COMP1110
Assignment 2
IQ Fit Design Outline
Group: wed11d

Group Members

Mingxuan Wang (u7202260)

Di Mou (u6553212)

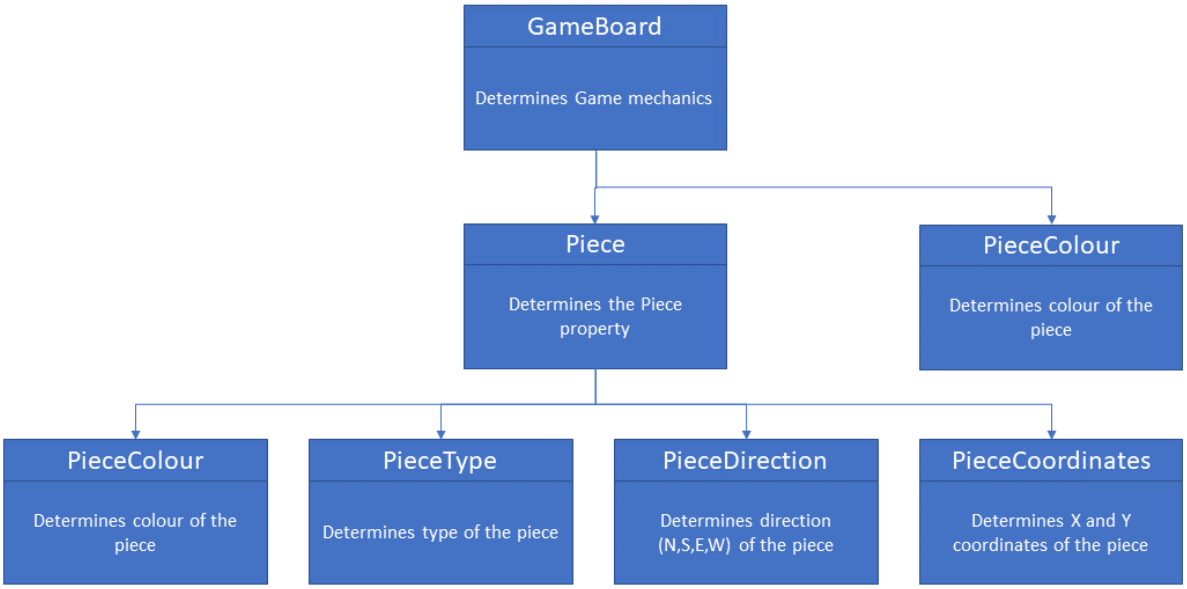
Jiwon Sin (u5666033)

Introduction

This outline is based on two meeting sessions (21.08.22 Friday and 23.08.20 Sunday).

During the two sessions, we discussed on how to approach this assignment and what are the needed Classes and methods for each class.

Below is a graphical representation of what we aim to do to make this assignment work.



Classes and Methods

1. `PieceColour`
    a. This class represents different colours of the pieces such as blue and yellow.
    b. Initially we thought there were 10 colours but there was a need to represent empty space for the initial board. Hence we added `noPiece` as one of the elements.
    c. `PieceColour` will be used by `PieceType` to create different shapes.
    d. List of methods implemented
        i. None now, since this class is merely an `enum` class with piece colours.
2. `PieceCoordinates`
    a. Upon looking at assignment speficications, it was evident that coordinates are needed.
    b. `PieceCoordinates` represents the piece's position on the board itself.
    c. List of methods implemented:
        i. `getXCoord()` : returns the `int` value of X coordinate of a piece.
        ii. `getYCoord()` : returns the `int` value of Y coordinate of a piece.
3. `PieceType`
    a. This class defines the specific properties of each piece's number of protrusions, colour and spine length.
    b. Initially, we wanted to put everything into Piece class only, but it seemed like it could get messy and complicated

c. Instead, `Piece` class will define the main properties of each Piece.
    i. For example, "r00N" has three main types, which are (1) Piece's type, (2) X and Y coordinates and (3) the direction the piece is facing.
    ii. Hence it would be easier to make a separate class that defines Piece's type, which comprises of (1) colour, (2) number of protrusions and (3) spine length, which distinguishes one piece from another.
d. List of methods implemented:
    i. `getChar(PieceType type)` : Converts PieceType to a char type.
    ii. `getProtrusion()` : converts placement to PieceType.
    iii. `fromChar(String placement)` : converts placement string to PieceType.
    iv. `getProtrusion()` : returns number of protrusion.
    v. `getColour()` : returns the colour of the piece.
    vi. `getSpineNum()` : returns the length of spine (whether 3 or 4).

4. `PieceDirection`

a. This `enum` class defines the direction in which the piece is facing.
b. Its members being `NORTH, EAST, SOUTH, WEST`.
c. `PieceDirection` class will be incorporated into Piece, when defining the unique Piece property.
d. List of methods implemented:
    i. `getChar()` : returns the char value that is equivalent to first letter of direction.
    ii. `getDirection()` : returns the Direction (NORTH, EAST, SOUTH, WEST).
    iii. `rotate()` : changes the symbol of the direction (needs more clarification of its implementation with group members).

5. `Piece`

a. The main class that creates `Piece` objects.
b. Each `Piece` will have 3 components as mentioned from above.
c. List of methods that needs to be implemented.
    i. `toPiece(String placement)` : Converts the string component, whether length == 4 or more to Piece.
    ii. `getXCoordinate(String placement)` : returns the value of X coordinate of this piece.
    iii. `getYCoordinate(String placement)` : returns the value of Y coordinate of this piece.
    iv. `updatesPiece()` : When the piece is placed on the board, left-topmost coordinate is updated to the piece (needs more thought into this).
    v. `toPlacement()` : converts the Piece into String placement,
    vi. `changePiece()` : (This method is optional) Change from one Piece to another.
    vii. `getSpaceOccupied()` : Checks the coordinates occupied by the piece. Every time a piece is placed, there will be empty spaces. This method updates the board on whether placement of specific piece occupies which coordinates.
d. Personal thoughts
    i. Would it be better by just making Piece class as an abstract class and make 10 different subclasses?

6. **`DifficultyLevel`** – optional, not discussed fully yet-
    a. This class is to set the difficulty level of objectives.
    b. After much research on web, the game objectives were divided into 5 different levels.
    c. Starter -> Junior -> Expert -> Master -> Wizard.
    d. Still debating whether to separate this into independent class or just add as a method within the `GameBoard` class.
    e. If it exists in independent class, it will divide the Game objectives into five different levels and player could choose the difficulty level.
    f. Also considering adding a method that gives the random number within that difficulty level.

7. `GameBoard`

- a. The class that will have methods that determines game mechanics.
- b. Methods that could be implemented:
    - i. `initialBoard()` : displays initial empty board, by using multidimensional arrays.
    - ii. `importBoard()` : Considering the Game objecetives comes with per-placed pieces, change the board state to suit the needs.
    - iii. `updateBoard()` : updates the board status whenever called. Its main purpose is to update the board when the player placed the piece or removed it from the board. If the `initialBoard's` array type is in `PieceType`, then chages from nP to whatever colour of the piece is. IF `initialBoard's` array type is in int, it will be in terms of coordinates (from (0,0) to (9,4)).
    - iv. `changeState()` : changes the board's state from `nP(noPiece)` to colour or vice versa.
    - v. `clearBoard()` : clears entire board to `intialBoard` or `importBoard` state.
    - vi. `checkOverlap()` : this method checks if an array of coordinates overlaps another piece's coordiantes.

8. Designs about GUI
    a. Ideally, we would like to add buttons the user can press to rotate the piece.
        i. Rotate right/ rotate left
    b. If difficulty level is implemented, it would be a good idea to include a horizontal slider that changes the difficulty of the board.
    c. Placement count could be implemented to challenge the user to clear the level by minimising the placements counts.