1、 系统简述

1) 系统功能描述

该系统用来模拟一个 80*80 的城市的出租车分配，城市的地图被简化水平方向或垂直方向的网格。该系统可以响应来自乘客的需求，并将需求合理分配至各个出租车。

2) 系统性能要求

测试机器性能要求，1GB 内存支持，CPU 有 2.5GHz 以上基准速度，推荐使用四核八线程的 CPU 最好是 I7 6700HQ 及以上。
需要在 200ms 内计算所有车辆下一次的方向，并完成所有车辆的移动。

3) 约束条件

约束条件请查看 readme.pdf 在此不再赘述。

2、 交互分析

1)

与系统有交互关系的对象为地图上一个能发出需求的人群或者说乘客，其中的属性有该请求的发出时间，发出请求的地址即出发点，所需要到达的地点即目的地。以及窗口期的抢单的出租车队列。

2)

识别待开发系统与上述第 1 步所识别对象之间的交互，给出交互动作，并从数据和时间两个方面分析交互特征。

交互动作： 出租车向乘客发出抢单需求
在窗口期末尾将乘客分派给合适的出租车。

数据特征：发出点和目的地坐标。

时间特征：只要输入，便会被响应。交互持续时间直到指令执行完毕。交互频率不定，有输入便会存在交互。存在多用户并行。

3) 对象识别与构造

本程序的主要数据集合有：
100 个出租车的出租车集合 taxiList
80*80=6400 个 point 的点图 pointMap
80*80=6400 每个位置上所停留的车辆集合 taxiHere[]
6400*6400 的邻接矩阵
6400*6400 的最短路径矩阵
动态存在的请求队列 requestQueue

本程序利用在 TaxiMap 类中提前声明所有的 point，然后从中获取现成的 point 的方法，避免了每次 new point 的内存损耗和时间流逝。
出租车同理在 Scheduler 类中一次性声明，保证了不会多 new。
利用以上方法，可以极大的降低内存占用和运行缓慢的可能。

类图见本文结尾图片。

4) 并发分析

本程序主要是关于出租车的调度和输入内容的处理的并发。本次作业所应该该出现的是多辆出租车运行抢单和输入请求的多者调度。

由于出租车数过多，基于线程安全和运行效率的考虑，本程序直接将 100 辆出租车放入了一个线程。这与多个出租车线程，每次操作都 synchronized 的方式并无本质区别。但是舍弃 synchronized 的过程可以提高效率，节省时间并且不会出现线程不安全。