Classes/Methods to Use for DFS

MoveDirection/CellStatus

To use the directions (Up, Down, etc.) without doing MoveDirection.Up, and just say Up, you need to include the MoveDirection enum like this:

Import static MoveDirection.*;

Mine is in a package called Part0, so for me I do Import static Part0.MoveDirection.*; but it depends on your directory structure.

Cell

Peek/MoveToNeighbor – You should use this to traverse the grid and to look at its neighbors. They are actually the same function but you should use peek to look at neighbors and moveToNeighbor to move, because its more readable. But they do the same thing, you give it a direction, and it fetches the cell in that direction. Returns null if you reached the boundary or gave negative number.

setAsBlocked/setAsUnblocked – Use this to change the status of the cell.

Maze

`createNewMaze` – You can't actually instantiate a Maze object normally, you need to use this to make a new maze.

`saveMaze` `– You need to save the generated maze afterward by calling this instance method of the maze object.`

`getCellAtCoordinates` `– You can use this method and pass in 0,0 to get the top left cell.`

TreeNode

`generateInitialNode` `– You should use this method to generate the initial treenode`

`TreeNode (Constructor) –` The definition is straightforward but you give the node the additional cost of the path of adding in this current treenode (which in our project should always be 1)

`appendChild` `– Use this method to add a child treenode to this treenode.`

`nextChild` `– Use this method to get the next child of the treenode.`

UnvisitedNodesTracker

You should make a new object for every new generated maze.

`markNodeAsVisited –` Remember to call this method every time you finish expanding a node or set it to blocked.

`getUnvisitedNode` `– Use this to obtain coordinates for a new unvisited node.`