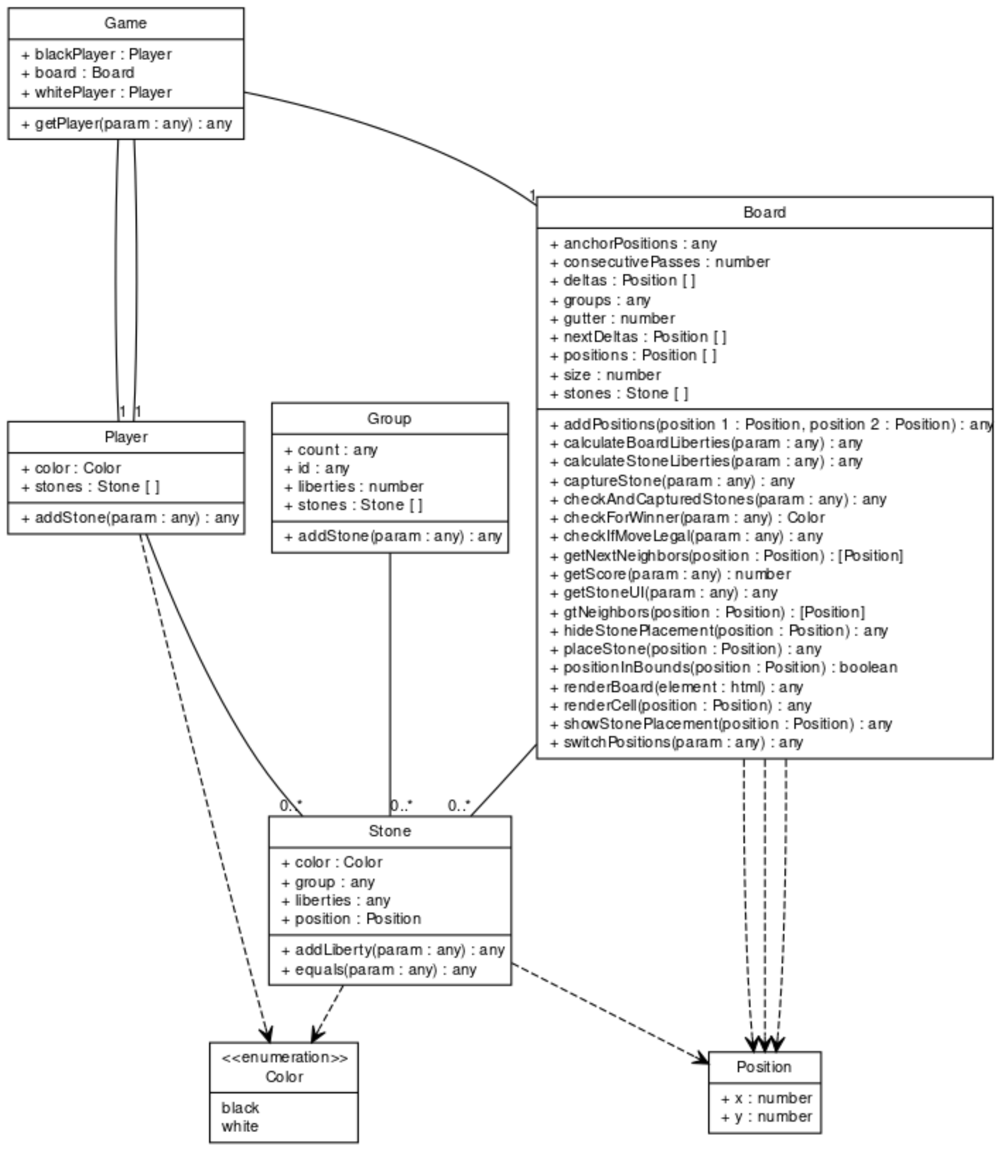
Summary

- The project is called GO or Weiqi.
- The project lets two players play the board game GO.
- The project requires 2 players to play GO. It can only be played offline and on a browser. It does not require internet connection and does not collect any information from the players.
- The game should have two main functions. First one should check if the stone placed is legal. If the move is legal, it should check if it captured any enemy stone. Second one should calculate the captured territory of both players and determine the winner of the game at the end game phase. If there is enough time, I am planning to add an AI player feature using reinforcement learning.
- The project is assigned to be done in 2 weeks. Each week's progress will be updated in this report.

Week 1:

- Stone:
  - Added property *liberties* that is an array of **Position** objects
  - Added property *group* that stores a reference to the **Group** object it belongs to
  - Added *equals* method that checks if two Stone objects are equal.
  - Added *addLiberty* method to add a liberty to a stone
- Player:
  - *placeStone* method now returns the first stone that is removed from the set
  - *addStone* method adds a **Stone** object to the set
- Group:
  - Added a static *count* property
  - Added *id* property
  - *stones* is now an array
  - *liberties* is now an array
  - Removed *calculateLiberties* method
- Game:
  - Added *getPlayer* method to return a Player object that pertains to a certain color
- Board:
  - Added a *groups* property that contains **Group** objects
  - Added *anchorPositions* property
  - Added *switchTurns* method that switches the *turn* property to the other **player** after a player makes their move
  - *calculateStoneLiberties, calculateBoardLiberties, checkAndCapturedStones, captureStone, checkIfMoveLegal, getStoneUI* methods added

Class Diagram:


Game
+ blackPlayer : Player
+ board : Board
+ whitePlayer : Player
+ getPlayer(param : any) : any
1
1 1
Board
+ anchorPositions : any
+ consecutivePasses : number
+ deltas : Position [ ]
+ groups : any
+ gutter : number
+ nextDeltas : Position [ ]
+ positions : Position [ ]
+ size : number
+ stones : Stone [ ]
+ addPositions(position 1 : Position, position 2 : Position) : any
+ calculateBoardLiberties(param : any) : any
+ calculateStoneLiberties(param : any) : any
+ captureStone(param : any) : any
+ checkAndCapturedStones(param : any) : any
+ checkForWinner(param : any) : Color
+ checkIfMoveLegal(param : any) : any
+ getNextNeighbors(position : Position) : [Position]
+ getScore(param : any) : number
+ getStoneUI(param : any) : any
+ gtNeighbors(position : Position) : [Position]
+ hideStonePlacement(position : Position) : any
+ placeStone(position : Position) : any
+ positionInBounds(position : Position) : boolean
+ renderBoard(element : html) : any
+ renderCell(position : Position) : any
+ showStonePlacement(position : Position) : any
+ switchPositions(param : any) : any
Player
+ color : Color
+ stones : Stone [ ]
+ addStone(param : any) : any
Group
+ count : any
+ id : any
+ liberties : number
+ stones : Stone [ ]
+ addStone(param : any) : any
0..*
0..*
0..*
Stone
+ color : Color
+ group : any
+ liberties : any
+ position : Position
+ addLiberty(param : any) : any
+ equals(param : any) : any
<<enumeration>>
Color
black
white
Position
+ x : number
+ y : number

Note: this is just for better visualization of the project and allow myself to keep track of the project

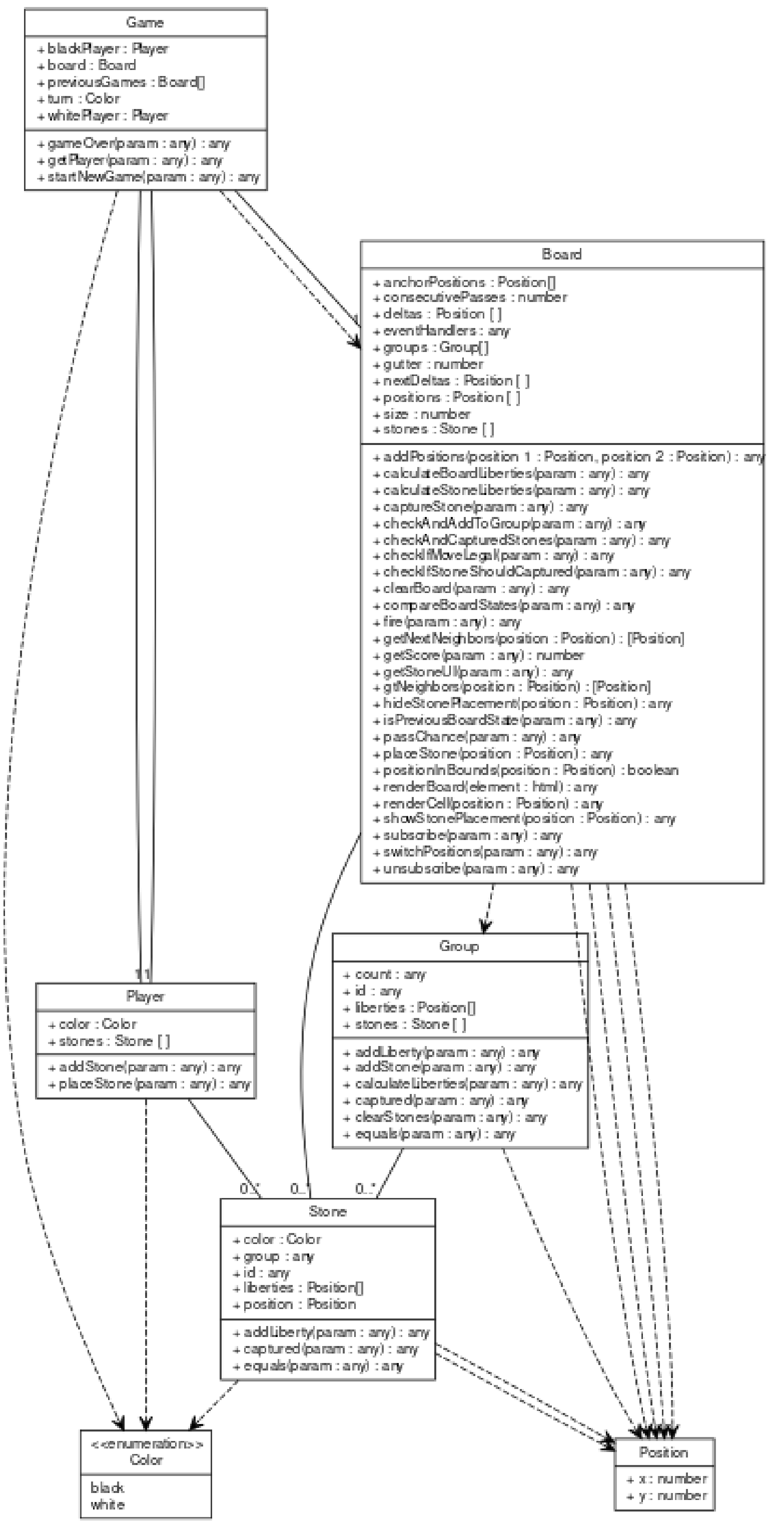
Plan for next iteration:

1. Integrate Groups into the game
2. Improve liberty calculation algorithm
3. Improve illegal move calculation algorithm
4. Scoring algorithm

Week 2:

- Observer pattern implemented
- Game over mechanism added
- Logic to determine positional superko added
- Logic to determine whether a suicide move is legal
- Grouping mechanism improved
- Game:
    - "turn": "property",
    - "previousGames": "property",
    - "getPlayer": "method",
    - "startNewGame": "method",
    - "gameOver": "method"
- Player
    - "addStone": "method",
    - "placeStone": "method"
- Stone
    - "captured":"method"
- Group
    - "calculateLiberties": "method",
    - "captured": "method",
    - "equals": "method",
    - "clearStones": "method"
- Board
    - "eventHandlers": "property",
    - "subscribe": "method",
    - "unsubscribe": "method",
    - "fire": "method",
    - "isPreviousBoardState": "method",
    - "compareBoardStates": "method",
    - "checkIfStoneShouldCaptured": "method",
    - "checkAndAddToGroup": "method",
    - "passChance": "method",
    - "clearBoard": "method"

Final class diagram:


Game
+ blackPlayer : Player
+ board : Board
+ previousGames : Board[]
+ turn : Color
+ whitePlayer : Player
+ gameOver(param : any) : any
+ getPlayer(param : any) : any
+ startNewGame(param : any) : any
Board
+ anchorPositions : Position[]
+ consecutivePasses : number
+ deltas : Position [ ]
+ eventHandlers : any
+ groups : Group[]
+ gutter : number
+ nextDeltas : Position [ ]
+ positions : Position [ ]
+ size : number
+ stones : Stone [ ]
+ addPositions(position 1 : Position, position 2 : Position) : any
+ calculateBoardLiberties(param : any) : any
+ calculateStoneLiberties(param : any) : any
+ captureStone(param : any) : any
+ checkAndAddToGroup(param : any) : any
+ checkAndCapturedStones(param : any) : any
+ checkIfMoveLegal(param : any) : any
+ checkIfStoneShouldCaptured(param : any) : any
+ clearBoard(param : any) : any
+ compareBoardStates(param : any) : any
+ fire(param : any) : any
+ getNextNeighbors(position : Position) : [Position]
+ getScore(param : any) : number
+ getStoneUI(param : any) : any
+ gtNeighbors(position : Position) : [Position]
+ hideStonePlacement(position : Position) : any
+ isPreviousBoardState(param : any) : any
+ passChance(param : any) : any
+ placeStone(position : Position) : any
+ positionInBounds(position : Position) : boolean
+ renderBoard(element : html) : any
+ renderCell(position : Position) : any
+ showStonePlacement(position : Position) : any
+ subscribe(param : any) : any
+ switchPositions(param : any) : any
+ unsubscribe(param : any) : any
1
1 1
Player
+ color : Color
+ stones : Stone [ ]
+ addStone(param : any) : any
+ placeStone(param : any) : any
Group
+ count : any
+ id : any
+ liberties : Position[]
+ stones : Stone [ ]
+ addLiberty(param : any) : any
+ addStone(param : any) : any
+ calculateLiberties(param : any) : any
+ captured(param : any) : any
+ clearStones(param : any) : any
+ equals(param : any) : any
0..*
0..1
0..*
Stone
+ color : Color
+ group : any
+ id : any
+ liberties : Position[]
+ position : Position
+ addLiberty(param : any) : any
+ captured(param : any) : any
+ equals(param : any) : any
<<enumeration>>
Color
black
white
1
Position
+ x : number
+ y : number

Third party code:

Vite was used as a build tool.
Lodash was used for Array/Object utilities.
Recycled and improved Observer pattern code from this article.
Inspiration for some of the **Board** class methods was drawn from Weiqi

Result & Conclusion:

The project is taking more effort than I initially planned. Weiqi is a complicated game, and there are some special rules (ex: superko) that are challenging and time consuming to implement. As a result, I implemented all the rules for the weiqi. However, I didn't implement the scoring feature and the AI player feature as I initially planned due to time constraints. Overall, I will say I am enjoying this project a lot. It is definitely a fascinating project that I am doing what I like.

Links:
https://github.com/nihs9958/CSPB3202