出租车调度程序需求分析文档

一、需求分析

我们这次作业的用户需求就是满足指导书里一切莫名其妙的要求，所以经分析可能需求如下，通过对文件的处理实现 Map 的初始化，在地图上创建 100 辆出租车供调遣，通过控制台输入来控制目标运动。

1.交互关系分析：

通过控制台输入，可实现同时大量请求涌入，所以请求数据就需要存储，而不能等待执行。这就意味着程序需要自己对大量数据进行自动处理，同时可以接收新的请求，所以需要输入和处理两个线程。而出租车自由运动，在没有请求的时候自行运动，且互不干扰，所以每个出租车都是一个线程。

2.数据管理

为了满足对数据的统一存储和管理，建立三个数据相关类，Request、Snapshot、和 RequestList，其中 Request 和 RequestList 为了满足输入需求而建立的，用于存放请求和其相应衍生数据；Snapshot 基本就是为了结果输出而建立，因为在运行中不需要记录某一时刻的特定信息，单纯为了作业输出要求而设。

3.设计思路：

此次作业思路很简单，主线程负责初始化现场和输入，每当有输入的时候，建立相应的Request和RequestList进行存储。对于新建立的输入，需要一个类进行处理，就是Scheduler，每来一个请求，他就会进行相应的操作，可能的话将其分配给某辆出租车。出租车平日里闲来无事瞎逛，当有配单的时候进入工作状态。

输入与执行为流水线式，清晰而系统，实现了良好的整体性。

二、程序实现的原则

1.SRP：single Responsibility Principle

每个类或方法都只有一个明确的职责，类职责：使用多个方法，从多个方面来综合维护对象所管理的数据，方法职责：从某个特定方面来维护对象的状态（更新、查询）。

2.OCP：open close principle

无需修改已有实现（close），而是通过扩展来增加新功能（open）

比如 Map 和 Gui，Gui 中有对地图的初始化和邻接表的建立，但是我需要更多的信息比

如路与路连通等，就在其基础之上建立自己的 Map 类。

3.LSP：liskov substitution principle

任何父类出现的地方都可以用子类来代替，并不会导致使用相应类的程序出现错误。因为本次作业思路较为清晰，所有类与类之间的职责都不尽相同，而且较扁平化，所以本次作业中未使用继承方法，所以不违背 LSP 原则。

4.ISP：interface segregation principle

同理，本次作业中类与类之间的相似性不是很大，接口定义显得没有必要，所以本次作业中未使用接口，避免了一个类中实现过多接口，所以不违背 ISP 原则。

5.DIP：Dependency Inversion Principle

依赖倒置原则，表示高层模块不应该依赖底层模块，二者都应该依赖其抽象。用 JAVA 中的表现描述就是，模块间的依赖应该通过抽象类和接口发生，实现类之间的依赖关系也是通过抽象类或接口产生，实现类之间不应发生直接的依赖关系。

本次作业中，实现了此原则，比如调度器和出租车之间进行交互，都是通过对 Request 数据类的访存实现的，高层类之间不直接互相操作。同时由于没有使用抽象类和接口，所以修改需求只需要改单链即可，比如调度器—请求，出租车—请求等。

6.责任分配原则

本次作业一共使用重要类 3 个，Begin 负责读写和总控，Scheduler 负责调度和分配，Taxi 负责执行和运动；辅助类 3 个，Element 负责宏定义，gui 负责

图形化输出卖萌，Output 负责输出写文件；数据存储类 3 个，Request 类负责存储请求和与该请求相关的衍生信息，RequestList 负责存储和管理 Request，Snapshot 用于记录某一时刻的状态。数据间相互调用，没有全知全能或者毫无用处者。

7.完整性原则

本次作业对于所有的需要输出的数据都有相应的处理。

8.信任原则

所有方法输出结果的确定是程序正确执行的根本，所以不存在问题。

9.懂我原则

命名清晰易懂。

10.层次化抽象原则

按照问题域逻辑关系来识别类，每个类的功能都与其名称有很强的对应关系，说明功能明确清晰抽象。

11.重用原则

本次作业中不同信息存放到不同的类中，避免了冗余。

12.局部化原则

本次作业中不同信息存放到不同的类中，避免了某一处数据改变而其他地方未出现联动的情况，不会出现控制耦合。

13.显示表达原则

使用 Element 类对常量进行定义，代码中几乎没有模糊的数字等令人难以理解的参数。