一. 需求分析

1. 系统简述

1) 系统功能描述

本系统实现了一个模拟的出租车抢单送单系统，能接收指定的地图信息和出租车信息，同时实时运行接收请求，在GUI中表现出100辆出租车及请求的执行过程。还支持随时的开/关道路指令，以及提供测试接口获得出租车状态位置等信息。

2) 系统性能要求

出租车送单过程中，实时更新基于路径长度和流量和的最短路径，如果指令数目过多，首先会导致GUI的卡顿，其次程序中出租车的500ms运行时间可能不够计算。因此需要较大的内存空间，并控制一次输入的请求数目。

3) 约束条件

a. 输入

运行以后，程序首先输出：“请输入配置文件路径：”，这时请输入有效的路径，如test.txt并按回车即可。关于配置文件格式，参考指导书，下面是一个正确的例子：

```
#test file#
#map
xxxxxxxx
#end_map
#flow
(0,0),(1,0),2
(0,0),(0,1),3
(0,1),(0,2),4
#end_flow
#taxi
0, 3, 1, (0,0)
1, 2, 5, (50,50)
2,2,3,(79,2)
4,2,6,(70,70)
#end_taxi
#request
[CR,(0,0),(20,11)]
[CR,(50,50),(33,44)]
[CR,(79,2),(77,46)]
[CR,(70,70),(55,58)]
[CR,(31,43),(50,50)]
[CR,(31,47),(50,50)]
[CR,(31,51),(50,50)]
[CR,(31,55),(11,50)]
```

```
[CR,(31,59),(50,50)]
[CR,(31,63),(0,0)]
[CR,(31,67),(50,50)]
[CR,(31,71),(50,50)]
#end_request
```

预设文件的格式正确性**完全由测试者保证**，格式出错导致的程序错误无效。提交的文件中有样例的预设配置文件test.txt，可供参考。
每一行的空格将在解析时去除，#map和#end_map中为80行的地图内容，每行有效字符数为80，在0,1,2,3中选择，具体请按照指导书的格式填写，测试者自己保证格式正确性和地图连通性，尤其是**最右侧不能向右连接，最下方不能向下连接，否则程序将出错**。
然后，#flow和#end_flow中间为预设的流量信息，格式为两个坐标以及流量值，以逗号分隔，注意该设定的流量值仅在前500ms有效，因为地图每500ms更新一次流量值。
之后，#taxi和#end_taxi中间为预设的出租车信息，分别是编号，状态，信用，位置，编号的涵义与指导书一致，0服务1接单2等待3停止。
最后，#request和#end_request中间为初始指令，将直接**不重复地**加入请求队列，格式为[CR,START,END]，其中START和END分别是起始位置和目标位置。
文件输入结束后，程序开始接受实时的控制台输入，在控制台可以进行地图道路的临时开启和关闭，以及发出新的乘客请求。具体规定为以下的请求格式：

```
[CR,(X1,Y1),(X2,Y2)]
[OP,(X1,Y1),(X2,Y2)]
[CL,(X1,Y1),(X2,Y2)]
```

分别代表乘客请求，开边请求和关边请求，乘客请求格式和文件输入的格式相同，开关边请求的后面的两个坐标代表边邻接的两个结点，须保证两个结点确实是相邻的才行。
若要结束程序的运行，在控制台输入END，此时程序将会等待最后一个出租车送完最后一单后结束。注意这是正常结束程序的**唯一方式**，否则你将无法得到程序运行的文件记录。

b. 输出

首先是控制台的输出：

1. 程序在收到无效请求，无效地图，无效文件时，输出提示。
2. 出租车成功被派单时，输出被派单的出租车和请求的编号。
3. 当某请求在达到抢单窗口后，没有找到合适的出租车时，也会输出其编号。
4. 当出租车到达请求发出地点接乘客上车时，输出两者编号。
5. 当出租车成功将乘客送到目的地时，也会输出两者编号。

更详细的信息在上文提到的文件输出中，如果程序正常结束（输入END后等待进程自动结束），就会在程序源目录下生成TaxiLog.txt。按照指导书要求，记录如下信息：

1. 文件记录每个请求的输入时刻，编号以及开始和结束位置。只有**有效且非重复的**请求会被记录并赋予唯一的编号。
2. 文件记录每个请求在抢单时间窗口内参与抢单的出租车信息，包括其编号，抢单时的位置，信用以及状态（只可能是WFS）。
3. 文件记录每个请求最终派给哪辆出租车，记录派单时间，出租车编号和位置。
4. 文件记录每个请求的载客出租车到达乘客位置的时刻和位置。也记录到达最后目的地的时刻和位置。
5. 文件记录每个请求被处理过程中，出租车路过的每个结点及路过时间。

6. 更多细节在最后的README中说明。

c. 测试接口

TestTool类提供指导书要求的两种测试接口，包括通过ID获知出租车状态和通过状态取得该状态的出租车集合。测试者可以用出租车数组创建新的实例调用之。两个方法如下：

```
public long[] queryTaxiById(int id);
public ArrayList<Integer> queryTaxiByState(int s);
```

第一个方法参数为出租车ID，输出的数组长度为4，分别代表查询时间，x坐标，y坐标，状态。状态的值定义如下：

```
class State
{
    public static int STOP = 0;
    public static int SERVE = 1;
    public static int WFS = 2;
    public static int PICK = 3;
}
```

第二个方法参数为出租车状态，同样必须是合法的状态值，返回所有满足该状态的出租车的编号序列。

2. 交互分析

1) 与系统有交互的关系的对象识别

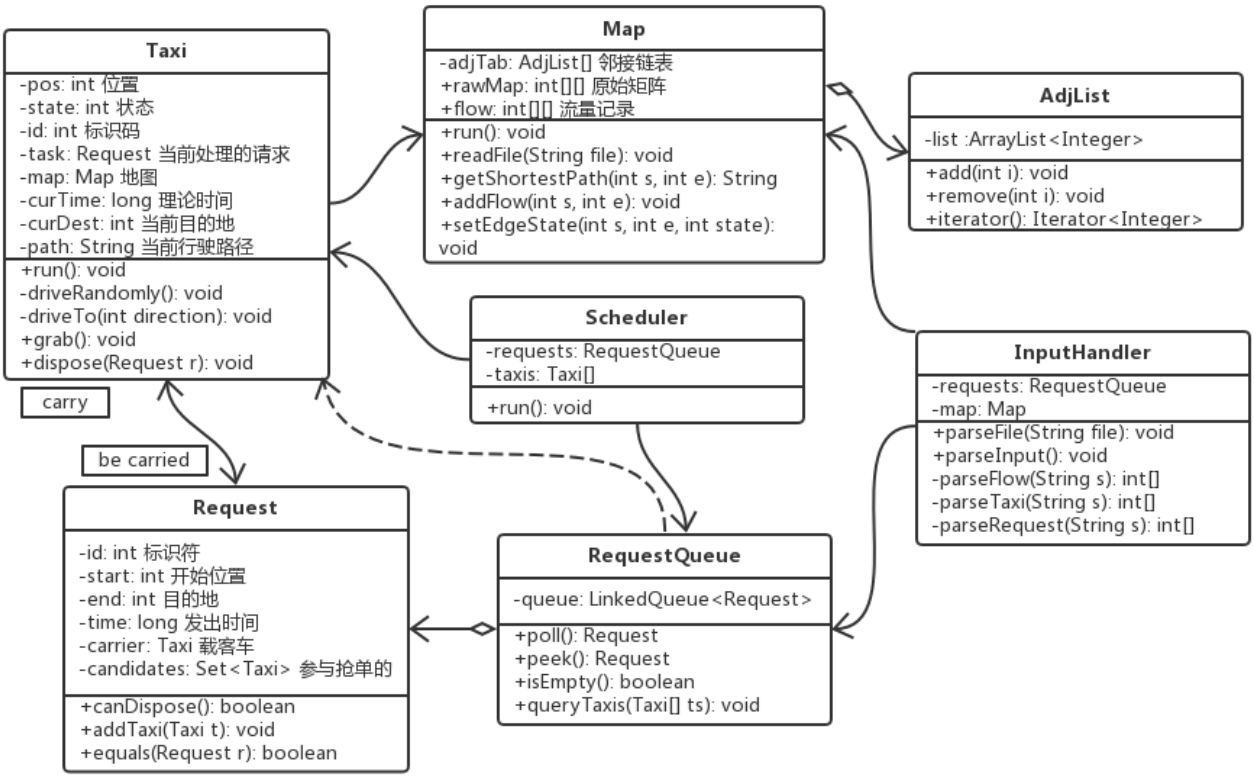
该出租车系统运行在一个80*80的地图上，并按照流量和与最短距离进行路线选择，因此有地图对象，具有连通图的边信息，各边的流量信息。
运行100辆出租车，并且系统根据其位置，信用来进行派单，同时出租车按照四个状态进行变换，因此有出租车对象，有位置坐标信息，信用，状态。
乘客可在某一位置发出请求，系统给予一定时间窗的抢单和派单处理，因此其维护有发出时间，开始位置，目的位置，以及参与抢单的出租车记录candidates和最终被派单的出租车记录carrier。
系统应当通过请求队列来获取和存储请求，因此有请求队列对象。

2) 识别待开发系统与上述对象的交互

出租车需要给出位置信息，状态信息，信用信息供系统进行抢单，派单的判定，因此需要对应的GET方法，同时系统需要对出租车进行派单，需要对应的dispose方法传递请求内容。
地图类对象需要给出两个位置之间的最短路径，优先级为路径长度，其次为路径流量和，因此需要getShortestPath方法，同时其需要提供进行开关边的接口setEdgeState用于实时的测试。
请求类对象除了常用的GET方法，还需要对抢单的出租车进行选择，根据其信用和状态，距离来选择最终的载者，如果没有合适的出租车，则该请求被丢弃，因此有canDispose方法。
请求队列对象提供对未处理请求的存储和获取，因此实现相应的offer，poll，peek方法即可，还需有判断是否为空，实现isEmpty方法。它还需要对请求进行轮询，判断抢单的动作是否执行，因此提供questTaxi供系统调用。

3) 对象识别与构造

由于系统一开始从文件中读取预设信息来初始化，而后从控制台不断读取输入，接收实时的请求进行处理。因此需要再设计一个InputHandler类进行这些功能的整合。
其次，本身所有的对于请求的处理虽然分散给了请求队列类和请求类，但是整体由调度器进行调用和轮询，需要一个Scheduler类进行统筹，但其负担已经很小，不会是GOD类。
同时指导书要求对程序的所有运行信息进行详细的记录，包括请求内容，抢单信息，送单路径，时间等，通过一个OutFuncs类提供一系列文件写操作进行统筹。
因此最终可以得到如下的一个UML类图：



需要说明的是，由于OutFuncs多提供静态方法，与其他类交互几乎没有，因此没有画上，AdjList为邻接链表，在程序中名字为EdgeSet，一些名字的差别是因为手误和思想的变化，比如questTaxi在上图中写成了queryTaxi等。

4) 并发分析

程序需要运行100辆出租车，还需要做到地图实时的对流量更新（每500ms），同时每个请求维护7500ms的抢单时间窗，不可避免地需要通过多线程程序实现。
每辆出租车按照自己的逻辑进行状态间的转换，因此开辟100个线程来模拟。地图需要对流量进行刷新，同样以一个线程模拟。调度器需要不断地对请求队列中的请求进行轮询，因此也需要开一个线程。
在上面这些类中，一个请求在未分配给出租车时，只会被调度器线程访问进行出租车的抢单和派单，在被分配给某出租车后，只会被该出租车线程访问，添加对应的记录，因此对于请求类方法不需要加同步锁。
对于Map对象，会同时被100个出租车线程访问，同时也会在控制台进行边的开关操作，因此也会被InputHandler访问，因此Map对象的方法需要进行同步控制。

二. ReadMe

一些细节

1. 使用JSFTool检查会出现一些莫名的Warning。在输入类的规格中采用了较概括的写法，因为其逻辑难以表述。
2. 程序输出的时间是以**100ms为单位**的。
3. 如果你在控制台输入的请求格式错误，导致了程序出现异常，请重新运行程序。**请一定保证请求的格式有效**，当然常见的错误是能够反映的。
4. 程序不支持控制台一行输入多个请求，请**一条一条输入**，或是复制粘贴多行输入以模拟一次输入多条请求的效果。
5. 重复的请求不输出任何提示，直接被忽略。格式无效的请求，包括开始位置和结束位置超范围等，会在控制台输出Invalid，但不会在文件记录中输出。文件TaxiLog.txt中**只记录有效的，加入到请求队列中**的请求。
6. 输出的记录中，乘客被接到的时间不等于到达该点的时间，实际上相差10，这是因为输出的时间加上了程序模拟的上车时间1000ms（相当于车暂停了1000ms），这点在下面的例子中说明。
7. 出租车在连续WFS状态20s后停止一次，因此接过单的出租车和未接过单的出租车停止时间是错开的。

记录文件例

```
Get new valid request #0: (0,0)---->(20,11) at 15297141774.7
--------------------------------------------------------------------------------------------------------
Request #0 is grabbed by following taxis:
Taxi #0: (1,1), WFS, credit is 2
--------------------------------------------------------------------------------------------------------
Taxi #0 carries Request #0 at 15297141851.2 in (1,1)
---->(1,1) at 15297141857.6---->(1,0) at 15297141863.5---->(0,0) at 15297141868.5
It arrives at (0,0) to pick at 15297141868.5
---->(1,0) at 15297141883.5---->(1,1) at 15297141888.5---->(1,2) at 15297141893.5---->.....
It finishes task at (20,11) at 15297142084.4
--------------------------------------------------------------------------------------------------------
```

若系统得到新的请求，则如第一部分输出单条请求的内容，包括编号，位置和时间。
达到7500ms的抢单窗口后（可能会有几十毫秒的计算误差），如第二部分一样，输出参与抢单的所有出租车信息，包括编号，位置，状态，信用，本例中只有0号车抢了单。注意这里输出的信用是包括了抢此单获得的1点信用的。
如果没有出租车抢单，则会输出：No taxi grabbed it!
成功派单后，请求将会输出出租车送单过程中的所有路线结点及到达时间，还有什么时候派单，接客，到达目的地。如本例第三部分，省略了一部分的路线（太长了懒得贴），路线包括两部分，第一部分是出租车从被派单的位置去往请求起始位置接单，第二部分是接到客人后，送往目的地址。
第三部分中有两个看似不符合常理的地方：

- 出租车被派单的时候是xxxx851.2，地点就在(1,1)，但是输出到达(1,1)的时间是后延了将近600ms的（xxxx857.6），这是因为被派单的时候，出租车虽然地点在(1,1)，但实际上已经在去往某相邻点的路上（WFS状态的动作），当他到达相邻点后，才进入PICK状态，进行路径的计算，计算出的最短路径正好需要通过(1,1)，因此回头重走一遍，所以多花了时间。
- 出租车到达请求发起地点的时间是xxxx868.5，到达下一个点的时间是xxxx883.5，相差15而不是5，因为接到乘客后需要暂停1s模拟上车的动作。