OO-第九次作业-Readme

2017/5/2

# 程序功能说明

模拟出租车的乘客呼叫与应答系统，在上次作业的基础上增加了流量要求以及JSF规格的编写。

# 程序运行所需环境

java环境：

java version "1.8.0\_73"

Java(TM) SE Runtime Environment (build 1.8.0\_73-b02)

采用的IDE是Eclipse Java Neon.

# 具体说明

## 3.1 程序的输入说明

大部分主要参考作业要求，**加黄色的内容为我自己强调说明的重要内容。**

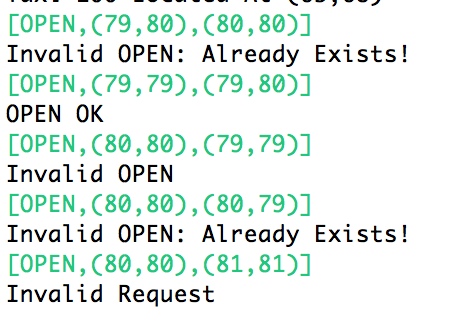
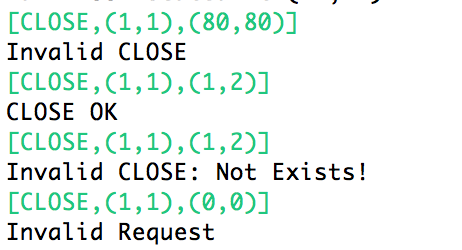
读入的地图信息请严格按照指导书要求，即一共有80行，每行在去除了空格回车和制表符(‘\t’)后为80个0到3之间的数字字符。当出现格式非法的地图信息时，会输出"Map Invalid!!!"并推出程序，否则输出"Map Is Okay"代表读入的地图是合法的。

按照指导书要求，请测试者保持地图的连通性。包括关闭道路后的情况。

读入的请求信息同样按照指导书的要求，标准格式为[CR,src,dst]，其中src和dst均为(x,y)的形式，除了空格外不允许出现其它的字符。例如合法的请求：

[CR,(80,80),(1,1)].

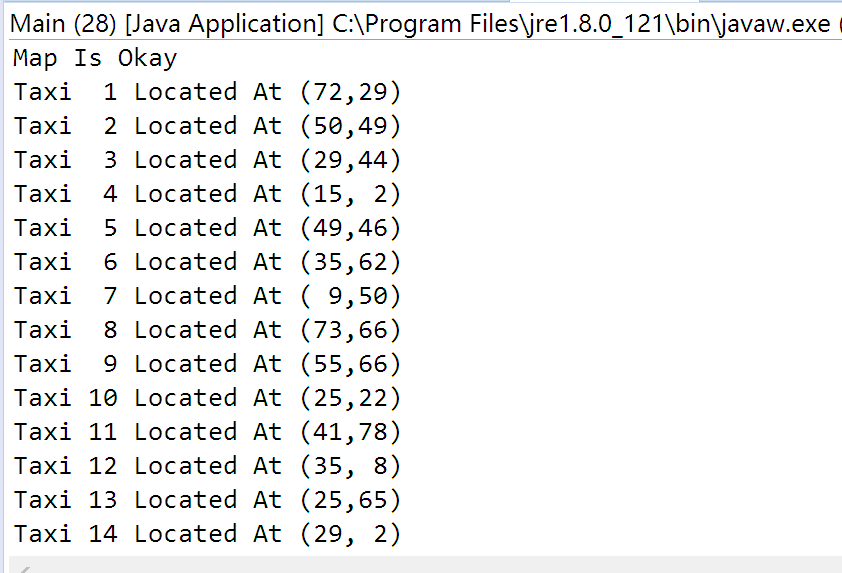
此外，为了提供可交互的提供道路临时关闭或打开功能，采用读入的方式来关闭打开道路。其中，打开一条道路的标准格式为[OPEN,src,dst]，关闭一条道路的标准格式为[CLOSE,src,dst]，其中src和dst均为(x,y)的形式，除了空格外不允许出现其它的字符。当要操作的两个节点不相邻时，会输出”Invalid CLOSE”或者”Invalid OPEN”。当关闭一条不存在的边时，会输出”Invalid CLOSE: Not Exists”；当打开一条已经存在的边时，会输出”Invalid OPEN: Already Exists”。两个可能的控制台的截屏如下：



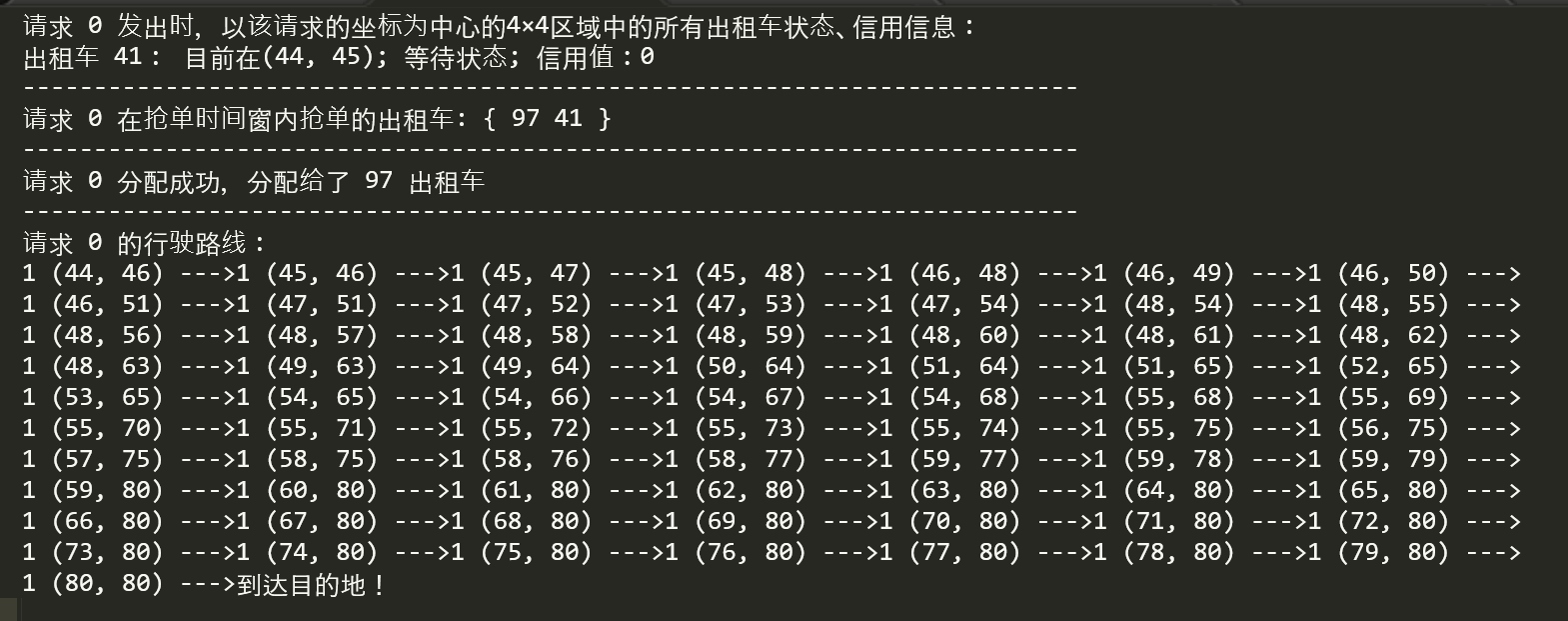
## 3.2 输出响应信息

总体见作业要求。

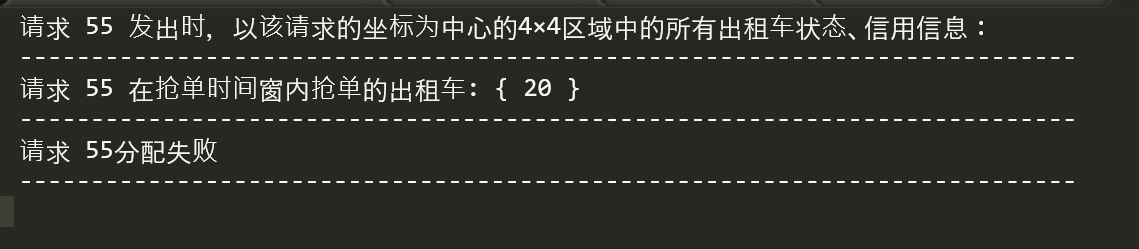
100辆出租车的起始位置信息通过随机化产生，然后会在控制台输出出来，例如图下的部分截图：



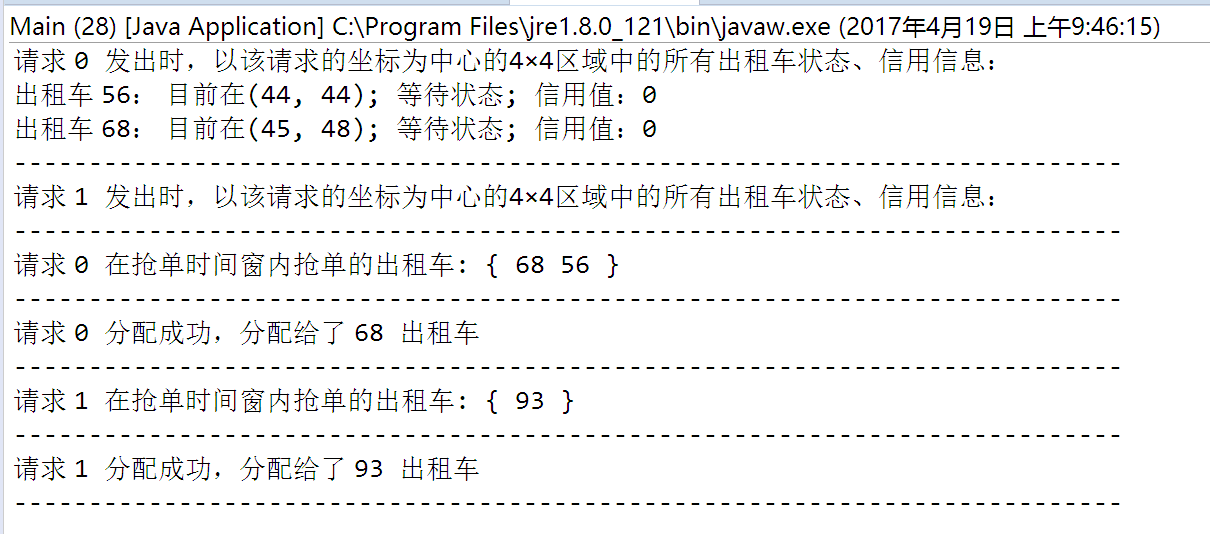
请求从0开始编号，每个请求都会产生一个logi.txt在工程文件夹下，其中i是请求的编号也就是第几个个请求。输出的信息包括四个部分，为指导书第三大点第4小点的要求：请求发出时，处于以请求src为中心的4×4区域中的所有出租车状态、信用信息；在抢单时间窗内所有抢单的出租车；系统选择响应相应请求的出租车；出租车响应相应请求过程中的实际行驶路径。可能的输出文件如下：



或者当缺失时的可能情形：



在控制台除了不会输出形式路线信息外，其它的在文件输出的都会在控制台显示。例如：



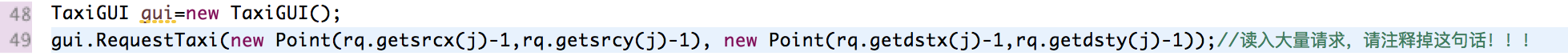
## 3.3一些重要的说明

1. 地图信息为Main类的第65行，测试者可以更改该文件名读取其它文件，默认工程文件下的map.txt为课程网站上的mytaxi里面的map.txt，然后工程文件夹下还提供了一个默认每个点和相邻点全相联的图map1.txt。

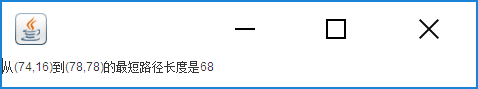


2. 在我的代码里面，地图是从(1,1)到(80,80)的，但为了输出到GUI的(0,0)到(79,79)，我传递给GUI上按照x坐标和y坐标都减1传递的。因此，相当于GUI上的(0,0)对应的我代码里的(1,1)。

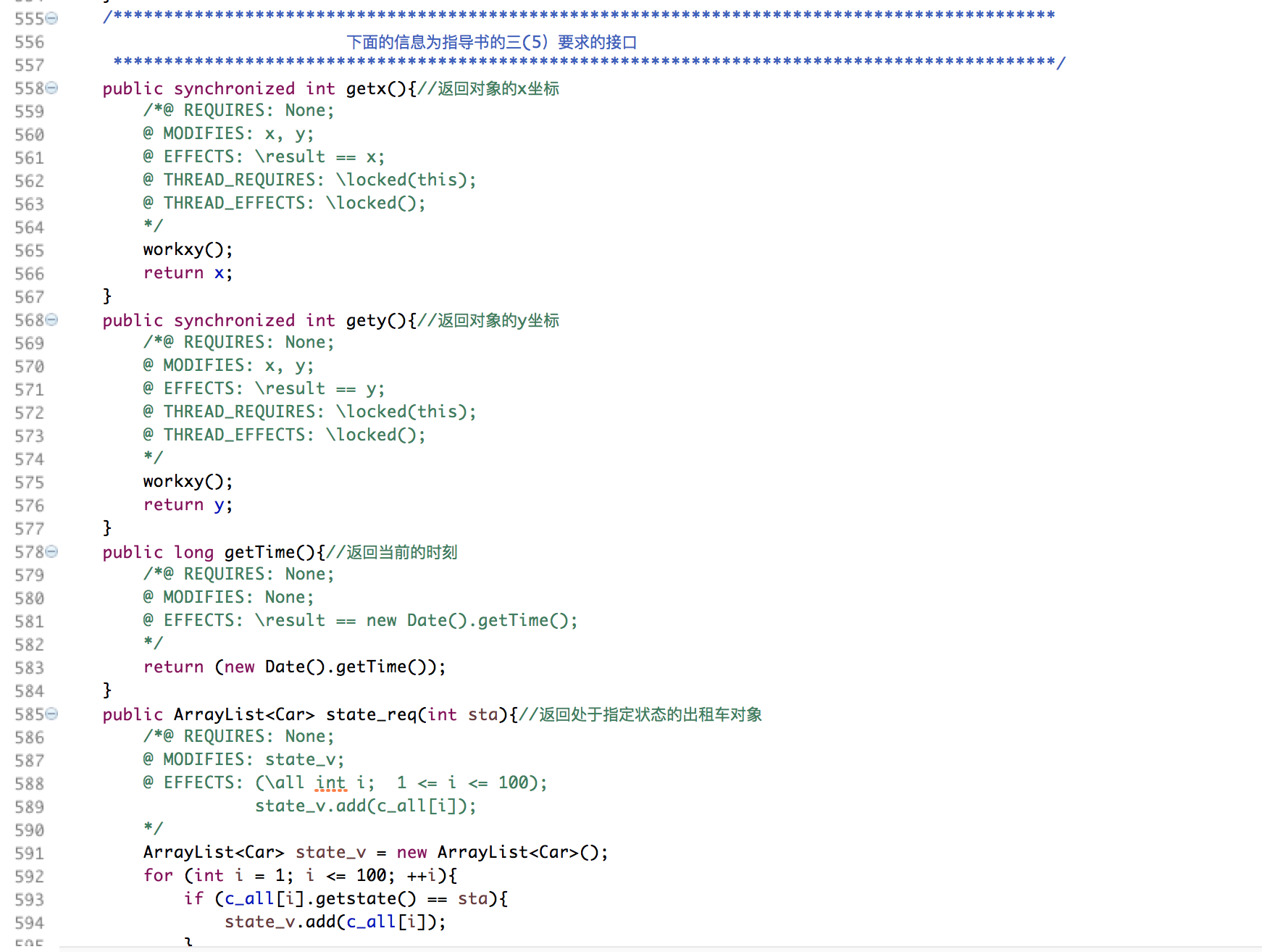
当读入大量的请求时，为了避免由于GUI的RequestTaxi的小窗口带来的大量的文件产生而产生的鬼畜现象导致系统卡顿，请测试者删去Dispatch类的第48、49行，如下图。



注释掉这句话后，这个小窗口就会消失，避免大量请求时的卡顿现象：



3. 关于指导书要求的测试接口查询指定出租车的状态，在我的Car类的最下面有几个函数，如图：



接口说明：

**public** **synchronized** **int** getx();//返回对象的x坐标

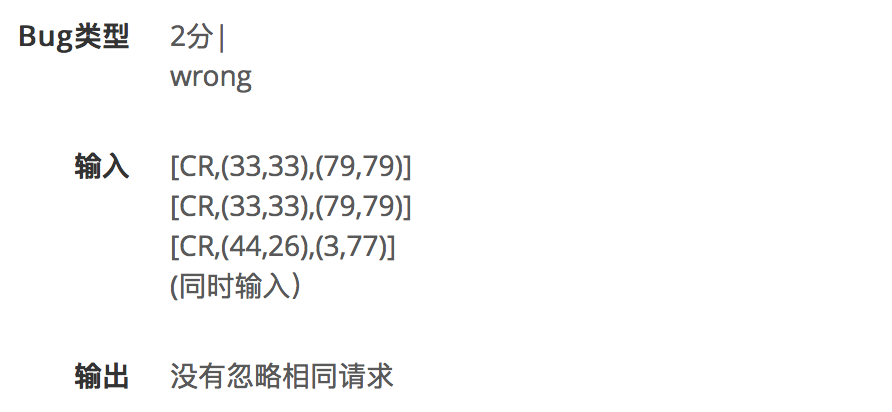
**public** **synchronized** **int** gety();//返回对象的y坐标

**public** **long** getTime();//返回当前的时刻

**public** ArrayList<Car> state\_req(**int** sta);//返回处于指定状态的出租车对象

# 上一次的bug修复

第一次的出租车作业我错了一个互测，没有错公测。互测的错误如下：



错误的原因是因为我没有忽略那个展示路径距离的小框框（从(0,0)到(0,1)的最短路径长度是1）GUI的展示的时间导致，导致后面的请求时间会比前面的长很多。因为我自己调试的时候都是注释掉那句话的，因此我自己测试的时候没有遇到过这个问题，而我交上去的时候是测试者看得到那个小框框的，因此导致了这个问题，其实主要是没考虑到GUI的展示需要较长时间，加上我每次我一读入就立刻展示了，因此出现了这个低级错误。是自己的疏忽与大意所致。

修改也很简单，挪一下展示小框框的代码的位置，不让它在请求输入之后就产生，而是在调度器线程里跑，这样就不会遇到向控制台复制进去多个请求判断不出来重复请求了。修改后的正确情况的部分截图：

