Escape Final Design
Mario Castro

Alpha Release:

My major design decisions used in the alpha release are as follows -

- Three different classes that implement Coordinate for each coordinate system.
- Three different classes that implement Board for each of the boards with their appropropriate coordinate systems.
- I created a functional interface BoardFactoryMethod and used a factory pattern so the board builder would know what type of board to create and how to create it from the information given in a BoardInitializer.
- I created an additional interface implemented in each of the boards that allow them to create coordinates of their given types.This interface also allows the setting of a location type and getting a location at a certain coordinate and the removal of a piece from a board. These methods serve to delegate these tasks to the board and kept the information of a board's actual implementation independent from anything that might use a Board
- I also created an additional interface for all the coordinates called XYCoordinate. This interface was to allow for a getX and getY method of a given coordinate.

Beta Release:

My major design decisions used in the beta release are as follows -

- I created an implementation of EscapeGameManager that is created via the escape game manager builder.
- I decided to use the existing BoardBuilder from the Alpha release and then adapt it for use by the EscapeGameBuilder. This way I would not have to rewrite and retest the working code that creates boards for use by the game.
- I chose to store each piece's attributes inside of the piece itself and added the necessary fields and methods inside of the Piece class to support that. I did this because I thought each piece ought to know about its own attributes.

- The BoardBuilder class that originally created the pieces and placed them was now responsible for properly constructing the pieces with their attributes when the board is initialized. I chose to do this here because it was already dealing with the creation of a piece, so it should also deal with injecting each pieces attributes
- To actually handle the usage of these attributes and their role in validating moves, I created a MoveValidator class to delegate the validation of moves away from the game manager.
- The MoveValidator class has a single purpose of checking that a move is valid, given a location to move from, a location to move to, the piece to move, and the board on which to move the pieces. This is done in a series of phases.
    - First I check if the moves pass universal rules such as that the piece isn't trying to move to a block, that there is a piece to move, etc
    - Then I check if a path exists for the piece given it's attributes to the desired location. This is done through DFS.
        - To know where a piece can pass through, I gave every Coordinate a function that returns a list of neighbors. This list of neighbors is then filtered using java streams inside of the MoveValidator down to only the valid neighbors and these neighbors are used for edges in pathfinding.
    - After the shortest path is found, the MoveValidator checks if the path is under or equal to each piece's max travel distance and returns the results of this check. If no path is found, the Move Validator's validate function returns false;

Gamma Release (Final):

My gamma release major design decisions are as follows -

- I replaced the MoveValidator class with an interface that would allow different validators to be injected into the GameManager depending on what was needed. This seemed like a good decision if any hypothetical future released should

require different forms of MoveValidation for different games. The validator used is injected via the constructor inside of the EscapeGameBuilder

- To validate the gamma release rules, I created an interface called EscapeRuleValidator. This interface would allow different methods of interpretation of the rules. The EscapeGameManager implementation has an object of EscapeRuleValidator and this object is injected in the EscapeGameBuilder. The EscapeRuleValidator is created via a static factory method that takes the set of rules from the game initializer.
- The EscapeRuleValidator keeps track of what player is allowed to make a move, the amount of turns, and the score of each player.