放在前面：

readme 中没有提到的一些细节，请以指导书和 issue 为准。

测试环境：

请使在搭载 windows10 系统的非 MAC 系列电脑上 JDK 版本 jdk-8u171-windows-x64 环境 中，使用 eclipse-java-oxygen-2-win32-x86_64 运行该程序。如果以上要求不满足，运行时 有可能出现未知错误。注：仅支持控制台输入，不支持使用文件输入（load 属于文件调用）和命令行输入。

输入：

1. The system initing! 之后可以在 sonsole 输入 Load 指令。
2. 系统加载地图等需要一定时间，请在"The system ready!"字样出现后，在输入请求。
3. Load 指令在程序最开始进行初始化的时候在 eclipse 控制台输入，**必须使用且仅能使用一次**。如果其中的 map 部分出错会直接结束整个程序，其他的出错会进忽略。如果 load 文件名出错或不存在，会进行反馈，并进行等待，直到正确的 load 指令输入，再进行初始化。
4. Load 的 map 和 light 必须是文件。
5. 本次指导书没有明确规定等的输入形式，也没有规定必须有默认灯图，所以没有准备默认灯图。请务必在 Load 的 file 中指定 load 的灯图文件 。
6. 再 load 之后仍然可以通过输入请求来增加订单。
7. 再 load 之后仍然可以通过输入指令来查看车的状态。
8. CheckTaxi 指令以$开头，后面直接跟数字。不$开头的进入后面的判断。
9. CkeckTaxi 指令输出的是出租车上一次移动后的状态，第一个时间是现实时间，第二个时间是上一个运动周期的出租车时间。
10. Load 指令请务必按照正确的格式输入，如果里面的文件输入格式错误，后面发生的 bug 可能无法预料，但不会 crash，也有可能直接触发系统 shut down。
11. Flow 的输入请严格遵守(x1,y1) (x2,y2) value 的格式即 value 和第二个点之前都有且仅有空格，括号内无空格。具体可以参见文末的样例。
12. 请求的输入支持前导 0，不支持前导+和负数，但是请不要输入>=80 或<0 的位置坐标。
13. 出发点目的地一致的请求算作无效请求。触发 INVALID。
14. 除非您计算机性能特别强大，否则请不要尝试一次运行太多指令，毕竟理论上有可能计算不完。
15. 如果想要关闭程序只需要将 GUI 的窗口点击关闭即可，程序会直接自动关闭。
16. 由于本次测试不测格式，所以不是所有的报错都会有输出，有可能直接忽略而不显示，也有一定可能直接触发系统结束。
17. 程序的所有输入数值请不要大于 100000，不要有前导+。
18. 总请求数大于 300 时，输入线程结束
19. 所有的文件名及路径中，请不要出现空格。
20. **默认地图的文件名为 map_ori.txt，如果##之间没有指定文件，则调用 map_ori.txt**

请保证此文件存在。否则当没有指定文件时系统结束。

输出：

1. 输出时，为每个有效的（包括有无被派单的请求）建立一个日志（也就是一个文件）里面记录与该请求有关的信息。具体的路径为，与 src 文件夹和 map.txt 同级。
2. 输出格式请同学测试一组查看，格式很直观，恕时间紧张不再多做解释。大概顺序依次为：请求本身，曾经抢过该单的所有出租车，派单的情况，前往乘客的路程，搭上乘客前往目的地的过程，最终到达的时间
3. 输出的参与抢单的车的状态会被认为是指最后派单时候的状态。
4. 出租车移动过程在节点上中输出的状态，除了接到乘客后休息一秒结束的节点以外。均值的是在该节点进行完状态转换之后的状态
5. 出租车移动过程在节点上中输出的时间是绝对时间，所以都没有划归到 100 的倍数，其主要价值在于差值。
6. 乘客上車后（1s 之后），才输出接到乘客的日志。到达乘客地点的时候不会产生日志。
7. 同质请求（100ms 内的起点终点相同者）会算作一个请求计入输入的请求数，分派其一个 ID，但是不会算数 console 中会输出同质提示，也不会为其建立日志。
8. 如果 load 的地方特殊，可能 sonsole 有一些输出优先于 The system ready!。这从事情请不要计较，毕竟 console 本来也不在检查范围。
9. 幽灵 GhostRequest 所有车统一输出到 request0.txt。但是由于整个过程不完整里面会缺失一些东西，请同学不要在意。
10. 为了防止信息输出不全，在 request 输出中，边界时间节点（状态进行变化）的输出会稍有一些累赘冗余。边界状态也不好把握，请同学见谅。

light 行为解释：

1. Light 必须在 load 中输入，**请保证此文件存在**，如果没有，系统自动结束。
2. 允许空格和 tab
3. 所有灯的颜色一致。这种行为允许且合理，请勿扣分！
4. 等必须建在度大于 2 的节点上，即丁字路口和十字路口。否则输出错误，且不建立此灯。

出租车行为解释：

1. 休息时间不会被打断。
2. 出租车的 20s 一睡，由于等红灯大概率会延长，但是除去等灯，保证总时间是 20s，即与指导书一致。
3. 判断抢单范围时是如果在半路上，视为在原路口才进行抢单。
4. 当出租车仍在某条路上（非路口）的时候，如果被派单，将继续完成此路段（即不会突然掉头）再进入服务状态，也因此派单时间（窗口关闭时间）和最终到达的时间的差很可能不是 500 的倍数。
5. 由于等灯，输出的两次节点信息之间可能不是 500。
6. 如果出现幽灵乘客的现象，根据状态执行[CR,(20,0)(10,0)]请求，此条请求的编号为 request0，在 request0.txt 中输出所有车辆的情况。（由于中途派单，故输出内容信息不完整。）同时，此条请求不存在分派等过程，永远不会消失。

7. 运行过程从道路塌陷，出租车将继续行驶至写一个路口。
8. 需要注意的是，由于这次的规定断路不会增加流量，所以当可追踪车经过断路的时候是很有可能回头的。

SetRoadStatus 行为解释：

1. SetRoadStatues 时，请在 console 输入命令，格式请严格遵守#(x1,y1) (x2,y2) status 的格式，即 status 和第二个点之前都有且仅有空，括号内无空格，具体可以参见文末的样例。
2. 只有第一个字符为'#'时才会进入 SetRoadStatues 命令的判定，否则系统按照请求进行识别。输入不合法的时候会报错，同时请务必保证开闭之后整张图仍是连通的，否则会出现奇怪的错误，比如直接程序结束。（显然这已经超出题目的要求）
3. 新开通的路径不会影响已经进入服务状态的出租车的路线，其最短路径不会更新。
4. 车在接上乘客和接到的单发生效果的时候，以及碰到现有最短路径上出现断路的时候，才会重新计算最短路径。
5. 对于已经进入服务状态的出租车，如果其最短路径上某条路断了，只有当出租车到达该断掉的路径时，才会重新计算新的最短路。

迭代器行为解释：

1. 本代码提供了迭代器及其相关函数比如 next, hasNext, hasPrevious, previous。同时在 console 添加了调用这些函数的指令。根据 issue 学姐的要求，实现此功能可是不再实现测试线程。
2. Console 中输入时，先通过“%n”获取车辆的迭代器，然后通过“%hasPrevious”或“hasNext”获取有无情况。再通过使用“%next”或“%previous”获取信息集合。最后使用“%print”打印获取的信息集合。每个 print 打印一单的所有输出信息。输出均输出到 iterator.txt 文件。
3. 文末给出了一个使用自功能的命令集合实例。
4. 此迭代器在队首时 hasNext 为真，hasPrevious 为假。在队末，hasNext 为假，hasPrevious 为真。在队首使用一个 next 即可使用一个 previous 回到队首。
5. 切换查看车辆后，如果不 next 就直接 print，会打印之前的内容。

派单行为解释：

1. 7.5s 窗口结束后，派单时，如果一辆车处于等待状态之外的状态，即使他是最合适的或者唯一抢单的，也不会被派单。
2. 当信用和距离一致时随机选择队伍中的第一个（约等于随机）

GUI 解释：

1. GUI 不能对 Load 产生的初始流量进行显示（其固有缺陷），但是出租车会对初始化的流量进行响应，即第一个 500s 内车能够观察到初始化的流量。
2. GUI 中不写 JSF

LSP 规则论述：

1．本代码使用的是抽象类，将所有与原来类不同的部分均用抽象方法表示，然后在两个子类分别重写。避免了对于父类的行为的改变。
2．虽然父类无法直接实例化，但是所有子类和父类的方法之间不存在冲突，无疑实现了“子类可以扩展父类的功能，但不能改变父类原有的功能”。
3．子类对于父类进行功能的扩大，但是没有扩大参数需求和减小结果限制。
4．同时显然实现了所谓继承的要求。

写在最后：

人生艰难，OO 不易，求同学手下留情 OTZ。文明六系，和谐 OO。

Load map.txt 样例：

#map
map_new.txt
#end_map

#light
light1.txt
#end_light

#flow
(11,2) (12,2) 5
(3,23) (2,23) 10
(25,10) (25,11) 15
(45,46) (45,47) 20
#end_flow

#taxi
No.1 0 2 (1,1)
No.2 0 2 (2,0)
No.3 0 2 (3,3)
No.4 0 2 (4,4)
No.5 0 2 (5,5)

No.6 0 2 (6,6)
#end_taxi

Console 输入样例：
请求：
[CR,(48,8)(6,53)]
[CR,(16,56)(59,1)]
[CR,(27,77)(26,57)]

checkTaxi 查询样例：
$5
$6
$7

道路通断样例：(这两条本来不一定是通的，取决于地图)
#(7,10) (8,10) 0
#(7,20) (8,20) 0

一组 Console 使用迭代器实例：
%14
%hasNext
%next
%print
%hasNext
%next
%print
%hasPrevious
%previous
%print
%hasPrevious
%previous
%print