# Class pos

//overview: pos记录地图上每一个点的位置，x表示横坐标，y表示纵坐标，x,y为非负整数。

# Class node

//overview: node记录箱子推动过程中每一步状态信息，形如(mx, my, bx, by, step, prev)。 mx, my分别表示人的横坐标、纵坐标；bx, by分别表示箱子的横坐标、纵坐标；step记录箱子走过的步数；prev记录箱子运动轨迹点的前序位置。

# Class PushBox

//overview: PushBox接收用户从控制台的输入，进行人的移动和箱子的移动。该类管理着地图map和箱子推进的路径，并记录箱子的当前位置和目标位置。m和n分别为地图的高和宽；q记录当前路径需要遍历的结点队列；start为人和箱子的起始位置。vis记录节点被访问过的状态（true or false）, dx取值{0, 0, 1, -1}, dy取值{1, -1, 0, 0}，表示人或箱子向上、下、左、右移动。在编写测试中，以上属性都可以在类外直接被访问（非私有属性）。

public boolean readmap()

//map结构：第1 行有2个正整数n和m（1<=n,m<=20）表示地图的行、列宽度；除第一行外的整数组成一个n\*m的矩阵。对于满足(n+2)>=i>=3的第i行中满足1<=j<=m的第j列，输入的整数只能是{0,1,2,3,4}中的某个取值，表示格子（i,j）的状态：0表示格子空闲；1 表示格子上放了不可移动的沉重货物（障碍物）；2 表示箱子的初始位置；3 表示箱子的目标位置；4 表示仓库管理员的初始位置。人和箱子的所在位置横、纵坐标，分别为(mx,my),(bx,by)

/\*@ REQUIRES: System.in!=null ;

@ MODIFIES: this;

@ EFFECTS:

(count(2)||count(3)||count(4)!=1)==>(( System.out(Illegal Input!))&& (\result==false));

(count(2)||count(3)||count(4)==1)==>((start.mx=this.mx)&&(start.my=this.my)&&(start.bx=this.bx)&&(start.by=this.by) && (\result==true));

\*/

public void pre(Queue<node> q)

// q为当前路径需要遍历的结点队列。

/\*@ REQUIRES: q;

@ MODIFIES: this;

@ EFFECTS:

(q==null)==> q.clear()

(q!=null)==>(\all 0=<i,j,k,l<=20, vis[i][j][k][l]=false)&&(\result==true))

\*/

public boolean check(int x, int y)

/\* @REQUIRES: this

@ MODIFIES: none;

@ EFFECTS:

(map[x][y]==1)||x>=m||y>=n)==>(\result==false)

(map[x][y]!=1||(0=<x<m&&0<=y<n))==>(\result==true)

\*/

public boolean bfs\_man(int sx, int sy, int ex, int ey, int bx, int by)

//判断人能否从当前点走到与箱子移动方向相反的那个相邻点

//q记录人从当前所在位置(sx, sy)到目标位置(ex, ey)需要经过的路径

/\* @REQUIRES: this

@ MODIFIES: none;

@ EFFECTS:

(check(sx,sy)==false||q==null)==>(\result==false)

((check(sx,sy)==true)&&

(\all q.node; q.node.bx!=bx||q.node.by!=by)==>(\result==true)

\*/

public void bfs()

//针对箱子的目标位置计算箱子的下一步移动方向，以及人能够按照所计算方向来推动箱子的期望位置，即与箱子将要行进方向相反的相邻位置。q记录箱子从当前位置到目标位置经过的路径节点

/\* @REQUIRES: this;

@ MODIFIES :this;

@ EFFECTS:

(q==null)==>(System.out(“No solution!”))

(\all q.node, 0=< q.bx =< m && 0=< q.by =< n)==>(System.out(q.size()&&q))

)

AF(c)={Queue<node>, map, vis,dx,dy,m,n, start}

Where Queue<node>=c.queue, map=c.map, dx,dy={-1,0,1}, start=c.start,20=>m,n>=1

不变式

对于Queue<node>中的所有结点q（即广度优先搜索路径上访问的所有结点）都有

0=<q.mx , q.bx =< m && 0=<q.my, q.by =< n