(int index){

init();

for(int i=1;i<cap;i++){

if(e[index][i]!=inf)

dis[i]=e[index][i];

}

dis[index]=0;

for(int i=0;i<cap;i++){

int u=-1;

double min=inf;

for(int j=1;j<cap;j++){

if(vis[j]==false&&min>dis[j]){

min=dis[j];

u=j;

}

}

if(u==-1)

break;

vis[u]=true;

for(int v=1;v<cap;v++){

if(vis[v]==false&&e[u][v]!=inf){

if(dis[v]>dis[u]+e[u][v]){

dis[v]=dis[u]+e[u][v];

pre[v]=u;

}

}

}

}

}

## 六、测试结果

### 登陆界面

该界面是登陆的界面，要求验证用户身份



图2 系统登录界面

当输入为管理员账户时，会提示是否以管理员身份进入系统。点否就是普通用户身份，点是则可以进入修改具体的景点信息。

当输入的是普通用户账户，则不会提示，直接进去导游系统，提供导游服务。



图3 登入提示

### 导游界面

登入成功后，可以看到如下界面



图4 导游图界面

开启景点展示模式之后，点击对应结点可以显示该节点的具体信息，如名字，具体介绍等。

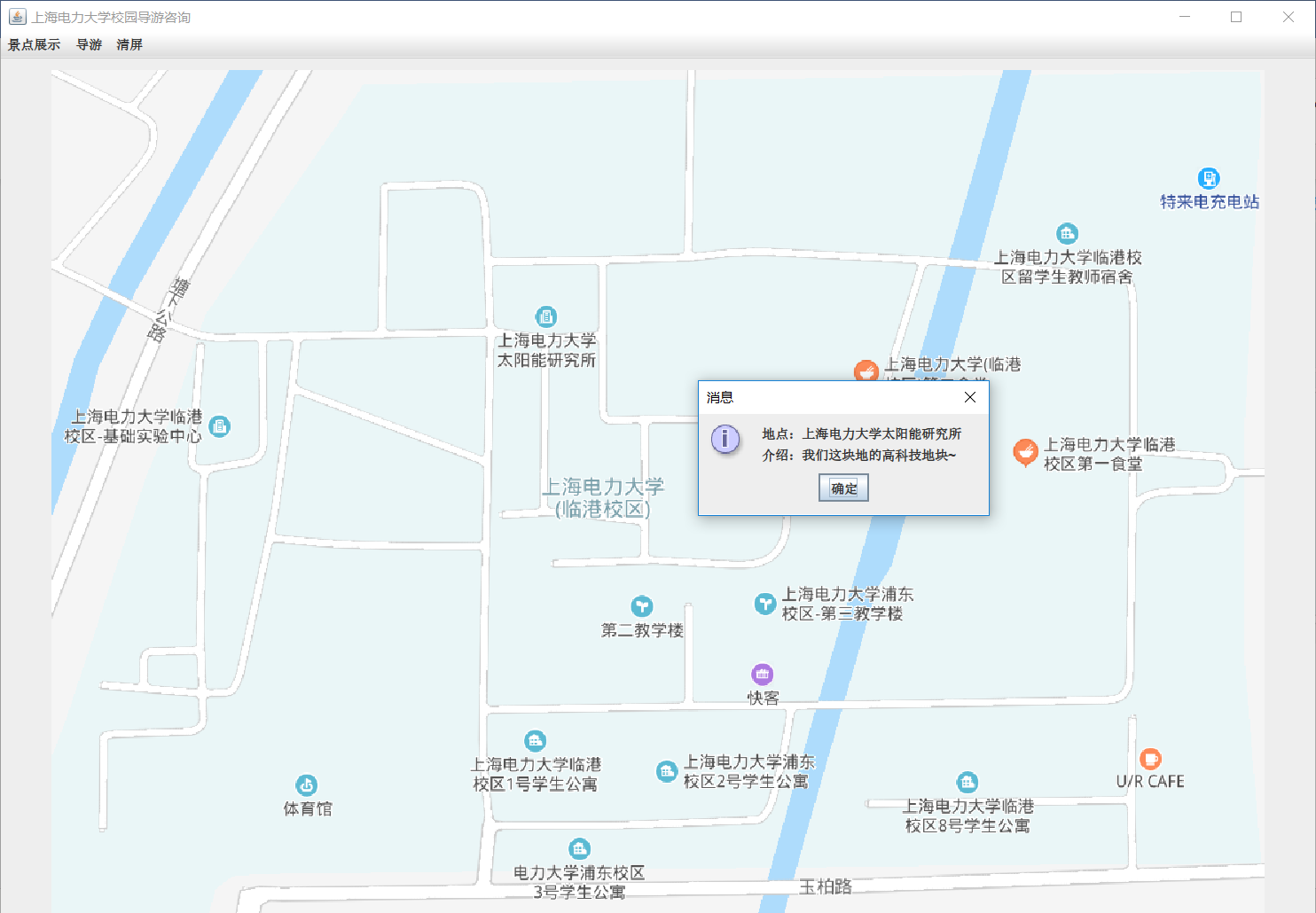


图5 景点介绍

当选择直接导游模式的时候，选择图片上两个景点，将会在界面上描绘出该最短路径，并在一秒钟之后提示是否清除路径。



图6 最短路径展示

如果选择多个景点，则可以生成通过这几个景点的最短路径，并描绘出来，并生成一份文字导游报告



图7 途径几个景点的最短路径

### 修改信息界面

选择进入管理员账号时，就会显示该界面。



图8 修改信息界面

点开修改景点信息，就能看到三个功能选项，选择对应功能选项。



图9 修改景点信息

选择一个功能后，会让用户选择一个景点位置。

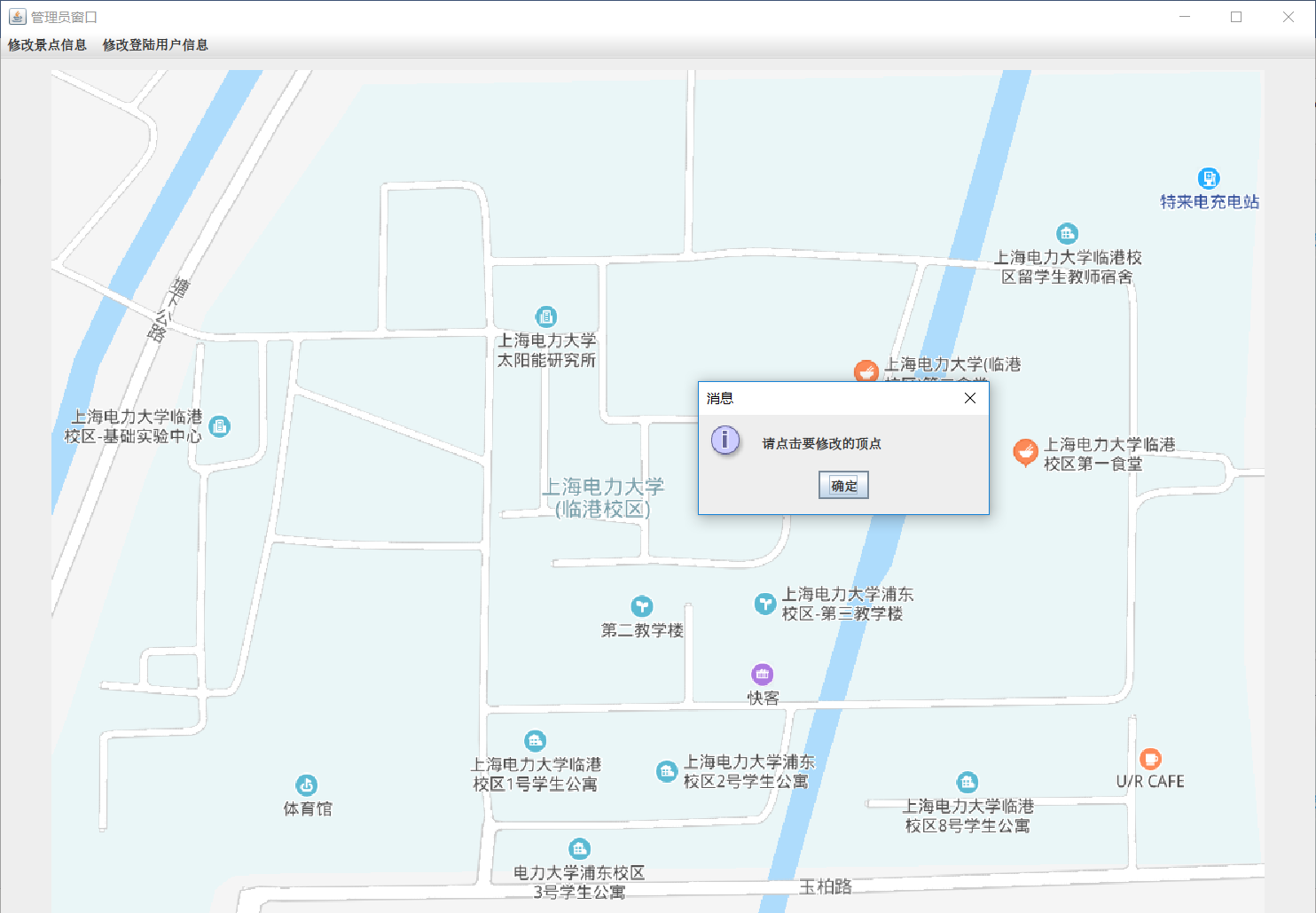


图10 选定需要修改的点

选定之后，会提示用户输入景点名字和具体信息等。其他两个功能同理。

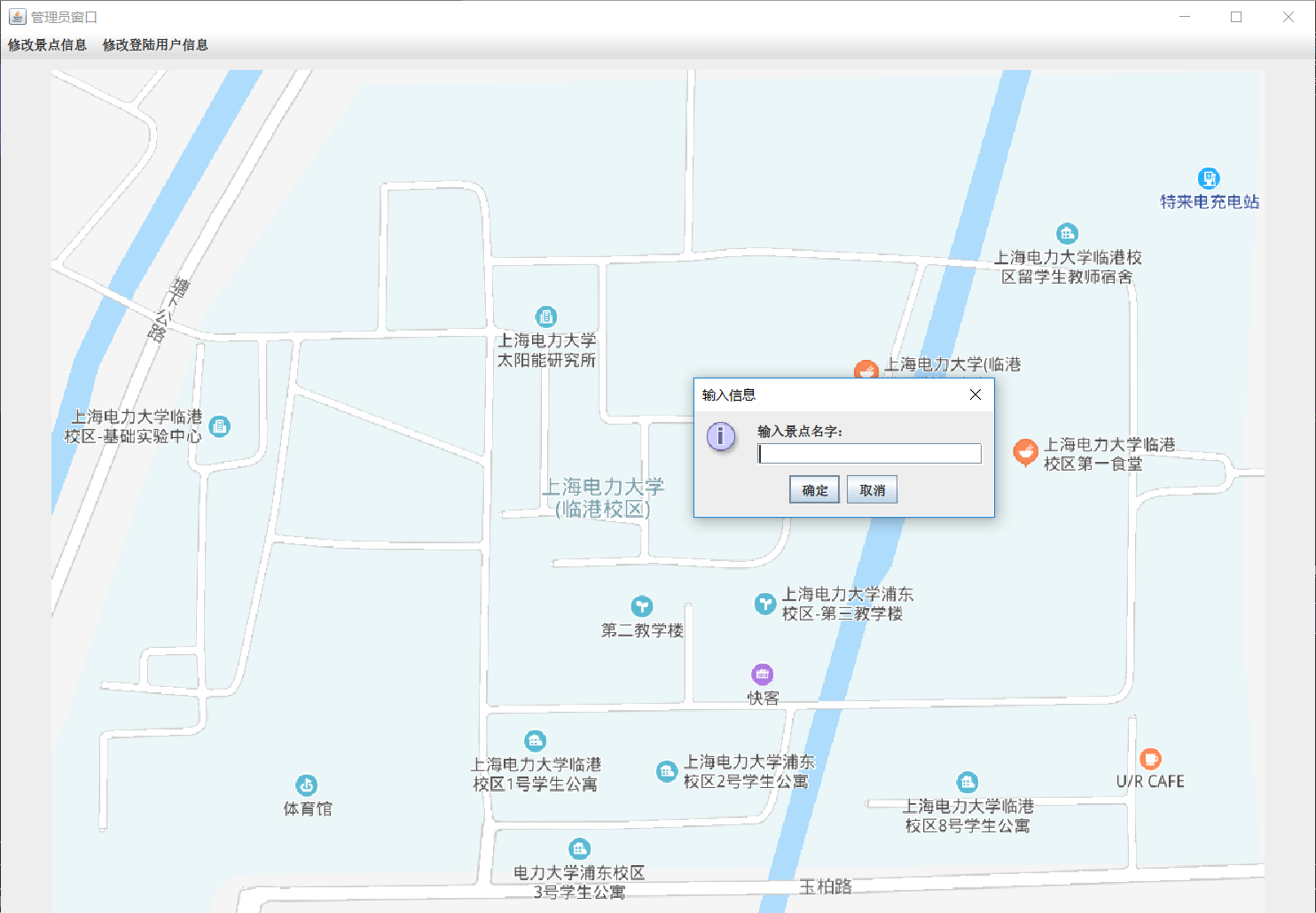


图11 输入信息

点开修改登陆用户功能后，会让用户选择功能。



图12 修改账户信息

选择添加账户，系统会一步一步引导用户输入信息。其他两个功能同理。

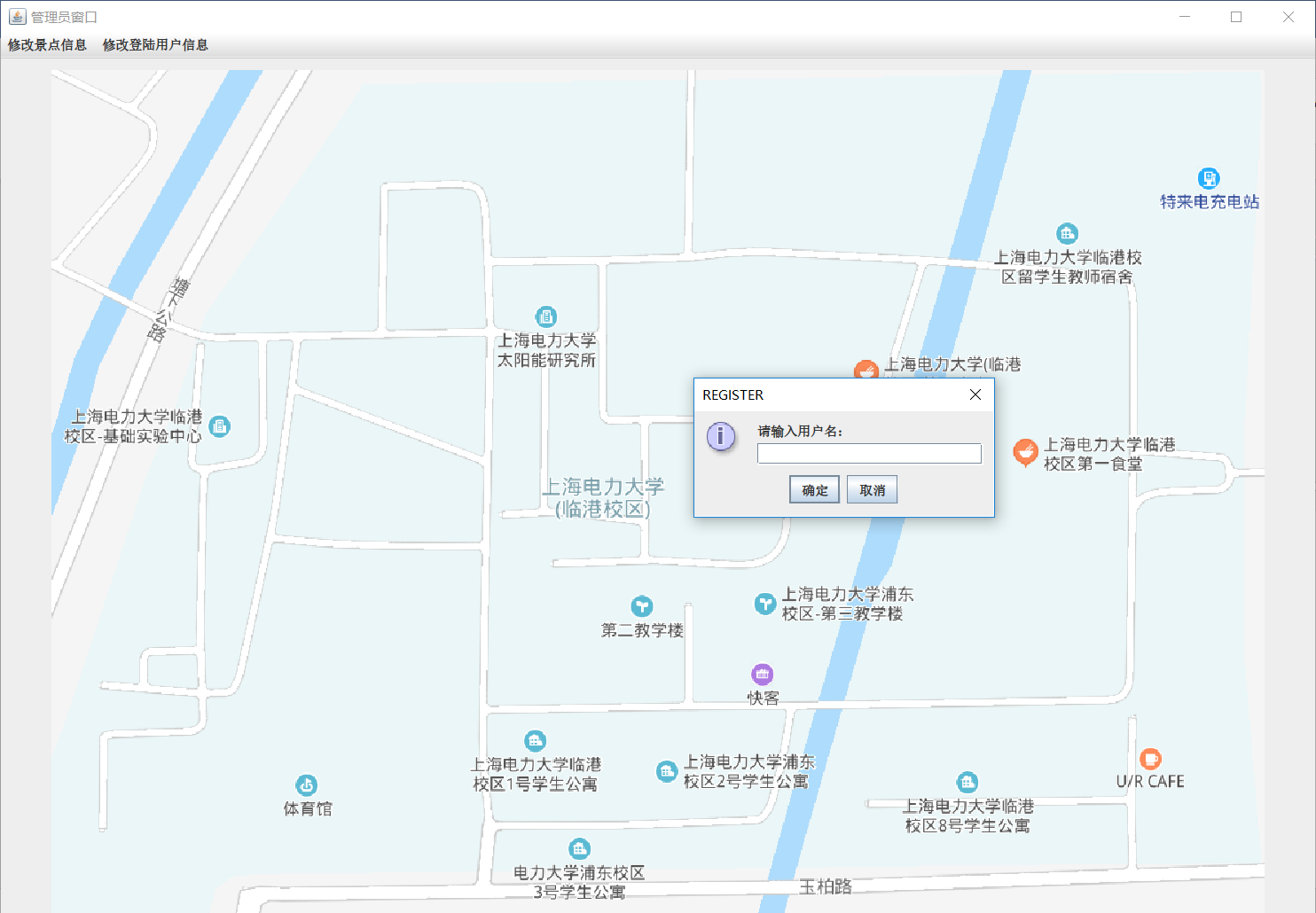


图13 输入信息

## 七 、附录：程序设计源代码

### welcomWindow.class:

package display**;**import dataStorage.connectSQL**;**import javax.swing.\***;**import java.awt.\***;**import java.awt.event.ActionEvent**;**import java.awt.event.ActionListener**;**public class welcomeWindow extends JFrame {  
 public welcomeWindow() {  
 ENTERButton.addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String u = textField1.getText()**;** String p = textField2.getText()**;** connectSQL temp = new connectSQL()**;** String args[] = {" "**,** "2"}**;** if (temp.insureUser(u**,** p) == **1**) {  
 int flag = JOptionPane.*showConfirmDialog*(null**,** "你是否想以管理员身份进入系统？"**,** "管理员账户登入"**,** JOptionPane.*WARNING\_MESSAGE*)**;** if (flag == JOptionPane.*YES\_OPTION*)  
 administerWin.*main*(args)**;** else  
 mainWin.*main*(args)**;** } else if (temp.insureUser(u**,** p) == **0**) {  
 mainWin.*main*(args)**;** } else {  
 JOptionPane.*showMessageDialog*(null**,** "Failed")**;** welcomeWindow.*setDefaultLookAndFeelDecorated*(true)**;** }  
  
 }  
 })**;** }  
  
 public static void main(String[] args) {  
 JFrame frame = new JFrame("上海电力大学校园导游系统登陆窗口")**;** frame.setContentPane(new welcomeWindow().panel1)**;** frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*)**;** frame.pack()**;** frame.setVisible(true)**;** }  
  
 private JPanel panel1**;** private JTextField textField1**;** private JPanel Login**;** private JButton ENTERButton**;** private JPanel Left**;** private JPanel Right**;** private JPanel Name**;** private JPanel Model**;** private JPanel School**;** private JPanel Yuan**;** private JPanel left2**;** private JPanel left1**;** private JPanel left3**;** private JPanel right1**;** private JPanel right2**;** private JPanel right3**;** private JLabel sc**;** private JLabel cs**;** private JTextField textField2**;** {  
// GUI initializer generated by IntelliJ IDEA GUI Designer  
// >>> IMPORTANT!! <<<  
// DO NOT EDIT OR ADD ANY CODE HERE!  
 $$$setupUI$$$()**;** }  
  
 */\*\*  
 \* Method generated by IntelliJ IDEA GUI Designer  
 \* >>> IMPORTANT!! <<<  
 \* DO NOT edit this method OR call it in your code!  
 \*  
 \** ***@noinspection*** *ALL  
 \*/* private void $$$setupUI$$$() {  
 panel1 = new JPanel()**;** panel1.setLayout(new BorderLayout(**0, 0**))**;** Login = new JPanel()**;** Login.setLayout(new com.intellij.uiDesigner.core.GridLayoutManager(**1, 2,** new Insets(**0, 0, 0, 0**)**,** -**1,** -**1**))**;** panel1.add(Login**,** BorderLayout.*SOUTH*)**;** Left = new JPanel()**;** Left.setLayout(new com.intellij.uiDesigner.core.GridLayoutManager(**3, 1,** new Insets(**0, 0, 0, 0**)**,** -**1,** -**1**))**;** Login.add(Left**,** new com.intellij.uiDesigner.core.GridConstraints(**0, 0, 1, 1,** com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER**,** com.intellij.uiDesigner.core.GridConstraints.FILL\_BOTH**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** null**,** null**,** null**, 0,** false))**;** left1 = new JPanel()**;** left1.setLayout(new FlowLayout(FlowLayout.*CENTER***, 5, 5**))**;** Left.add(left1**,** new com.intellij.uiDesigner.core.GridConstraints(**0, 0, 1, 1,** com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER**,** com.intellij.uiDesigner.core.GridConstraints.FILL\_BOTH**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** null**,** null**,** null**, 0,** false))**;** final JLabel label1 = new JLabel()**;** label1.setText("USER")**;** left1.add(label1)**;** left2 = new JPanel()**;** left2.setLayout(new FlowLayout(FlowLayout.*CENTER***, 5, 5**))**;** Left.add(left2**,** new com.intellij.uiDesigner.core.GridConstraints(**1, 0, 1, 1,** com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER**,** com.intellij.uiDesigner.core.GridConstraints.FILL\_BOTH**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** null**,** null**,** null**, 0,** false))**;** final JLabel label2 = new JLabel()**;** label2.setText("PASSWORD")**;** left2.add(label2)**;** left3 = new JPanel()**;** left3.setLayout(new FlowLayout(FlowLayout.*CENTER***, 5, 5**))**;** Left.add(left3**,** new com.intellij.uiDesigner.core.GridConstraints(**2, 0, 1, 1,** com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER**,** com.intellij.uiDesigner.core.GridConstraints.FILL\_BOTH**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** null**,** null**,** null**, 0,** false))**;** final JLabel label3 = new JLabel()**;** label3.setText("ENSURE")**;** left3.add(label3)**;** Right = new JPanel()**;** Right.setLayout(new com.intellij.uiDesigner.core.GridLayoutManager(**3, 1,** new Insets(**0, 0, 0, 0**)**,** -**1,** -**1**))**;** Login.add(Right**,** new com.intellij.uiDesigner.core.GridConstraints(**0, 1, 1, 1,** com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER**,** com.intellij.uiDesigner.core.GridConstraints.FILL\_BOTH**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** null**,** null**,** null**, 0,** false))**;** right1 = new JPanel()**;** right1.setLayout(new FlowLayout(FlowLayout.*CENTER***, 5, 5**))**;** Right.add(right1**,** new com.intellij.uiDesigner.core.GridConstraints(**0, 0, 1, 1,** com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER**,** com.intellij.uiDesigner.core.GridConstraints.FILL\_BOTH**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** null**,** null**,** null**, 0,** false))**;** textField1 = new JTextField()**;** textField1.setColumns(**20**)**;** right1.add(textField1)**;** right2 = new JPanel()**;** right2.setLayout(new FlowLayout(FlowLayout.*CENTER***, 5, 5**))**;** Right.add(right2**,** new com.intellij.uiDesigner.core.GridConstraints(**1, 0, 1, 1,** com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER**,** com.intellij.uiDesigner.core.GridConstraints.FILL\_BOTH**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** null**,** null**,** null**, 0,** false))**;** textField2 = new JTextField()**;** textField2.setColumns(**20**)**;** right2.add(textField2)**;** right3 = new JPanel()**;** right3.setLayout(new FlowLayout(FlowLayout.*CENTER***, 5, 5**))**;** Right.add(right3**,** new com.intellij.uiDesigner.core.GridConstraints(**2, 0, 1, 1,** com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER**,** com.intellij.uiDesigner.core.GridConstraints.FILL\_BOTH**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** null**,** null**,** null**, 0,** false))**;** ENTERButton = new JButton()**;** ENTERButton.setText("ENTER")**;** right3.add(ENTERButton)**;** Name = new JPanel()**;** Name.setLayout(new FlowLayout(FlowLayout.*CENTER***, 5, 5**))**;** panel1.add(Name**,** BorderLayout.*NORTH*)**;** final JLabel label4 = new JLabel()**;** label4.setText("Campus tour guide consultation")**;** Name.add(label4)**;** Model = new JPanel()**;** Model.setLayout(new FlowLayout(FlowLayout.*CENTER***, 5, 5**))**;** panel1.add(Model**,** BorderLayout.*CENTER*)**;** School = new JPanel()**;** School.setLayout(new com.intellij.uiDesigner.core.GridLayoutManager(**1, 1,** new Insets(**0, 0, 0, 0**)**,** -**1,** -**1**))**;** Model.add(School)**;** sc = new JLabel()**;** sc.setEnabled(true)**;** sc.setHorizontalAlignment(**0**)**;** sc.setHorizontalTextPosition(**0**)**;** sc.setIcon(new ImageIcon(getClass().getResource("/display/校徽.png")))**;** sc.setText("")**;** sc.setVerticalAlignment(**0**)**;** School.add(sc**,** new com.intellij.uiDesigner.core.GridConstraints(**0, 0, 1, 1,** com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER**,** com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** new Dimension(**200, 200**)**,** new Dimension(**200, 200**)**,** new Dimension(**400, 400**)**, 0,** false))**;** Yuan = new JPanel()**;** Yuan.setLayout(new com.intellij.uiDesigner.core.GridLayoutManager(**1, 1,** new Insets(**0, 0, 0, 0**)**,** -**1,** -**1**))**;** Model.add(Yuan)**;** cs = new JLabel()**;** cs.setHorizontalAlignment(**0**)**;** cs.setHorizontalTextPosition(**0**)**;** cs.setIcon(new ImageIcon(getClass().getResource("/display/院徽章.png")))**;** cs.setText("")**;** Yuan.add(cs**,** new com.intellij.uiDesigner.core.GridConstraints(**0, 0, 1, 1,** com.intellij.uiDesigner.core.GridConstraints.ANCHOR\_CENTER**,** com.intellij.uiDesigner.core.GridConstraints.FILL\_NONE**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_SHRINK | com.intellij.uiDesigner.core.GridConstraints.SIZEPOLICY\_CAN\_GROW**,** new Dimension(**200, 200**)**,** new Dimension(**200, 200**)**,** new Dimension(**400, 400**)**, 0,** false))**;** }  
  
 */\*\*  
 \** ***@noinspection*** *ALL  
 \*/* public JComponent $$$getRootComponent$$$() {  
 return panel1**;** }  
  
}

### SwingConsole.class:

package display**;**import javax.swing.\***;**public class SwingConsole {  
 public static void run(final JFrame f**,**final int width**,**final int height**,**String title){  
 SwingUtilities.*invokeLater*(new Runnable() {  
 @Override  
 public void run() {  
 f.setTitle(title)**;** f.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*)**;** f.setSize(width**,**height)**;** f.setVisible(true)**;** }  
 })**;** }  
}

### mainWin.class:

package display**;**import dataStorage.\***;**import graphAlgorithm.shortestPath**;**import javax.swing.\***;**import java.awt.\***;**import java.awt.event.\***;**import java.util.Vector**;**import java.util.concurrent.TimeUnit**;**public class mainWin extends JFrame {  
 private Point Temp=new Point()**;** private boolean watchFlag =false**;** private int clickCount=**0,** modern =**0;** private final int OFF=**0,**RESET=**0;** private final int DISPLAYMODERNON=**0,**DISPLAYMODERNOFF=**1,**DISPALYMODERN=**0;** private final int GUIDE=**1,**TWOPOINTMODERN=**2,**GREATPOINTMODERN=**3;** private final int CLEAR=**2,**CLEARMODERNON=**4;** private Vector<Point> set=new Vector()**;** private Vector<Point> line=new Vector()**;** private shortestPath path=new shortestPath()**;** private JPanel basic=new JPanel()**;** private JPanel menu=new JPanel()**;** private allPointData data=new allPointData()**;** private ensureViewPointData temp=new ensureViewPointData()**;** private JPanel Graph=new JPanel(){  
 private Vector<Point> a=new Vector()**;** public void setLocation(Point e){  
 if(e==null){  
 a.clear()**;** return**;** }  
 Temp=e**;** a.add(e)**;** }  
 @Override  
 public void paintComponents(Graphics g) {  
 super.paintComponents(g)**;** ImageIcon a=new ImageIcon(getClass().getResource("黄旗子.png"))**;** Image image=a.getImage()**;** for(int i=**0;**i<this.a.size()**;**i++){  
 g.drawImage(image**,**this.a.get(i).x**,**this.a.get(i).y-**20,**this)**;** }  
 }  
 @Override  
 public void paintAll(Graphics g) {  
 super.paintAll(g)**;** paintComponents(g)**;** for(int i=**0;**i<line.size()-**1;**i++){  
 g.drawLine(line.elementAt(i).x**,**line.elementAt(i).y**,**line.elementAt(i+**1**).x**,**line.elementAt(i+**1**).y)**;** }  
 line.clear()**;** }  
 }**;** private JLabel map=new JLabel()**;** private JMenuBar menuBar=new JMenuBar()**;** private JMenu Menu[]={  
 new JMenu("景点展示")**,** new JMenu("导游")**,**new JMenu("清屏")  
 }**;** private JMenuItem[] item={  
 new JMenuItem("开启展示模式")**,** new JMenuItem("关闭展示模式")**,** new JMenuItem("直达路线")**,** new JMenuItem("途径多个景点")**,** new JMenuItem("清空全屏")  
 }**;** public mainWin(){  
 setLayout()**;** setMenuItem()**;** addMenuItemListener()**;** setMenuBar()**;** setGraph()**;** setJMenuBar(menuBar)**;** basic.add(Graph)**;** add(BorderLayout.*CENTER***,**basic)**;** }  
 public void clear(){  
 Graph.setLocation(null)**;** Graph.paintComponents(Graph.getGraphics())**;** JOptionPane.*showMessageDialog*(null**,**"已清除")**;** }  
 public void clean(){  
 Graph.setLocation(null)**;** Graph.paintComponents(Graph.getGraphics())**;** }  
 public void cleanCounter(){  
 try {  
 TimeUnit.*SECONDS*.sleep(**1**)**;** int n=JOptionPane.*showConfirmDialog*(null**,**"即将清除界面中的路径信息，是否继续？")**;** if(n==JOptionPane.*OK\_OPTION*){  
 clear()**;** }else {  
 JOptionPane.*showMessageDialog*(null**,**"您可以随时在菜单栏中清除页面")**;** }  
 } catch (InterruptedException ex) {  
 ex.printStackTrace()**;** }  
 }  
 public void messageOfView(MouseEvent e){  
 returnViewPointMessage temp2=new returnViewPointMessage()**;** try{  
 Point p=temp.realViewPoint(e.getPoint())**;** JOptionPane.*showMessageDialog*(null**,**"地点："+temp2.getViewPointMessage(p).getName()+"\n"+"介绍："+temp2.getViewPointMessage(p).getTxt())**;** }catch (NullPointerException l){  
 JOptionPane.*showMessageDialog*(null**,**"没有点在景点上哦~")**;** }  
 }  
 public void clickInPoint(MouseEvent e){  
 Point p=temp.realViewPoint(e.getPoint())**;** if(p==null)  
 JOptionPane.*showMessageDialog*(null**,**"没点到景点上哦~")**;** else {  
 set.add(p)**;** Graph.setLocation(e.getPoint())**;** Graph.paintComponents(Graph.getGraphics())**;** clickCount++**;** }  
 }  
 public void disPlaySimplyPath(){  
 int start=data.returnIndex(set.elementAt(**0**))**;** int end=data.returnIndex(set.elementAt(**1**))**;** line.clear()**;** line=path.shortPath(start**,**end)**;** Graph.paintAll(Graph.getGraphics())**;** path.toPrintMessage()**;** clickCount=RESET**;** set.clear()**;** cleanCounter()**;** }  
 public void disPlayablePath(){  
 Vector<Integer> clickPoint=new Vector()**;** for(int i=**0;**i<set.size()**;**i++){  
 clickPoint.add(data.returnIndex(set.elementAt(i)))**;** }  
 line.clear()**;** line=path.shortPathInGreatPoint(clickPoint)**;** Graph.paintAll(Graph.getGraphics())**;** path.toPrintMessage()**;** clickCount=RESET**;** set.clear()**;** cleanCounter()**;** }  
 public void setMenuItem(){  
 Menu[DISPALYMODERN].add(item[DISPLAYMODERNON])**;** Menu[DISPALYMODERN].add(item[DISPLAYMODERNOFF])**;** Menu[GUIDE].add(item[TWOPOINTMODERN])**;** Menu[GUIDE].add(item[GREATPOINTMODERN])**;** Menu[CLEAR].add(item[CLEARMODERNON])**;** }  
 public void addMenuItemListener(){  
 item[DISPLAYMODERNON].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 watchFlag =true**;** }  
 })**;** item[DISPLAYMODERNOFF].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 watchFlag =false**;** }  
 })**;** item[TWOPOINTMODERN].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 clean()**;** modern =TWOPOINTMODERN**;** }  
 })**;** item[GREATPOINTMODERN].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 clean()**;** modern =GREATPOINTMODERN**;** JOptionPane.*showMessageDialog*(Graph**,**"鼠标左键加点，右键结束")**;** }  
 })**;** item[CLEARMODERNON].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 clear()**;** }  
 })**;** }  
 public void setMenuBar(){  
 menuBar.add(Menu[DISPALYMODERN])**;** menuBar.add(Menu[GUIDE])**;** menuBar.add(Menu[CLEAR])**;** }  
 public void setLayout(){  
 basic.setLayout(new FlowLayout())**;** menu.setLayout(new FlowLayout())**;** Graph.setLayout(new FlowLayout())**;** }  
 public void setGraph(){  
 map.setIcon(new ImageIcon(getClass().getResource("map.png")))**;** Graph.add(map)**;** Graph.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e)**;** if(watchFlag ==true){  
 messageOfView(e)**;** }  
 if(modern ==TWOPOINTMODERN){  
 clickInPoint(e)**;** if(clickCount==**2**){  
 modern =OFF**;** disPlaySimplyPath()**;** }  
 }  
 else if(modern ==GREATPOINTMODERN){  
 if(e.getButton()==**1**){  
 clickInPoint(e)**;** }  
 if(e.getButton()==**3**){  
 modern =OFF**;** disPlayablePath()**;** }  
 }  
 }  
 })**;** }  
 public static void main(String afgs[]){  
 SwingConsole.*run*(new mainWin()**,1200,1000,**"上海电力大学校园导游咨询")**;** }  
}

### administerWin.class:

package display**;**import dataStorage.connectSQL**;**import dataStorage.ensureViewPointData**;**import javax.swing.\***;**import java.awt.\***;**import java.awt.event.ActionEvent**;**import java.awt.event.ActionListener**;**import java.awt.event.MouseAdapter**;**import java.awt.event.MouseEvent**;**public class administerWin extends JFrame {  
 private JPanel base**;** private JPanel Main**;** private JMenuBar tool**;** private final int VIEW=**0,**NUMBER=**1;** private final int ADDVIEW=**0,**CHANGVIEW=**1,**DELETEVIEW=**2;** private final int ADDNUMBER=**3,**CHANGELOGINPASSWORD=**5,**DELETENUMBER=**4;** private boolean flag=true**;** private connectSQL data=new connectSQL()**;** private JMenu[] menus={  
 new JMenu("修改景点信息")**,** new JMenu("修改登陆用户信息")  
 }**;** private JMenuItem[] items={  
 new JMenuItem("增加景点")**,**new JMenuItem("修改景点信息")**,**new JMenuItem("删除景点信息")**,** new JMenuItem("添加登陆账号")**,**new JMenuItem("注销账号")**,**new JMenuItem("修改账号密码")  
 }**;** private JLabel graph**;** public administerWin(){  
 init()**;** setLayout()**;** setItemsListener()**;** setItems()**;** setMenuBar()**;** setGraph()**;** setGraphListener()**;** Main.add(base)**;** add(BorderLayout.*CENTER***,**Main)**;** }  
 public void init(){  
 base=new JPanel()**;** Main=new JPanel()**;** tool=new JMenuBar()**;** graph=new JLabel()**;** }  
 public void setLayout() {  
 base.setLayout(new FlowLayout())**;** Main.setLayout(new FlowLayout())**;** }  
 public void setItemsListener(){  
 items[ADDVIEW].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 JOptionPane.*showMessageDialog*(null**,**"敬请期待")**;** }  
 })**;** items[CHANGVIEW].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 flag=false**;** JOptionPane.*showMessageDialog*(null**,**"请点击要修改的顶点")**;** }  
 })**;** items[DELETEVIEW].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 JOptionPane.*showMessageDialog*(null**,**"敬请期待")**;** }  
 })**;** items[ADDNUMBER].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String user=getUser()**;** String password=getPassword()**;** data.addUser(user**,**password)**;** }  
 })**;** items[CHANGELOGINPASSWORD].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String user=getUser()**;** String password=getPassword()**;** data.changePassword(user**,**password)**;** }  
 })**;** items[DELETENUMBER].addActionListener(new ActionListener() {  
 @Override  
 public void actionPerformed(ActionEvent e) {  
 String user=getUser()**;** String password=getPassword()**;** data.deleteUser(user**,**password)**;** }  
 })**;** }  
 public void setItems(){  
 menus[VIEW].add(items[ADDVIEW])**;** menus[VIEW].add(items[CHANGVIEW])**;** menus[VIEW].add(items[DELETEVIEW])**;** menus[NUMBER].add(items[ADDNUMBER])**;** menus[NUMBER].add(items[CHANGELOGINPASSWORD])**;** menus[NUMBER].add(items[DELETENUMBER])**;** }  
 public void setMenuBar() {  
 tool.add(menus[VIEW])**;** tool.add(menus[NUMBER])**;** setJMenuBar(tool)**;** }  
 public void setGraph(){  
 graph.setIcon(new ImageIcon(getClass().getResource("map.png")))**;** base.add(graph)**;** }  
 public void setGraphListener(){  
 graph.addMouseListener(new MouseAdapter() {  
 @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e)**;** if(flag==false){  
 flag=true**;** ensureViewPointData temp=new ensureViewPointData()**;** Point real=temp.realViewPoint(e.getPoint())**;** if(real==null){  
 JOptionPane.*showMessageDialog*(null**,**"没有点在景点上~")**;** }else {  
 data.changeViewPointMessage(real)**;** }  
 }  
 }  
 })**;** }  
 public static void main(String args[]){  
 SwingConsole.*run*(new administerWin()**,1200,1000,**"管理员窗口")**;** }  
 public String getUser(){  
 return JOptionPane.*showInputDialog*(null**,**"请输入用户名：\n"**,**"REGISTER"**,**JOptionPane.*INFORMATION\_MESSAGE*)**;** }  
 public String getPassword(){  
 return JOptionPane.*showInputDialog*(null**,**"请输入密码：\n"**,**"REGISTER"**,**JOptionPane.*INFORMATION\_MESSAGE*)**;** }  
}

### allPointData.class:

package dataStorage;  
  
import java.awt.\*;  
import java.util.HashMap;  
import java.util.Map;  
import java.util.Vector;  
  
public class allPointData {  
 public Vector<pointInformation> data;  
 private Map<Integer, Point> hashTable=new HashMap();  
 private Map<Point,Integer> reHash=new HashMap();  
 private Map<Integer, Integer> getPointType=new HashMap();  
 public allPointData(){  
 connectSQL temp=new connectSQL();  
 data=temp.returnAllPointData();  
 buildHashTable();  
 }  
 private void buildHashTable(){  
 for(int i=0;i<data.size();i++){  
 hashTable.put(data.elementAt(i).getIndex(),data.elementAt(i).getPosition());  
 getPointType.put(data.elementAt(i).getIndex(),data.elementAt(i).getType());  
 reHash.put(data.elementAt(i).getPosition(),data.elementAt(i).getIndex());  
 }  
 }  
 public Point returnPoint(int index){  
 return hashTable.get(index);  
 }  
 public int returnIndex(Point click){  
 return reHash.get(click);  
 }  
 public int returnThePointType(int index){  
 return getPointType.get(index);  
 }  
}

### connectSQL.class:

package dataStorage**;**import javax.swing.\***;**import java.awt.\***;**import java.sql.\***;**import java.util.Vector**;**public class connectSQL {  
 private int cap=**0;** private int viewCap=**0;** private final String rootUser="hy4165yh@gmail.com"**,**rootPassword="hyong430fbing"**;** private String driverName="com.mysql.jdbc.Driver"**;** private String url="jdbc:mysql://127.0.0.1:3306/viewPoint?characterEncoding=UTF-8"**;** private String user="root"**,**password="123456"**;** private String selectMessageCommand="select \* from pointMessage"**;** private String getSelectMessageTable="select \* from viewMessage"**;** private String getUserMessage="select \* from user where userName="**;** private String getGraphTable="select \* from graphTable"**;** private String getRotutePointMessageTable="select \* from rotutePointMessage"**;** private String insertCommand="insert into pointMessage values"**;** private String insertCommandIntoRotuteMessageTable="insert into rotutePointMessage values"**;** private String insertUserMessage="insert into user values"**;** private String deleteUser="delete from user where userName="**;** private Connection con**;** private Statement cmd**;** private Vector<pointRowData> sentMessage=new Vector()**;** private Vector<viewMessage> sentViewMessage=new Vector()**;** private int clickCounter=**0;** private int viewPointNumber=**0,**allPointNumber=**0;** public void initClickCounter(){  
 clickCounter=**51;** }  
 public void conSQL(){  
 try{  
 Class.*forName*(driverName)**;** }catch (ClassNotFoundException e){  
 e.printStackTrace()**;** System.*out*.println("Fail!")**;** }  
 }  
 public void builtCon(){  
 try {  
 con=DriverManager.*getConnection*(url**,**user**,**password)**;** cmd=con.createStatement()**;** }catch (Exception e){  
 System.*out*.println("Fail!")**;** }  
 }  
  
 //Raw click point data  
 private Vector<pointRowData> getAllRowMessage(){  
 sentMessage.clear()**;** try{  
 ResultSet rs=cmd.executeQuery(selectMessageCommand)**;** while(rs.next()==true){  
 pointRowData temp=new pointRowData()**;** temp.setClickPointX(rs.getInt(**1**))**;** temp.setClickPointY(rs.getInt(**2**))**;** temp.setViewPointX(rs.getInt(**3**))**;** temp.setViewPointY(rs.getInt(**4**))**;** sentMessage.add(temp)**;** }  
 rs.close()**;** }catch (Exception e){  
 System.*out*.println(e.toString())**;** }  
 return this.sentMessage**;** }  
 public Vector<pointRowData> loadData(){  
 conSQL()**;** builtCon()**;** return getAllRowMessage()**;** }  
  
 //View point message , such as name of view , read in china  
 private Vector<viewMessage> getAllViewMessage(){  
 sentViewMessage.clear()**;** try{  
 ResultSet rs=cmd.executeQuery(getSelectMessageTable)**;** while (rs.next()){  
 viewMessage temp=new viewMessage()**;** temp.setPointX(rs.getInt(**1**))**;** temp.setPointY(rs.getInt(**2**))**;** temp.setViewName(rs.getString(**3**))**;** temp.setTxt(rs.getString(**4**))**;** sentViewMessage.add(temp)**;** }  
 }catch (SQLException e){  
 JOptionPane.*showMessageDialog*(null**,**"ERROR!")**;** }  
 return sentViewMessage**;** }  
 public Vector<viewMessage> loadViewMessage(){  
 conSQL()**;** builtCon()**;** return getAllViewMessage()**;** }  
  
 //Register raw data in SQL  
 public int tableLimit(){  
 int max=**0;** conSQL()**;** builtCon()**;** try {  
 ResultSet rs=cmd.executeQuery(getSelectMessageTable)**;** while (rs.next()){  
 if(max<rs.getInt(**5**))  
 max=rs.getInt(**5**)**;** }  
 }catch (SQLException e){  
 System.*out*.println(e.toString())**;** }  
 return max++**;** }  
 public void addViewPointPosition(Point click){  
 conSQL()**;** builtCon()**;** for(int i=-**5**\***2;**i<**6**\***2;**i++)  
 for(int j=-**5**\***2;**j<**6**\***2;**j++){  
 String massage="('"+((int)click.getX()+i)+"','"+((int)click.getY()+j)+"','"+((int)click.getX())+"','"+((int)click.getY())+"')"**;** try{  
 cmd.executeUpdate(insertCommand+massage)**;** System.*out*.println("Successful!")**;** }catch (SQLException e){  
 System.*out*.println(e.toString())**;** }  
 }  
 }  
 public void addRotePointPostion(Point click){  
 conSQL()**;** builtCon()**;** String message="('"+((int)click.getX())+"','"+((int)click.getY())+"','"+(clickCounter)+"')"**;** clickCounter++**;** try {  
 cmd.executeUpdate(insertCommandIntoRotuteMessageTable+message)**;** System.*out*.println("Successful!"+clickCounter)**;** }catch (SQLException e){  
 System.*out*.println(e.toString())**;** }  
 }  
 public void changeViewPointMessage(Point click){  
 conSQL()**;** builtCon()**;** String name=JOptionPane.*showInputDialog*(null**,**"输入景点名字："**,**"输入信息"**,**JOptionPane.*INFORMATION\_MESSAGE*)**;** String txt=JOptionPane.*showInputDialog*(null**,**"请输入介绍："**,**"输入信息"**,**JOptionPane.*INFORMATION\_MESSAGE*)**;**System.*out*.println(click.toString())**;** String command1="update viewMessage SET name='"+name+"' where pointX="+click.x**;** String command2="update viewMessage SET txt='"+txt+"' where pointX="+click.x**;** try {  
 cmd.executeUpdate(command1)**;** cmd.executeUpdate(command2)**;** }catch (SQLException e){  
 JOptionPane.*showMessageDialog*(null**,**"ERROR!")**;** }  
 }  
  
 //To get AllPointData part  
 private Vector<pointInformation> getViewPointData(){  
 Vector<pointInformation> temp=new Vector()**;** try{  
 ResultSet rs=cmd.executeQuery(getSelectMessageTable)**;** while (rs.next()){  
 pointInformation a=new pointInformation()**;** Point v=new Point()**;** v.setLocation(rs.getInt(**1**)**,**rs.getInt(**2**))**;** a.setPosition(v)**;** a.setIndex(rs.getInt(**5**))**;** a.setType(**1**)**;** temp.add(a)**;** }  
 }catch (SQLException e){  
 JOptionPane.*showMessageDialog*(null**,**"ERROR!")**;** }  
 return temp**;** }  
 private Vector<pointInformation> getRotutePointData(){  
 Vector<pointInformation> temp=new Vector()**;** try{  
 ResultSet rs=cmd.executeQuery(getRotutePointMessageTable)**;** while (rs.next()){  
 pointInformation b=new pointInformation()**;** Point a=new Point()**;** a.setLocation(rs.getInt(**1**)**,**rs.getInt(**2**))**;** b.setPosition(a)**;** b.setIndex(rs.getInt(**3**))**;** b.setType(**2**)**;** temp.add(b)**;** }  
 }catch (SQLException e){  
 System.*out*.println(e.toString())**;** }  
 return temp**;** }  
 private Vector<pointInformation> combineData(){  
 Vector<pointInformation> temp=new Vector()**;** temp.addAll(getViewPointData())**;** viewPointNumber=temp.size()**;**viewCap=viewPointNumber+**1;** temp.addAll(getRotutePointData())**;** allPointNumber=temp.size()**;**cap=allPointNumber+**1;** return temp**;** }  
 public Vector<pointInformation> returnAllPointData(){  
 conSQL()**;** builtCon()**;** return combineData()**;** }  
  
 //To get main algorithm of data  
 public double[][] getGraphData(){  
 conSQL()**;** builtCon()**;** int e[][]= getViewPointLinkData()**;** int g[][]= getRotutePointLinkData()**;** e= combineToGraph(e**,**g)**;** double[][] dis=initDisArray()**;** dis=generateDisArray(dis**,**e)**;** return dis**;** }  
 private int[][] getViewPointLinkData(){  
 allPointData temp=new allPointData()**;** int e[][]=new int[cap][cap]**;** for(int i=**1;**i<viewCap**;**i++){  
 int index=**0;** double min=**5000;** for(int j=viewCap**;**j<cap**;**j++){System.*out*.println(j+" ")**;** if(temp.returnPoint(i).distance(temp.returnPoint(j))<min){  
 min=temp.returnPoint(i).distance(temp.returnPoint(j))**;** index=j**;** }  
 }  
 e[i][index]=e[index][i]=**1;** }  
 return e**;** }  
 private int[][] getRotutePointLinkData(){  
 int[][] e=new int[cap][cap]**;** try{  
 ResultSet rs=cmd.executeQuery(getGraphTable)**;** while (rs.next()){  
 e[rs.getInt(**1**)+**16**][rs.getInt(**2**)+**16**]=e[rs.getInt(**2**)+**16**][rs.getInt(**1**)+**16**]=**1;** }  
 }catch (SQLException l){  
 System.*out*.println(l.toString())**;** }  
 return e**;** }  
 private int[][] combineToGraph(int[][] e**,** int[][] g){  
 for(int i=**0;**i<cap**;**i++){  
 for (int j=**0;**j<cap**;**j++){  
 e[i][j]+=g[i][j]**;** }  
 }  
 return e**;** }  
 private double[][] initDisArray(){  
 double dis[][]=new double[cap][cap]**;** for(int i=**0;**i<cap**;**i++)  
 for(int j=**0;**j<cap**;**j++)  
 dis[i][j]=**0x3fffffff;** return dis**;** }  
 private double[][] generateDisArray(double[][] dis**,**int[][] e){  
 allPointData temp=new allPointData()**;** for(int i=**0;**i<cap**;**i++){  
 for (int j=**0;**j<cap**;**j++){  
 if(e[i][j]==**1**){  
 Point a**,**b**;** a=temp.returnPoint(i)**;** b=temp.returnPoint(j)**;** dis[i][j]=a.distance(b)**;** }  
 }  
 }  
 return dis**;** }  
  
 //Login Message  
 public int insureUser(String user**,**String password){  
 conSQL()**;** builtCon()**;** try{  
 ResultSet rs=cmd.executeQuery(getUserMessage+"'"+user+"';")**;** if(rs.next()==false)  
 return -**1;** if(user.equals(rootUser)&&rs.getString(**2**).equals(password)){  
 return **1;** }else if(rs!=null){  
 if(rs.getString(**2**).equals(password))  
 return **0;** }  
 }catch (SQLException e){  
 return -**1;** }  
 return **3;** }  
 public void addUser(String user**,**String password){  
 conSQL()**;** builtCon()**;** try{  
 cmd.executeUpdate(insertUserMessage+"('"+user+"','"+password+"')")**;** JOptionPane.*showMessageDialog*(null**,**"注册成功！")**;** }catch (SQLException e){  
 JOptionPane.*showMessageDialog*(null**,**"ERROR!")**;** }  
 }  
 public void changePassword(String user**,**String password){  
 conSQL()**;** builtCon()**;** String updatePassword="update user SET password='"+password+"' where userName='"+user+"'"**;** try{  
 int flag=cmd.executeUpdate(updatePassword)**;** if(flag==**0**)  
 JOptionPane.*showMessageDialog*(null**,**"没有这个用户！更改失败。")**;** }catch (SQLException e){  
 JOptionPane.*showMessageDialog*(null**,**"ERROR!")**;** }  
 }  
 public void deleteUser(String user**,**String password){  
 int userType=insureUser(user**,**password)**;** if(userType==**1**)  
 JOptionPane.*showMessageDialog*(null**,**"管理员账户不可删除")**;** else if(userType==**0**){  
 String reInsure=JOptionPane.*showInputDialog*(null**,**"再次确认是否删除，请输入密码：\n"**,**"是否删除？"**,**JOptionPane.*INFORMATION\_MESSAGE*)**;** userType=insureUser(user**,**reInsure)**;** if(userType==**0**){  
 try {  
 cmd.executeUpdate(deleteUser+"'"+user+"'")**;** JOptionPane.*showMessageDialog*(null**,**"删除成功！")**;** }catch (SQLException e){  
 JOptionPane.*showMessageDialog*(null**,**"ERROR")**;** }  
 }else{  
 JOptionPane.*showMessageDialog*(null**,**"取消删除")**;** }  
  
 }  
 }  
  
 //  
 public int getViewPointNumber(){  
 return viewPointNumber**;** }  
 public int getAllPointNumber(){  
 return allPointNumber**;** }  
 public connectSQL(){  
 initClickCounter()**;** }  
}

### ensureViewPointData.class:

package dataStorage**;**import java.awt.\***;**import java.util.HashMap**;**import java.util.Map**;**import java.util.Vector**;**public class ensureViewPointData {  
 private Map<Point**,**Point> hashTableToEnsureViewPoint=new HashMap()**;** public ensureViewPointData(){  
 builtHashTable()**;** }  
 public Point realViewPoint(Point clickPosition){  
 Point temp=new Point()**;** try{  
 temp=hashTableToEnsureViewPoint.get(clickPosition)**;** }catch (Exception e){  
 System.*out*.println("Fail")**;** }  
 return temp**;** }  
 private Vector<pointRowData> getAllPointMessage(){  
 connectSQL temp=new connectSQL()**;** return temp.loadData()**;** }  
 private Vector<Point> getClickPoint(Vector<pointRowData> allData){  
 Vector<Point> clickPoint=new Vector()**;** for(int i=**0;**i<allData.size()**;**i++){  
 Point temp=new Point(allData.elementAt(i).getClickPointX()**,**allData.elementAt(i).getClickPointY())**;** clickPoint.add(temp)**;** }  
 return clickPoint**;** }  
 private Vector<Point> getViewPoint(Vector<pointRowData> allData){  
 Vector<Point> viewPoint=new Vector()**;** for(int i=**0;**i<allData.size()**;**i++){  
 Point temp=new Point(allData.elementAt(i).getViewPointX()**,**allData.elementAt(i).getViewPointY())**;** viewPoint.add(temp)**;** }  
 return viewPoint**;** }  
 private void setConnect(Vector<Point> clickPoint**,**Vector<Point> viewPoint){  
 for(int i=**0;**i<clickPoint.size()**;**i++){  
 hashTableToEnsureViewPoint.put(clickPoint.elementAt(i)**,**viewPoint.elementAt(i))**;** }  
 }  
 private void builtHashTable(){  
 Vector<pointRowData> temp=getAllPointMessage()**;** Vector<Point> clickPoint=getClickPoint(temp)**;** Vector<Point> viewPoint=getViewPoint(temp)**;** setConnect(clickPoint**,**viewPoint)**;** }  
}

### pointInformation.class:

package dataStorage**;**import java.awt.\***;**public class pointInformation {  
 private Point position**;** private int index**;** private int type**;** private final int view=**1;** private final int rotute=**2;** public pointInformation(){  
 position=new Point()**;** index=**0;** type=**0;** }  
 public void setPosition(Point position){  
 this.position=position**;** }  
 public void setIndex(int index){  
 this.index=index**;** }  
 public void setType(int type){  
 this.type=type**;** }  
 public Point getPosition(){  
 return position**;** }  
 public int getIndex(){  
 return index**;** }  
 public int getType(){  
 return type**;** }  
 public boolean isRotute(){  
 return type==rotute**;** }  
 public boolean isView(){  
 return type==view**;** }  
}

### pointRowData.class:

package dataStorage**;**public class pointRowData {  
 private int clickPointX**,**clickPointY**;** private int viewPointX**,**viewPointY**;** pointRowData(){  
 this.clickPointX=this.clickPointY=this.viewPointY=this.viewPointX=**0;** }  
 public void setClickPointX(int x){  
 clickPointX=x**;** }  
 public void setClickPointY(int y){  
 clickPointY=y**;** }  
 public void setViewPointX(int x){  
 viewPointX=x**;** }  
 public void setViewPointY(int y){  
 viewPointY=y**;** }  
 public int getClickPointX(){  
 return clickPointX**;** }  
 public int getClickPointY(){  
 return clickPointY**;** }  
 public int getViewPointX(){  
 return viewPointX**;** }  
 public int getViewPointY(){  
 return viewPointY**;** }  
}

### returnViewPointMessage.class:

package dataStorage**;**import java.awt.\***;**import java.util.HashMap**;**import java.util.Map**;**import java.util.Vector**;**public class returnViewPointMessage {  
 private Map<Point**,**txtMessage> hashTabelViewMessage=new HashMap()**;** private Vector<viewMessage> getAllData(){  
 connectSQL temp=new connectSQL()**;** return temp.loadViewMessage()**;** }  
 private Vector<Point> getPointMessage(Vector<viewMessage> allData){  
 Vector<Point> temp=new Vector()**;** for(int i=**0;**i<allData.size()**;**i++){  
 Point tempPoint=new Point(allData.elementAt(i).getPointX()**,**allData.elementAt(i).getPointY())**;** temp.add(tempPoint)**;** }  
 return temp**;** }  
 private Vector<txtMessage> getTxtMessage(Vector<viewMessage> allData){  
 Vector<txtMessage> temp=new Vector()**;** for(int i=**0;**i<allData.size()**;**i++){  
 txtMessage tempTxt=new txtMessage()**;** tempTxt.setName(allData.elementAt(i).getViewName())**;** tempTxt.setTxt(allData.elementAt(i).getTxt())**;** temp.add(tempTxt)**;** }  
 return temp**;** }  
 private void buildHashTable(){  
 Vector<viewMessage> allData=getAllData()**;** Vector<Point> key=getPointMessage(allData)**;** Vector<txtMessage> values=getTxtMessage(allData)**;** for(int i=**0;**i<key.size()**;**i++){  
 hashTabelViewMessage.put(key.elementAt(i)**,**values.elementAt(i))**;** }  
 }  
 public returnViewPointMessage(){  
 buildHashTable()**;** }  
 public txtMessage getViewPointMessage(Point click){  
 return hashTabelViewMessage.get(click)**;** }  
}

### txtMessage.class:

package dataStorage**;**public class txtMessage {  
 private String name**;** private String txt**;** public void setName(String name){  
 this.name=name**;** }  
 public void setTxt(String txt){  
 this.txt=txt**;** }  
 public String getName(){  
 return name**;** }  
 public String getTxt(){  
 return txt**;** }  
}

### viewMessage.class:

package dataStorage**;**public class viewMessage {  
 private int pointX**;** private int pointY**;** private String viewName**;** private String txt**;** public void setPointX(int x){  
 pointX=x**;** }  
 public void setPointY(int y){  
 pointY=y**;** }  
 public int getPointX(){  
 return pointX**;** }  
 public int getPointY(){  
 return pointY**;** }  
 public void setViewName(String name){  
 viewName=name**;** }  
 public void setTxt(String txt){  
 this.txt=txt**;** }  
 public String getViewName(){  
 return viewName**;** }  
 public String getTxt(){  
 return txt**;** }  
}

### shortestPath.class:

package graphAlgorithm**;**import dataStorage.\***;**import javax.swing.\***;**import java.awt.\***;**import java.util.Vector**;**import java.util.\***;**public class shortestPath {  
 private double e[][]**;** private connectSQL dataCon=new connectSQL()**;** private int cap=**0,**viewPointCap=**0;** private final double inf=**0x3fffffff;** private int[] pre**;** private boolean[] vis**;** private double[] dis**;** private allPointData getPoint**;** private returnViewPointMessage getViewMessage=new returnViewPointMessage()**;** Vector<String> allMessage=new Vector()**;** public shortestPath(){  
 getPoint=new allPointData()**;** dataCon.returnAllPointData()**;** e =dataCon.getGraphData()**;** cap=dataCon.getAllPointNumber()+**1;** viewPointCap=dataCon.getViewPointNumber()+**1;** pre=new int[cap]**;**System.*out*.println(cap+" \*\* "+viewPointCap)**;** vis=new boolean[cap]**;** dis=new double[cap]**;** }  
 public void init(){  
 //popIndex.clear();  
 for(int i=**0;**i<cap**;**i++){  
 pre[i]=i**;** dis[i]=inf**;** vis[i]=false**;** }  
 }  
 public void Dijkstra(int index){  
 init()**;** for(int i=**1;**i<cap**;**i++){  
 if(e[index][i]!=inf)  
 dis[i]=e[index][i]**;** }  
 dis[index]=**0;** for(int i=**0;**i<cap**;**i++){  
 int u=-**1;** double min=inf**;** for(int j=**1;**j<cap**;**j++){  
 if(vis[j]==false&&min>dis[j]){  
 min=dis[j]**;** u=j**;** }  
 }  
 if(u==-**1**)  
 break**;** vis[u]=true**;** for(int v=**1;**v<cap**;**v++){  
 if(vis[v]==false&&e[u][v]!=inf){  
 if(dis[v]>dis[u]+e[u][v]){  
 dis[v]=dis[u]+e[u][v]**;** pre[v]=u**;** }  
 }  
 }  
 }  
 }  
 public Vector<Point> shortPath(int start**,**int end){  
 Dijkstra(start)**;** Vector<Point> path=linkPath(start**,**end)**;** allMessage.add(dis[end]+"")**;** allMessage.addAll(returnViewName(path))**;** return path**;** }  
 private Vector<String > returnViewName(Vector<Point> path){  
 Vector<String > nameSet=new Vector()**;** for(int i=path.size()-**1;**i>=**0;**i--){  
 txtMessage temp=getViewMessage.getViewPointMessage(path.elementAt(i))**;** if(temp==null)  
 continue**;** nameSet.add(temp.getName())**;** }  
 return nameSet**;** }  
 private Vector<Point> linkPath(int start**,**int end){  
 Vector<Point> path=new Vector()**;** while (pre[end]!=end){  
 path.add(getPoint.returnPoint(end))**;** end=pre[end]**;** }  
 path.add(getPoint.returnPoint(end))**;** path.add(getPoint.returnPoint(start))**;** return path**;** }  
 public Vector<Point> shortPathInGreatPoint(Vector<Integer> pointIndex){  
 Vector<Point> path=new Vector()**;** boolean visit[]=new boolean[viewPointCap]**;** int index=pointIndex.elementAt(**0**)**;** double distance=**0;** for(int i=**0;**i<pointIndex.size()**;**i++){  
 Vector<Point> temp=new Vector()**;** int start=index**;** visit[index]=true**;** Dijkstra(index)**;** double min=inf**;** for(int j=**0;**j<pointIndex.size()**;**j++){  
 if(visit[pointIndex.elementAt(j)]==false){  
 if(min>dis[pointIndex.elementAt(j)]){  
 min=dis[pointIndex.elementAt(j)]**;** index=pointIndex.elementAt(j)**;** }  
 }  
 }  
 int end=index**;** temp=linkPath(start**,**end)**;** distance+=dis[end]**;** for(int j=temp.size()-**1;**j>=**0;**j--){  
 path.add(temp.elementAt(j))**;** }  
 allMessage.add(dis[end]+"")**;** allMessage.addAll(returnViewName(temp))**;** }  
 for(int i=**0;**i<**3;**i++)  
 allMessage.remove(allMessage.size()-**1**)**;** allMessage.add(distance+" \n")**;** return path**;** }  
 public void toPrintMessage(){  
 String allData=new String()**;** for(int i=**0;**i<allMessage.size()**;**i++){  
 if(i%**3**==**0**&&i==allMessage.size()-**1**)  
 allData+="总"**;** if(i%**3**==**0**)  
 allData+="距离："+allMessage.elementAt(i)+"\n"**;** if(i%**3**==**1**)  
 allData+="从："+allMessage.elementAt(i)+"\n"**;** else if(i%**3**==**2**)  
 allData+="到："+allMessage.elementAt(i)+"\n\n"**;** }  
 JOptionPane.*showMessageDialog*(null**,**allData)**;** allMessage.clear()**;** }  
}