Model

Maze + <u>X</u>: int + Y: int + RATE OF ONEWAY : double PATH : int SHORTEST PATH : int WAY RIGHT : int + WAY LEFT : int + WAY UP : int + WAY_DOWN : int START : int + WALL int GOAL · int - DEFAULT_WIDTH: int DEFAULT HEIGHT : int DEFAULT WIDTH2 : int DEFAULT HEIGHT2 : int X MARKER : int[] + Y MARKER : int [] + random : Random start : Point goal : Point - width · int height : int map : int[][] isSolutionShown : boolean ⊥ Maze() + resetStartAndGoal() : void - setStartAndGoal() : void - getAdiacent(point : Point) : ArravList<Point> + breadthFirstSearch() : void breadthFirstSearch(node: Point, unvisited: HashSet<Point>, visited: HashSet<Point>, queue: LinkedList<Point>): void findPath() : void solve(): void - getSmallestDAdjacent(adjacentList : ArrayList<Point> .d : HashTableChain<Point, Integer>) : Point - getSmallestDVertex(d : HashTableChain<Point, VmS : Integer>.HashSet<Point>) : Point + getStart() : Point egetGoal() : Point e getWidth(): int e aetHeiaht() : int getSameRegion(x : int, y : int, allSet : HashTableChain<String,Point> , set : HashTableChain<String, Point>) : int isLeftPath(x:int, y:int):boolean isLeftPath(point : Point) : boolean isRightPath(x:int, y:int):boolean + isRightPath(point : Point) : boolean + isDownPath(x : int, y : int) : boolean + isDownPath(point : Point) : boolean + isUpPath(x : int. v : int) : boolean + isUpPath(point : Point) : boolean + isPlayerAtGoal(point : Point) : boolean - verticalClone(x:int, y:int, visited: HashTableChain<Point, Boolean>, pathType:int): void - horizontalClone(x : int, y : int, visited : HashTableChain<Point, Boolean>, pathType : int) : void - distance(p1 : Point, p2 : Point) : double + draw(gr : Graphics) : void + hideSolution() : void showSolution(): void + makeTraffic() : void





