面向对象第十一次课程作业指导说明

一、程序功能

如指导书中所说，该程序可以实现对百辆出租车进行配单操作的功能，可实现道路的开闭功能，红绿灯的创建功能，和 VIP 小车的创建。

二、程序运行所需环境和使用规范

运行环境：与课程提供的 JDK 和 Eclipse 相同即可。

运行步骤：将全部.java 导入工程文件，从属 my_Taxi 包，该工程文件中需要有 JRE System Library，之后运行 run Begin 文件，即可在控制台输入 Request 或 Road 或 VIP 或 Test 信息。

三、正确输入格式和使用操作说明

1. 控制台输入：

①Request 语句的使用

要求需满足如下格式，”[CR,(x1,y2),(x2,y2)]”。

其中 CR 为标志符，无实际意义，(x1,y1)为请求发出时乘客所在位置，(x2,y2)为乘客想要去往的目的地，每个数字都是一到两位，即 0~99。

在上述格式中，需完全符合，不满足对空格或制表符的容错，如果不同会显示 Not Match 字样提示。

对于满足格式，但是超出地图大小的点坐标，会提示 INVALID。

对于同 100ms 内，输入的出发地、目的地均相同的请求，会提示 Same Request。

对于有效请求会给出 Matches 样式提示。

输入结束在最后会说明。

②Road 语句的使用

要求需满足如下格式，[RR,(x1,y1),(x2,y2),OPEN]或[RR,(x1,y1),(x2,y2),CLOSE]。

其中，RR 为标志符，无实际意义，(x1,y1)为道路的一段，(x2,y2)为道路的另一端，对于可能的错误输入，会有一定的判断。

③VIP 与双向迭代器

该程序中，双向迭代器的使用不需要修改代码，在控制台输入即可。双向迭代器只能访问 VIP 出租车接单的 Request，而不是所有抢单的 Request。

提供6种输入类型，其中number为VIP小车编号，范围从0-29：

[VIP,number,Next] 将下标往后移一个，如果成功，控制台提示“VIP-number go Next to Request-whichone”，如果不成功，控制台提示“VIP-number can’t go Next”；

[VIP,number,Previous] 将下标往前移一个，如果成功，控制台提示“VIP-number go Previous to Request-whichone”，如果不成功，控制台提示“VIP-number can’t go Previous”；

[VIP,number,hasNext] 查询下标是否可以往后移一个，如果可以，控制台提示“VIP %d has Next”，如果不可以，控制台提示“VIP %d doesn’t have Next”；

[VIP,number,hasPrevious] 查询下标是否可以往前移一个，如果可以，控制台提示“VIP %d has Previous”，如果不可以，控制台提示“VIP %d doesn’t have Previous”；

[VIP,number,PrintNow] 打印当前光标所在位置对应的下一个，如果可以，控制台提示“Print current request”，并输出到文件VIP-number中，如果不可以，控制台提示“Can't print current request”；

[VIP,number,PrintAll]将光标移回最开始的位置，然后将所有的订单输出到文件中。

关于光标位置的说明，A Request-1 B Request-2 C，如果一辆车接了两个单，则光标有ABC三个位置。

假如光标在A，hasPrevious为false，hasNext为true；假如光标在C，hasPrevious为true，hasNext为false。

Next和Previous指的是，将光标移动到下一个或上一个位置。

最开始光标在A，此时PrintNow，是指输出A后面的请求；假如光标在C，则输出失败。

无论光标在哪里，PrintAll都先将光标移动到A处，然后到最后一个请求之后，途中将所有请求输出。

④TestThread访问

这是之前说要实现的访问接口，我放在控制台输入了。格式为[Test,numberS,Status]或[Test,numberN,Number]，其中numberS为0-3的数字，numberN为0-99的数字。

[Test,numberS,Status]可以根据状态查询车辆编号，具体numberS代表什么下面会说。

[Test,numberN,Number]根据车辆编号，可以查询车辆位置和车辆信誉。

2.地图文件输入说明

本次地图输入与loadFile相统一，由loadFile完成地图的初始化。

A.地图采用文件读取的方式录入，运行时自动读入。

B.需要存放80*80的矩阵，具体要求与指导书相同。

C.需要测试者自己判断地图是否连通，如果遇到不连通的题图，可能导致的后果无法设想。

D.文件fileInfo.txt放置在与Project工程文件相同的目录下即可。

E.文件的格式如下，提交的代码中有样例fileInfo.txt供参考。格式需要完全相同，否则会报File IO Error；除了map和light之外，没有对空行和空格等的容错机制。

```
#map
必须是连通图
#end map
#light
红绿灯图
#end light
#flow
(x1,y1),(x2,y2),value
#end flow
#taxi
Number,Status,Credit,(X,Y)
#end taxi
#request
[CR,(X1, Y1),(X2, Y2)]
[CR,(X3, Y3),(X4, Y4)]
#end request
```

如上表所示，map和light为80*80矩阵；flow格式为(x1,y1),(x2,y2),value；taxi格式为Number,Status,Credit,(X,Y)；request格式与控制台相同，为[CR,(X1, Y1),(X2, Y2)]。其中value和credit不超过5位数。

对于light的额外说明，由于对指导书理解不同，本程序对所有红绿灯分别随机初始状态，即所有红绿灯不统一，但是同时变化。

对于红绿灯不合理的放置位置，会忽略并在控制台给出提示。

车流量的计算为新的计算方式。

对于 Status 的额外说明，采用 int 表示，对应数字如下：

```
1.      public final static int Serving = 1;
2.      public final static int Picking = 3;
3.      public final static int Waiting = 2;
4.      public final static int Stop = 0;
```

个人理解 Picking 和 Serving 在此种情况下相同，均为随机产生一个请求让某辆出租车跑，且不会输出结果进 result.txt。但是应要求，对于 VIP 汽车的 GhostRequest，可以通过双向迭代器访问并输出。

4.输出结果的说明

运行后会在程序所在的目录下生成一个文件，result.txt，存放着所有的请求以及相应的处理。

文件最开始记录该次模拟的红绿灯周期。

对于未被响应请求，输出结构为”Systime:%d Request:[CR,(%d,%d),(%d,%d)] Not Picked”，对于被响应请求，输出结果包含请求自身、请求时间、出发地、目的地、响应车辆编号、当时参与抢单的所有车辆信息、接人路径与送达路径。

四、特殊情况和边界可能

1.由于请求数量过多导致的处理时差

使用系统时间不可避免的就是运算时差，可能出现的时间差不对问题希望测试者能够冷静合理分析，同时理性判断，双方均容易接受。

2.关于程序结束

输入结束，需要输入 END，来停止所有线程的运动，待所有请求都执行完毕后会提示 Simulation Over，此时可见 result.txt。当观察到 Gui 的地图上的出租车不再运动时，即可点击 Terminal 终端程序。

3.某些要求的个人理解

个人对寻找流量最小的最短路径的理解是实时的，所以因为流量是时刻动态变化的，所有出租车每次运动都会重新计算，寻找当前流量最小的最短路径，因此对计算资源要求可能偏大。

对于接单位置和实际出发位置，由于不能要求出租车与请求输入完全同步，所以需要等

该车走完当前那一步,才能去响应接单,因此可能出现接单位置与出发位置相隔一格的情况。

对于每次运动完成时间，当数据量过大的时候，由于计算资源较大，不能保证所有出租车都能快速计算出最短路径，所以若出现累计误差导致的偏差，希望谅解 TAT。

由于本次指导书说明很多地方存在歧义,很多地方的理解与处理方式可能与测试者略有不同，如果出现疑似理解偏差的问题，希望到时候能够多多交流，谢谢。

4.关于 JSF

个人理解放在了方法名的前面；对于很多方法的 Requires，由于自带检查，所以多数情况下为 None；如果有不能通过 JSFTool 的地方，希望测试同学可以给予理解，并且受累人工阅读，辛苦了！

5.关于 repOK

个人理解是对该对象内的成员变量进行判断，如果有误返回 false，否则认为对象正确返回 true。

6. 关于 Overview

个人理解是描述该类的作用,以及重要方法和重要成员变量,如有疏漏,多多包涵 TAT。

7.关于 VIPTaxi 和 Taxi 的 LSP 关系论证

里氏替换原则有以下四点内含和要求：

A．子类必须完全实现父类的方法

在子类中，父类的方法均被重写或继承，该条实现。

B．子类可以有自己的个性

VIP 出租车有自己对路径判断是否可以通过的方法。

C．覆盖或实现父类的方法时输入参数可以被放大

因为 VIP 出租车和普通出租车成员变量基本一致，增加的成员变量只有两个 int 和一个 ArrayList,用于双向迭代器的实现,同时新增的不需要通过构造函数初始化。对于位置、声誉等数据的范围，子类与父类相同，不存在缩小的问题，Requires 范围一致。

D．覆盖或实现父类的方法时输出结果可以被缩小

成员变量只对 number 和 type 的判断出现了改变，普通出租车要求 type 为 0 或 1，number 在 0~99 中间；而 VIP 要求 type 为 1，number 在 0~29 之间，并不会改变父类 repOK 的正确性。

8.之前的 BUG

A.上次没有支持 fileInfo 内 Map 的空格和制表符容错；

B.上次忘记写随机运动时车辆找流量最小的路径了；

C.上次忘记在 readme 中列出上上次 Bug；

D.上次由于找最短距离算法有误，导致配单时不是配给声誉最高且最近的车辆