

ReadMe

一、 运行环境

Java SE 1.8, Eclipse neon 2

二、 输入规范

城市地图需要满足《00 第七次作业指导书 V3.3》2.1 节中关于城市地图的定义。如果不满足程序将会提示错误并结束。城市地图文件命名为 `map.txt`，位于 Eclipse 中相应的工程文件目录下，如：D:\Eclipse\00\map.txt，如果在该目录下未能找到地图文件，程序将会提示错误并结束。

红绿灯信息输入文件命名为 `light.txt`，位于 Eclipse 中响应的工程文件目录下，如：D:\Eclipse\00\light.txt，如果在该目录下未找到文件，程序将会提示错误并退出。文件中只能包含 0 和 1，其中 0 表示无红绿灯控制，1 表示有红绿灯控制，如果文件中某个为 1 的点不是十字路口或丁字路口，则程序会提示错误并将其改为 0 再读入。

用户乘车请求可以从控制台输入，每行输入一条指令，输入样例为：[CR,src,dest]，其中 CR 为标识符不能省略，src 与 dest 不可相同，src 和 dest 分别表示请求发出位置的坐标和目的地的坐标，如：(20,34)，如果以地图左上角为坐标原点，x 轴正方向向右，y 轴正方向向下建立坐标系，则纵坐标在前，横坐标在后，即按照矩阵的元素的标识方法标识位置信息。对坐标(x,y)，x,y 的范围均为[1,80]的闭区间，即从(1,1)开始标号。此处应当说明，原本 GUI 中位置是从(0,0)开始标号的，即对于点(x,y)，GUI 窗口中显示为(x-1,y-1)。由于不能修改 gui.java 的内容，所以不做调整，希望测试者能够注意。

道路修改请求从控制台输入，每行一条道路修改指令，输入样例为：[Close,src,dest]，其中 Close 为标识符不能省略，src 与 dest 为有道路直接相连的相邻两点，src 与 dest 定义同上。一行的任意位置可以添加任意数量的空格。

三、 输出规范

程序会为所有有效请求按时间顺序编号，同时发出的请求按输入先后顺序编号，编号从 0 开始。程序会将同一个请求的执行过程信息输出到一个单独的文本文件中，文件名为请求的编号，格式为 txt，位置在对应的 Eclipse 工程文件目录下的 LogInfo 文件夹中。每个请求的输出内容包括以下几点：

1. 具体的请求信息。发出时间(以 ms 为单位)、起始位置、目标位置。坐标标识方法

同输入中的规定。

2. 请求发生时，在请求范围内的出租车信息。如果没有，会输出“没有在范围内的出租车!”。
3. 抢单时间窗口关闭时，所有已抢单的出租车信息。如果没有，会输出“没有抢单的出租车!”。
4. 最终抢到请求的出租车。如果没有成功分配，会输出“没有可分配的出租车!”，并且，**程序会认为本请求已经处理完成**，不会再有之后的其他输出内容。
5. 前往请求发生位置的路径和行驶距离。**路径以从起始位置（含）到目标位置（含）的点的列表表示**，位置表示方法同输入中的规定。
6. 前往请求目标位置的路径和行驶距离。表示方法同上。

其中，**出租车的输出信息**包括：

1. 编号。从 0 开始的出租车编号。
2. 状态（status）。Stop 表示停止服务，Serving 表示正在服务，Waiting 表示等待服务，Picking 表示接单状态。
3. 信用度（credit）。
4. 当前位置（position）。标识方法同上。

四、 错误提示

在遇到错误时，程序会**将错误信息输出到控制台**中。具体错误及处理方法如下：

1. 地图文件不存在。此时程序会输出"**Map file does not exist!**"并结束。
2. 地图文件不合规范。此时程序会输出"**Invalid map file!**"并结束。
3. 红绿灯控制文件不存在或不合规范（不包含非交叉路口有红绿灯控制的情况）。此时程序会输出"**Wrong light file!**"并结束。
4. 红绿灯控制文件中非交叉路口的值为 1。此时程序会输出"**Wrong light control in (i,j)!**"并将 1 改为 0 后继续执行,其中**(i,j)**表示控制出错的点。
5. 请求不合规范（包含请求发出地与目的地相同）。此时程序会输出"**Wrong Request!**"并忽略此条错误请求。
6. 相同请求（同时同地同目的地）。此时程序会输出"**Same Request!**"并忽略此条错误请求
7. 程序运行错误。程序会输出相关 Exception 信息并结束。

五、 测试说明

编写测试程序时请使用 Test.java 中提供的方法。具体说明如下：

1. public static void sendRequest(Vector<String> strings)

此方法可以从测试程序中读取用户请求。**Strings** 为输入的字符串队列，每个请求单独存放在一个字符串中。请求规范与输入规范中的要求相同本函数对不合法输入的处理同上。

2. public static void traceTaxi(String pathname, Taxi taxi, long lasttime, long stoptime)

此方法跟踪某一辆出租车的运行状态并输出到文件中。**pathname** 为输出文件的路径，**taxi** 为要跟踪的出租车，**lasttime** 为跟踪时间，超出这一时间将不再跟踪，**stoptime** 为两次输出之间的间隔。每次输出的出租车状态信息与输出规范中的说明相同。

3. public static Vector<Taxi> findTaxi(int sta)

此方法会返回所有状态为 **sta** 的出租车队列，**sta** 为要找的出租车现在的状态，0 表示停止服务，1 表示正在服务，2 表示等待服务，3 表示接单状态。

测试前请保证 eclipse 的**编码方式为 UTF-8 格式**。以上所有输出方法，所有方法均为**静态方法**，并且如果输出目标文件已经存在，均**不会覆盖原有内容**。如果需要保证覆盖，请在程序运行前删除原有内容。