# 课程设计：校园导航系统

目录：

需求分析---------------------------------------------------------------------------------------------------------1

概要设计---------------------------------------------------------------------------------------------------------2

详细设计---------------------------------------------------------------------------------------------------------7

调试分析-------------------------------------------------------------------------------------------------------18

课程设计总结-------------------------------------------------------------------------------------------------19

1. **需求分析**

新生入学需要一定的时间才能熟悉河海大学（江宁校区）的地理信息。对于起航周，新生需要自主查询一些教室、办公楼的地理位置信息，才能知晓对应活动的地点。同样对于一些校外参观人员，若想参观某一特定地点，在一般的APP导航中是无法找到详尽到校园内某一处的地点的。为此，我小组编写校园导航系统，为新生与参观人员提供校内主要地点的信息资料，并可以查询所给出的主要地点中，任意两点间的最短路径。

1. 地图信息存储(金刘宇、苗凯龙)
2. 利用GraphMartix类来存储地图信息。地图中的每一个顶点都是一处地标/建筑物。
3. 顶点的数据类型规定为Position类，包含编号、名称、功能介绍三个信息。
4. 地图中的边代表了从该点到其临界点所需的时间长短，边权重的信息存储在GraphMartix类中的arcs[ ][ ]邻接矩阵中。利用Dist类形成的Dist数组来存放以某一点Start为起点时，初始化的Dist数组（利用arcs[start][i] i=[0,n] n代表地图结点数量）。

2． 查询地标/建筑物的基本信息(刘生伟)

提供getInfo( )函数，用户通过输入地标名称，查询GraphMartix对应的顶点表Vex中对应的建筑物/地标信息，返回该点编号、名称、功能信息。

1. 列出全部地点（刘生伟）

提供getAllPositon( )函数，向用户返回系统中所存储的主要地标的名称。

1. 计算某两点间的最短信息（刘生伟）

提供PathCalculator类，利用其中Dijkstra算法，计算某两点间通行时间最短的一条路径。

1. 地标名转换地标Id（刘生伟）

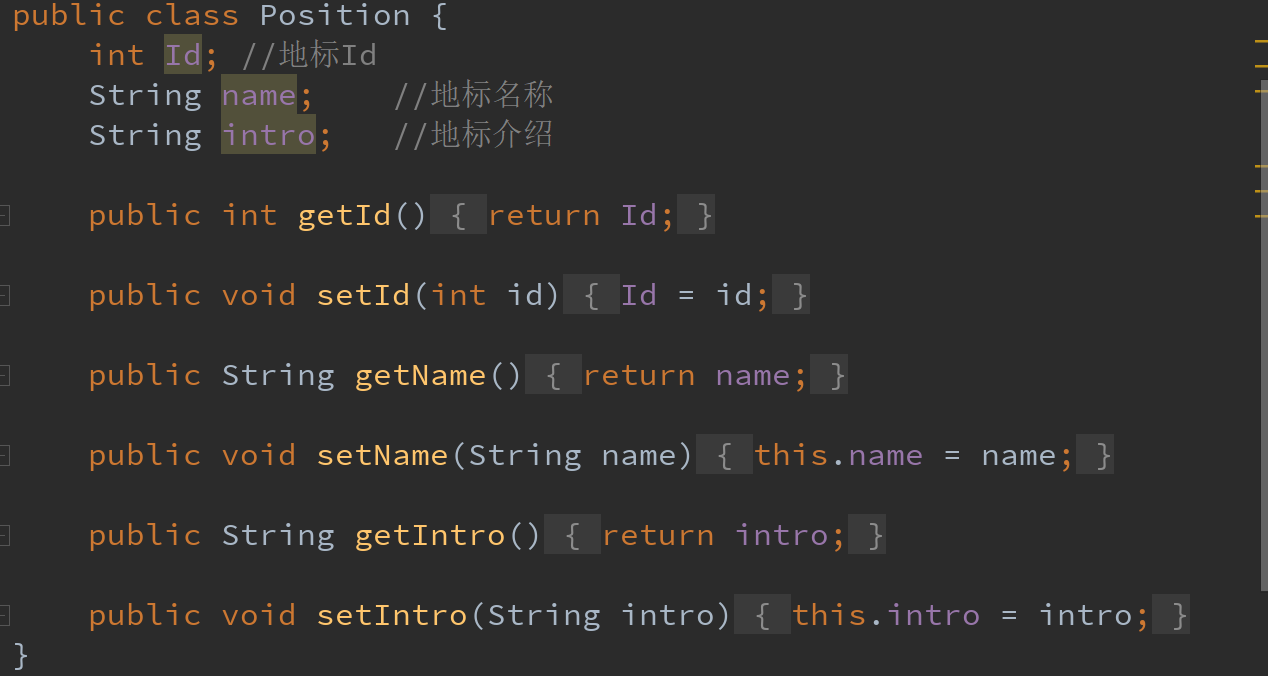
提供convert函数，通过输入地标名，返回地标名对应的序号，为Pathcalculator服务。

1. 用户界面（刘生伟）

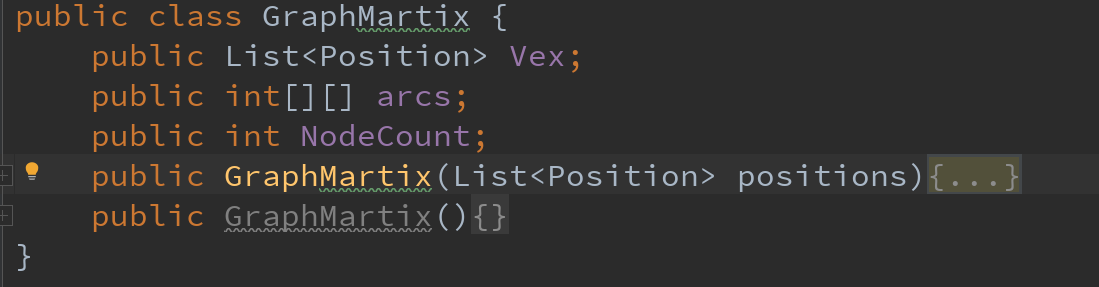
通过控制台输出一个简单的用户交流界面。

1. **概要设计**
2. 地图信息存储

I. 图中顶点Position定义



II. 图类 GraphMartix的定义

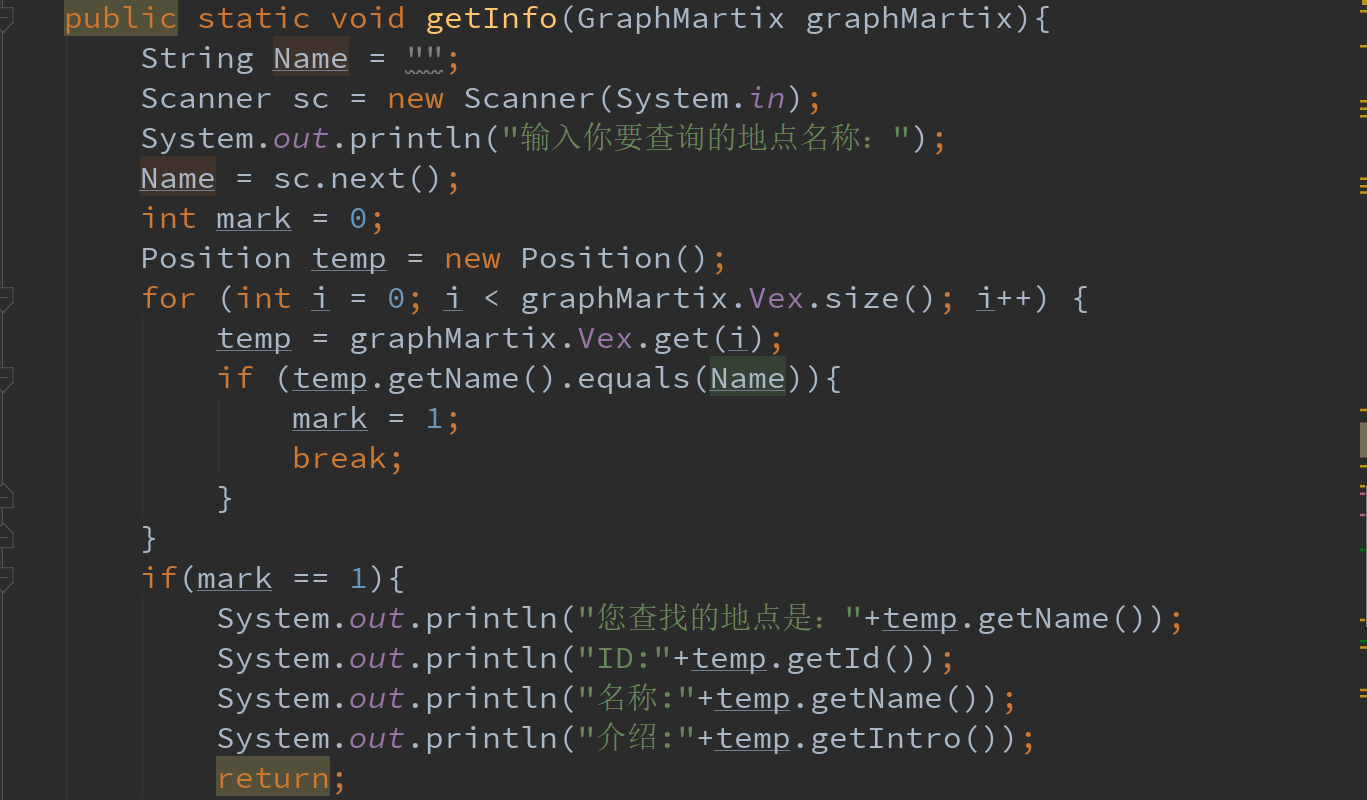


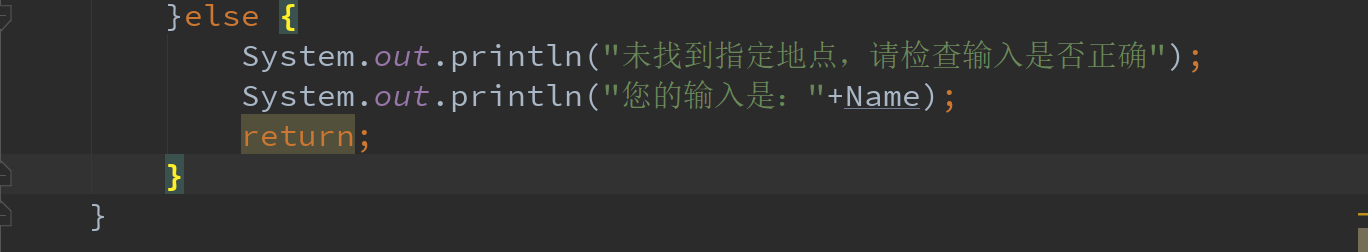
III. Dist数组定义



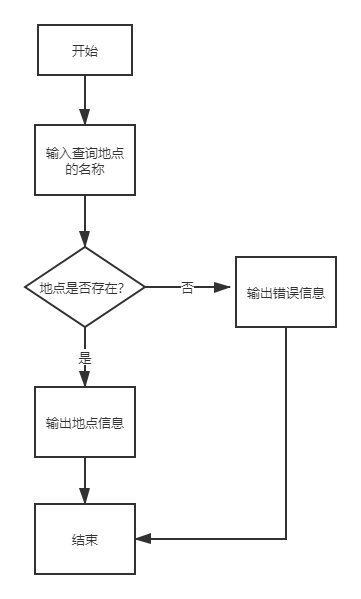
1. getInfo( )函数

源代码：



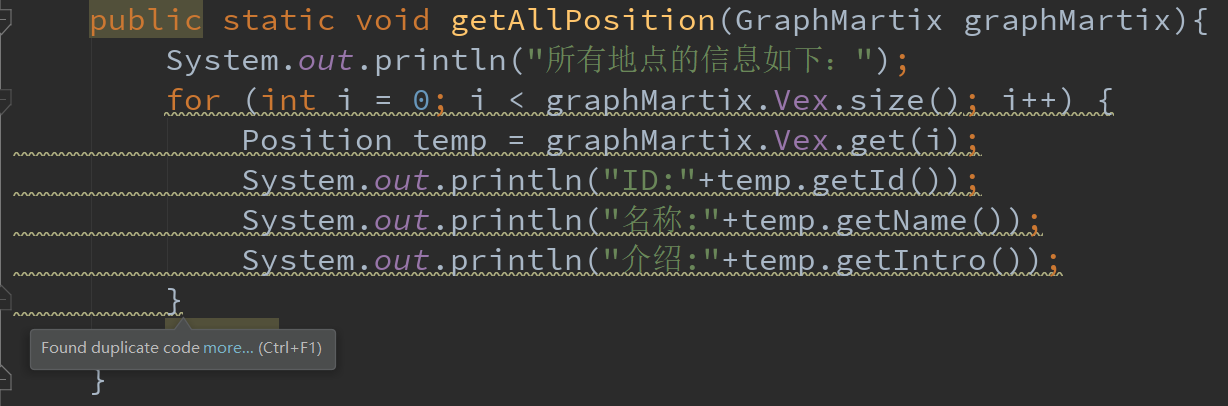


流程图:

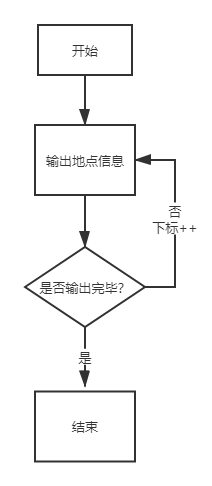


1. getAllPositon( )函数

源代码：

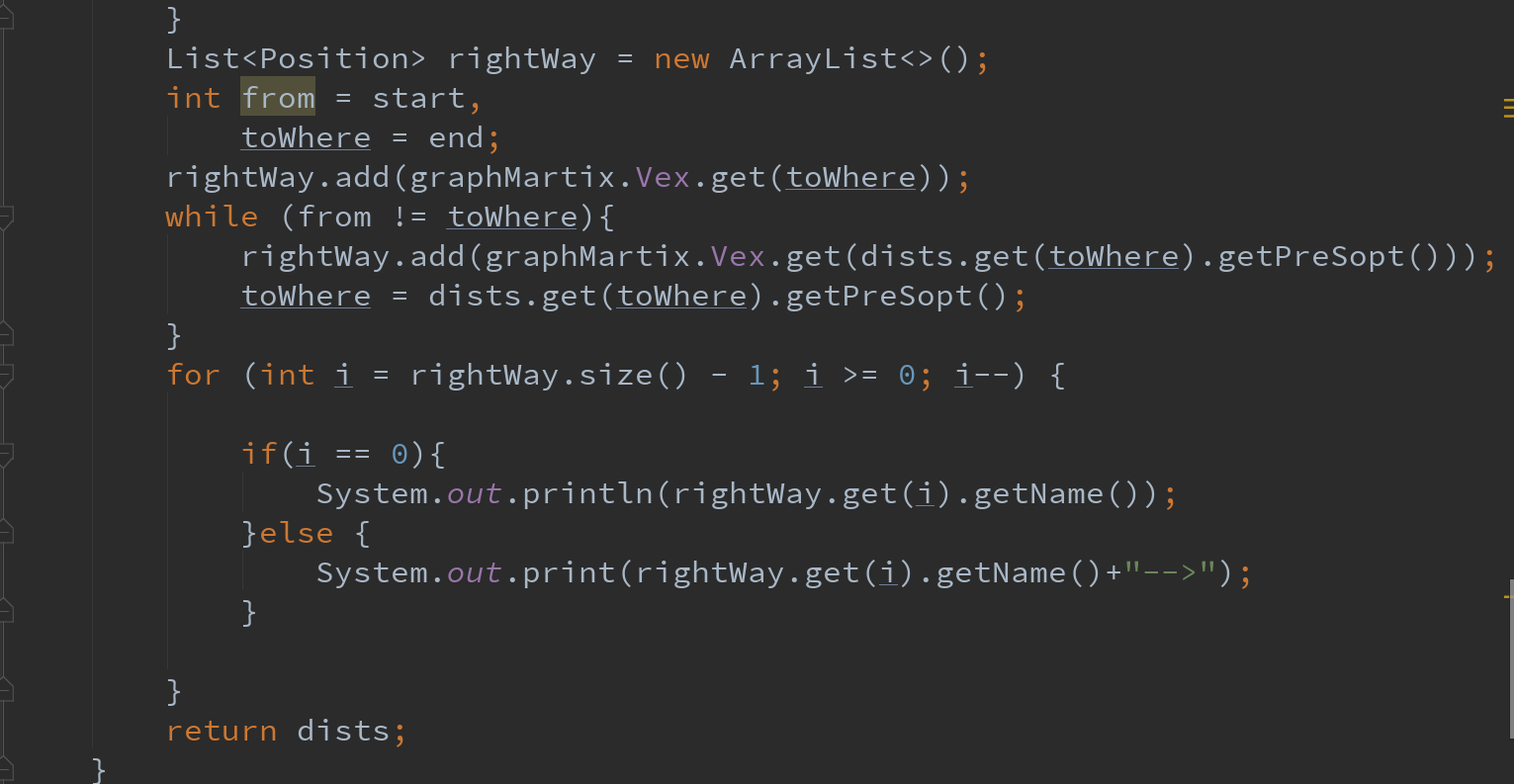
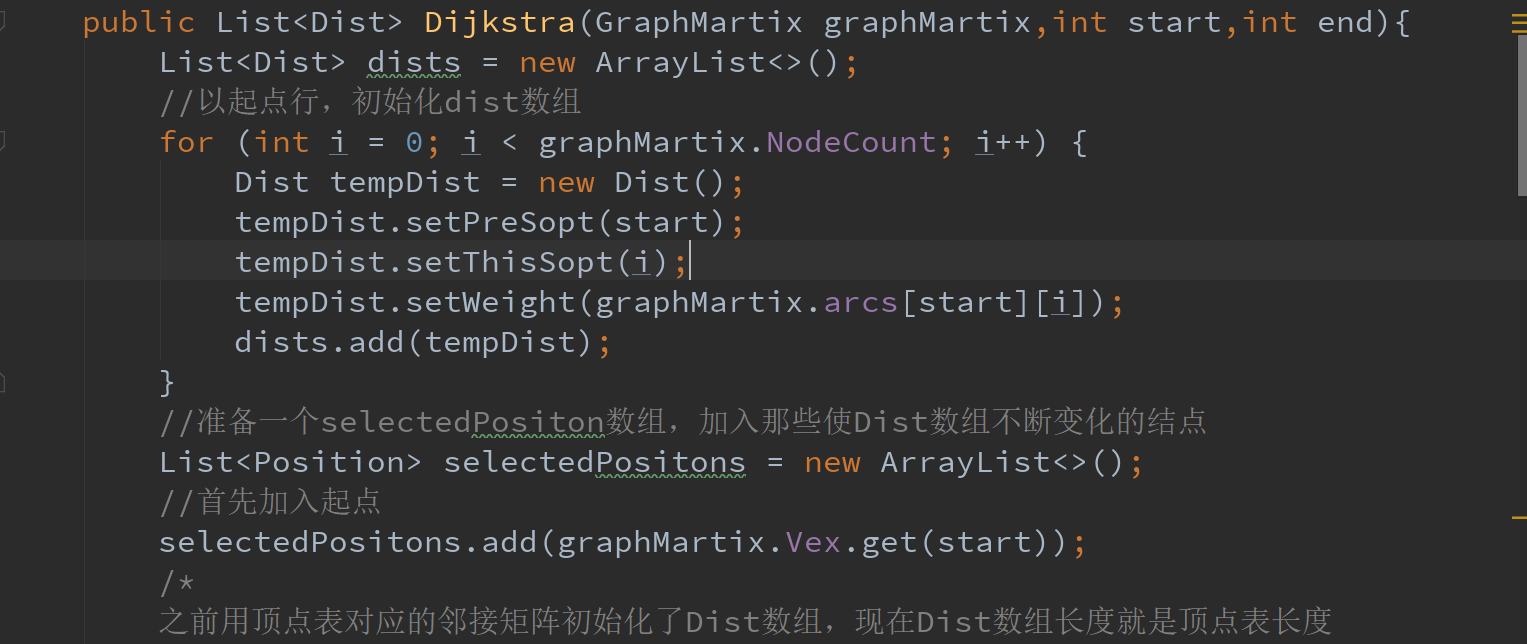


流程图：

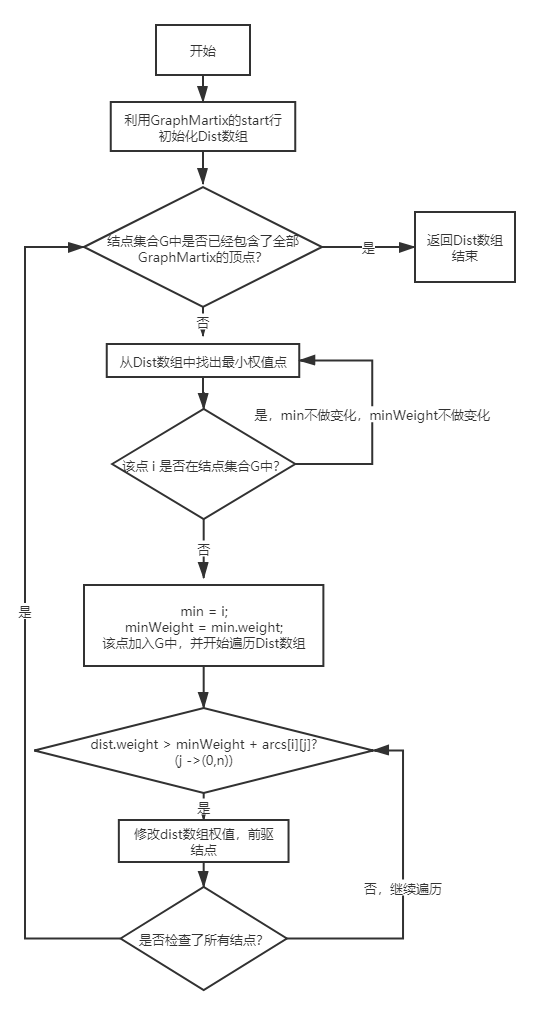


1. getQuickestWay( )函数

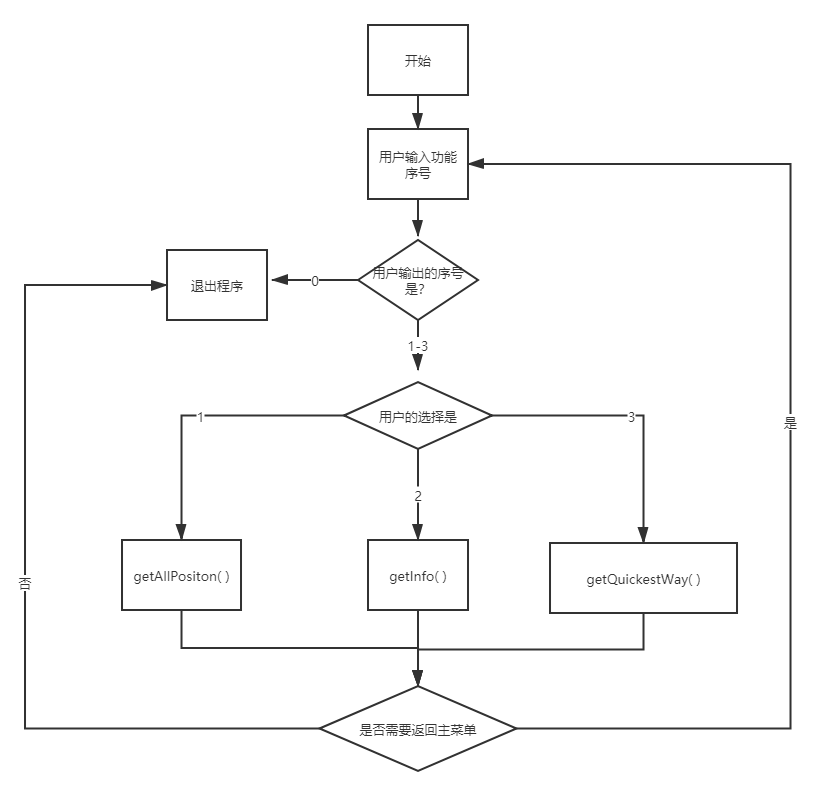
源代码



程序流程图：



1. **详细设计**
2. 源程序代码流程图



1. 源程序代码

SchoolNav.java

import Bean.GraphMartix;

import Bean.Position;  
import Utils.PathCalculator;  
import Utils.PositionIniter;  
  
import java.util.List;  
import java.util.Scanner;  
  
*/\*\*  
 \* Created with IntelliJ IDEA.  
 \*  
 \** ***@Auther:*** *Piweiii  
 \** ***@Date:*** *2020/09/07/16:39  
 \** ***@Description:*** *\*/*public class SchoolNav {  
 public static void main(String[] args) {  
 PositionIniter psit = new PositionIniter();  
 List<Position> positions = psit.initPos();  
 GraphMartix graphMartix = new GraphMartix(positions);  
 PathCalculator pclr = new PathCalculator();  
 Scanner sc = new Scanner(System.*in*);  
 int choice = 0,  
 start = 0,  
 end = 0;  
 while (true){  
 System.*out*.println("=================欢迎使用河海大学校园导航系统==================");  
 System.*out*.println("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 0.退出系统 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|");  
 System.*out*.println("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 1.查询所有地点 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|");  
 System.*out*.println("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 2.输入地点名称查询信息 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|");  
 System.*out*.println("|\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* 3.查询某两点的最短路径 \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*|");  
 System.*out*.println("==========================================================");  
 System.*out*.println("输入功能序号：");  
 choice = sc.nextInt();  
 switch (choice){  
 case 0:  
 System.*out*.println("感谢您的使用");  
 return;  
 case 1:  
 *getAllPosition*(graphMartix);  
 break;  
 case 2:  
 *getInfo*(graphMartix);  
 break;  
 case 3:  
 System.*out*.println("输入起点：");  
 String from = sc.next();  
 System.*out*.println("输入终点：");  
 String toWhere = sc.next();  
 start = *convert*(graphMartix,from);  
 end = *convert*(graphMartix,toWhere);  
 pclr.Dijkstra(graphMartix,start,end);  
 }  
 }  
 }  
 public static void getAllPosition(GraphMartix graphMartix){  
 System.*out*.println("所有地点的信息如下：");  
 for (int i = 0; i < graphMartix.Vex.size(); i++) {  
 Position temp = graphMartix.Vex.get(i);  
 System.*out*.println("ID:"+temp.getId());  
 System.*out*.println("名称:"+temp.getName());  
 System.*out*.println("介绍:"+temp.getIntro());  
 }  
 return;  
 }  
  
 public static void getInfo(GraphMartix graphMartix){  
 String Name = "";  
 Scanner sc = new Scanner(System.*in*);  
 System.*out*.println("输入你要查询的地点名称：");  
 Name = sc.next();  
 int mark = 0;  
 Position temp = new Position();  
 for (int i = 0; i < graphMartix.Vex.size(); i++) {  
 temp = graphMartix.Vex.get(i);  
 if (temp.getName().equals(Name)){  
 mark = 1;  
 break;  
 }  
 }  
 if(mark == 1){  
 System.*out*.println("您查找的地点是："+temp.getName());  
 System.*out*.println("ID:"+temp.getId());  
 System.*out*.println("名称:"+temp.getName());  
 System.*out*.println("介绍:"+temp.getIntro());  
 return;  
 }else {  
 System.*out*.println("未找到指定地点，请检查输入是否正确");  
 System.*out*.println("您的输入是："+Name);  
 return;  
 }  
 }  
  
 public static int convert(GraphMartix graphMartix,String Name){  
 Position temp = new Position();  
 for (int i = 0; i < graphMartix.Vex.size(); i++) {  
 temp = graphMartix.Vex.get(i);  
 if(temp.getName().equals(Name)){  
 return i;  
 }  
 }  
 return -1;  
 }  
}

Utils包：

(1).PositionIniter.java

package Utils;

import Bean.Position;  
  
import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* Created with IntelliJ IDEA.  
 \*  
 \** ***@Auther:*** *Piweiii  
 \** ***@Date:*** *2020/09/07/16:40  
 \** ***@Description:*** *\*/*public class PositionIniter {  
 int NodeCount = 10;  
 String[] postionName = {"新食堂",  
 "十二舍",  
 "西操场",  
 "图书馆",  
 "致用楼",  
 "超市",  
 "致高楼",  
 "二十三舍",  
 "勤学楼",  
 "博学楼"};  
 String[] postionIntro = {"新食堂提供各种各样的美食，价格便宜。",  
 "男生宿舍，休养生息。",  
 "你可以在这里锻炼身体。",  
 "你可以在这里借阅书籍，自习，喝咖啡与休息。",  
 "你可以在这里上公共课，借用教室展开社团活动。",  
 "你可以采购生活的必需品。",  
 "你可以在这里上选修课，借用模拟法庭。",  
 "女生宿舍，休养生息。",  
 "你可以在这里找到信息学部的老师，借用实验室。",  
 "你可以在这里上专业课，享受有空调的教室。"};  
 int[] numbers = {1,2,3,4,5,6,7,8,9,10};  
 public List<Position> initPos(){  
 List<Position> positions = new ArrayList<>();  
 for (int i = 0; i < NodeCount; i++) {  
 Position position = new Position();  
 position.setId(numbers[i]);  
 position.setName(postionName[i]);  
 position.setIntro(postionIntro[i]);  
 positions.add(position);  
 }  
 return positions;  
 }  
}

(2).PathCalculator.java

package Utils;

import Bean.Dist;  
import Bean.GraphMartix;  
import Bean.Position;  
  
import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* Created with IntelliJ IDEA.  
 \*  
 \** ***@Auther:*** *Piweiii  
 \** ***@Date:*** *2020/09/07/20:29  
 \** ***@Description:*** *\*/*public class PathCalculator {  
 public List<Dist> Dijkstra(GraphMartix graphMartix,int start,int end){  
 List<Dist> dists = new ArrayList<>();  
 //以起点行，初始化dist数组  
 for (int i = 0; i < graphMartix.NodeCount; i++) {  
 Dist tempDist = new Dist();  
 tempDist.setPreSopt(start);  
 tempDist.setThisSopt(i);  
 tempDist.setWeight(graphMartix.arcs[start][i]);  
 dists.add(tempDist);  
 }  
 //准备一个selectedPositon数组，加入那些使Dist数组不断变化的结点  
 List<Position> selectedPositons = new ArrayList<>();  
 //首先加入起点  
 selectedPositons.add(graphMartix.Vex.get(start));  
 /\*  
 之前用顶点表对应的邻接矩阵初始化了Dist数组，现在Dist数组长度就是顶点表长度  
 所以，当selectedPosition集合大小小于Dist数组时，证明Dijkstra算法没有完成  
 需要继续添加顶点，演化Dist数组  
 \*/  
 while (selectedPositons.size() < dists.size()){  
 //每次开始寻找最小权，都先将min和minWeight做一个初始化  
 int min = 0; int minWeight = 9999999;  
 //遍历整个Dist数组，才能找到最小权  
 for (int i = 0; i < dists.size(); i++) {  
 /\*  
 该循环的条件是：当该点的权值小于最小权，  
 并且该点不在selectedPosition数组中，  
 将其选作下一个待加入selectedPosition集合的元素，  
 方法是改变min的下标。  
 \*/  
 if(dists.get(i).getWeight() < minWeight &&  
 !selectedPositons.contains(graphMartix.Vex.get(i))){  
 min = i;  
 minWeight = dists.get(i).getWeight();  
 }  
 }  
 /\*  
 退出for循环，证明找到了在之前演化之后的Dist数组中，最小的一个权值  
 将该权值对应的顶点加入到selectedPosition中，表明将从该点，继续  
 演化Dist数组  
 \*/  
 selectedPositons.add(graphMartix.Vex.get(min));  
 for (int i = 0; i < dists.size(); i++) {  
 if(dists.get(i).getWeight()  
 > minWeight + graphMartix.arcs[min][i]){  
 dists.get(i).setPreSopt(min);  
 dists.get(i).setWeight(minWeight + graphMartix.arcs[min][i]);  
 }  
 }  
/\* //打印Dist数组的演化过程  
 for (int i = 0; i < dists.size(); i++) {  
 System.out.print("("+dists.get(i).getPreSopt()+  
 ","+dists.get(i).getThisSopt()+")"+  
 dists.get(i).getWeight()+" ");  
 }  
 System.out.println();\*/  
 }  
 List<Position> rightWay = new ArrayList<>();  
 int from = start,  
 toWhere = end;  
 rightWay.add(graphMartix.Vex.get(toWhere));  
 while (from != toWhere){  
 rightWay.add(graphMartix.Vex.get(dists.get(toWhere).getPreSopt()));  
 toWhere = dists.get(toWhere).getPreSopt();  
 }  
 for (int i = rightWay.size() - 1; i >= 0; i--) {  
  
 if(i == 0){  
 System.*out*.println(rightWay.get(i).getName());  
 }else {  
 System.*out*.print(rightWay.get(i).getName()+"-->");  
 }  
  
 }  
 return dists;  
 }  
}

Bean包

(1).Dist.java

package Bean;

*/\*\*  
 \* Created with IntelliJ IDEA.  
 \*  
 \** ***@Auther:*** *Piweiii  
 \** ***@Date:*** *2020/09/08/21:18  
 \** ***@Description:*** *\*/*public class Dist {  
 int preSopt;  
 int thisSopt;  
 int weight;  
  
 public int getWeight() {  
 return weight;  
 }  
  
 public void setWeight(int weight) {  
 this.weight = weight;  
 }  
  
 public int getThisSopt() {  
 return thisSopt;  
 }  
  
 public void setThisSopt(int thisSopt) {  
 this.thisSopt = thisSopt;  
 }  
  
 public int getPreSopt() {  
 return preSopt;  
 }  
  
 public void setPreSopt(int preSopt) {  
 this.preSopt = preSopt;  
 }  
}

(2).GraphMartix.java

package Bean;

import java.util.ArrayList;  
import java.util.List;  
  
*/\*\*  
 \* Created with IntelliJ IDEA.  
 \*  
 \** ***@Auther:*** *Piweiii  
 \** ***@Date:*** *2020/09/07/19:06  
 \** ***@Description:*** *\*/*public class GraphMartix {  
 public List<Position> Vex;  
 public int[][] arcs;  
 public int NodeCount;  
 public GraphMartix(List<Position> positions){  
 int nodeCount = positions.size();  
 NodeCount = nodeCount;  
 Vex = new ArrayList<>();  
 arcs = new int[NodeCount][NodeCount];  
 for (int i = 0; i < NodeCount; i++) {  
 Vex.add(positions.get(i));  
 arcs[i][i] = 0;  
 }  
 //初始化邻接矩阵  
 //第一行  
 arcs[0][1] = arcs[1][0] = 500;  
 arcs[0][2] = arcs[2][0] = 200;  
 arcs[0][3] = arcs[3][0] = 9999999;  
 arcs[0][4] = arcs[4][0] = 9999999;  
 arcs[0][5] = arcs[5][0] = 9999999;  
 arcs[0][6] = arcs[6][0] = 9999999;  
 arcs[0][7] = arcs[7][0] = 9999999;  
 arcs[0][8] = arcs[8][0] = 9999999;  
 arcs[0][9] = arcs[9][0] = 9999999;  
 //第二行  
 arcs[1][2] = arcs[2][1] = 50;  
 arcs[1][3] = arcs[3][1] = 9999999;  
 arcs[1][4] = arcs[4][1] = 500;  
 arcs[1][5] = arcs[5][1] = 9999999;  
 arcs[1][6] = arcs[6][1] = 9999999;  
 arcs[1][7] = arcs[7][1] = 9999999;  
 arcs[1][8] = arcs[8][1] = 9999999;  
 arcs[1][9] = arcs[9][1] = 9999999;  
 //第三行  
 arcs[2][3] = arcs[3][2] = 1000;  
 arcs[2][4] = arcs[4][2] = 9999999;  
 arcs[2][5] = arcs[5][2] = 9999999;  
 arcs[2][6] = arcs[6][2] = 9999999;  
 arcs[2][7] = arcs[7][2] = 9999999;  
 arcs[2][8] = arcs[8][2] = 9999999;  
 arcs[2][9] = arcs[9][2] = 9999999;  
 //第四行  
 arcs[3][4] = arcs[4][3] = 100;  
 arcs[3][5] = arcs[5][3] = 9999999;  
 arcs[3][6] = arcs[6][3] = 50;  
 arcs[3][7] = arcs[7][3] = 9999999;  
 arcs[3][8] = arcs[8][3] = 500;  
 arcs[3][9] = arcs[9][3] = 9999999;  
 //第五行  
 arcs[4][5] = arcs[5][4] = 250;  
 arcs[4][6] = arcs[6][4] = 200;  
 arcs[4][7] = arcs[7][4] = 9999999;  
 arcs[4][8] = arcs[8][4] = 9999999;  
 arcs[4][9] = arcs[9][4] = 9999999;  
 //第六行  
 arcs[5][6] = arcs[6][5] = 350;  
 arcs[5][7] = arcs[7][5] = 9999999;  
 arcs[5][8] = arcs[8][5] = 9999999;  
 arcs[5][9] = arcs[9][5] = 9999999;  
 //第七行  
 arcs[6][7] = arcs[7][6] = 1000;  
 arcs[6][8] = arcs[8][6] = 9999999;  
 arcs[6][9] = arcs[9][6] = 9999999;  
 //第八行  
 arcs[7][8] = arcs[8][7] = 9999999;  
 arcs[7][9] = arcs[9][7] = 9999999;  
 //第九行  
 arcs[8][9] = arcs[9][8] = 100;  
 }  
 public GraphMartix(){  
  
 }  
}

(3).Position.java

package Bean;

*/\*\*  
 \* Created with IntelliJ IDEA.  
 \*  
 \** ***@Auther:*** *Piweiii  
 \** ***@Date:*** *2020/09/07/16:36  
 \** ***@Description:*** *\*/*public class Position {  
 int Id; //地标Id  
 String name; //地标名称  
 String intro; //地标介绍  
  
 public int getId() {  
 return Id;  
 }  
  
 public void setId(int id) {  
 Id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public String getIntro() {  
 return intro;  
 }  
  
 public void setIntro(String intro) {  
 this.intro = intro;  
 }  
}

Test包：

Test.java

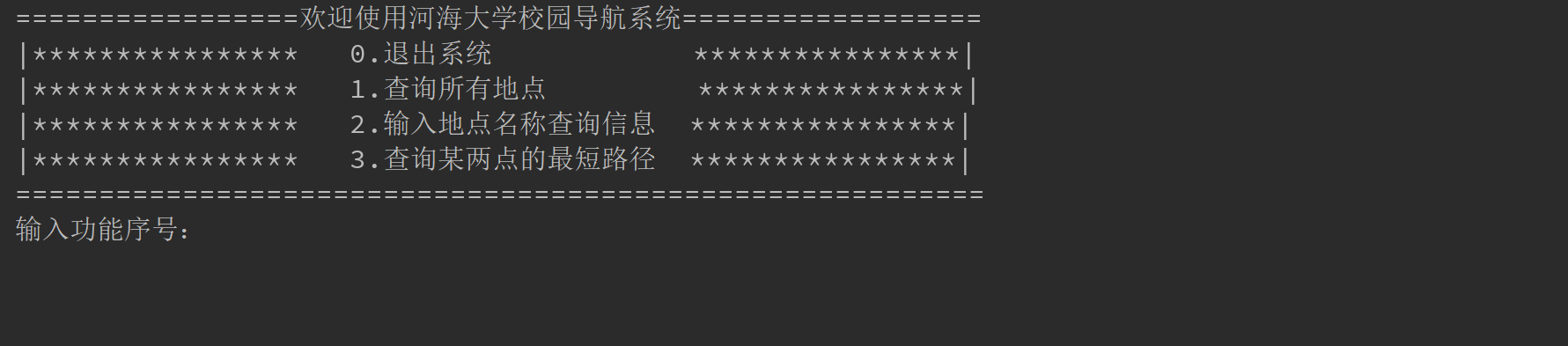
package Test;

import Bean.Dist;  
import Bean.GraphMartix;  
import Bean.Position;  
import Utils.PathCalculator;  
import Utils.PositionIniter;  
  
import java.util.List;  
  
*/\*\*  
 \* Created with IntelliJ IDEA.  
 \*  
 \** ***@Auther:*** *Piweiii  
 \** ***@Date:*** *2020/09/08/21:12  
 \** ***@Description:*** *\*/*public class Test {  
 public static void main(String[] args) {  
 PositionIniter psit = new PositionIniter();  
 List<Position> positions = psit.initPos();  
 GraphMartix graphMartix = new GraphMartix(positions);  
 PathCalculator pclr = new PathCalculator();  
 List<Dist> dists = pclr.Dijkstra(graphMartix,3,0);  
 for (int i = 0; i < graphMartix.Vex.size(); i++) {  
 Position temp = graphMartix.Vex.get(i);  
 System.*out*.println("ID:"+temp.getId());  
 System.*out*.println("名称:"+temp.getName());  
 System.*out*.println("介绍:"+temp.getIntro());  
 }  
 for (int i = 0; i < graphMartix.NodeCount; i++) {  
 for (int j = 0; j < graphMartix.NodeCount; j++) {  
 if(j == graphMartix.NodeCount - 1){  
 System.out.println(graphMartix.arcs[i][j]);  
 }else {  
 System.out.print(graphMartix.arcs[i][j]+" ");  
 }  
 }  
 }  
 System.out.println("dist数组初始化测试");  
 for (int i = 0; i < dists.size(); i++) {  
 System.out.println("pre:"+dists.get(i).getPreSopt()+  
 " this:"+dists.get(i).getThisSopt()+  
 " weight:"+dists.get(i).getWeight());  
 }  
  
 }  
}

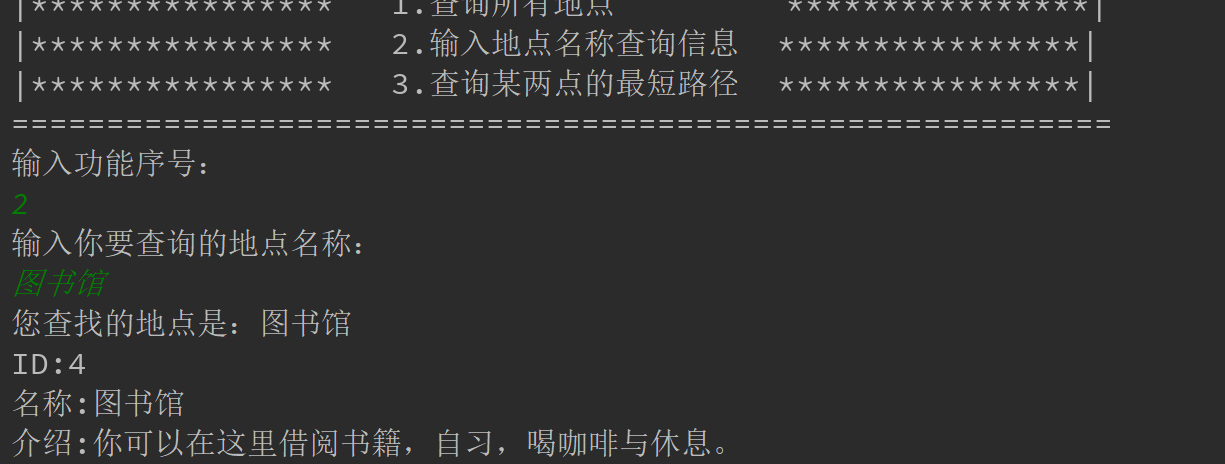
1. **调试分析**

Part 1：数据测试

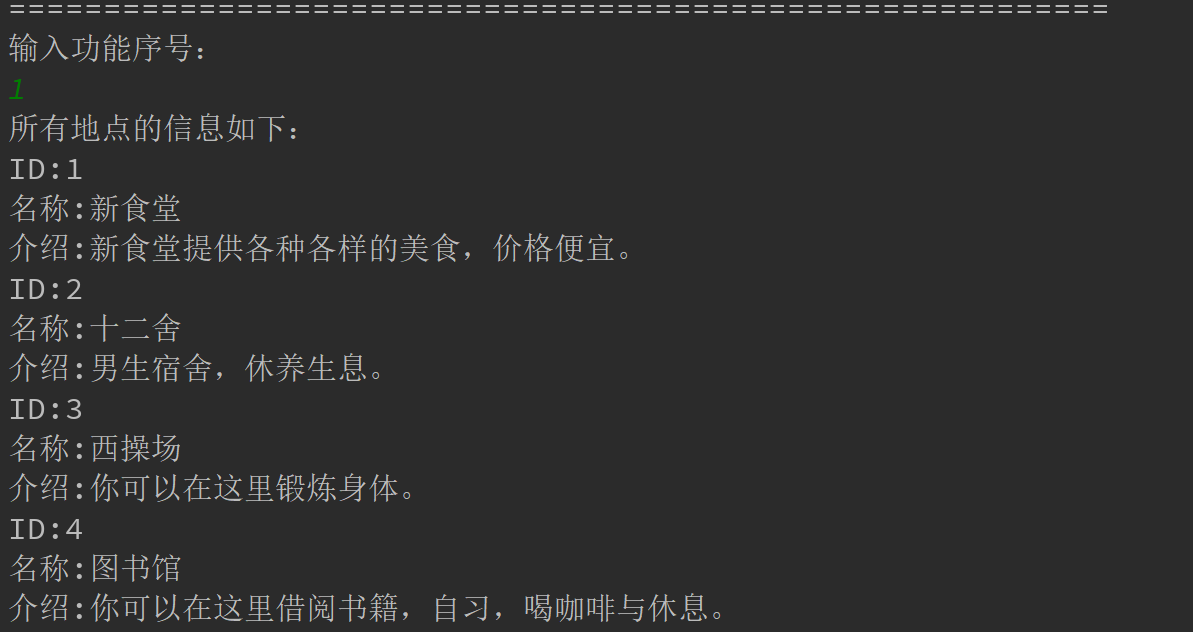
主菜单：



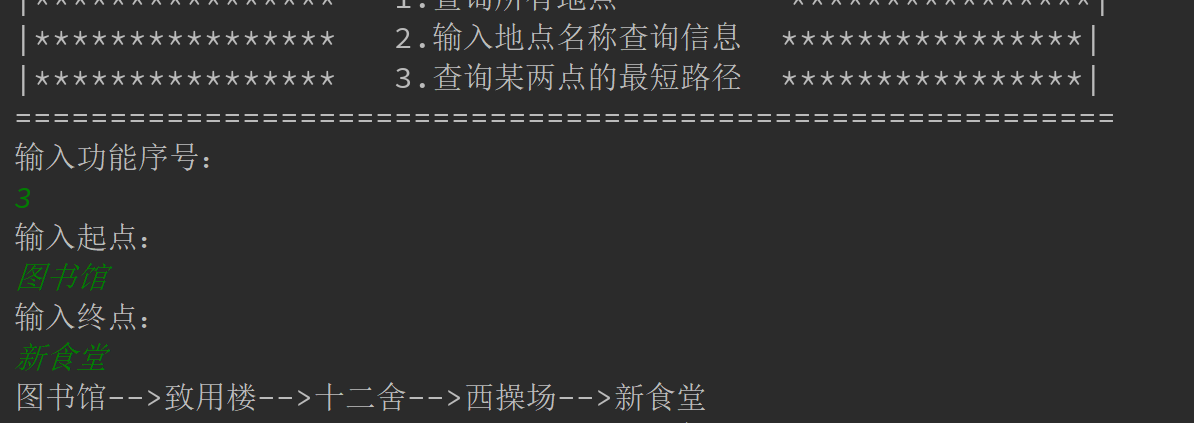
1. 查询地标/建筑物的基本信息



1. 列出全部地点



1. 计算某两点间的最短信息



Part2：问题与解决

1. 两点间的最短距离

单纯的遍历Graph是一个不明智的选择，所以我们采用了Dijkstra算法解决该问题。

其中，Java中没有结构体数组，所以例如Dist数组等无法正常表示，只能借用List类来实现这一功能。

1. 输入

我们不能要求用户知晓每一个地点的序号，我们设想用户只知道地点名称，例如用户输入地名，通过convert函数，获得该地名的对应编号，通过顶点编号调用Dijkstra方法是容易的。同时这样也是用户友好的。

1. 输出

结果的输出是一个Dist数组，但是用户很难读懂其中蕴含的信息，我们只能通过一种办法：在计算最短路径的过程中，将每个结点保存在List中，最后通过逆序输出每个结点的名称，并将这种输出规格化。

1. **课程设计总结**

**通过本次课设，我们熟练的掌握了Dijkstra算法，明白了整个算法的过程。同时在软件设计方面，采用了互联网传统的Web开发模式---MVC模式，我们利用各类Bean存储我们需要的数据类型，利用List将它们组织起来。利用Controller中的Utils的方式，准备我们需要解决问题的工具。利用控制台充当用户界面，负责显示用户交互界面。不足点在于我们没有使用图形化界面，这也是因为课程时间太过紧张。总的来说，我们每个人在负责项目的过程中，都掌握了自己所负责方面的一些技巧与知识，同时也对编程与程序开发的理解更进了一步。**

**附录：运行截图**

