需求分析

（一）本次作业主要包括三部分对象，分别为地图、乘客和出租车。下面就三个对象简单分析一下它们的职能。

1. 地图：

大小为 80x80 的网格。

地图为连通图，且只有相邻的两点才有可能有边相连，边为双向边。

出租车、顾客分布在点上。其中出租车的起始坐标通过随机产生，其后按照一定的运行规则运动。

2. 顾客

每个请求包含起始点坐标、目的地坐标，每次有 3 秒的时间留给出租车们来进行抢单。在这 3 秒钟，符合条件的出租车会像该顾客发送抢单请求，3 秒后选择一个最优的顾客来进行行驶到目的地。如果不存在一辆可以搭载顾客的出租车，则该请求者将不能坐车。

3. 出租车（Taxi）：

出租车有 100 辆，最初随机出现在地图上。

出租车具有信用，初始值为 0，每抢单一次成功信用值加一，每次将客服送到目的地信用值加 3。

出租车具有不同的状态，对应不同的行为。状态分为停止运行、等待服务、接单状态、服务状态

停止服务 ：在这 1 秒内停止；其后变为服务状态或者等待状态。

等待状态 ：在地图上随机行驶，只要在用户的请求窗口内便可以抢单。如果被顾客选中，那么装填变为接单状态。每等待状态形成 20 秒变进入一下停止服务状态。

接单状态 ：从出租车的位置以最短路径开到顾客的起始位置。随后进入运行状态。

运行状态 ：以最短路径从顾客的起始位置行驶到终止位置。随后进入等待状态。

（二）线程设计

从上面的类的职能分析我们可以看到，出租车和乘客是我们这次工程的主要核心，而它们可以经由调度器连接。调度器能分析出每个顾客的请求，并对所有出租车进行筛选，最后将请求分配给最合适的出租车执行。

根据 SOLID 原则的单一责任原则，我们应该将每一个类的责任划分到尽可能小的单元，因此我决定开这么几个类：Car 类进行模拟出租车的运动、Request 类分析每个请求的合法

性并提取其中的信息、Requestadd 类将合法的请求加入队列、RequestQue 类储存合法的请求、Dispatch 类对所有已经接受到的合法请求会搜索距离乘客 2*2 范围内的出租车并进行分配。每个类的功能都没有重叠。

程序采用多线程设计，每个 Car 为一个线程，Requestadd 为一个线程，Dispatch 为一个线程。为了保证线程的安全，在 RequestQue 和 Car 的部分方法加上了 synchronized。

最终的类图如下所示：