| Board | |
|---|---|
| • Display the grid of the board on the screen<br>• Display the numbers and letters for each intersection on the screen<br>• Store each of the pieces on the screen<br>• Check if each move is legal before accepting a move by using the rules class<br>• Keep track of who the current player is so that the AI can query the board<br>• Return a game state of the board after a certain move is applied<br>• Store all of the legal possible moves by using the rules class<br>• Check if a player has won the game in a given board state | • Piece<br>• Go Rules<br>• Monte Carlo Class<br>• Game class |

| Piece | |
|---|---|
| • Store the position of the piece on the board<br>• Store the colour of the piece<br>• Display the representation of the piece on the screen | • Colour<br>• Board |

| Colour | |
|---|---|
| • Store the colour of a piece on the screen<br>• Store as an empy colour if the piece has not been placed yet | • Piece |

| player_turn | |
|---|---|
| • Store who's turn it is supposed to be | • Main file that runs the game loop |

| Go Rules | |
|---|---|
| • Return if a move is legal<br>• Go through each rule and check that a move complies by it<br>• Be able to find all of the places on the board where a legal move can be played so that the ai is able to make a move | • Piece<br>• Board |

| Monte Carlo Tree | |
|---|---|
| • Calculate the best move from the current position<br>• Play out random games from the current position to try and find the best move<br>• Store all of the previous game states to be able to remember what the best move was | • Board |

| Game | |
|---|---|
| • Run the game loop<br>• Check for inputs from the user<br>• Display the UI<br>• Make sure that the correct screen is being displayed, eg. Main menu, game and game over<br>• Render the Game board<br>• Alter who's turn it is after each move | • Board<br>• Piece<br>• Main function |