OO\_10说明文档

1. 程序功能功能说明：

程序模拟出租车的乘客呼叫与应答系统，采用多线程的设计，并结合了相应的GUI即时显示结果。

1. 程序运行环境与运行指令规范

程序运行环境为Java(TM) SE Runtime Environment (build 1.8.0\_73-b02)，运行IDE为Intellij Idea 2017.1.2。如果用命令行编译并运行，编译指令为javac -encoding utf-8 \*.java，运行指令为java TaxiSys。最后输出结果保存在Log.txt文件中。

1. 程序注意事项

由于官方GUI的性能问题，在同时输入过多请求的时候可能产生程序运行的迟缓现象，又因为本次作业不强制要求使用GUI，故程序取消了部分GUI效果，包括红绿灯变化以及用户请求范围的显示。

程序提供了一些Taxi类中的接口函数来查询单个具体的出租车的当前状态，它们是：getId():返回出租车id，getX():返回出租车当前所在行，getY()返回出租车当前所在列（注意这里的行列都是以1开始，以80结束的数字），getState()获取出租车当前运行状态，getCredit()获取出租车当前信用值。程序还在TaxiManager中提供接口查询多个出租车状态，queryTaxi(int \_state):查询状态为\_state的所有出租车，返回一个ArrayList，而其中的getState(int i)返回编号为i的出租车的状态。至于出租车当前时刻，则调用getCurTime()即可返回当前毫秒数。

对于红绿灯状态的查询，在TrafficLight类中提供了getEastWest(int index)与getSouthNorth(int index)两个方法来查询点index对应的位置的红绿灯状况，如果输入的index非法则只能返回2，如果输入的index合法则返回0、1或者-1，分别表示该点没有红绿灯、此时该方向上红灯亮、此时该方向上绿灯亮。

程序规定所有的红绿灯变化的周期统一为TrafficLight类初始化时随机产生的一个介于[50,100]的值。

1. 程序输入说明

程序可以接受 [CR,(srcX,srcY),(dstX,dstY)]形式的输入，输入之间的空格都会被替换，因此接受含有空格的输入。其中的坐标值(srcX,srcY,dstX与dstY)都是以1开始，以80结束的形式表示，这与以后的路径显示和车辆位置记录中所采用的是同一种表示形式。此外，程序还接受一行不超过10个请求输入，例如[CR,(srcX,srcY),(dstX,dstY)]; [CR,(srcX,srcY),(dstX,dstY)]...其中要求请求之间用分号隔开，而且注意程序采取的分隔方式是严格按照分号分隔，两个分号之间、第一个分号与输入行首位之间、最后一个分号与输入行末尾之间都被认为是一个请求。此外，程序接受输入”end”作为整个程序结束的标志。

对于开闭路请求，程序支持用[CP, (srcX,srcY),(dstX,dstY)]表示关闭点(srcX,srcY)到(dstX,dstY)的路径，而[OP, (srcX,srcY),(dstX,dstY)]表示打开点(srcX,srcY)到(dstX,dstY)的路径。其中关闭路径不能是关闭本来就不存在的路径，起止点也不能相同。同理打开路径也不能打开本来就存在的路径或者根本不相邻的两点之间的路径。在关闭路径的时候，程序是等路径上的车都通过了之后才会真正切断路径，而程序是在200ms末将车瞬移到目的地的，因此可能存在在GUI上看似切断了路径但是车还是移动了过去的情况，因为切断路径的时刻处于200ms的中间，而此时从理论上讲车在路上，所以依然会移动过去。此外，程序对于增删路改变了点的路口数的情况，不再对红绿灯进行调整， 比如说，原先一个丁字路口被删掉一条边变成了只有两条边，那么此时该点的红绿灯还在，这可以解释为，我们的开闭路请求只是临时开闭路，因此无需改变最开始设置的红绿灯。

所有关于坐标的输入（形如”(x,y)”）均支持前导零和正号。

输入的地图文件中允许有空行，地图的连通性需要由测试者保证。开闭路之后地图的连通性也需要由测试者保证。输入的交通信号灯文件的正确性也应该由测试者保证。如果地图文件或者信号灯文件不正确，那么程序会在报错之后自动退出。

1. 程序输出说明

程序对于格式错误的输入会报错，并且对于和以前的请求重复或者起始点重复的输入也会报错。程序会调用本次官方提供的GUI显示车辆与乘客状态。程序还会产生一个总的记录文件，记录文件中记录了每个用户刚刚发出请求时，周围车辆的状态，信用(Customer#i First Logging: )。以及抢单时间窗口内所有抢过单的车辆(Taxis tried to grap Customer#i's order:)，最后被选中的车辆(Taxi chosen by Customer#i : Taxi#j)以及车辆将乘客从起点送到终点的路径(Path found for Customer#i)。不同的乘客之间用一串短横线隔开，由于出租车完成每个乘客的请求的顺序不一定和乘客请求产生顺序一致，因此输出的乘客编号不一定是从小到大排列的。注意乘客假如乘不到车那么这些记录就是不完全的。建议用Sublime Text等编辑器打开输出的Log.txt文件，那样显示效果会更好。

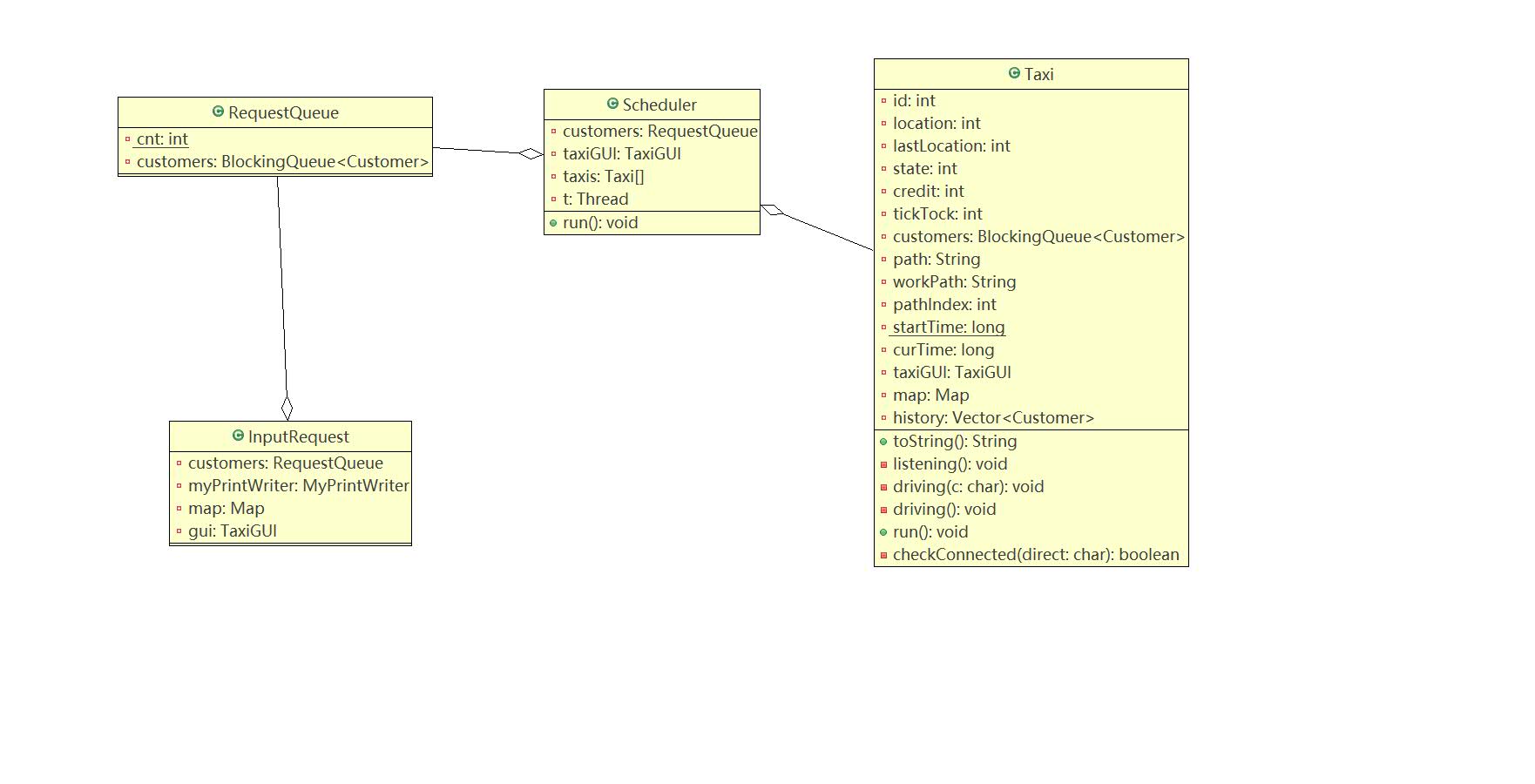
1. 需求分析

本次作业要求设计的是一个模拟出租车与乘客交互的系统，根据实际，我们可以自然地抽象出这样几个类：Taxi,Customer,Map，分别表示出租车，乘客，地图，红绿灯类。此外还需要一些辅助的类来调度整个系统，如请求队列类，输入类，调度器类

1. 地图能处理给定的地图文件，得到地图文件的邻接表表示形式。便于后面处理最短路。地图还提供对边和流量操作的封装，出租车只需发出请求增加流量，而流量的更新与边的开闭以及线程的同步均由地图完成，因此地图也是一个线程(为了定时更新流量信息)。
2. 出租车作为一个可以run的对象，能在没有任务时根据相应规则到处游走，在有任务时及时响应并完成任务。出租车的状态转换规则指导书已经说明，此处不再赘述。
3. 乘客可以抽象为一个请求，包含起始位置与终止位置以及产生时间。
4. 红绿灯作为一个独立的对象，需要以初始化时确定的随机值作为周期变换灯光，而且要做到全地图的灯光同时改变。此外其灯光需要有控制出租车等待的能力。
5. 请求队列负责装载用户的请求。
6. 输入类提供将控制台输入转换为用户请求的能力，并与请求队列协同。此外本次输入类增加了处理开闭路请求的能力，可以根据需要改变地图中路径状态。
7. 调度器类负责处理请求队列中的用户请求。

为此，程序采用SOLID原则中的单一职责原则，每个类都只管理自己的事，例如出租车只负责导航与运行，乘客只负责上车等。

根据生活经验与本次指导书要求，我们可以设计出如下应答方式：乘客在发出请求的瞬间产生一个呼叫区域，这个区域持续3s，3s内任何到达该4x4区域(包括边界)的车辆都会给该乘客发请求(投简历)，乘客在3s后作出决定，从简历箱中选出此刻能响应的最优出租车，然后呼叫，出租车便立即去往乘客所在地，将其接走，送到目的地后重新回到等待服务状态。具体的关系如下：



总而言之，是调度器类从用户请求队列中不断取出到时间的请求，将其放入合适的出租车，出租车线程会根据输入的请求自动运行，而输入线程在另一边可以将输入的请求放入该请求队列，实现了测试的需求。