Design Document for Assignment 3

Mengxing Wang
Jerin Joseph

Class Creation:

In the previous assignment we created two abstract classes called Game and Board. These two classes would be inherited by all the games. The class inheriting the Game class would focus on running the game and performing functions like enter player names, choose different moves and quit the game. The class inheriting the Board class would focus on the logic of the board and is different based on the requirements. The common logic for the board class is to print the board and to check if the board is solved (i.e. if the game is completed).

In this assignment, we create two more common classes called Tile and Piece which would be inherited by classes for all the games. The piece class stores the location of players and logic of its movement whereas the tile class stores different pieces of a board depending on the game.

Quoridor Game:

For the quoridor game, we created a Quoridor class which is inheriting the Game class. This class has the logic to add players' names and select a move. Based on the move, it performs the functions in the QuoridorBoard class. The QuoridorBoard class is inheriting the Board class. This class has logic to print the board in the terminal, to update the board with player and wall pieces and also has logic for block prevention and illegal player movement prevention.

The quoridor PawnPiece class is used to distinguish the player piece in the board. It is represented by symbols A and B for player 1 or player 2. The WallTile class is used to store the edges which have a wall.

To prevent blockage, we added additional rules to the game like horizontal walls and vertical walls would not coincide with each other. This rule made the implementation of the code to verify if a path is blocked easier. To relate the walls with the cells of the board, we mapped the walls with the two adjacent cells which have a common edge. The wall would be between the two cells. This helped in analysing if a wall can be placed or not or if a player moving from one cell to the other, can move or not depending if there is a wall in the common edge.

Scalability of Project:

All the current games have common elements like a board, players, score board and a timer. There are separate common classes which contain common logic which can be reused for future games instead of creating new classes. Eg: When we created the new game quoridor, we utilized the existing score and timer class to implement the scoreboard logic.

Since all the games created require a board to be played, we created a board class which has common methods like displayBoard() or isSolved() which can be implemented in any of the games as these functionalities are common for all the games. Other features which are unique to a class are written separately in the board class of the game.

Role breakdown:

Mengxing: Implemented the logic for the movement of the players and added the foundational code for running the game like creation of the map, able to place walls and added win condition to game
Jerin: Tested the working of the game, updated the logic for wall placement, player movement blocking logic and close path prevention logic. Updated the game to include colors for the wall and created the UML diagram, README and design documentation.

Github link: https://github.com/jerinjoseph121/oop_lab_assignment_3
Note: The creation of this repository was done later, so a lot of work in the code would not be shown in the commits.