1.Point类

// Overview： 点（x，y）x为行值，y为列值，用于表示地图上一个点的具体位置

// invariant： 0 <= x < MapInfo.MAP\_ROW && 0 <= y < MapInfo.MAP\_COLUMN

// (MapInfo.MAP\_ROW,MapInfo.MAP\_COLUMN分别表示地图的行数和列数)

// 表示对象： short x, short y

// 抽象函数: AF(c) = (x,y) x = c.x y = c.y

过程规格：

1.1  
// constructor

/\*

\* MODIFIES x,y

\* EFFECTS: x = y = 0

\*/

public Point()

/\*

\* MODIFIES x,y

\* EFFECTS: 用两个short型的数字为x和y赋值

\*/

public Point(short x, short y)

1.2

//EFFECTS： 判断一个点是否符合不变式，是返回true，否返回false

public boolean repOK ()

1.3

//EFFECTS： 判断一个点是否在列表里,若在则返回第一个下标值，否则返回-1

public static int isInLinkedList (Point p, LinkedList<Point> pl )

1.4

/\*

\* EFFECTS：判断点test是否在以点p为中心长度为1的十字架上，上1， 下2， 左3， 右4，若不在十字架上，则返回-1

\*/

public static int dirInCross ( Point p, Point test )

1.5

// EFFECTS： 随机产生一个点（x，y）：0 <= x < maxX; 0 <= y < maxY

public static Point getRandomPoint( short maxX, short maxY )

1.6

//EFFECTS： 得到左边的点

public Point getLeftPoint ()

1.7

// EFFECTS： 得到右边的点

public Point getRightPoint ()

1.8

// EFFECTS： 得到上面的点

public Point getUpPoint ()

1.9

// EFFECTS：得到下面的点

public Point getDownPoint ()

1.10

/\*

\* EFFECTS: 重写的Point 的 toString 方法

\* (non-Javadoc)

\* @see java.lang.Object#toString()

\*/

public String toString()

1.11

/\*

\* EFFECTS：重写的equals方法，两个点的x，y值都相等的时候，认为这是同一个点，返回true，否则返回false

\*/

public boolean equals ( Point p )

2.Taxi类

// Overview：出租车用于响应乘客的请求，具有四种运行状态，出租车每次只能响应一个请求

// 表示对象： short id, Point lastPosition, Point position, TaxiSatte state, int credit,

// PassengerReq pr, boolean called

// 抽象函数 ： AF(c) = (id,lastPosition,position,state,credit,pr,called)

// id = c.id, lastPosition:出租车上次所在点的坐标， position = c.position, credit = c.credit,

// state: in { WAITTING, READY,RUNNING, REST }

// pr:乘客的请求（当出租车状态是READY或者RUNNING时），null（其他状态）;

// called: true（当出租车状态是READY或者RUNNING时），false（其他状态）

// invariant： if state == WATTING or state == REST: called == false && pr == null

过程规格

2.1

// constructor

/\*

\* MODIFIES：出租车的所有属性

\* EFFECTS：创建一辆出租车编号为id的出租车，并且对它的所有属性进行初始化，

\* 其中position是在地图上随机选择的一个非平面交叉路口的点，初始状态下为WATTING。

\*/

public Taxi( short id )

2.2

// EFFECTS： 判断是否符合不变式，是则返回true,否则返回false

public boolean repOK ()

2.3

/\*

\* MODIFIES： pr, called

\* EFFECTS: 将乘客请求传给出租车，并将called属性置为true

\*/

public synchronized void call ( PassengerReq req )

2.4

/\*

\* MODIFIES：position

\* EFFECTS： 通过基于车流量考虑的最短路径到达目的地点end的过程

\*/

public void goDestination ( Point end )

2.5

/\*

\* MODIFIES： Flow.roadFlow， position， lastPosition

\* EFFECTS：考虑红绿灯，若为需要等待信号灯的情况，则等待，将车开到点p，同时对车流量进行计数

\*/

public void goPoint( Point p)

2.6

/\*

\* EFFECTS：休息1s

\*/

public void rest ()

2.7

/\*

\* EFFECTS：在等待服务的状态下随便走一个格子

\*/

public void randomRun ()

2.8

/\*

\* EFFECTS： 出租车的信用值增加n

\*/

public void addCredit ( int n )

2.9

\* EFFECT:返回一个布尔值。若出租车处于WAITTING状态则为true，否则为false

\*/

public boolean isTaxiWaitting ()

2.10

/\*

\* 测试者可以使用

\* 可修改

\* EFFECTS：返回出租车的信息

\*/

public String taxiInfo ()

3.MapInfo类

// Overview： 利用info数组来存储从地图文件中读取的关于地图的信息，用cross数组来存储从文件中读取的交叉路口信息

// 用cost存储每个点与其邻接边的信息

// 表示对象： short[][] info, boolean[][] cross, boolean[][] cost

// 抽象函数：AF(c) = (info,cross,cost) info[i][j] in {0,1,2,3} cross[i][j] = true:1,false:0;

// (以上两个二维数组取决于读入的文件)

// cost[i][j]:j方向上是否有道路，是true，否false；（j方向：0上，1下，2左，3右）

// invariant： 地图是连通的 && info[i][j] in {0,1,2,3}

过程规格：

3.1

// EFFECTS： 判断是否符合不变式，是则返回true，否则返回false

public boolean repOK ()

3.2

/\*

\* 可修改

\* 从文件中读入地图

\* 读入的地图是含空格的

\* MODIFIES： MapInfo.info

\* EFFECTS：从地图文件中读入地图的信息，将其存到MapInfo.info数组中

\*/

public static void readMap()

/\*

\* 可修改

\* 从文件中读入地图

\* 读入的地图是不含空格的

\* MODIFIES： MapInfo.info

\* EFFECTS：从地图文件中读入地图的信息，将其存到MapInfo.info数组中

\*/

public static void readMap2()

3.3

/\*

\* 可修改

\* 读入的是带空格的文件

\* MODIFIES：MapInfo.cross

\* EFFECTS：从文件中读入交叉路口的信息，存到Mapinfo.cross数组中，平面交叉：true，立体交叉：false

\*/

public static void readCross ()

/\*

\* 可修改

\* 读入的是不带空格的文件

\* MODIFIES：MapInfo.cross

\* EFFECTS：从文件中读入交叉路口的信息，存到Mapinfo.cross数组中，平面交叉：true，立体交叉：false

\*/

public static void readCross2 ()

3.4

/\*

\* MODIFIES： MapInfo.cost

\* EFFECTS：初始化6400\*4的矩阵用于表示邻接矩阵 下标 ：0上 1下 2左 3右

\* 值： true道路存在， false道路不存在

\*/

public static void initCost()

3.5

/\*

\* EFFECTS： 用BFS算法判断地图是否为连通图，是则返回true，否则返回false

\*/

public static boolean testMap()

3.6

/\*

\* EFFECTS：找到数组dist中最小的且s[i] == false的点，返回其下标

\*/

public static short minDist(boolean[] s, short[] dist)

3.7

/\*

\* EFFECTS: 通过广搜算法（BFS）算出start到end的最短路径，返回的是最短路径的长度

\*/

public static int calShortestLen ( Point start, Point end )

3.8

/\*

\* EFFECTS：计算点start到end应该走的第一步（在同样是最短路径的情况下，车流量小的优先），返回接下来走的点

\*/

public static Point getNextStep ( Point start, Point end

3.9

/\*

\* EFFECTS：得到一个随机邻点（上下左右选一个）

\*/

public static Point getRandomNbrPoint ( Point p )

3.10

/\*

\* EFFECTS： 得到某个点的附近区域的点集

\*/

public static LinkedList<Point> NbrPoints ( Point p )

3.11

\*

\* 打开一条道路

\* REQUIRES:点p1，p2必须是相邻的点

\* MODIFIES：MapInfo.cost， Mapinfo.info， MapInfo.Flow.roadFlow

\* EFFECTS: 对应的MapInfo.cost的值置为1,对应的MapInfo.Flow.roadFlow的值置为0,对应的MapInfo.info的值做相应的改变

\*/

public static void openRoad( Point p1, Point p2 )

3.12

\*

\* 打开一条道路

\* REQUIRES:点p1，p2必须是相邻的点

\* MODIFIES：MapInfo.cost， Mapinfo.info， MapInfo.Flow.roadFlow

\* EFFECTS: 对应的MapInfo.cost的值置为1,对应的MapInfo.Flow.roadFlow的值置为0,对应的MapInfo.info的值做相应的改变

\*/

public static void openRoad( Point p1, Point p2 )

3.13

/\*

\* EFFECTS:i,j是在地图内的某点的坐标（i,j） 通过（i，j）计算index

\*/

public static short calIndex(short i, short j)

3.14

/\*

\* EFFECTS：计算点p的索引值index

\*/

public static short calIndex(Point p)

3.15

/\*

\* EFFECTS：通过 index 计算 Point的坐标

\*/

public static Point calPoint(short index)

3.16

/\*

\* EFFECTS：给定一个地图中点p1,p2,给出p2在p1的cost数组中对应位置的costIndex

\*/

public static int getCostIndex ( Point p1, Point p2 )

3.17

/\*

\* EFFECTS：判断一个点的上边是否有路,有返回true，否则返回false

\*/

public static boolean hasUp ( Point p )

3.18

/\*

\* EFFECTS：判断一个点的下边是否有路,有返回true，否则返回false

\*/

public static boolean hasDown ( Point p )

3.19

/\*

\* EFFECTS： 判断一个点的左边是否有路,有返回true，否则返回false

\*/

public static boolean hasLeft( Point p )

3.20

/\*

\* EFFECTS： 判断一个点的右边是否有路,有返回true，否则返回false

\*/

public static boolean hasRight( Point p )

3.21

/\*

\* EFFECTS：判断该点的道路数量是否为3或4，是则说明这是一个交叉口，返回true，否则返回false

\*/

public static boolean isCrossRoad ( Point p )

4.Flow类

// Overview：用一个short型的二维数组来存储道路流量信息

// 表示对象: short[][] roadFLow

// 抽象函数： AF(c) = roadFLow (roadFlow[i][j]表示索引值为i的点的j方向的车流量，j方向：0上，1下，2左，3右)、

// invariant：roadFlow[i][j] >= 0 for all (i,j)

过程规格

4.1

// constructor

// EFFECTS:构造一个记录车流量的进程

public Flow()

4.2

// EFFECTS：判断是否符合不变式，是则返回true，否则返回false

public static boolean repOK ()

4.3

/\*

\* EFFECTS：初始化道路流量信息数组

\*/

public static void initRoadFlow ()

4.4

/\*

\* 车流量+1

\* REQUIRES: 点start和点end必须都在地图内

\* MODIFIES： None

\* EFFECTS： 对应的roadFlow增加1，表示经过的车流量增加

\*/

public static void addFlow( Point start, Point end )

4.5

/\*

\* 车流量-1

\* REQUIRES: 点start和点end必须都在地图内

\* MODIFIES： None

\* EFFECTS： 对应的roadFlow减少1，表示经过的车流量减少

\*/

public static void subFlow( Point start, Point end )

4.6

\*

\* EFFECTS： 返回道路（p1,p2）的实时车流量,若该道路不存在则返回-1

\*/

public static short getRoadFlow ( Point p1, Point p2 )

5.TrafficLight类

//Overview: 这个类是用来实现红绿灯的控制，用一个short的二维数组来表示红绿灯

// 表示对象：short[][] tflight

// 抽象函数：AF(c) = tflight

//( tflight[i][j] 表示坐标为（i,j）点 处的红绿灯状况，

// 1表示上下方向可以通行，左右方向被阻塞 -1表示左右方向可以通行，上下方向被阻塞,0表示该点不存在红绿灯)

// invariant： tflight[i][j] in {0,1,-1}

过程规格

5.1

// constructor

/\*

\* MODIFIES：Traffic.tflight

\* EFFECTS：利用MapInfo类中的信息对tflight数组进行初始化，当一个点是平面交叉且有三条或四条道路的时候

\* 为它在tflight相应的位置随机设置红绿灯（值设成1或-1），否则不设置红绿灯（值设成0）

\*/

public TrafficLight ( )

5.2

// EFFECTS：判断是否符合不变式，是则返回true，否则返回false

public static boolean repOK ()

5.3

//mutator

/\*

\* MODIFIES：TrafficLight.tflight

\* EFFECTS：改变红绿灯的状态，红灯变绿灯，绿灯变红灯

\*/

public static void changeLightState ()

5.4

/\*

\* EFFECTS：根据出租车的位置和它下一步要去的点以及红绿灯，判断出租车是否需要等待

\* 1.任何情况下右转都不需要等待 ；2.直行、左转和掉头的时候，看将要去的路：是绿灯不等待，红灯等待

\*/

public static boolean waitLight (Point lastPosition, Point position, Point p )

6.PassengerReq类

// Overview：乘客请求，用于向请求处理线程发送请求

// 表示对象：fromPoint, toPoint

// 抽象函数：AF(c) = (fromPoint,toPoint)

// 输入 的字符串分割后的第1，2位为fromPoint.x,fromPoint.y 第3，4位为toPoint.x,toPoint.y

// invariant： fromPoint.repOK == true && toPoint.repOK() == true

过程规格

6,1

// constructor

// EFFECTS: 根据输入的字符串获取请求的信息，对fromPoint和toPoint进行初始化

public PassengerReq ( String s )

6.2

/\*

\* EFFECTS:检查一个请求是否合法（即是否符合不变式）,合法返回true，非法则返回false

\*/

public boolean repOK ()

7.Despatcher类

// Overview：调度器类，对一个请求进行调度，进程持续windowTime时间，

// 为请求寻找合适的出租车，若找不到则提示寻车失败

// 表示对象： PassengerReq pr

// 抽象函数： AF(c) = pr pr = c,pr

// invariant： pr.repOK()

过程规格

7.1

// constructor

// EFFECTS: 利用乘客请求构造一个调度器线程

public Despatcher ( PassengerReq pr )

7.2

/\*

\* EFFECTS:检查否符合不变式,是返回true，否则返回false

\*/

public boolean repOK ()

7.3

/\*

\* REQUIRES：tArray是待选择的出租车队列

\* MODIFIES：tArray

\* EFFECTS：选择最优的出租车

\*/

public Taxi chooseTaxi ( ArrayList<Integer> tArray )

7.4

/\*

\* EFFECTS：返回出租车队列中credit最高值

\*/

public static int getMaxCredit ( ArrayList<Integer> lk ) {

int max = 0;

for( int i = 0; i < lk.size(); ++i ) {

Taxi t = TaxiSystem.taxis[lk.get(i)];

if( t.credit > max ) {

max = t.credit;

}

}

return max;

}

7.5

/\*

\* EFFECTS： 返回出租车与请求起始点的路程的最近的出租车列表

\*/

public static ArrayList<Integer> MinDistList( ArrayList<Integer> tArray, PassengerReq req )