需求分析

系统简述

系统功能描述

类与其功能描述：

- `Main`：主类，程序的进入口，负责输入线程和出租车线程的初始化以及调度，以及请求队列的初始化和地图的加载。
- `Request`：请求线程类，负责记录一个请求的出发点与目的地以及请求的时间，并具有计算某个点到其他点最短距离的方法，方便决定接单的出租车。run方法中先会沉睡三秒，在这三秒内如果有能够响应的出租车则会从所有能够响应的出租车里面找到当前满足条件的出租车来进行接单操作。无论有没有出租车能够接单，三秒后都会结束。
- `Taxi`：出租车线程类，具有模拟出租车行为的各种方法，同样具备计算某个点到其他点最短距离的方法。按照指导书上的内容来模拟行为，包括处于等待服务状态时随机选路，随机选择最短路径等。
- `InputHandler`：输入线程类，负责进行输入并对输入的请求进行合法性判断，还会将合法的请求加入到请求队列里面进行处理，并同时开启当前的请求线程。
- `RandomNumGenerator`：随机数生成器类，用来生成初始化Taxi类地址的随机数。
- `RequestQueue`: 请求队列，出租车从里面选择还生效的请求来进行处理。
- `gui`：已经提供好的类。
- `mapInfo`:提供好的类进行改写后的产物，负责进行地图的读入与处理。

系统性能要求

约束条件

详见`readme.pdf`

交互分析

`mapInfo`类负责打开文件读入地图，并将其做好处理之后存储下来；`InputHandler`负责读入指令，将指令加入请求队列并开启指令线程。`Taxi`类从请求队列中读取尚未失效的指令，并根据自身的坐标判断自己能否响应该指令。若能则将自己加入到该指令的预备抢单队列之中。`Request`请求线程建立三秒钟之后，若自身的预备抢单队列不为空则说明有出租车能够响应这条指令，再进一步判断将请求派发给哪辆出租车之后完成使命，结束线程。

以上为大致的数据传输路线与轮廓。

为了保证线程的安全性，在所有用到请求队列的代码块处以及对请求队列的所有方法都加上了`synchronized`，对`Taxi`类的各种方法也都加上了`synchronized`。

设计思路

本次作业的设计思路较为简单。生产者`InputHandler`负责生产出请求，将请求放入请求队列中供消费者`Taxi`来处理。整体来说是一个相互交互与综合的过程。

`Taxi`类从请求队列中读取尚未失效的指令，并根据自身的坐标判断自己能否响应该指令。若能则将自己加入到该指令的预备抢单队列之中。若自身的属性出发地与目的地存在，则进入接单状态，直到到达出发地接到顾客后才会变为服务状态。然后到达目的地后就会变回停止状态，$1s$后回到等待服务状态。

每个类的作用以及分工都很明确，因此采用高内聚低耦合的设计理念进行设计。