Maze Program

Part I

You will be creating a maze program.
The maze design will be stored in a text file.
The design must then be read in by your program and stored in a 2-dimensional array (contains Walls).
List of objects that must be coded:

- Explorer – This is the playable character that will move through the maze. Explorer must have a move() method (which will be called in the keyPressed method in your runner) and store the current position as a Location.
- Location – This will be used to keep track of Locations (Explorer and Maze will both use this object)

Some things to keep in mind:

- User must press keys to make the Explorer traverse the maze. Explorer cannot move through walls.
- Program must display the number of moves it took to finish the maze. The count must be modified after each move.
- An announcement must be made when the maze has been completed. “Great Job!” or “It looks like you were lost for a while. Glad you found the way out.”
- You must declare initial starting position for the Explorer.

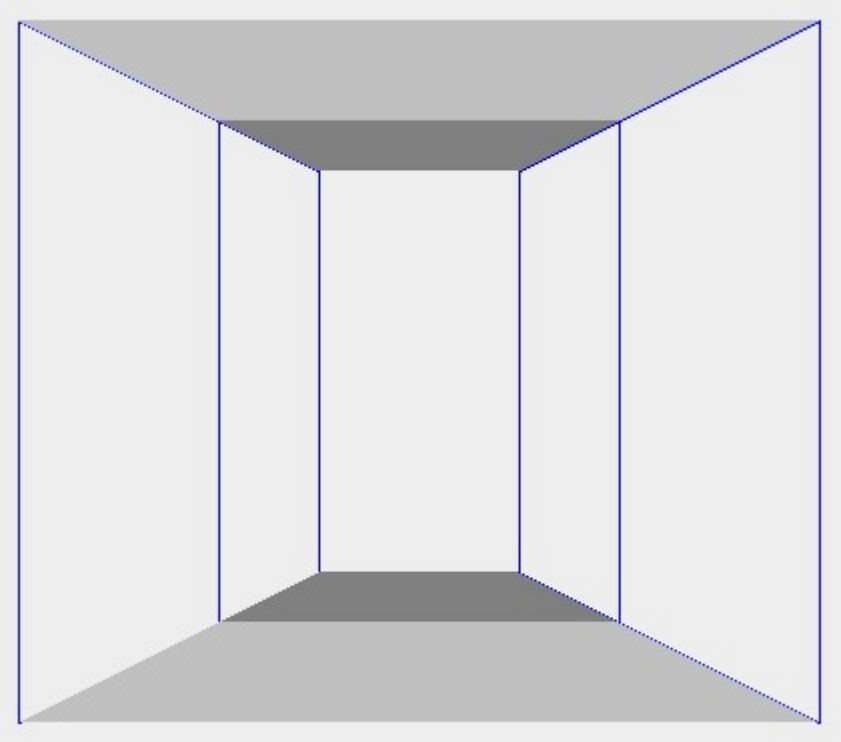
Programming Tasks:

- Declare a global 2-dimensional array (or an ArrayList of ArrayList) of Wall that will be used to store all of the wall locations in the maze. Make sure that your maze design is standardized so that your array can be initialized for any design you produce without producing errors.
- Complete a setBoard method.
    - Initialize all the Wall(s) in the board. (This needs to be called first in your runner constructor because this will prevent a null pointer exception from happening at the beginning when you run your program. The program must have something to draw.)
    - Inside of the text-file loop, you need to chop each line from the text file into single characters, identify if they are walls or empty spaces and store them in the array.
    - If you code your Explorer's character in the maze design (which would be the easiest design), instantiate your Explorer with a Location.
- Complete the paintComponent method:
    - This is where you draw the board and the character. Use fillOval and fillRect to draw the maze with the character inside it.
- Use the key pressed events to control movement of your character.
    - Movement forward MUST always be the up arrow. The left and right arrows are to turn your character 90 degrees left or right. Do not code the down key. If you want to go back the way you just came from, you have to turn around before you walk back that direction. This is important, because in the 3D version, it won't make any sense to code a traditional up/down/left/right 2D maze movement method.

PART II

Redraw your maze in 3D. You must always draw at least 3 spaces ahead of your moveable character. You need to create a new object:

- Wall – This is an object used to store a Wall. All wall positions will be stored using the Location class. Each wall will be stored in an ArrayList of Wall objects.



This is just an example of what the lines can look like.

Make use of a wall object (walls, floor, ceiling) that would retain the coordinates of the shape as a Polygon to make it easy to draw the polygons that represent the walls, floors, or ceilings.

Because you set movement based on the up arrow always being the key to move forward, you should have an easier time of it when it comes to drawing your maze.

<u>Grading:</u>

- 2D maze perfectly functional – 30%
- 3D maze perfectly functional – 40%
- Modify 3D maze with add-ons would get the final 30% (each add-on tacks on 10% - so 3 would get you to 100%)

Maze Add-On Ideas

When you're finished with a playable basic 3D version, you must code in three gameplay additions. These additions are to make it more like an adventure.

Keys and Doors: Place a key in the maze and a doorway to another maze. You could tie a bunch of mazes together to turn it into an adventure. Leave one maze, enter another.

Keys and Treasure Chests: Place a key in the maze that is used to unlock a treasure chest. The treasure can't be opened unless you have the key.

Traps and Puzzles:

- Traps that'll cause some serious damage
- Puzzles. Pull a some switches that will open a passage that hides something.

Monsters that chase you: It's like an "anti-Explorer". Just code in an AI that will close the gap on you.

Flashlight/Torch: Change how far you can see in the maze. Code it to dim as you walk (just change the colors to get progressively darker) until you find a new power source.

Ability to drop bread crumbs (keep a limit on crumbs) or spray paint/tag a wall or floor so that you know where you have been.

These are just ideas. You can add just about anything. Just keep in mind that sound is not an add-on worthy of one of three "10%"s. It just isn't much code to add it in (although, it is pretty cool for atmosphere.)