**Player(Heuristic H):**Object representing a player. This object is capable of making moves depending on the board position based on its own heuristic.

Attributes:

Heuristic H: ????? what is this? function? object?

Tree T: Tree with all the possible moves in the game

Node lastMove: Stores this player's last move (reference to a node in tree T).

Methods:

*Player.Play*(list Node.Board) returns list Node.Board

*Player.ApplyHeuristic*(Heuristic H, Tree T) returns Tree

*Player.createChildren*(Node N): returns Tree

*Player.checkWin*(list Node.Board) returns Boolean

*Player.checkDraw*(list Node.Board) returns Boolean

*Player.checkLose*(list Node.Board) returns Boolean

*Player.move*(int Node.Depth) returns Boolean

*Player.printBoard*(list Node.Board)

*Player.Play*(list Node.Board) returns list Node.Board  
Checks among the children of Player.lastMove for the one that matches the value of Node.Board (opponents move) and moves to that node. Then decides which move to make based on the values of the values of the heuristic. Updates the value of lastNode with the chosen node and returns lastNode.Board.

*Player.createChildren*(node n): returns Tree  
Creates a tree with all possible moves from the current root node.

*Player.applyHeuristic*(Heuristic h, Tree T): returns Tree  
Goes through the whole tree, calculates and applies the value of the heuristic to each node. It will make use of the functions checkWin(), checkDraw(), checkLose() to find the values of the leafs. Returns the new tree with updated values at each node.

*Player.checkWin*(Node.Board) returns Boolean  
Checks the board to see if the current player won.

*Player.checkDraw*(Node.Board) returns Boolean  
Checks the board to see if the game is a draw.

*Player.checkLose*(Node.Board) returns Boolean  
Checks the board to see if the current player lost.

*Player.move*(int Node.Depth) returns Boolean  
Given a node depth, it checks if it is the current players turn.

*Player.printBoard*(list Node.Board)  
Prints on the console the given board.

**Node():**Object representing a node on a tree or graph.

Attributes:

int depth: Depth on the tree at which this node is located

list board: List of the positions on the board (current state of the game)

list children: List of children of this node

node parent: Reference to parent node

int value: Heuristic value for this node

list action: Action which applied to the parent returns this node. [int position, int simbol]

int gameResults: Integer storing Win, Draw, Lose, None to check if it is a leaf and then the result of the game.

*automaticGame*(Player P1, Player P2): returns integer = {1,0,-1}

Executes an automatic game between Player P1 and Player P2. It assumes P1 makes the first move. Returns and integer, 1 if P1 won, 0 if it is a draw, -1 if P2 won.

It can either check each board game as it is passed between the players to see if a someone won, or wait until a player claims it won or it is a draw.