文件结构：

Src 文件夹：源代码

路线图文件夹：地铁线路及时间

以及 Jar 文件和需要挪到 C 盘下运行的 map 文件夹（版本 1.8）

使用说明：

输入站点名，至少两个，用空格分割

将得到完整路线，用时和运行时间

如输入： 滴水湖 花桥

得到： 滴水湖 -> Line16 -> 罗山路 -> Line11 -> 花桥

用时共 156 分钟

系统运行时间为：4 毫秒

已解决问题：

十号线，十一号线和二号线换乘的部分

四号线为环线的部分

输出 324*324 种组合用时 37 秒，平均一个 0.37 毫秒

Classes：

Node // 每个站点

BinaryHeap // 用来快速寻找最小值的堆

Map // 构建的地铁图

Path // 使用 Dijkstra 算法求最短路径

Class Details:

Node:

```
Public int ID; // 站点序号独一无二
Public String Name; // 站点名称
public boolean Visited; // 是否访问过
public ArrayList<String> Next; // 相邻站台的名称
public ArrayList<Node> NextNode; // 相邻站台节点
public ArrayList<Integer> Distance; // 相邻站台到达所需要的时间
public ArrayList<Integer> Line; // 所属线路
public ArrayList<Node> Parent; // 记录路径
public int Time; // 起始点到目前为止的时间
```

BinaryHeap:

```
public ArrayList<Node> heap; //  储存站点的动态数组
public void buildBinaryHeap(); //  辅助构建堆
public void BinaryHeapify(int i); //  下沉排序
public void UpHeapify (int i); //  上浮排序
public Node extractMin(); //  取出最小值并返回
public int getIndex (Node node); //  输出元素位置
```

Map:

```
public ArrayList<Node> M; //  大地图
public HashMap<String, Node> Name2Node; //  名称对站台的映射
public void setNodeVisted (); //  初始化访问状态
public boolean isNewNode (Node temp); //  判断是否为新节点
public int getID (Node temp); //  寻找旧站点 ID
public void addNode (Node temp); //  添加一个新站点
public void constructNodeMap () throws FileNotFoundException; //  构造地图
public int getLine (String s1, String s2); //  寻找公共线路
public void constructRoadMap (); //  构造路线网络图
```

Path:

```
public Map map; //  地图
public HashMap<String, Node> name2node; //  名字对节点的映射
public BinaryHeap BH; //  堆
public String Route; //  路径
public int ShortestTime; //  用时
public ArrayList<Node> Known; //  已知节点数组
public void search (String from, String to); //  搜索路径
public int getLine (Node n1, Node n2); //  寻找公共路径
public void getRoute (String from, String to); //  得到公共路径
```