COM2108 Functional Programming Assignment

Design

When designing my solution for this project I used a top-down approach. As suggested in the brief I decided that designing several *Tactics* for my solution would be optimal. Those *Tactics* could then be implemented in several different varieties to form a functional *Player.*

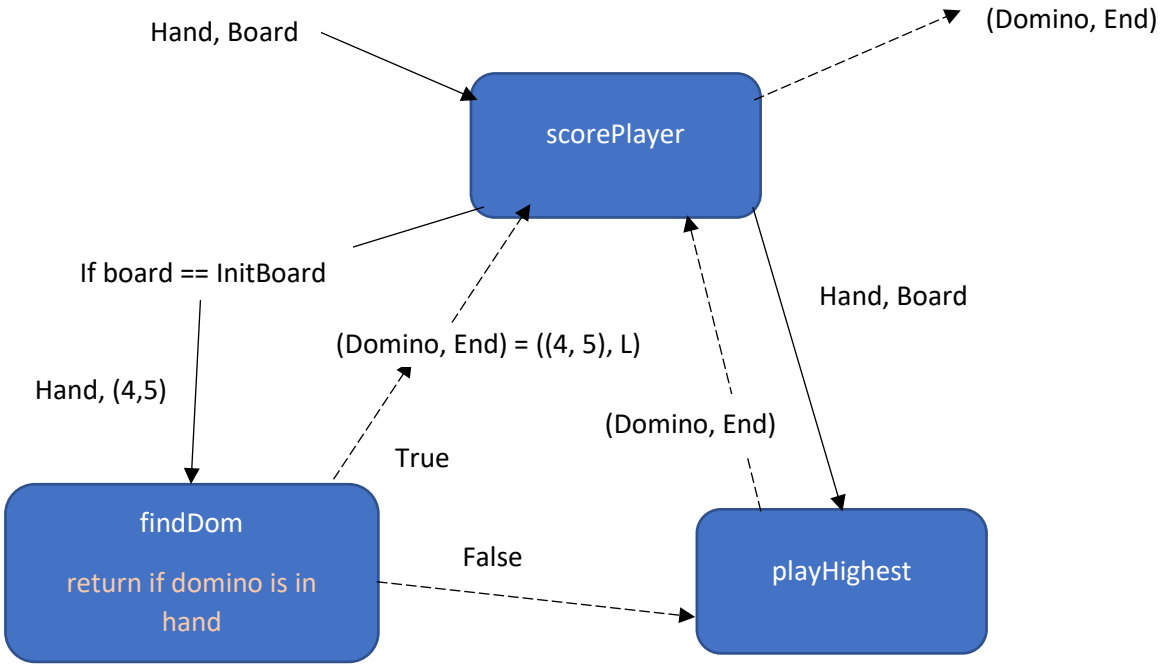
Firstly, I constructed a list of the *Tactics* I would implement:

- **firstDrop** – If the board is empty and the player has (4,5), play this as it scores high and if difficult to get a high score from.
- **playHighest** – Play the highest scoring domino in players hand.
- **playDefensive** – Play the domino in players hand that has on average lowest score out of the remaining dominoes
- **playOptimal** – Assess both the score achieved and average playable score of new board for each domino in hand and use this metric to pick the optimal play
- **playMajority** – Play the domino that will result in a board giving the highest amount of playable dominoes in hand.
- **nearWin** – Find a play that will result in the player winning the round (61 points). If this is not possible attempt to land on 59 points
- **blockWin** – Try to find a move that will block the opponent winning when they are on a high score

After deciding on these *Tactics,* I was then able to produce design diagrams for how my match players would operate.

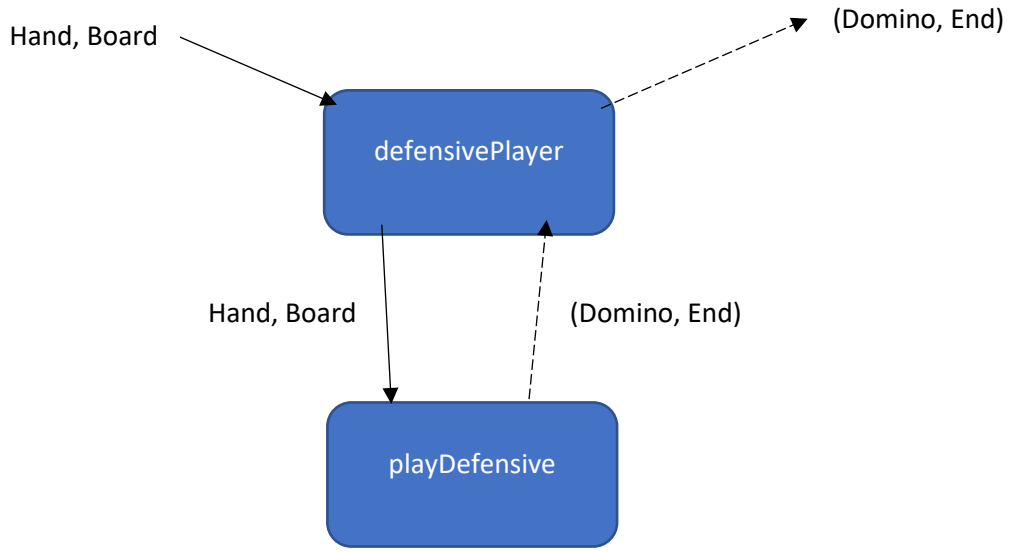
scorePlayer

- Implements the **firstDrop** and **playHighest** *Tactics*



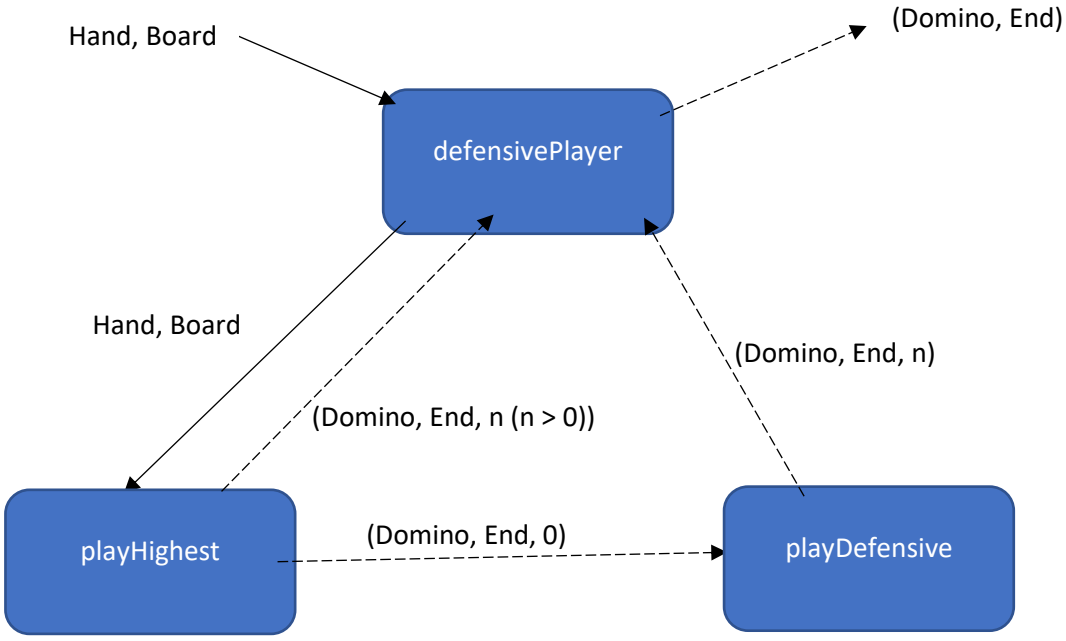
defensivePlayer

- Implements the **playDefensive** *Tactic*



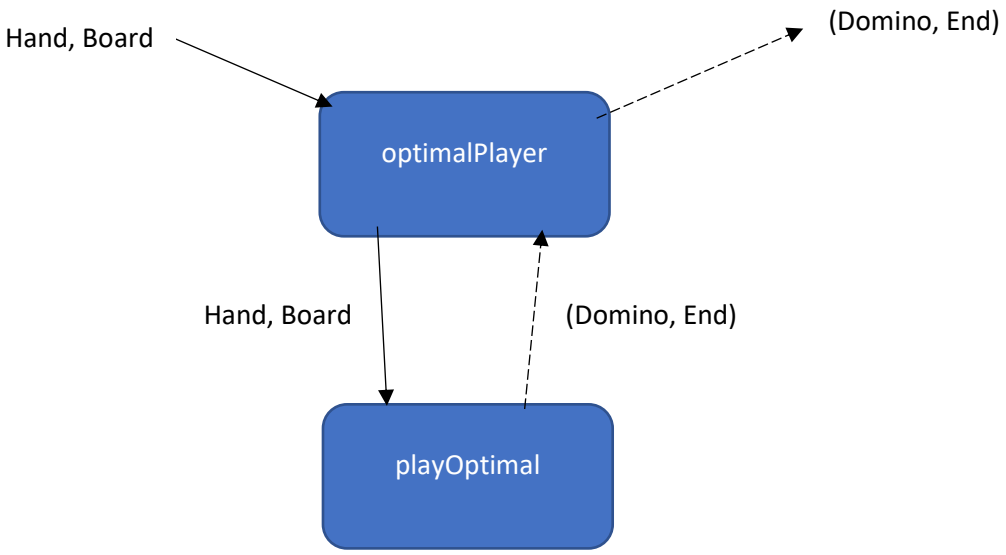
tacticalPlayer

- Uses the **playHighest** *Tactic*, unless it returns a move gaining 0 points then use **playDefensive.**



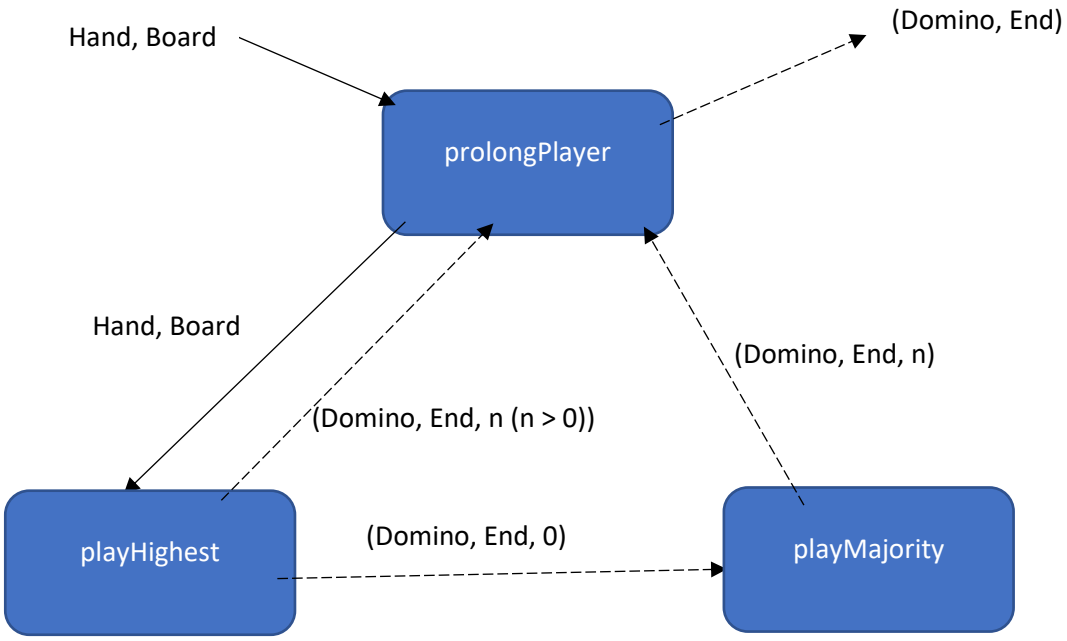
optimalPlayer

- Implements **playOptimal** *Tactic*



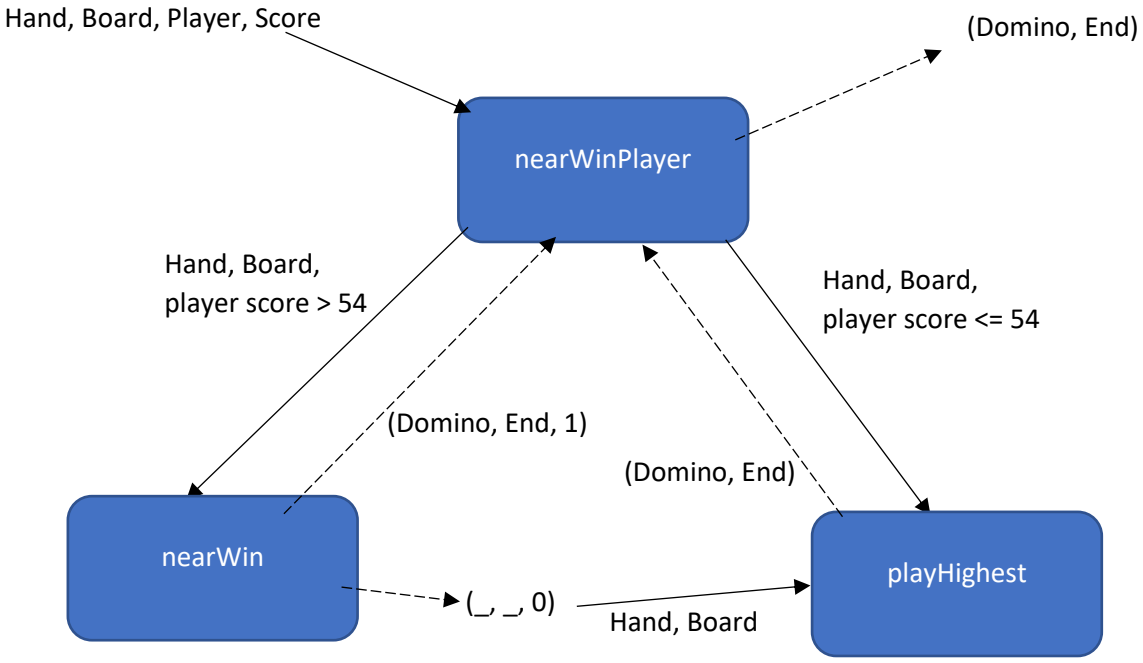
prolongPlayer

- Uses the **playHighest** *Tactic*, unless it returns a move gaining 0 points then use **playMajority.**



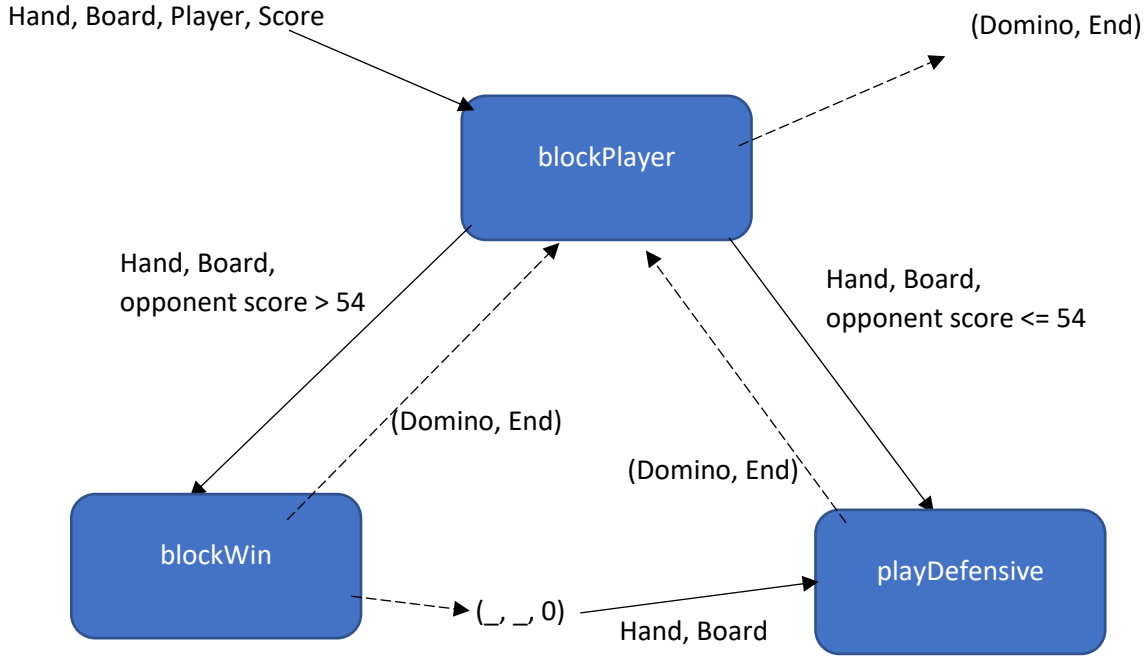
nearWinPlayer

- If player is nearing 61 points, use **nearWin** to try and find a winning play, otherwise use **playHighest.**



blockPlayer

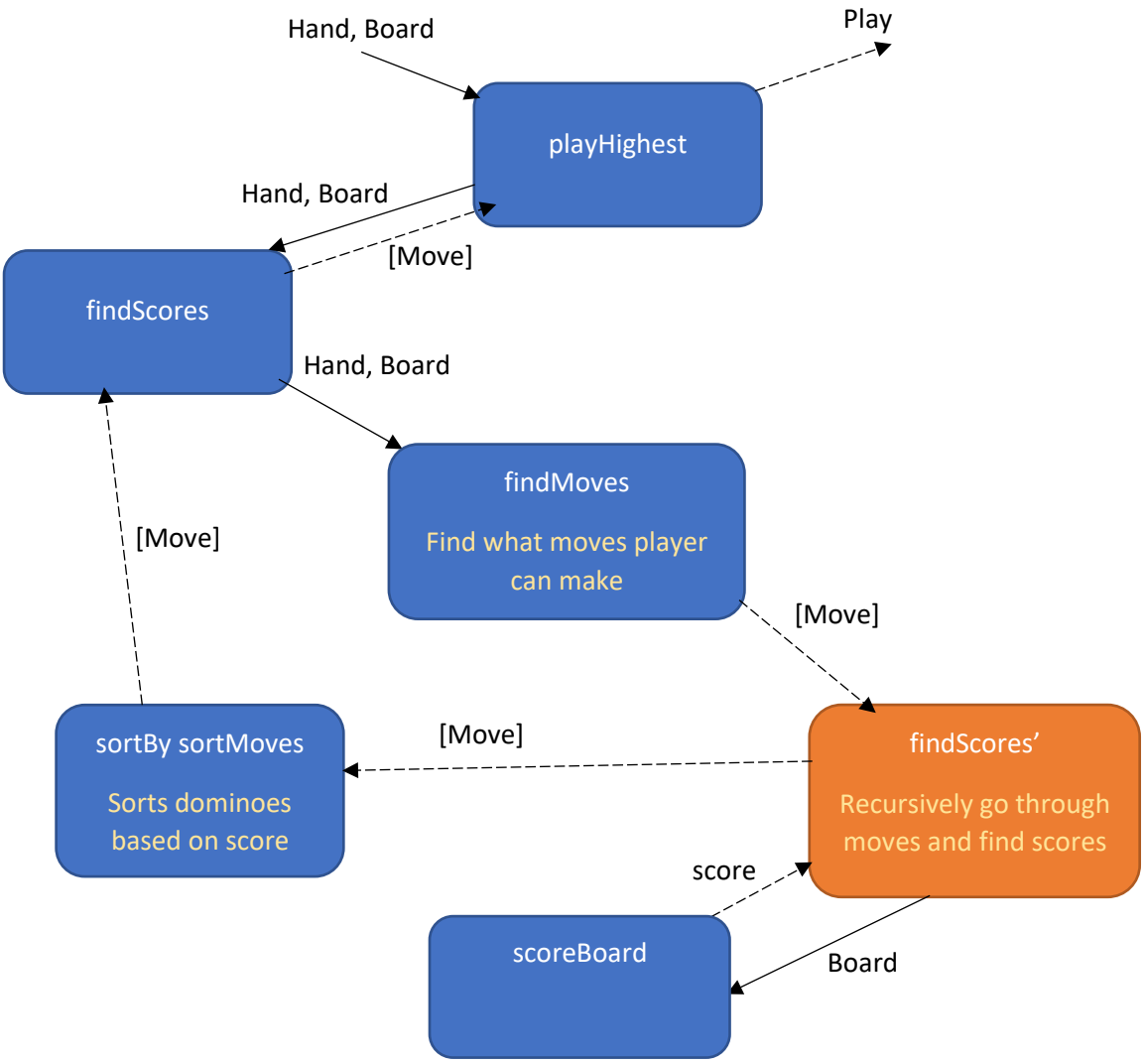
- If opponent is nearing 61 points, use **blockWin** to prevent them from winning, otherwise use **playDefensive.**



Following my top-down design structure, I began to build design diagrams on how the *Tactics* would be created.

firstDrop – this *Tactic* will be built into the player, making use of a 'findDom' function which checks if the Domino input is in the Hand input.

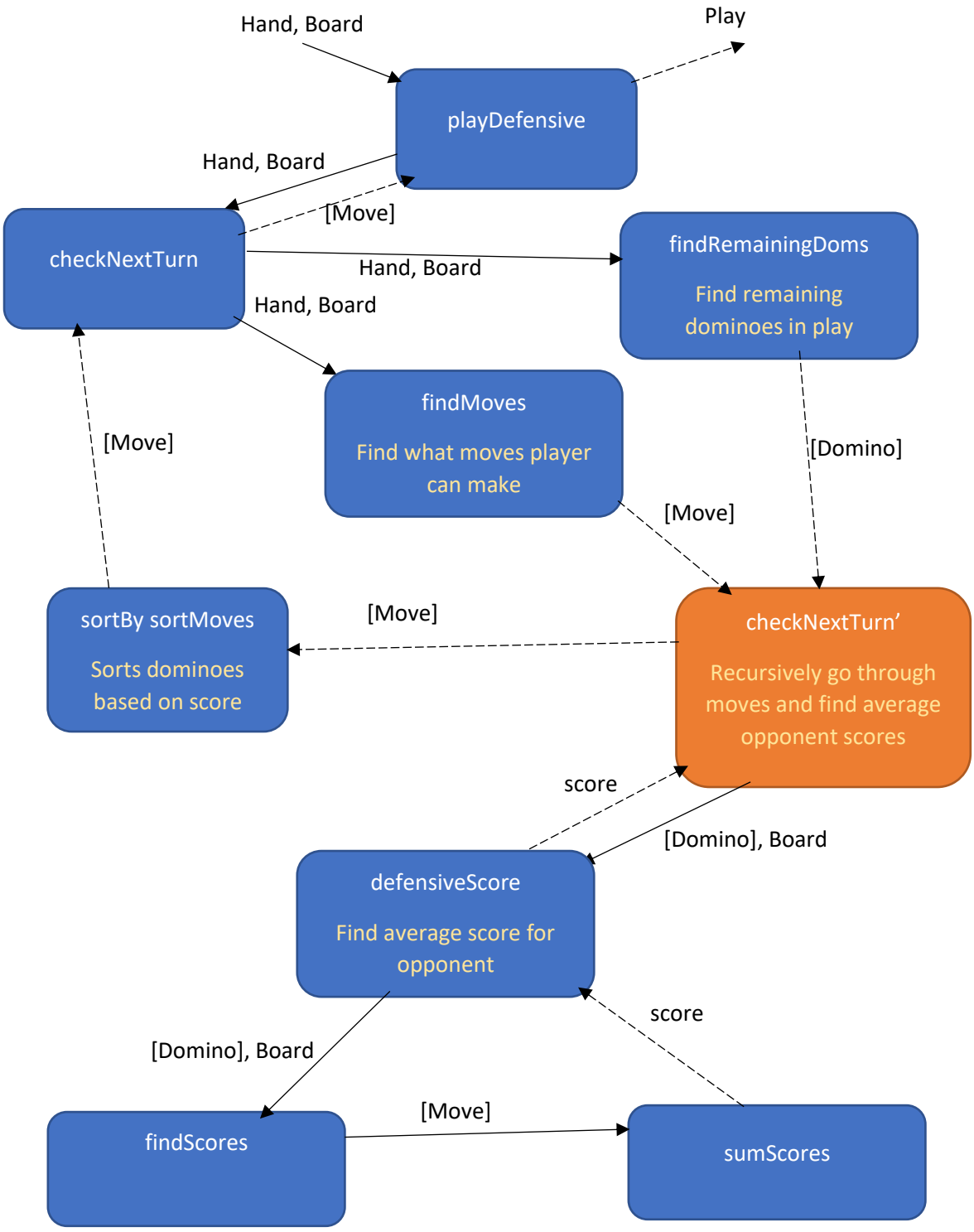
playHighest



Process:

- Find all possible moves the player can make
    - Score each board each move creates
        - Sort the moves based on score
            - Return the highest scoring move

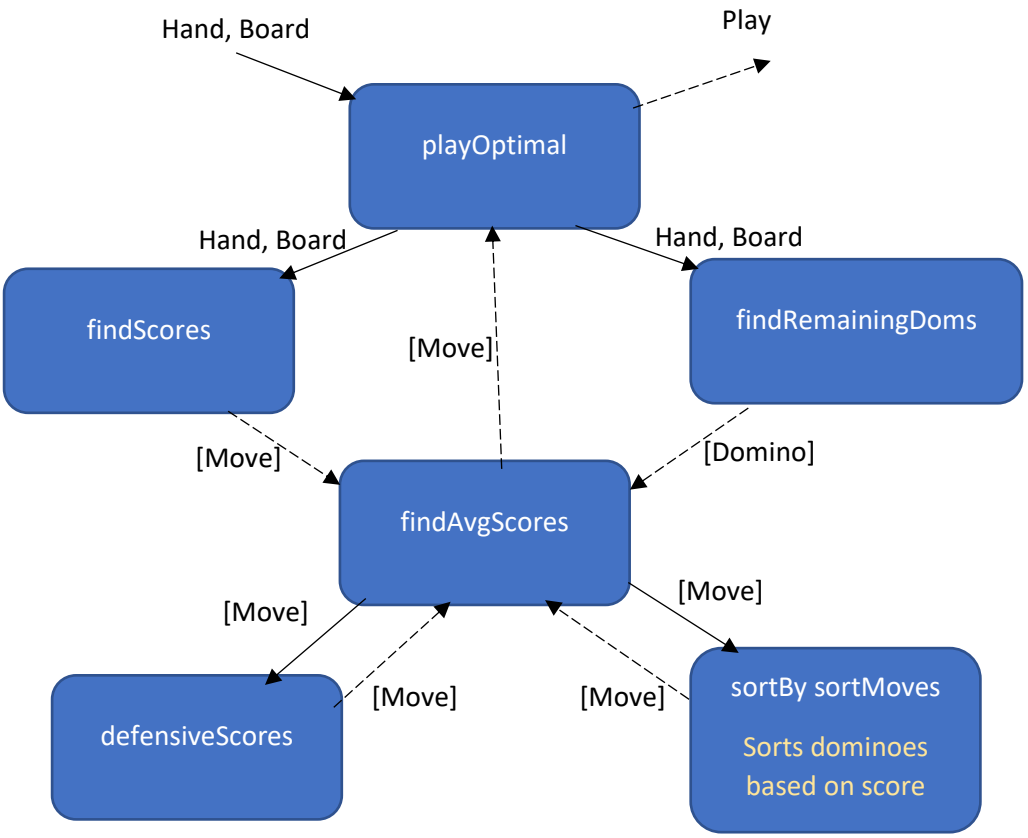
playDefensive



Process:

- Find all possible moves the player can make
- Find all remaining dominoes that opponent may have
    - Find defensive scores for each move
        - Find opponent score for each remaining domino on new board
        - Sum scores to find average
            - Sort moves by score metric
                - Return highest scoring play

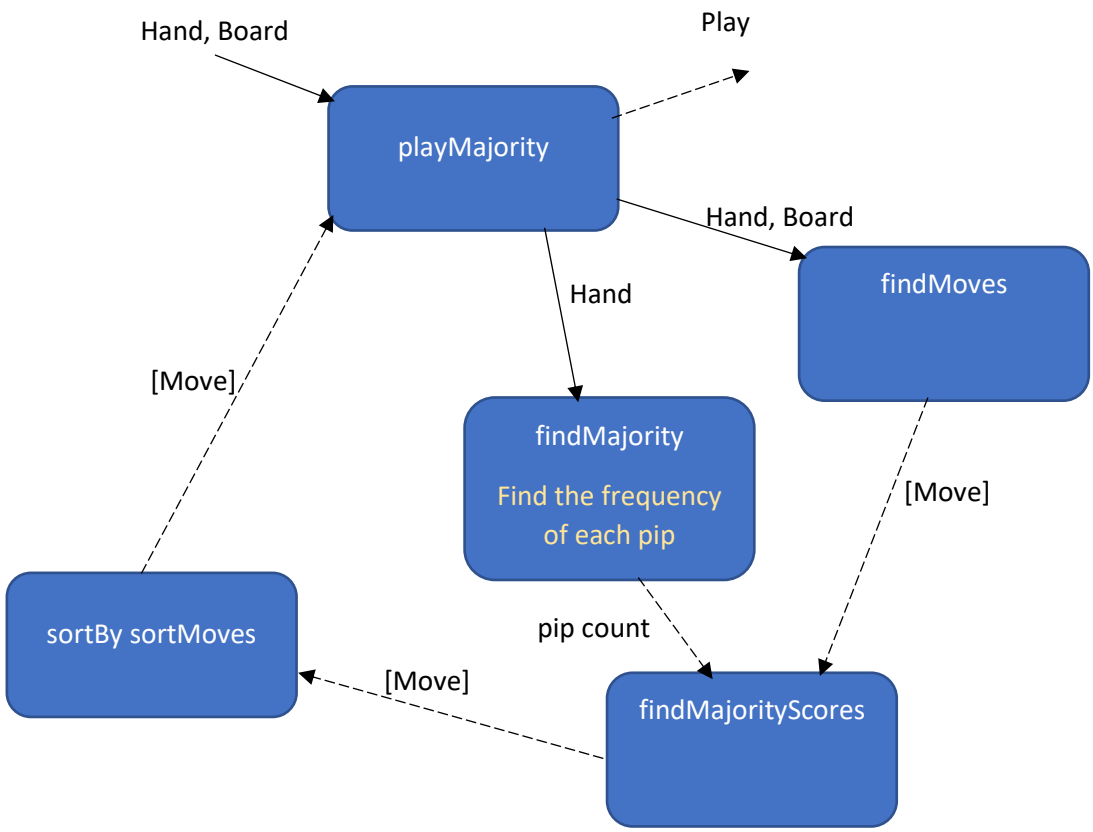
playOptimal



Process:

- Find all possible moves the player can make
- Find all remaining dominoes that opponent may have
    - Find scores for each move player can make
    - Find defensive scores for each move
        - Find opponent score for each remaining domino on new board
        - Sum scores to find average and take away from what score domino gives (times by weight)
    - Sort the moves based on score
        - Return the highest scoring move

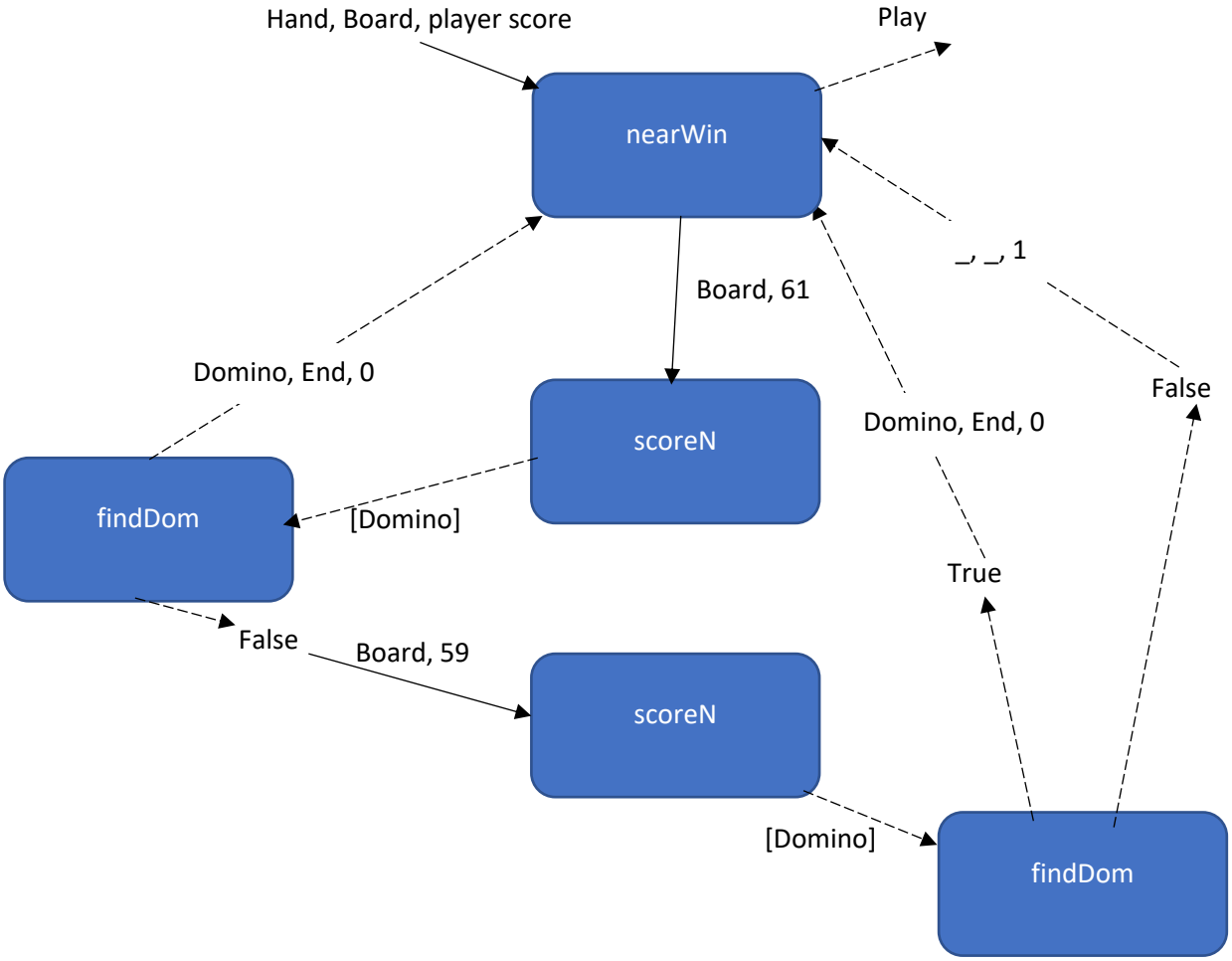
playMajority



Process:

- Find all possible moves the player can make
- Find frequency of each pip in players hand
    - Find score metric for each play based on frequency of the pips on the end of new board in hand
        - Sort plays by score metric
            - Return highest scoring play

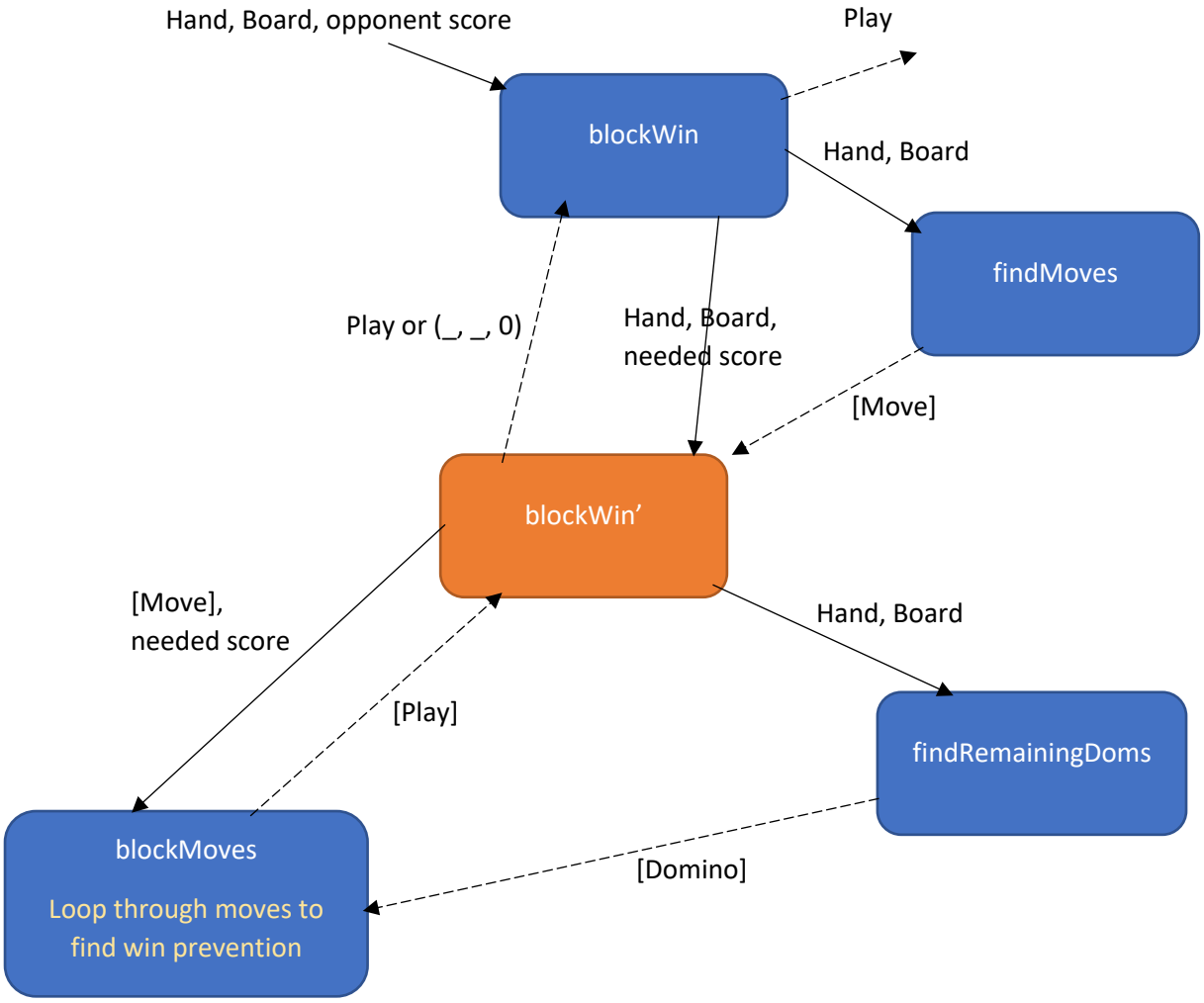
nearWin



Process:

- Find all dominoes that will gain a winning score of 61
    - Check hand for dominoes
        - Return first winning domino found
- If no winning dominoes, find all dominoes that will gain a score of 59
    - Check hand for dominoes
        - Return first domino scoring 59
- If no dominoes return play with score of 0 to indicate to player that score of 61 or 59 cannot be obtained

blockWin



Process:

- Find all moves player can make
- Find remaining dominoes opponent could have
    - Create a list of plays that will prevent the opponent from gaining a winning score of 61
        - Return the first play in list
    - Else return a play with score of 0 to indicate to player that no blocking move is possible