COM2108 Programming Assignment: Design

1. Introduction

In this document I present my design of players, which are meant to play the game of dominoes as implemented by the *DomsMatch.hs* module. First, I present the idea behind my designs and its general structure. Deeper design choices and ideas behind implementation are presented in the attached design diagrams.

As a person, who is completely new to the game of 5s-and-3s dominoes, my design of the players is hugely influenced by the types of players which you generally encounter in any real-life board game. Hence, my players fall under categories such as:

- Tries to score as high as possible, disregarding any other options;
- Tries to protect themselves from losing as much as possible, often losing the goal of winning;
- Focuses on sabotaging other players in hopes of winning.

2. General structure

Main focus of my player implementations is the use of *Tactics*. *Tactics* are functions, which pick a domino to play from the player's hand, based on different constraints and factors. *Tactics* can either be definitive or not and are mainly based on the hints provided in the brief. In the first case, they always return a *Domino* to play and an *End* to play it at. However, in the latter case they can return *Just (Domino, End)* or *Nothing*, if it was unsuccessful. Hence, *Tactics* are either defined by *Tactic* or *TacticUnsure*.

The players have their own characteristics, which are defined by the choice of *Tactics* at different points of the game.

2.1. Players

2.1.1. highestScoringPlayer

Plays the highest scoring domino available in hand. Adapted from the Stage 2 of the programming assignment warm up.

2.1.2. reachesForGoalPlayer

Plays the highest scoring domino available in hand, until it is near the ending of the game. Then, the player tries to play the domino which either wins the game or puts the player into the best position to win the game. Also, if starts first, checks if a "popular choice" domino can be played.

2.1.3. avoidsKnockingPlayer

Plays the dominoes which have the lowest risk of the player getting blocked. Does this by playing the doubles early, playing dominoes which have the highest frequency of pips left in hand and choosing dominoes, which have the highest probability of keeping the player from knocking in the next turn.

2.1.4. saboteurPlayer

Always tries to play a domino, which is the most problematic for the opponent.

2.1.5. beginnerPlayer

Tries to mimic a reasonable beginner in any game by combining the aforementioned players into one. If starts first, tries to place the best available domino. During the middle of the game, tries to play dominoes which are frequent in hand by pip value or dominoes which will not leave the player blocked. Although, if those choices do not bring a desirable score, plays the highest scoring domino. In the end of the game, the player considers the scores of themselves and the opponent. Accordingly to those scores, either tries to block the opponent, stitch the game or win it by playing dominoes for each of the occasions.

2.2. Tactics

2.2.1. playHighestScoring

Chooses a domino, which would score the most if placed.

2.2.2. playSafe

Chooses a highest scoring safe domino by traversing through the *Hand*, sorted by scores and picking the first one, which is not dangerous and has the highest probability of leaving the player unblocked for the next turn.

2.2.3. playSafeDouble

Chooses a highest scoring double from the *Hand*. The double has to be safe to play as well.

2.2.4. playFrequent

Chooses the domino whose pip value has the greatest frequency in hand thus raising the chances of the player having something to play at the next turn.

2.2.5. playExactScore

Chooses such domino, which brings the score to the winning score- 61.

2.2.6. playSecondBest

Chooses such domino, which brings the score to the best position to win- 59.

2.2.7. playFirstMove

If the board is empty, looks for the "popular first choice" domino (5,4) or (4,5) and plays it if available.

2.2.8. playOpponentKnocking

Looks for the domino, which has the highest chance of leaving the opponent blocked and thus make them knock on the next turn.

2.2.9. playStitching

Looks for a domino, which has the highest chance of leaving both players knocking and hence ending the round.

3. Design Diagrams

3.1. Tactics

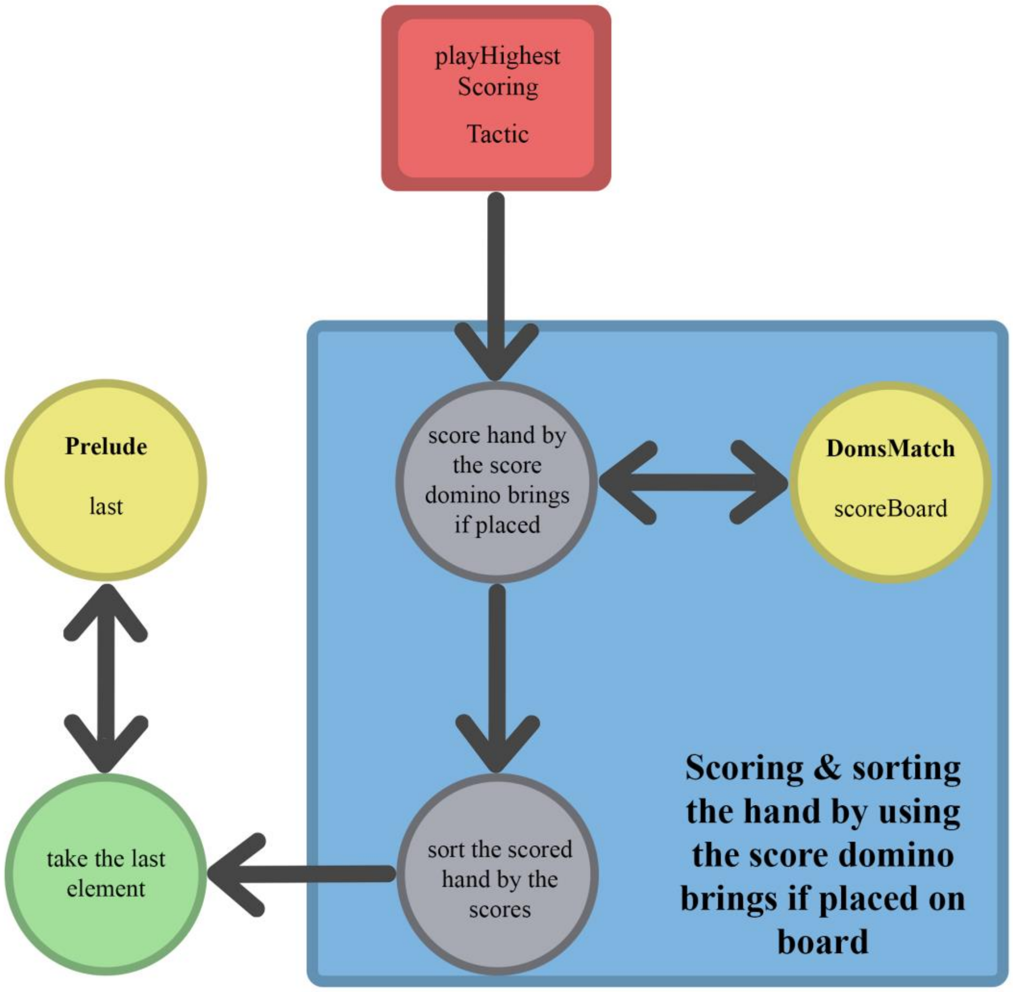
The designs are presented as flowcharts to capture the main flow and the framework of the functions. Each grey circle in the flowchart is a function that should be implemented. Green circle denotes the end of a *Tactic* and at that point a domino is returned. Although, if a *Tactic* is of type *TacticUnsure*, *Nothing* could be returned at that stage if no domino is found satisfying the condition.

Double-pointing arrows denote a function, which is used as a helper function by another function *(e.g. a function is traversing through a list and uses another function to evaluate elements of that list)*. Data flows one way and then comes back the same way and the path could be taken multiple times if needed.

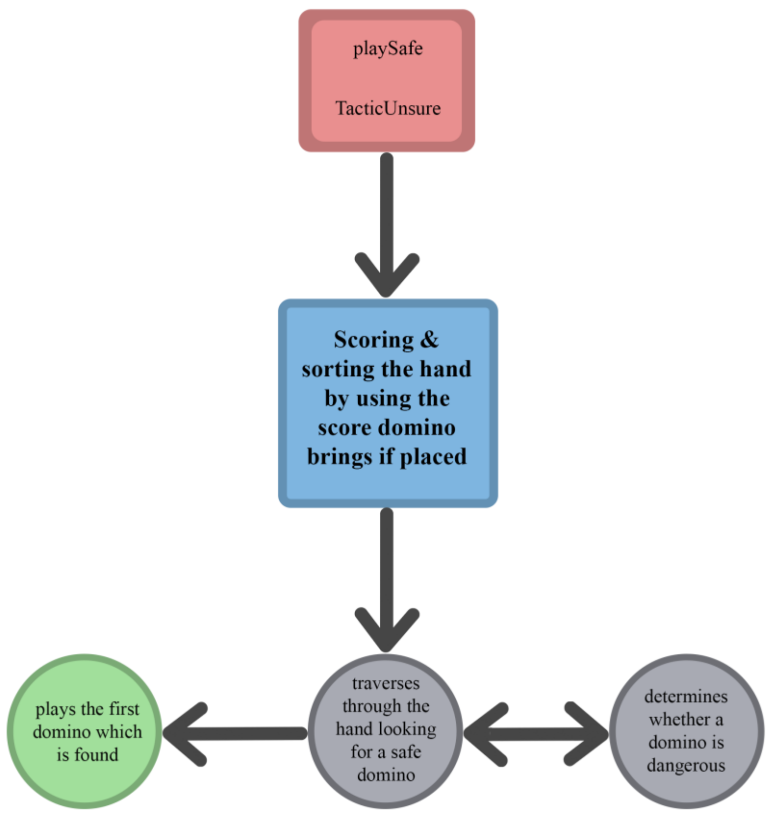
Yellow circles indicate functions, that are already provided either in a library *(e.g. Prelude)* or *DomsMatch.hs*.

Finally, I group some reusable functions from one *Tactic* and use them in another *Tactic* later. To avoid repetition in the diagram, I use those groups instead of the flowchart transitions to improve readability of the diagrams. Those function-flow groups are denoted in blue.

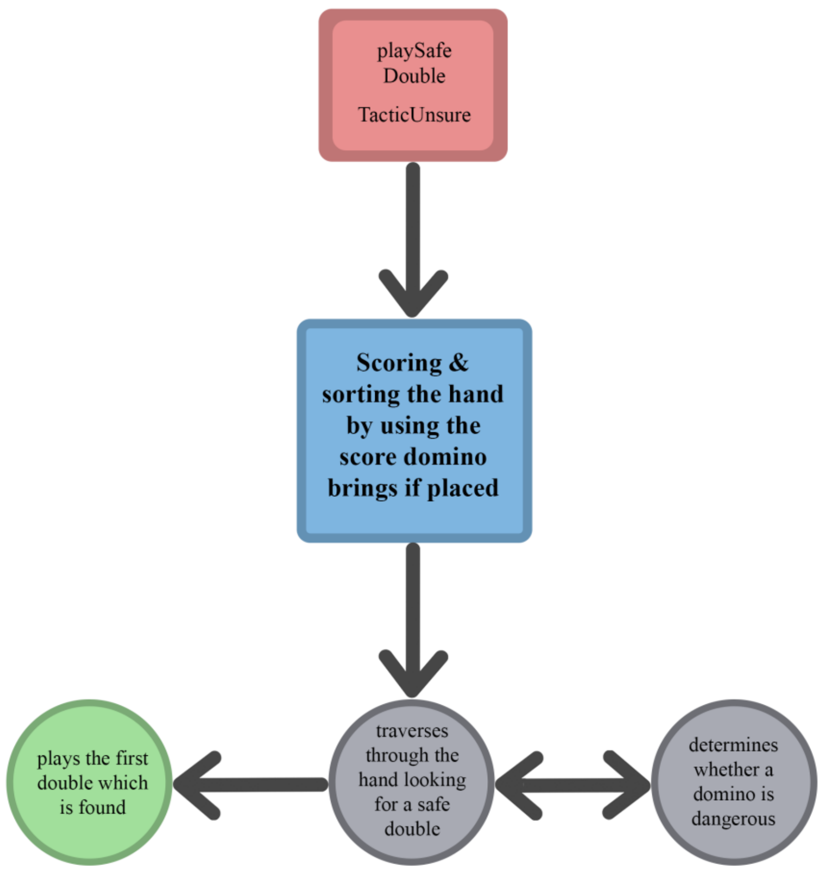
3.1.1. playHighestScoring


playHighest Scoring
Tactic
score hand by the score domino brings if placed
DomsMatch
scoreBoard
Prelude
last
sort the scored hand by the scores
take the last element
Scoring & sorting the hand by using the score domino brings if placed on board

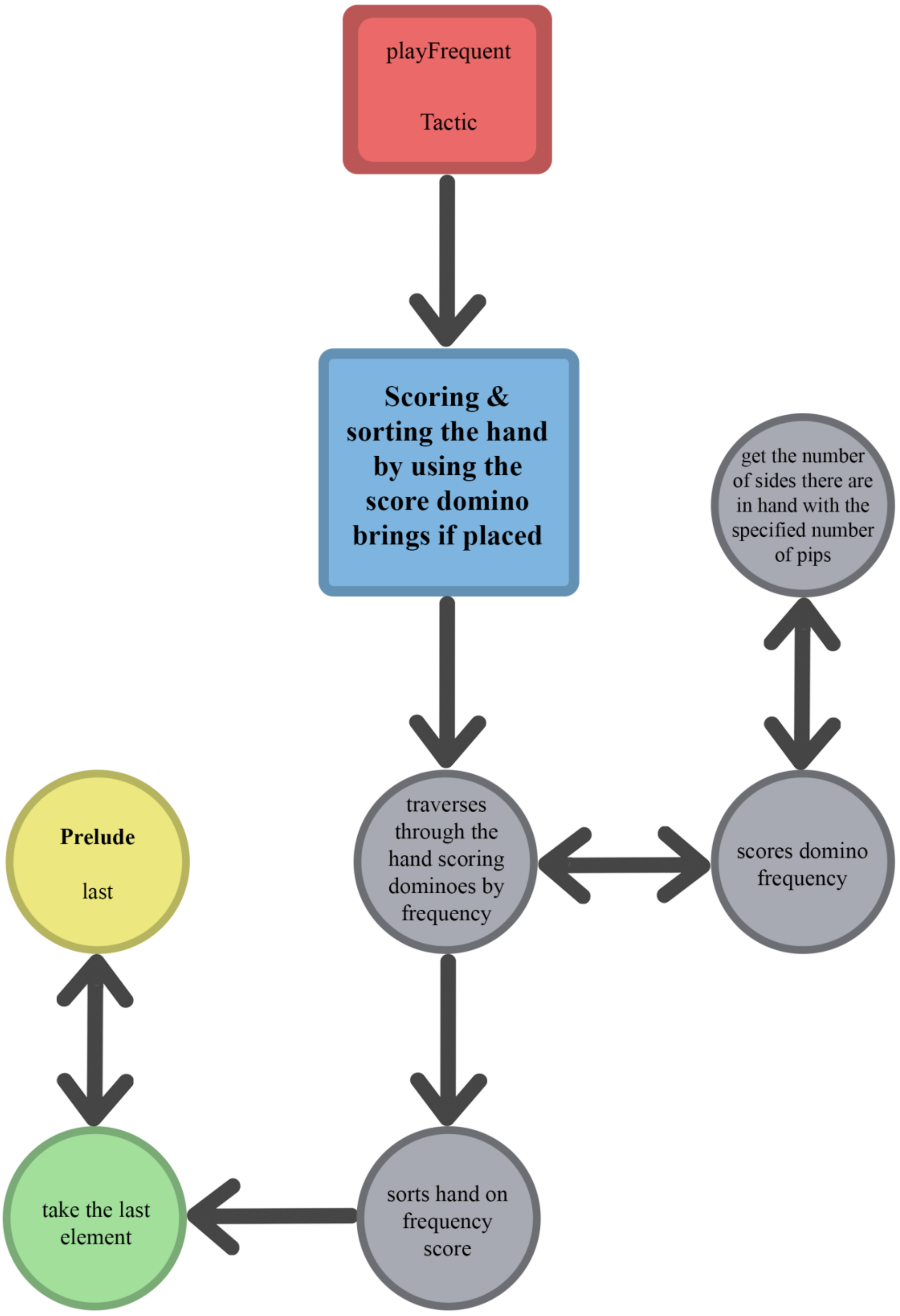
3.1.2. playSafe


playSafe
TacticUnsure
Scoring & sorting the hand by using the score domino brings if placed
plays the first domino which is found
traverses through the hand looking for a safe domino
determines whether a domino is dangerous

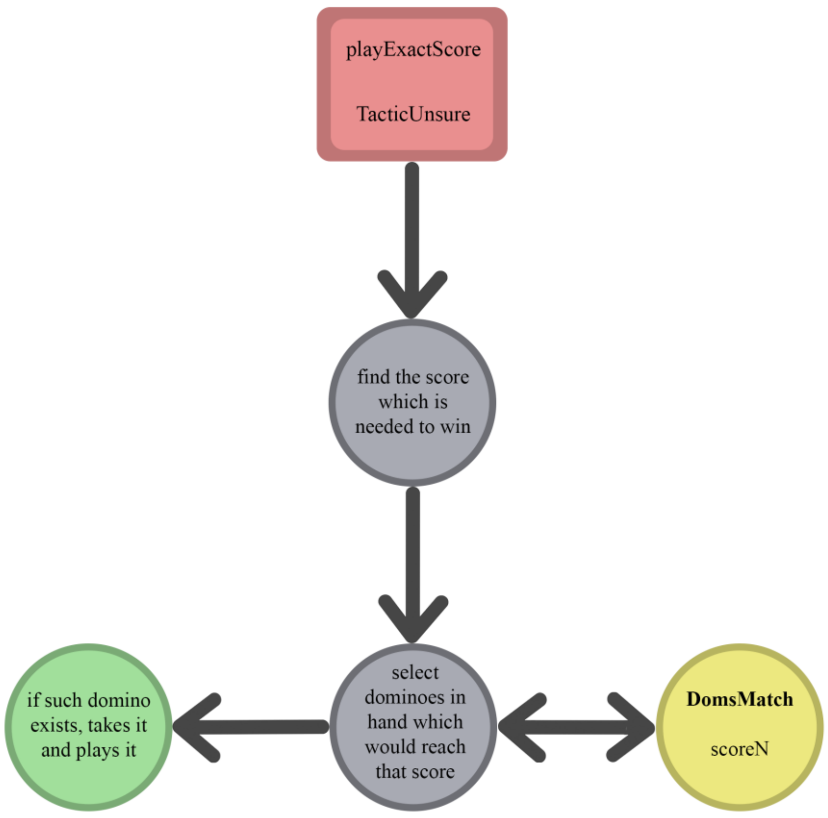
3.1.3. playSafeDouble


playSafe Double
TacticUnsure
Scoring & sorting the hand by using the score domino brings if placed
plays the first double which is found
traverses through the hand looking for a safe double
determines whether a domino is dangerous

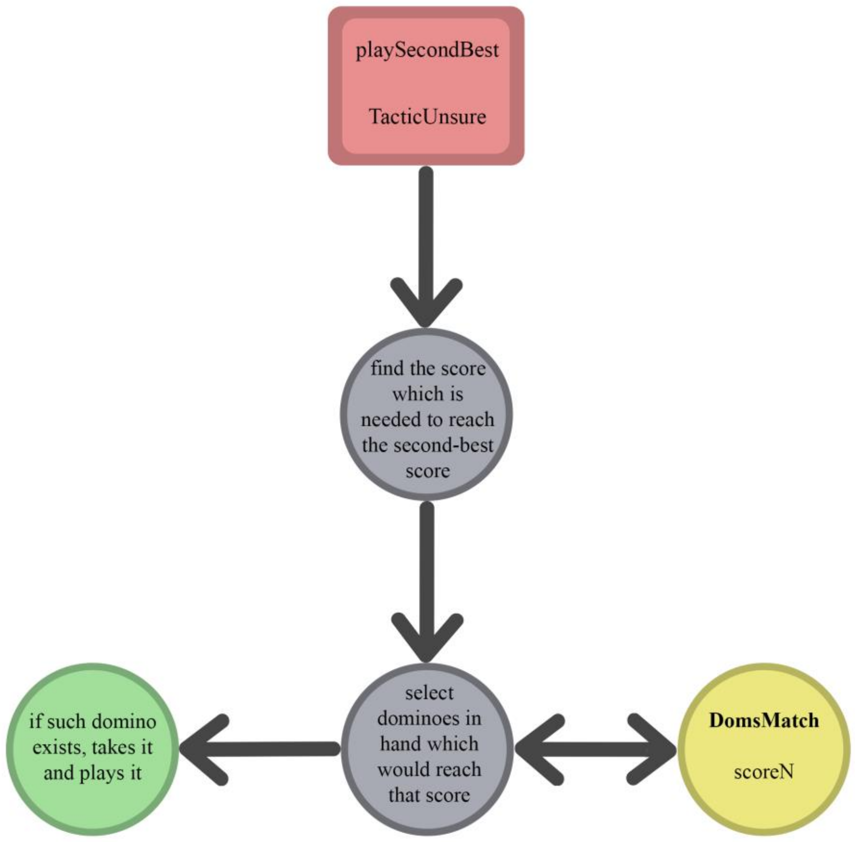
3.1.4. playFrequent


playFrequent
Tactic
Scoring & sorting the hand by using the score domino brings if placed
get the number of sides there are in hand with the specified number of pips
traverses through the hand scoring dominoes by frequency
scores domino frequency
Prelude
last
sorts hand on frequency score
take the last element

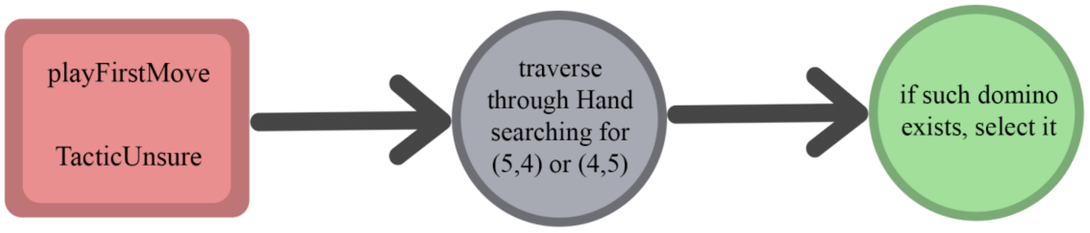
3.1.5.playExactScore


playExactScore
TacticUnsure
find the score which is needed to win
select dominoes in hand which would reach that score
if such domino exists, takes it and plays it
DomsMatch
scoreN

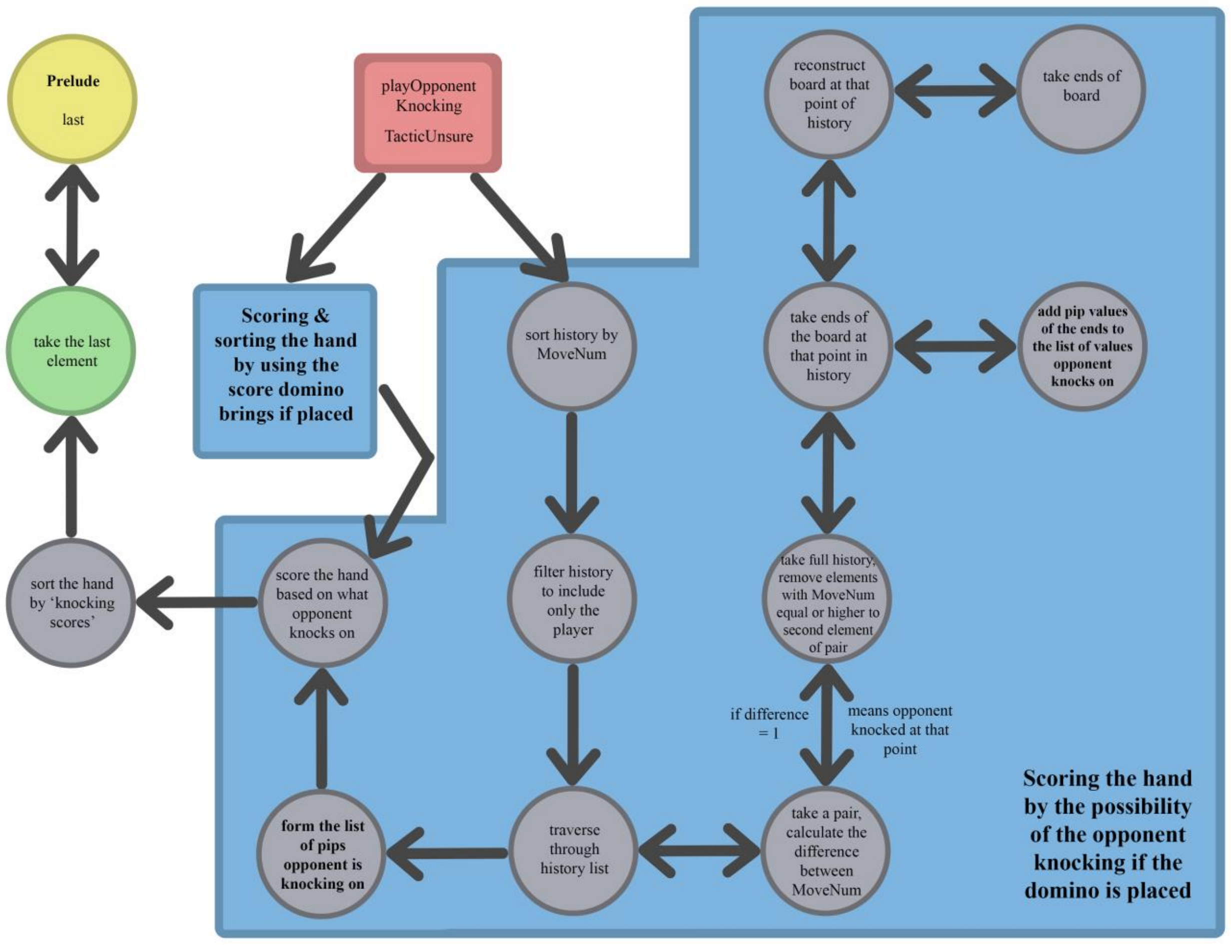
3.1.6. playSecondBest


playSecondBest
TacticUnsure
find the score which is needed to reach the second-best score
select dominoes in hand which would reach that score
if such domino exists, takes it and plays it
DomsMatch
scoreN

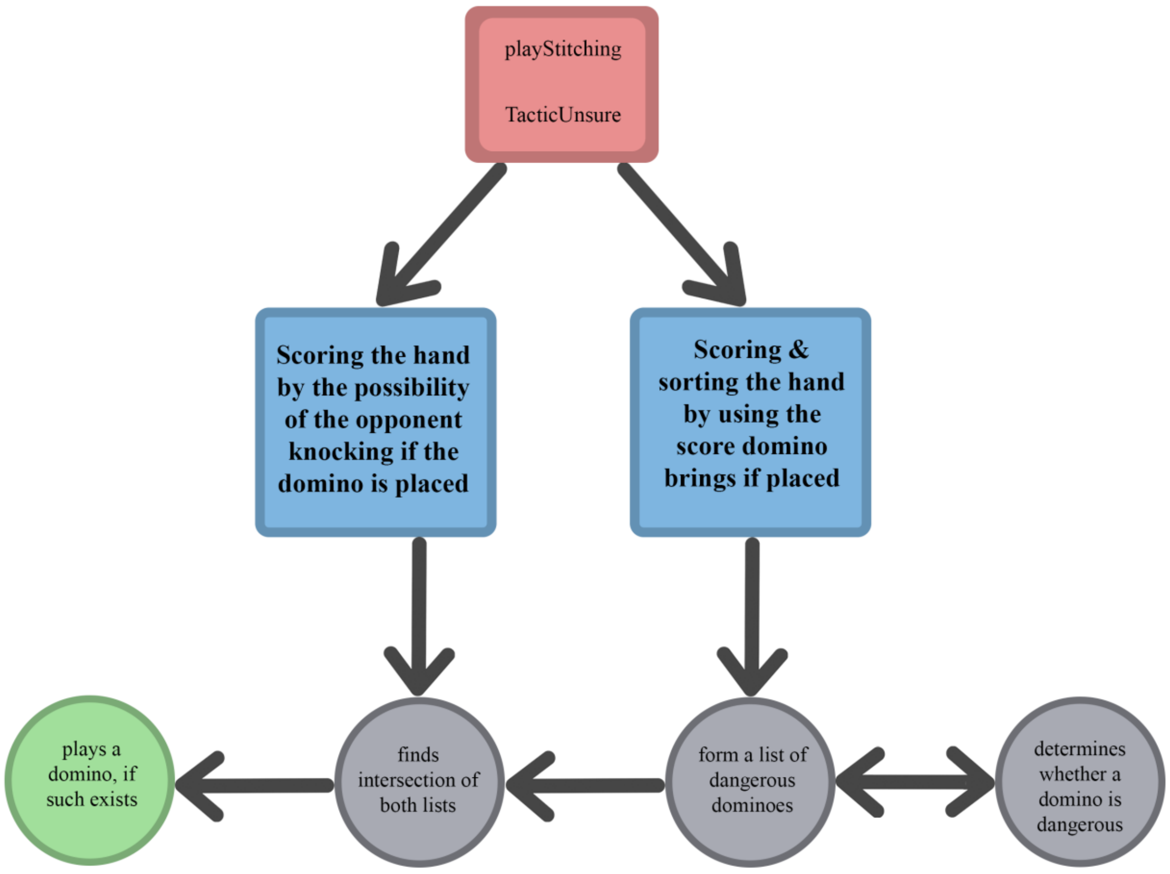
3.1.7. playFirstMove


playFirstMove
TacticUnsure
traverse through Hand searching for (5,4) or (4,5)
if such domino exists, select it

3.1.8.playOpponentKnocking


Prelude
last
take the last element
sort the hand by 'knocking scores'
playOpponent Knocking
TacticUnsure
Scoring & sorting the hand by using the score domino brings if placed
score the hand based on what opponent knocks on
form the list of pips opponent is knocking on
sort history by MoveNum
filter history to include only the player
traverse through history list
reconstruct board at that point of history
take ends of board
take ends of the board at that point in history
add pip values of the ends to the list of values opponent knocks on
take full history, remove elements with MoveNum equal or higher to second element of pair
if difference = 1
means opponent knocked at that point
take a pair, calculate the difference between MoveNum
Scoring the hand by the possibility of the opponent knocking if the domino is placed

3.1.9. playStitching

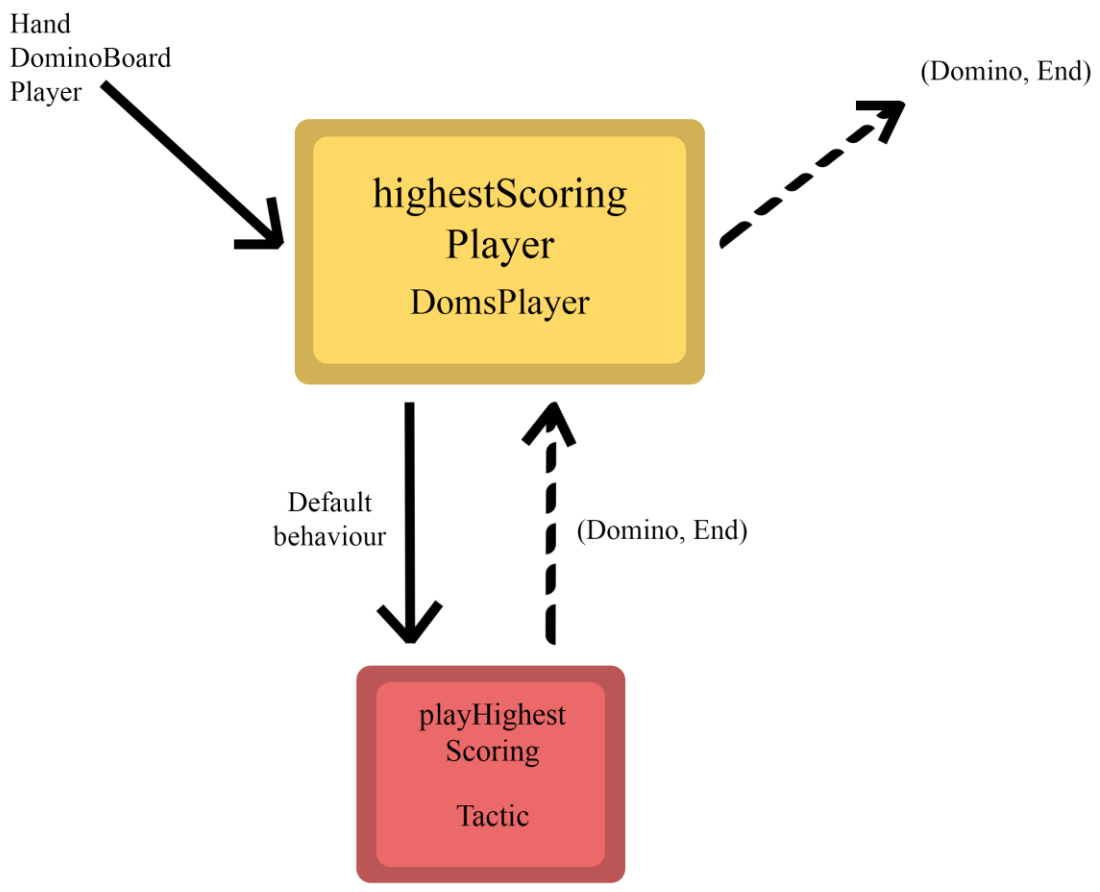

playStitching
TacticUnsure
Scoring the hand by the possibility of the opponent knocking if the domino is placed
Scoring & sorting the hand by using the score domino brings if placed
plays a domino, if such exists
finds intersection of both lists
form a list of dangerous dominoes
determines whether a domino is dangerous

3.2. Players

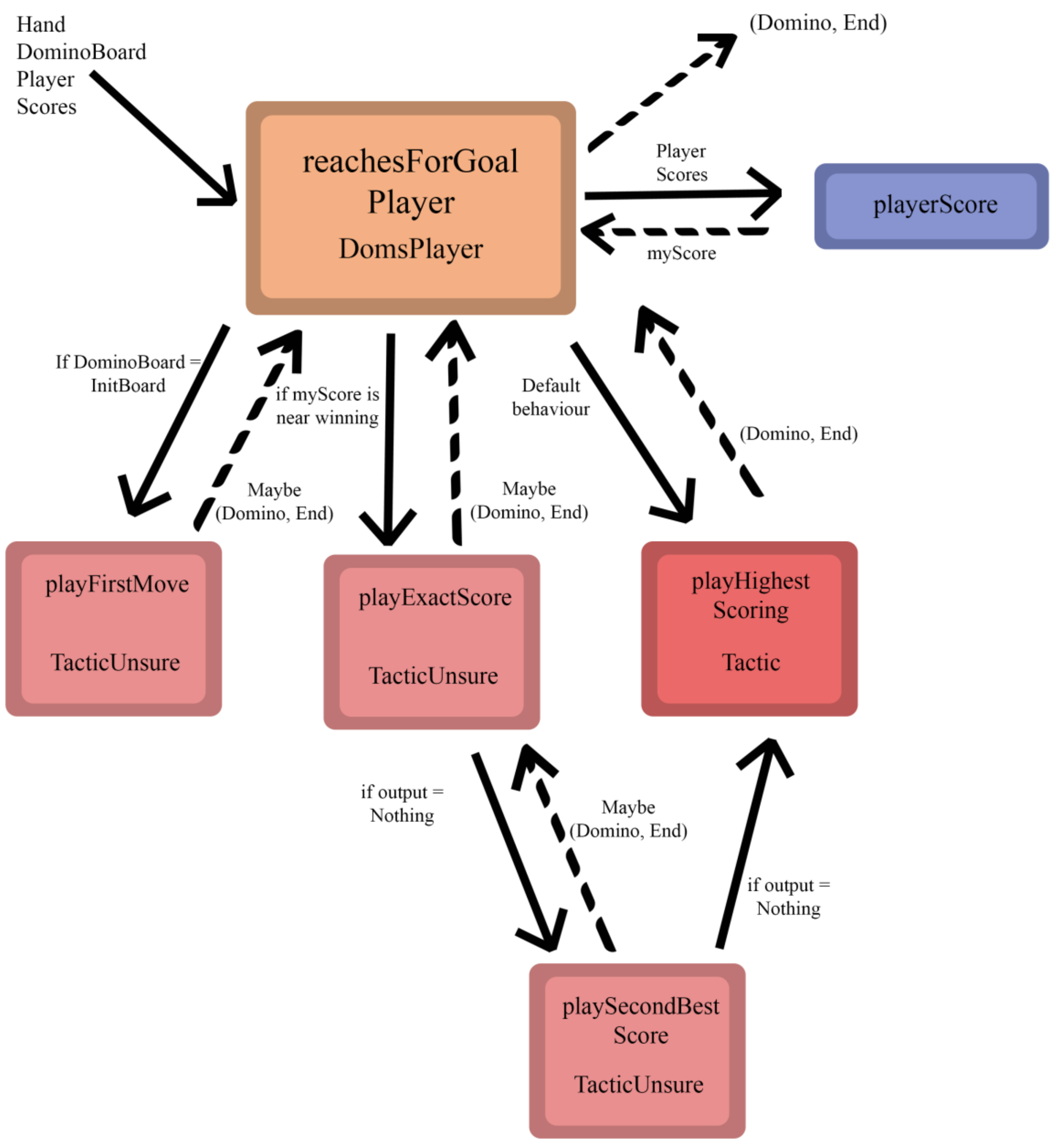
The solid arrows denote logic flow- they act as if statements, hence a tactic is equipped based on a certain condition. Thus, the diagrams are meant to be read from left to right- if a path is followed the path on its left could not be followed or returned *Nothing*. For some players, if no paths work out, the *default behaviour* path is followed and it is guaranteed to always return a domino.

The dashed arrows act as a return statement and they denote what is returned by a function to the domino player function.

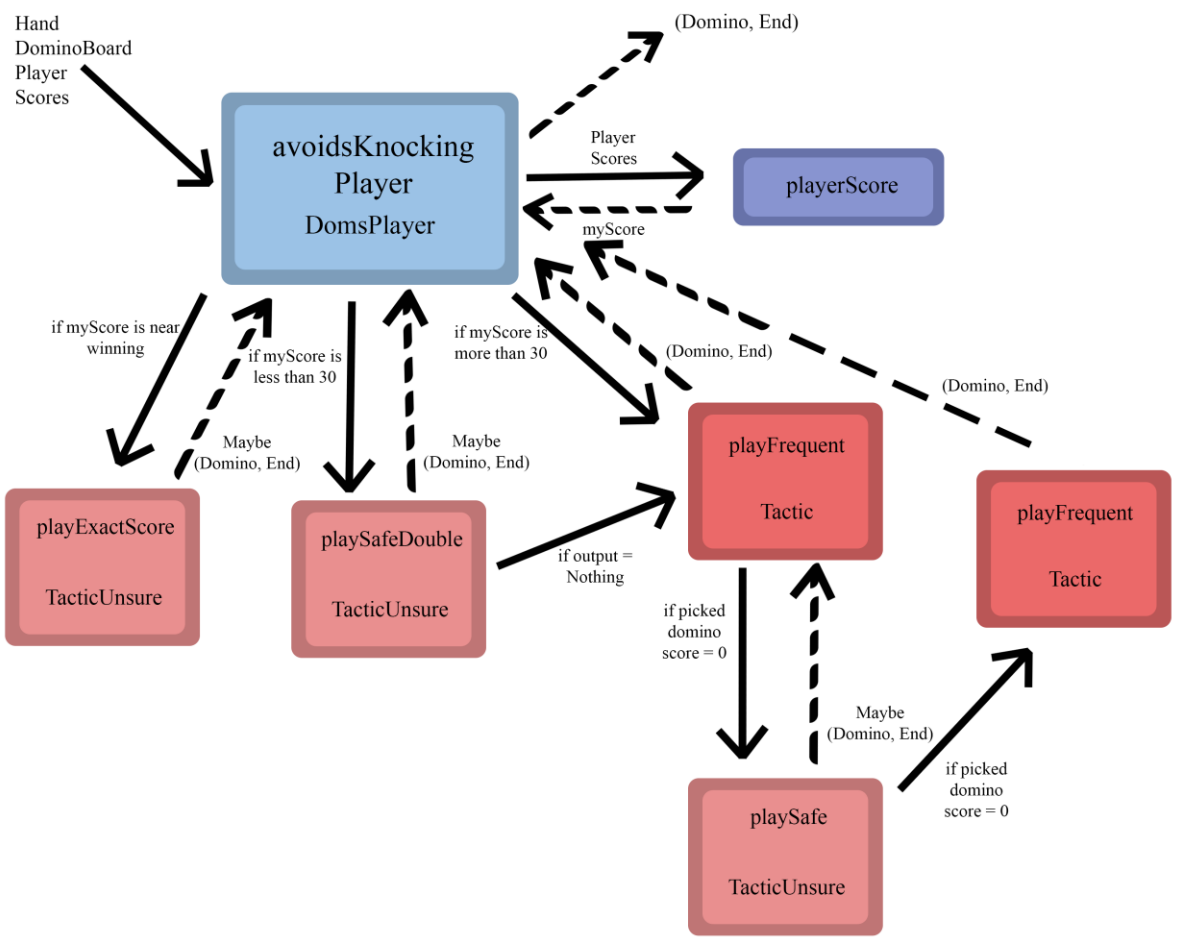
3.2.1. highestScoringPlayer



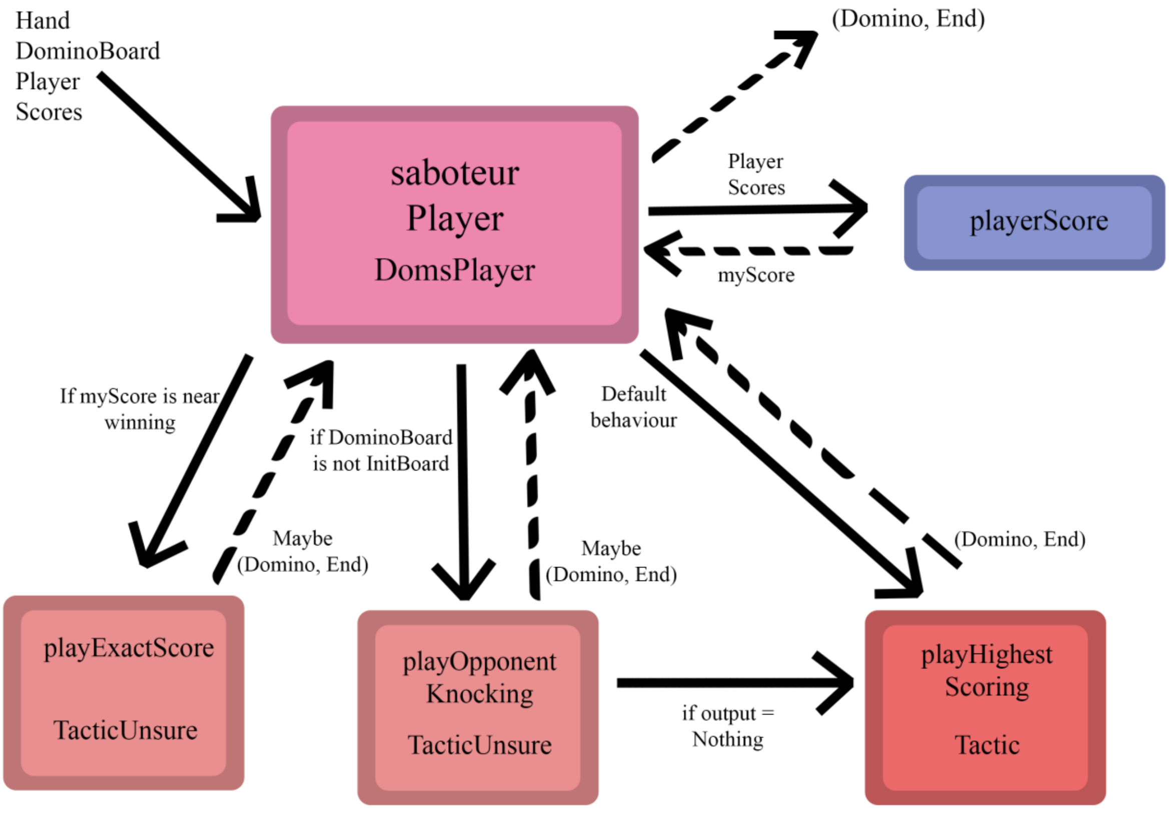
3.2.2. reachesForGoalPlayer



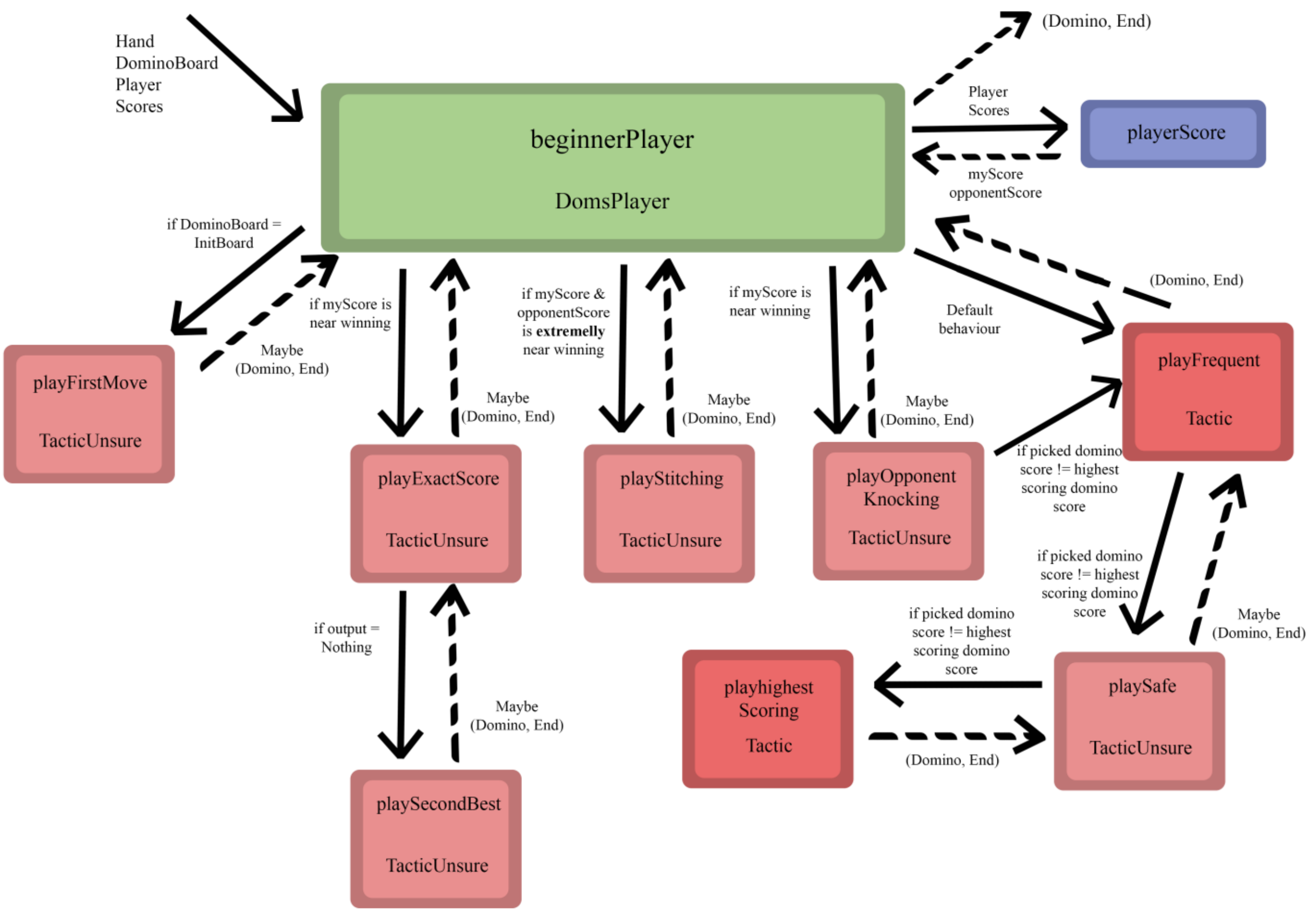
3.2.3.avoidsKnockingPlayer



3.2.4. saboteurPlayer



3.2.5. beginnerPlayer


Hand
DominoBoard
Player
Scores
beginnerPlayer
DomsPlayer
(Domino, End)
Player
Scores
playerScore
myScore
opponentScore
if DominoBoard = InitBoard
if myScore is near winning
Maybe (Domino, End)
playFirstMove
TacticUnsure
if myScore & opponentScore is **extremelly** near winning
if myScore is near winning
Default behaviour
(Domino, End)
Maybe (Domino, End)
Maybe (Domino, End)
Maybe (Domino, End)
playFrequent
Tactic
playExactScore
TacticUnsure
playStitching
TacticUnsure
playOpponent Knocking
TacticUnsure
if picked domino score != highest scoring domino score
if picked domino score != highest scoring domino score
if output = Nothing
Maybe (Domino, End)
if picked domino score != highest scoring domino score
Maybe (Domino, End)
playhighest Scoring
Tactic
playSafe
TacticUnsure
(Domino, End)
playSecondBest
TacticUnsure