设计文档

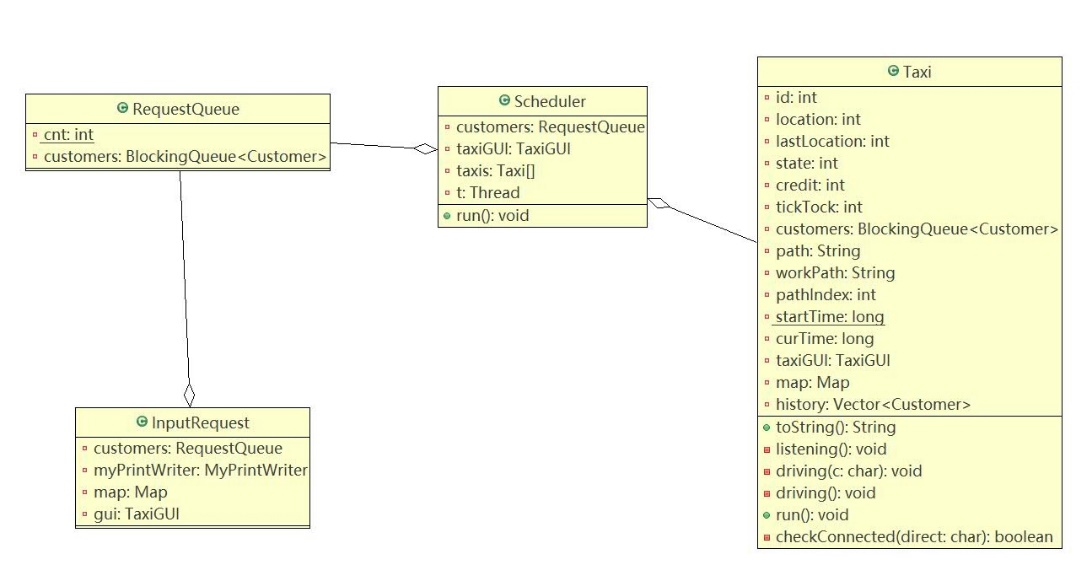
1. 需求分析

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. 可追踪出租车继承自出租车，能够在每次完成服务之后记录服务，并且提供返回迭代器以供迭代的接口。可追踪出租车还能行走在程序运行过程中被关闭的道路。
4. 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。
5. 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。
6. 请求队列负责装载用户的请求。
7. 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。
8. 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。

1. 验证LSP

在本次作业中，新增的TrackableTaxi相比于父类Taxi多了属性：

private ArrayList<Record> records;

该属性不会影响父类的行为。

此外，新增的TrackableTaxi相比于父类，主要是重写了void finishedTask(Customer c)方法，该方法的作用是在完成一个请求的时候记录整个服务过程。因此该方法不会影响在调度类Scheduler以及出租车管理类TaxiManager中的行为(因为该方法是protected类型)，所以外部的类根本无法访问这个方法，故不会影响子类在调度以及管理过程中的行为。而另一个重写的repOK()方法则更不会影响调度了。

综上，任何父类出现的地方都可以由子类替代，满足LSP原则。