# UML for maze_generator

| **maze_generator** |
| --- |
| • import java.util.Stack<br>  ○ For the maze generation algorithm utilized in this class, we will need the Stack data structure, thus its methods must be imported.<br>• cell[][] maze<br>  ○ Holds our maze as a two-dimensional array of cells.<br>• Stack<cell> path<br>  ○ The stack will hold an arrangement of cells that will denote the current path you are taking.<br>• cell current<br>  ○ A cell that marks where you currently are on the board.<br>• Character dude<br>  ○ The character that is invoked once the maze generation is over.<br>• int x<br>  ○ Current xcor within array [y][x]<br>• int y<br>  ○ Current ycor within array [y][x]<br>• int newX<br>  ○ Used for transitions with midX<br>• int newY<br>  ○ Used for transitions with midY<br>• int midX<br>  ○ Used for walls in between cells<br>• int midY<br>  ○ Used for walls in between cells<br>• void setup()<br>  ○ Sets up the maze two-dimensional array.<br>• void draw()<br>  ○ After creating the maze array, this portion of the method creates the actual path that takes you from one starting point to one end point.<br>• Void keyPressed()<br>  ○ If the key ‘k’ is pressed, the generation is stopped. |

**Character**

- color col;
  - Variable for the color
- int lives
  - Counts the amount of lives you have left
- int xpos
  - Marks the current x-position of the character
- int ypos
  - Marks the current y-position of the character
- int xperm
  - Marks the original x-position of the character (important for spawning)
- int yperm
  - Marks the original x-position of the character (important for spawning)
- void printCircle()
  - A display method for the circle based on position and color.
- boolean isAlive()
  - Uses the amount of lives to check whether or not you are alive.
- void moveKey(int i)
  - If…
    - i = 0: move up
    - i = 1: move down
    - i = 2: move left
    - i = 3: move right
- void interact()
  - If you are not on the path, you lose a life and you respawn. This is dictate by the color of the floor that you are on.
- void run(int i)
  - A method that invokes…
    - printCircle()
    - moveKey(i)
    - interact()

| **Cell** |
| --- |
| ● protected color c<br>○ The color of the cell, which plays a role in what each cell represents separately. For example, a green cell would denote the path while the black cells would denote a wall.<br>● protected boolean unvisited<br>○ Marks whether or not a cell was visited in the maze generation algorithm.<br>● protected int x<br>○ X-coordinate of the center of the cell.<br>● protected int y<br>○ Y-coordinate of the center of the cell.<br>● color getColor()<br>○ Accessor method for the color of the cell.<br>● void setColor(color col)<br>○ Mutator method for the color of the cell.<br>● int getX()<br>○ Accessor method for the x-position.<br>● int getY()<br>○ Accessor method for the y-position.<br>● void visit()<br>○ Changes the color of the cell to blue, which shows that it is in the midst of being a part of the maze generation.<br>○ Sets the unvisited boolean to false, because it is now visited.<br>● void backTrack()<br>○ Changes the color of the cell to green, in order to confirm that it is part of the finalized maze.<br>● boolean unvisited()<br>○ Accessor method for the unvisited boolean. |

| **Wall extends Cell** |
| --- |
| ● x and y are set to inputted values in the constructor.<br>● The color is set to black.<br>● The wall starts and remains unvisited. |