Updated Program Specification

1. At the end -
    a. When run, the first thing that happens is a panel pops up asking the user to fill in specifications for the size, type of maze, an integer for the key, and if they want enemies or not.
    b. After putting in their specifications, the user can click start and a new frame will pop up
    c. Visual display will either be a kind-of-3D rendering of a maze or a 2D birds eye view of a maze. The user can move through the maze to solve it
    d. After solving the 3D maze, the player will be able to input their name and time into a leaderboard, tracking all the best times for that size of maze
2. User input
    a. The user will be able to select the options for their maze from a start screen by clicking on buttons in the window. Options include maze dimensions, 2d or 3d, the integer maze key, and if they want enemies or not. When playing the game, the user will be able to use the keys on the keyboard to control their movements in the maze. (arrow keys to move forward/right/back/left and a/d to turn left/right for the 3d maze).
3. How will the program respond
    a. The graphics will change with a user inputted movement according to the location of the walls around them for the 3d maze.
4. What purpose does it serve
    a. It is a game

PART 2

- Classes
    - Main - runs the method
        - Methods
            - public static void main(String[] args) - main method runs the program
    - Cell - a cell in the maze grid
        - Member variables
            - boolean unvisited - if the cell has been visited (while making the maze)
            - boolean Path - if the cell is part of the solution from the start point to the end point
            - boolean UpWall - if the cell has a wall above it
            - boolean LeftWall - if the cell has a wall to its left
            - boolean RightWall - if the cell has a wall to its right
            - boolean DownWall - if the cell has a wall below it
            - boolean enemy - if the cell has an enemy in it
            - boolean heart - if the cell has a heart in it
            - int life - how much life the heart or enemy has

        - Constructors
            - public Cell() - initializes all the member variables for a blank cell
        - No methods
    - Maze - a representation of the maze → inputting the dimensions and a key (for a random object) will generate a maze accordingly
        - Member variables
            - Cell[][] mazeGrid - stores the cells in the maze
            - int dimensions - the dimensions of the maze (rows x columns for a square maze)
            - int key - initializing a random object to a specific key
            - boolean ifEnemies - if the player wants enemies in the maze
            - Random rand - random object to help generate the maze
            - int xPos - the current x position of the maze generator
            - int yPos - the current y position of the maze generator
            - int xStart - the random x coordinate of the starting location of the maze generator
            - int yStart - the y coordinate of the random starting location of the maze generator
            - BacktrackerDS ds
        - Constructors
            - public Maze(int dimensions, int key, boolean ifEnemies) - initializes the dimensions, the key, the enemies, and creates a blank mazeGrid
        - Methods
            - public void createMaze() - creates the maze according to dimensions and key
            - private void ChooseNeighborSolution() - This method is used when finding the solution, it chooses a neighbor that hasn't been visited and there is no wall between the cells, It will backtrack to the previous cell if it can't move to one of its neighbors
            - private boolean[] getNeighborSolution() - This method returns a boolean array telling the ChooseNeighborSolution() method which cells it can move to
            - public void ChooseNeighbor() - This method is used when creating the maze, it chooses a neighbor that hasn't been visited, It will backtrack to the previous cell if it can't move to one of its neighbors
            - public boolean[] getNeighbors() - This method returns a boolean array telling the ChooseNeighbor() method which cells it can move to
    - BacktrackerDS - a linked list that allows the maze to know the path it has already taken so far and how to backtrack to a cell that has unvisited neighbors
        - Member variables

        - Node end - stores the end of the linked list
        - int length - the length of the data structure
    - Constructors
        - public BacktrackerDS() - initializes end to null and length to 0
    - Methods
        - public void append(Boolean toAppend, int X, int Y) - appends a cell to the data structure
        - public void pop() - pops the last item out of the data structure
        - public int returnX(int X) - returns the x coordinate of the last cell in the data structure
        - public int returnY(int Y) - returns the y coordinate of the last cell in the data structure
        - public int getLength() - returns the length of the data structure
- Node - node for the backtrackerDS (linked list node) representing a cell
    - Member variables
        - boolean visited - if the cell has been visited
        - int x - the x coordinate of the cell
        - int y - the y coordinate of the cell
        - Node next - the next node in the data structure
    - Constructors
        - public Node(boolean visited, int x, int y) - initializes a node with those parameters
- 2DGraphicInterface extends JPanel implements KeyListener - this class creates the GUI where the 2D maze is displayed
    - Member variables
        - Cells[][] mazeGrid - the maze
        - int boxSize - the pixel size of each cell in the maze to determine the frame size with
        - int xPosition - the x position of the player in the maze
        - int yPosition - the y position of the player in the maze
        - boolean solution - if the solution should be displayed
        - JFrame frame2D - the JFrame where the maze is displayed
    - Constructors
        - public 2DGraphicInterface(Cells[][] mazeGrid) - initializes the maze grid
    - Methods
        - public void paintComponent(Graphics g) - draws the maze and the player
        - public void keyReleased(KeyEvent e) - This method uses the KeyListener to update the player's position when they press one of the arrow keys or displays the solution when they press '1'.
        - keyTyped(KeyEvent e) - need to implement with KeyListener but not actually used

            - keyPressed(KeyEvent e) - need to implement with KeyListener but not actually used
    - 3DGraphicInterface extends JPanel implements KeyListener - generates an interactive depiction of the 3D maze based on the user specifications chosen in the main menu
        - Member variables
            - Cells[][] mazeGrid - the maze
            - int difficulty - number of steps until solution
            - int direction - (0 for north, 1 for east, 2 for south, 3 for west) - direction the player is facing
            - int xPosition - the x coordinate of the player
            - int yPosition - the y coordinate of the player
            - boolean[] displayWalls - boolean array containing statements for the graphics of different walls inside the maze
            - JFrame frame3D - the JFrame that displays the maze
            - int life - how much health the player has (out of 100)
            - int[] farcells - array of cells that are in view for the player
            - boolean winner - if the user has completed the maze
        - Constructors
            - public 3DGraphicInterface(Cells[][] grid, int difficulty) - initialize member variables at the start of the maze
        - Methods
            - public void paintComponent(Graphics g) - draws the floor, the mist, the walls, the compass, enemies, and health to the screen
            - public void enemy(Graphics g, int ovalXY, int ovalW, int ovalH, int xp1, int xp2, int xp3, int yp1, int yp2, int yp3, int xp4, int xp5, int xp6, int xp7, int xp8, int xp9, int yp7, int yp8, int yp9, int xp10, int xp11, int xp12, int ovalx1, int ovaly1, int ovalw1, int ovalh1,int eyeX, int eyeY, int eyeWH, int eyeX1, int eyeX2, int eyeY1, int eyeWH1, int eyeX3, int arcX, int arcY, int arcW, int arcH, int eyeX4,int eyeY2, int eyeWH2, int eyeX5, int tX, int tY, int tW, int tH, int tX1) - draws the enemies
            - public void drawHeart(Graphics g, int x1, int x2, int x3, int y1, int y2, int y3, int y4) - draws the hearts
            - public void compass(Graphics g) - draws the compass in the bottom left corner
            - private int EnemyorHealth - method that determines whether an enemy or a health is drawn
            - private void walls - method that draws the walls of the maze
            - private Color shadingForBackWalls() - method that determines the shading of the back walls
            - private Color shadingForLeftWalls() - method that determines the shading of the left walls

- private Color shadingForRightWalls() - method that determines the shading of the right walls
- private void mist (Graphics g) - method that draws the mist
- private void floor(Graphics g) - method that draws the floor (makes the floor a darker red as the player gets closer to the finish)
- public boolean[] whichWallsToDraw - method that determines shading/which walls to draw
- private boolean[] left(boolean[] walls) - creates boolean array for which walls to draw if player is facing left
- private boolean[] right(boolean[] walls) - creates boolean array for which walls to draw if player is facing right
- private boolean[] up(boolean[] walls) - creates boolean array for which walls to draw if player is facing up
- private boolean[] down(boolean[] walls) - creates boolean array for which walls to draw if player is facing down
- private boolean WallOnOff(int wall, int Xuse, int Yuse) - turns walls on or off
- public void keyPressed(keyEvent e) - tracks if the player presses the space bar to either attack the enemy or heal
- public void keyReleased(keyEvent e) - player users the arrow keys to move and a or d to turn (triggers when key is released)
- public void keyTyped(keyEvent e) - need to implement with KeyListener but not actually used

- FrontPageGraphics extends JPanel implements KeyListener, MouseListener, ActionListener - the front page allows the player to customize the maze by choosing its dimensions, the type of maze, a custom key, and enemies if they choose
  - Member variables
    - int dimensions
    - int typeOfMaze - 2 for a 2D maze, 3 for a 3D maze
    - int key - if the user wants to input a key for a preset maze
    - JFrame frame - the frame that serves as the GUI
    - boolean check - if the start button is pressed to continue the program
    - boolean ifEnemies - if the user wants enemies
    - int maxDimensions - the maximum dimensions the maze can have
  - Constructors
    - public FrontPageGraphics() - default constructor that initializes the initial screen with the frame and buttons
  - Methods
    - No methods
- ImagePanel extends JComponent - a class that stores an image and acts as a JComponent so the main menu can set its background to an image

        - Member variables
            - Image image - the image passed in that will serve as the background for the main menu
        - Constructors
            - public ImagePanel(Image image) - initializes the image
        - Methods
            - protected void paintComponent(Graphics g) - displays the image on the main menu
    - Leaderboard - generates the GUI for the leaderboard and manages reading and writing to the text files where the leaderboard is stored
        - Member variables
            - int dimensions - dimensions of the maze
            - ArrayList<Player> leaderboard - an arraylist to store the information of the leaderboard with player objects
            - String leaderboardPath - a string to store the file path of the different leaderboards given the dimensions
        - Constructors
            - public Leaderboard(int dimensions, double time) - the constructor reads the information in the leaderboard text file and generates a GUI to display it. Here, the user can also submit their time to the leaderboard and the constructor will write their data to the text file
        - Methods
            - private void addPlayer(String name, double time) - adds a player to the leaderboard ArrayList in the correct spot when they submit their time and updates the leaderboard text file
            - private String convertTime(double time) - converts the player's time in milliseconds to the form of minutes:seconds:deciseconds to be displayed on the leaderboard
    - Player - stores one player's name and time as one element in the leaderboard ArrayList
        - Member variables
            - String name - the player's submitted name
            - double time - the player's time in milliseconds
        - Constructors
            - public Player(String name, double time) - initializes a player given their name and time
        - Methods
            - No methods

Class interactions

- When the main method is run, a new instance of FrontPageGraphics is created
- When the start button is clicked, the variables within the FrontPageGraphics object is used to initialize an instance of Maze, creating the maze with those dimensions and random key

    - Within Maze, BacktrackerDS is used to create the maze, which extends OrderedCollection and uses the Node class (similar to a linked list)
    - Maze also uses the Cell class to create a 2D representation of the maze
- After the maze is created, depending on the type of maze the player requested (variable in FrontPageGraphics) either 2DGraphicInterface or 3DGraphicInterface is used
- After the user completes the 3D maze, a new instance of the Leaderboard class is created from the main method, displaying the leaderboard GUI