Version 1

Group: Foxtrot

Maze Program Version: 1.0

Our basic idea for this version is to create the gameboard, the piece spots, and the buttons and to have them all in the correct positions.

Tiles: We are not ready to implement movable tiles, so this will be saved for future versions.

Gameboard: Gridbag layout was difficult to learn, but Buck figured out how to set the pieces in the correct position. Getting these set up and working was simple once we understood the GridBagConstraints function, especially the tile pieces. We also changed the background color to a less vibrant color.

Buttons: Getting the buttons to work was simple enough, as we coded the public class for the buttons, since they will be used throughout the project. The exit button works as intended, but the start and reset button still do nothing other than output to the console. These other buttons will be addressed in future versions.

Sidebars: These were created alongside the Game Board. They are just JPanles right now but will eventually become more and hold the pieces that we need.

Entire Project

Planning for Group Foxtrot:

Create a maze game that has tiles that need to be arranged in a path to win.

Win Condition: To create a win condition for the maze game, a checker needs to be implemented that verifies if the tiles are arranged the correct way for all possible win conditions. This will allow players to know when they have successfully completed the game.

Window Size: Setting a minimum or static window size for the game is important to provide a consistent display experience for players, so that even when resizing it will make sense to the user. This will help ensure that the game can be played on various screen sizes, and accommodate players screen size preferences..

Moving Tiles: The tiles in the game need to be designed to be movable and rotatable within the game board. We will achieve this by creating code that allows the tiles to change positions. We will have to keep track of all current positions of tiles and their rotations, which will eventually be connected with a win checker to determine if everything is in its correct position.

Tile Randomizer: A tile randomizer should be added to the game to create different paths each time it is played. We will do this by having a class that randomizes the rotation of each individual tile piece and which position in the "sidebar" it will be located. It should make sure that every new game looks different to the user.

Win Screen: The win screen should be designed to display when the player has successfully arranged the tiles in the correct path. This will provide a visual confirmation of the player's success and create a sense of accomplishment for them. It will allow them to exit the game or start a new game if they please.

UML Diagram: The UML should be reviewed and be updated as necessary to ensure it accurately reflects the design and implementation of the game for each step of the way.

Bug Fixes: Regular bug fixes should be performed to maintain the stability and functionality of the game. This will ensure that players have a smooth experience while playing the game and that the code looks nice, readable, and maintains Object Oriented principles.

Version 2
Group: Foxtrot
Maze Program Version: 2.0

The idea for this version was to add tile pieces that could be moved from the side bars to the board.
We also fixed issues brought up from the previous programming assignment.

Tile:
   Added game tiles. Currently these are Jpanels that use a mouse listener for when they are clicked.
   These display a number and color when implemented.
Tile Functionality:
   Use one click on the left button on the mouse to select the tile, and use one click on the a playbox to place
   the tile in its slot.
Playbox:
   This is a layered JPanel that creates the starting points for the tiles. This gives us a great deal of flexibility
   since we can create borders on one layer and cover them with a slightly larger tile.
   Added grid lines to show where the tiles can be placed.
Board:
   This is also a set of layered JPanels that provides a play area for the tiles.
   Added grid lines to show where tiles can be played.
Buttons:
   Fixed the "New" button so it now displays "New Game" and calls for an action of the same name.
   Removed the Println commands.
All Documents:
   Removed tabs and checked for line length.
   Edited documentation
UML Diagram:
   Fixed the way interfaces were displayed.
   Added new classes and methods to the diagram.

Rest of the Project:
Tile:
   Create a rotation method, add an outline or shadow to the tiles to better indicate which one is
   selected, create a design on the tiles for the maze game.
   For rotation, we will likely have to draw something on the Jpanel and move the components
   according to the rotation value.
Win Conditions:
   Still need to determine what win conditions will be and how they will be checked. This will deal with
   both tile placement and rotation. So a method will have to check whenever the board is full and another
   whether the tiles are in a position to win or not.The win rotations and placements could be stored in a
   data structure or a text document. We will have to discuss this more in-depth later.
   We should also add a pop up or animation that lets the user know they've won and asks them if they
   Would like to start a new game, restart, or quit.
Buttons:
   Need to make sure New Game and Reset work. There will likely be methods to clear the board and fill

each tile holder with a random tile. Something needs to hold the default positions and rotations for tiles when a game starts. This may be as simple as keeping the values in an array and moving the pieces back to the correct positions when reset is pressed.

Tile Randomizer:

This will be used when New Game is pressed. This will assign each tile a random spot when the game starts.

Game Window:

We still need to look into making the window scaleable or have a size adjustment option. This will have to be checked on multiple monitors.