1. 输入

（1） 地图：地图的输入文件是 map.txt,路径就是程序执行时的默认路径，测试者也可以在 Map 类中自行更改。地图文件有 80*80 个点,在这里规定有效的地图输入 txt 文件只能有 80 行，如果存在第 81 行且改行都是空白符也会被判错。每一行可以包含任意个数的空格和退格符。地图的左上角坐标定义为（0,0）而不是（1,1）

若输入不符合要求则输出提示：`Map initialization failed`

若输入的不是连通图则输出提示:`Map is not a connected graph`

出现以上两种错误，程序都会终止。

（2） 乘客请求格式：[CR,(srcx,srcy),(dstx,dsty)]

这里的标识符我设置为 CR，一个乘客请求一行，且中间不允许出现空格或退格符之类的空白符。其中每个点的坐标范围是[0,79]

有两种错误提示信息：INVALID: [request] 和 SAME: [request]

对于地图和乘客请求中出现的数字通通不支持前导 0 和前导符号。

2. 输出：

（1）当一个请求完成时，程序会同时将相应的信息输出在控制台和 Result.txt 文件中，改文件的路径也为默认路径。一个输出样例如下：

```
[VIP,(1,1),(6,6)]
T :149244129(100ms)  srcPoint: (1,1)  dstPoint: (6,6)
请求发出时，4x4区域的出租车信息：
Taxi:  ID:83 status: wait credit: 0 point: (1,2)
Taxi:  ID:67 status: wait credit: 0 point: (3,1)
Taxi:  ID:71 status: wait credit: 0 point: (3,1)
抢单窗口内所有出租车
83 67 71 69 94 30 80 59 53 54 18 36 45 82 78 46 24 41 86
响应请求的出租车：67
响应请求过程中的行驶路径：
(3,1) (2,1) (1,1) (1,2) (2,2) (3,2) (3,3) (3,4) (3,5) (3,6) (3,7) (3,8) (4,8) (4,7) (4,6) (4,5)
```

其中第一行为请求的发出时间和起始地点，T 的单位是 100ms.

最后一行的行驶路径包括出租车从开始接单到乘客所在地的路径，和从乘客

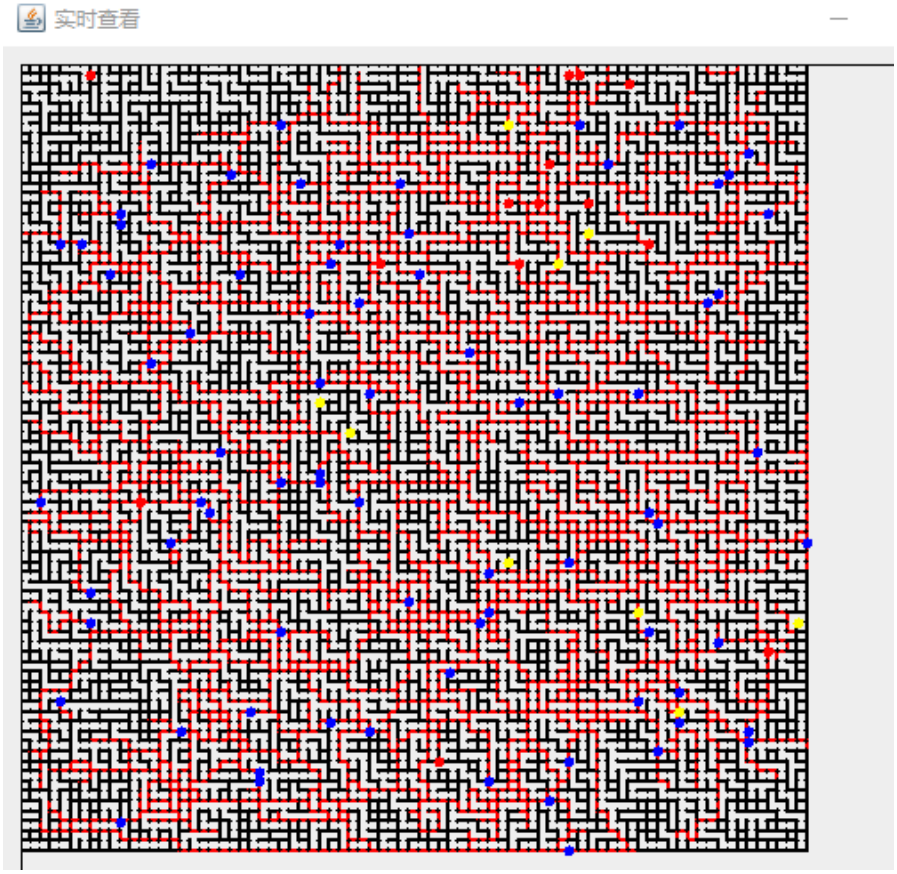
所在地到目标地址的路径。

（2）当一个请求抢单失败时，程序会同时在控制台和 Result.txt 文件中输出相应的提示信息。

（3）控制台界面的输出样例

```
[CR,(19,42),(15,14)]
T :149250475(100ms)  srcPoint: (19,42)  dstPoint: (15,14)
请求发出时，4x4区域的出租车信息：
抢单窗口内所有出租车
90
响应请求的出租车： 90
响应请求过程中的行驶路径：
(18,43) (18,42) (18,41) (19,41) (19,42) (19,41) (19,40) (19,39) (19,38) (18,38) (18,37
```

（4）GUI 界面的输出样例：



上图是我的程序在运行过程中的一张截图，如果 GUI 有错，和最终的程序 bug 判定无关。

3. 程序编码格式： utf8

4. 测试接口：

测试接口都实现在 ControlCenter 类里，并且指导书中只说明了要写两种测试接口，测试接口如下：

（1） `public CarInfo getCarInfo(int ID)`

该方法通过输入一个出租车的编号（编号为 0-99）来获取其相关信息，函数返回的是一个 CarInfo 对象。如果出出租车的编号不合法，则返回一个空指针。

CarInfo 记录了出租车的如下信息：

```
int ID;
int status;
int credit;
int x;
int y;
long t;
```

其中 status 是出租车的状态，其中 0 表示停止运行，1 表示服务，2 表示等待服务，3 表示接单状态。T 直接获取系统时间，单位是 ms.

使用方法：我已经对该方法进行了包装，使用时直接在控制台输入命令：car:ID ID 是要访问的车的编号，范围是 0 到 99。输入的命令不允许有多余的空白符。一个输入输出样例如下：

```
car:1
T: 1492525794623   Taxi:  ID: 1   status: wait   credit: 0   point: (9,15)
car:0
T: 1492525796998   Taxi:  ID: 0   status: wait   credit: 0   point: (31,66)
```

（2）

```
public ArrayList<Integer> getSpecifiedCar(int status)
```

该方法通过输入一个出租车的状态返回具有改状态的出租车的编号的 ArrayList.如果输入的状态不合法，则会返回一个空指针。

使用方法：我已经对该方法进行了包装，使用时直接在控制台输入命令：status:d d 可以为 0,1,2,3.其中 0 表示停止，1 表示服务，2 表示等待服务，3 表示接单。输入指令时不允许有多余空白符，否则会被判为无效输入。

```
status:2
wait: 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
status:1
serve: not found
```

（3）

```
public synchronized void add(RequestFile request,String str)
```

改方法是向请求队列加入一个请求，RequestFile 是一个记录请求信息的类，str 是从控制台输入的请求本身。

5. 测试线程

测试线程我已经实现了向请求队列发送请求这一部分，它是 Customer 类的一个实例化。可以从控制台读入请求，请求的输入格式见第一大条输入说明。测试者可以在这个类中再完成其他测试功能。

6. 执行过程中的限制与说明：

（1） 成功服务顾客则信用加 3，这里对成功服务顾客的时间节点定义为到达目的地并且停止运行 1s.

（2） 若同时有多个请求选定 1 辆车作为接单车，但是程序在执行的时候肯定有先来后到，只选择其中一个请求有效，其他请求记为无车相应并输出。

（3） 关于 gui 包 :本人在测试的时候使用了 void RequestTaxi(Point src, Point dst)方法，但是如果一次输入很多条请求，在调用该方法的时候程序会 crash,所以在最终版本中并没有使用该方法。个人认为这是 gui 包函数的问题，并不是本人程序的问题。

(4) 关于轰炸测试：我尝试过一次同时输入 300 条请求，程序能够正常相应，如果测试者在测试过程中由于 JVM 内存泄漏而 crash，建议自行修改内存上限，这并不能算程序的问题。

(5) 一辆出租车被选择作为某个请求的接单车的瞬间，它很可能还处于运动过程中，必须等其移动完一格再开始执行请求。

(6) 关于 3s 窗口的边缘时间问题 :个人认为程序无法及其精准的卡主边缘时间 0s 和 3s,比如说在 3s 整拍一张出租车信用的快照的过程中，或者在计算出租车距离的过程中，出租车的状态也是在不断改变的。而且强制把请求的发出时间转化为以 100ms 为单位也是不精确的。所以对于测试者来说请不要在 0s 或者 3s 节点附近卡 bug。

7.readme 里可能有些地方说的并不全，还希望测试者多多体谅，相互理解。

‘’