功能说明：

程序模拟出租车的乘客呼叫与应答系统，采用多线程的设计，并结合了相应的 GUI 即时显示结果。

程序运行所需环境和运行指令规范：

本程序需要在 java 虚拟机环境下运行，乘客请求从控制台输入，输出结果保存在工程目录下的 log. txt 文件中。

运行指令为:

```
javac TaxiSystem.class
java TaxiSystem
```

注意事项

1.由于官方 GUI 的性能问题，当有过多请求同时存在时，会导致 GUI 很卡，甚至出租车的路径不连续（GUI 显示的问题），所以实际运行情况以 log.txt 中的数据为准。

2.本程序中所有的时间是相对程序开始的时间，所以是从 0 开始的，但是因为每个线程启动的时间并不相同，所以如果电脑性能较差，或者一次性请求过多，会导致每个线程的初始时间误差大于 100ms，最终导致他们的初试时间不同，但从自己的实验来看，还没出现过这种情况。

3.对于在 7500ms 的窗口内抢单的出租车，一但出租车抢单，信用度立即+1，而不是等到 7500ms 之后才+1。

4.本程序是每个出租车线程 sleep 之后才更新其位置信息的，所以在抢单时获取的其位置信息很可能是其所在道路后向的节点的位置。

5.本程序最开始出租车的初始状态为停止状态。

6.针对预设场景，程序在运行开始前，需要测试者输入预设场景的文件名，如 test.txt，如果不设置预设场景，则输入 NO，程序会加载默认的 map.txt。

预设场景的格式参考指导书，并以目录下的 init.txt 为准。

7.如果出现中文乱码的情况，请到窗口->首选项->工作空间中更改编码为 utf-8。

输入说明

所有有关坐标的记录都是从 0 开始的，如地图大小为 80*80，则不存在(80,80)这个点。

请求的从控制台输入:

程序可以接受 [CR,(srcX,srcY),(dstX,dstY)]形式的输入，输入之间的空格都会被替换，因此接受含有空格的输入，但数字前不支持+号。本程序一行只能输入一个请求，多余一个整行都作为无效请求。此外，程序接受输入”end”作为整个程序结束的标志，注意大小写。

打开关闭道路的输入:

打开道路的指令:[OPEN,(srcX,srcY),(dstX,dstY)]

关闭道路的指令:[CLOSE,(srcX,srcY),(dstX,dstY)]

打开关闭道路后需要测试者保证地图的连通性。同时需要测试者保证道路两个点之间有存在边的可能性。

程序输出说明

对于每个请求，其处理过程记录在 log.txt 中，如果像验证请求处理的正确性，请阅读 log.txt。下面分别是三个请求的处理过程样例：

```
==========================================
发出时刻: 8300ms
出发地: (7,20)
目的地: (7,34)
#所有抢单的出租车的信息:
车辆编号: 64        车辆位置: (5,22)      车辆状态: PICK        车辆信用: 2
#没有出租车接单
==========================================
发出时刻: 9700ms
出发地: (15,57)
目的地: (63,31)
#没有出租车接单
------------------------------------------
```

```
========================================
发出时刻: 3300ms
出发地: (53,60)
目的地: (54,74)
#所有抢单的出租车的信息:
车辆编号: 90        车辆位置: (58,61)    车辆状态: IDLE        车辆信用: 1
车辆编号: 53        车辆位置: (51,55)    车辆状态: IDLE        车辆信用: 1
#被派单的车辆运行信息:
车辆编号: 90        派单时的车辆位置坐标: (58,61)  派单时刻: 10800ms
达到乘客位置的时刻: 14000ms  乘客位置坐标: (53,60)
中间点 时刻: 15000ms        坐标: (53,60)
中间点 时刻: 15500ms        坐标: (53,61)
中间点 时刻: 16000ms        坐标: (53,62)
中间点 时刻: 16500ms        坐标: (52,62)
中间点 时刻: 17000ms        坐标: (52,63)
中间点 时刻: 17500ms        坐标: (52,64)
中间点 时刻: 18000ms        坐标: (53,64)
中间点 时刻: 18500ms        坐标: (53,65)
中间点 时刻: 19000ms        坐标: (53,66)
中间点 时刻: 19500ms        坐标: (53,67)
中间点 时刻: 20000ms        坐标: (54,67)
中间点 时刻: 20500ms        坐标: (54,68)
中间点 时刻: 21000ms        坐标: (54,69)
中间点 时刻: 21500ms        坐标: (54,70)
中间点 时刻: 22000ms        坐标: (54,71)
中间点 时刻: 22500ms        坐标: (54,72)
中间点 时刻: 23000ms        坐标: (54,73)
到达目的地时刻: 23500ms      到达目的地坐标: (54,74)
========================================
```

测试接口说明

5) 要求提供**测试接口来按照出租车查询状态信息**，状态信息包括**查询时刻、出租车当前坐标、当前所处状态**。

6) 要求提供**测试接口来按照状态查询出租车**，返回所有在查询时刻处于相应状态的**所有出租车编号**。

指导书中所说的提供的测试接口在 Scheduler 类的最底部，我已经简单的实现了这两个功能，测试者可以看着两个方法的实现过程构造自己的测试方法。

```
public String queryOneTaxi(int No)//按照出租车查询状态信息，状态信息包括查询时刻、出租车当前坐标、当前所处状态。
public ArrayList<Integer> queryTaxisNowithState(TaxiState ts)//返回所有在查询时刻处于相应状态的所有出租车编号
```