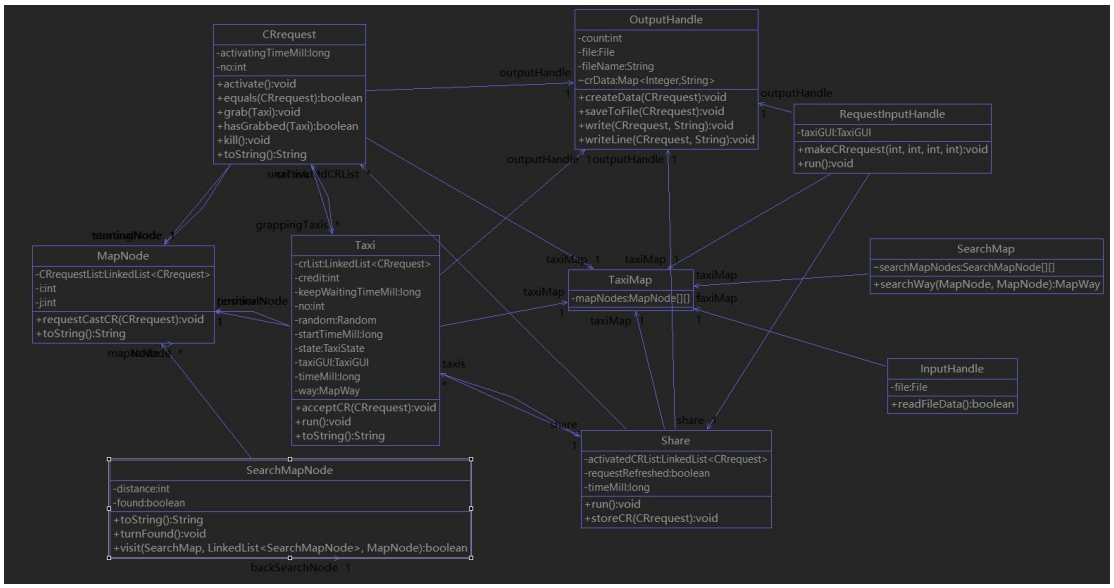
需求分析文档

一、需求分析

1. 地图读取的实现
2. 出租车移动的实现
3. 请求发出、接收与执行的实现

二、设计思路

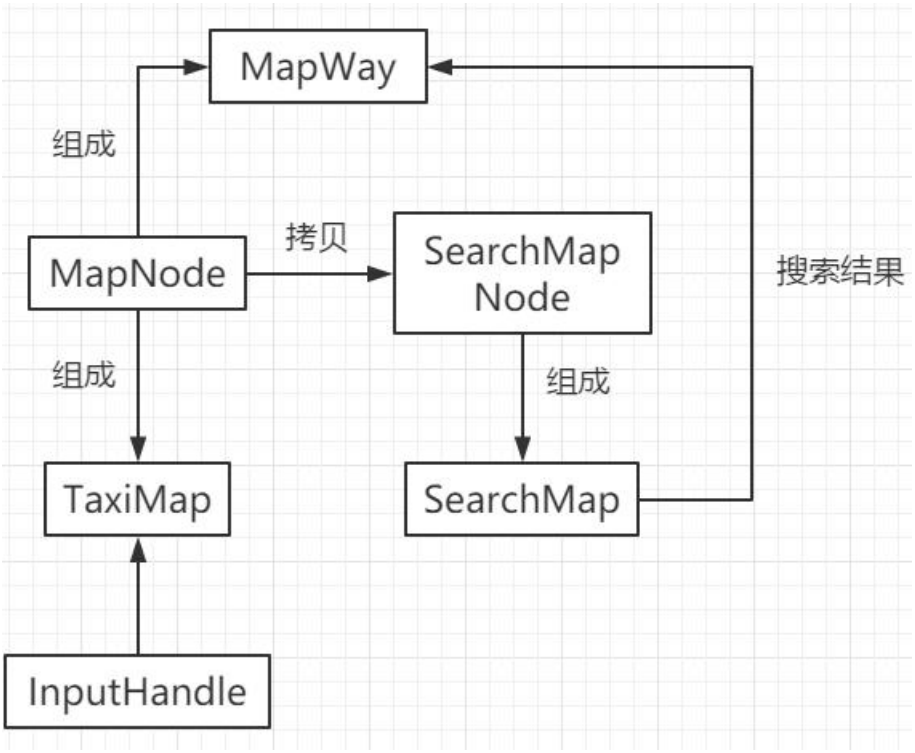
这次作业的思路呢其实十分的简单，我们用一张类图就可以将其完全展现出来：



（好吧连我自己都不信……）

其实上面这幅图之所以看起来复杂，是因为它把所有类之间的关系都浓缩在了一张图里。但实际上，一个庞大的类关系是由无数组小的类关系组成的，因此我们只要提取其中的某些类，进行分组，就可以将他们的关系简化。那么下面，我将对具体实现的思路进行整理和总结。

1. 地图相关



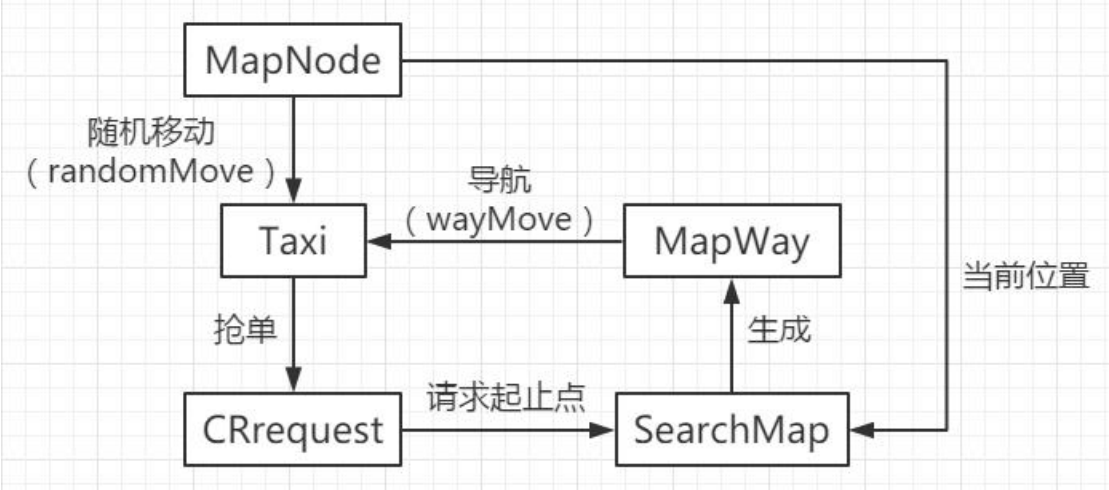
这幅图简要地概括了程序中和地图相关的五种类：TaxiMap（地图类）、MapNode（地图节点类）、SearchMap（搜索图类）、SearchMapNode（搜索图节点类）、MapWay（地图路径类）。这五个类各自拥有特定的职责，搞定了有关地图的一切，能够引导出租车移动、寻找最短路径和计算距离。可以说，它们是出租车的“眼睛”。

而在这五个类中，又可以根据它们的使用方法或特性分为两种：一种是长期使用的 TaxiMap 和 MapNode，而另一种是类似一次性用品的 SearchMap、SearchMapNode 和 MapNode。因此在整个程序中，前者既不需要也不允许存在多个本质相同的实例，而后者则可以存在多个复制品，且被利用完毕后应当及时销毁以节约空间资源。

首先，我们来分析前者。所谓图和节点之间的关系，可概括称为宏观和微观的关系。TaxiMap 可以看做为一个管理 MapNode 的类，它以上帝视角给这些 node 搭桥牵线，还可以根据坐标定位到具体 node。而 MapNode 只记录了邻接点，能够告知在这个点上的出租车可以往哪里走（出租车的移动其实就是在微观世界中穿梭的过程）。

而余下的部分都是和寻路相关的功能。当某个方法需要查找路径的时候，就把起点和终点传递给 SearchMap 的实例，接下来 SearchMap 广度搜索访问（visit）SearchMapNode 们，访问到的节点会记录之前访问的结点，直到访问到终点（terminalNode）。SearchMap 会根据记录，将从起点到终点的这些节点们封装成路径 MapWay，返回给原方法。

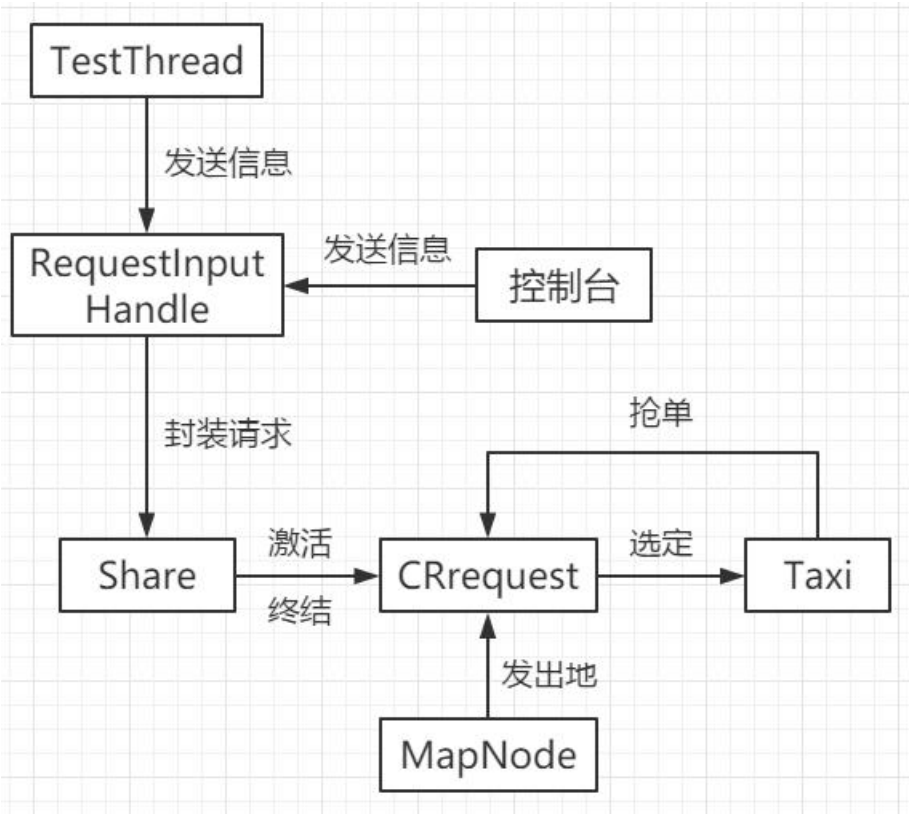
2. 出租车相关



出租车主要需要实现两种行为：移动和抢单。虽然在 Taxi 类中存在着 searchWay 方法，但这本质上是将信息传递给 SearchMap 处理。Taxi 具有随机移动和路径移动两种移动方式，前者只需根据当前节点位置进行随机（MapNode 可以访问到邻接点），后者则按照获取到的 MapWay 进行移动，而 MapWay 的可靠性则由 SearchMay 保证。

Taxi 每次移动前后或者是停止运动后都会对 Taxi 周围的 4*4 区域进行遍历，以接收请求广播，如果收到了某个还未被抢单请求的广播，则抢单。如果 Taxi 成功抢到单，CRrequest 便会加入到 Taxi 的待执行请求队列中，此时 Taxi 可以获取到 CRrequest 的起始地点（startingNode）和终止地点（terminalNode），再结合 Taxi 自身的位置（positionNode），调用 SearchMap 中的方法得到 MapWay，使其引导自己寻找乘客和服务乘客。

3. 请求相关



请求的处理是这次作业的重点。输入请求的方式有两种，一种是通过控制台，另一种是

编写测试线程，调用 putCRrequest 方法实现请求的输入。无论哪种，最终请求的封装都是在 RequestInputHandle 中进行的，它再将封装后的请求（CRrequest）传递给 Share。

Share 相当于 CRrequest 的管理者，它储存有当前所有的请求，并将其分为两种：未激活请求与激活请求。Share 是个线程，它每过 100ms 便会刷新请求队列，将所有到达 3s 的激活请求终结，并将所有未激活请求激活，将其全部放入激活请求队列，再清空未激活请求队列。可以看出，激活和终结是两个相对的概念，象征着一条请求的生命周期。

那么什么是激活呢？所谓激活，就是请求开始放送请求广播。在哪里放送请求广播呢？毫无疑问，应该在 MapNode 中，因为它是“地点”的象征。当 Share 激活某一条请求的时候，请求会把自己放入自己所在 MapNode 的请求队列中，以便 Taxi 的搜索。而当 3s 过后，请求就会被终结，这时候请求会选定一个符合要求的 Taxi 作为被执行的对象，然后将自己从 MapNode 中移除。

4. LSP 原则的满足（新增）

本次新出租车重写了父类的一些方法，我将针对这些方法进行论述。

searchWay：搜索路径只考虑存在的边 -> 搜索路径可以考虑被关闭过的边了！属于 EFFECTS 的扩展。

randomMove：随机移动只考虑存在的边 -> 随机移动可以移动到被关闭过的边了！属于 EFFECTS 的扩展。

saveToTaxi：压根不能保存 -> 可以保存了！属于 EFFECTS 的扩展。

LSP 原则是指“任何父类出现的地方都可以使用子类来代替，并不会导致使用相应类的程序出现错误”。对以上三个方法的重写，都属于功能性的扩展，故满足 LSP 原则。