COMP 6721
Aymeric Grail
#26810020
aymeric.grail@gmail.com

# Project Report

---

## General notes

### Sources

The game has been developed in Java 1.8, using IntelliJ
To start the game, simply run the “controller.StartGame” class.
The README.md file contains information on some of the game properties and other information useful for the A.I.

### Playing

Enter the name of the players. Any player whose name starts with “AI ” will be an automated player (eg: AI Google Deep Mind).
The game allows 2 AI player to play against each others.
The game is over when:

- Any player successfully managed to play a polarized ladder
- Or all positions on the board have been played

If no polarized ladder is present when the game is over, then it’s a tie.

### Testing

A test class covers the GameController, ensuring the manual game works. That is, it tests that players are able to play on the board, and that winners are properly detected.
The A.I. is not covered by automated tests at the moment.

---

## The Program

### UI:

The game has a very simple console UI. The console is hosted by the GameController to update the board, and is also passed to HumanPlayer so it can interacts with the users when they take turn.

## AI:

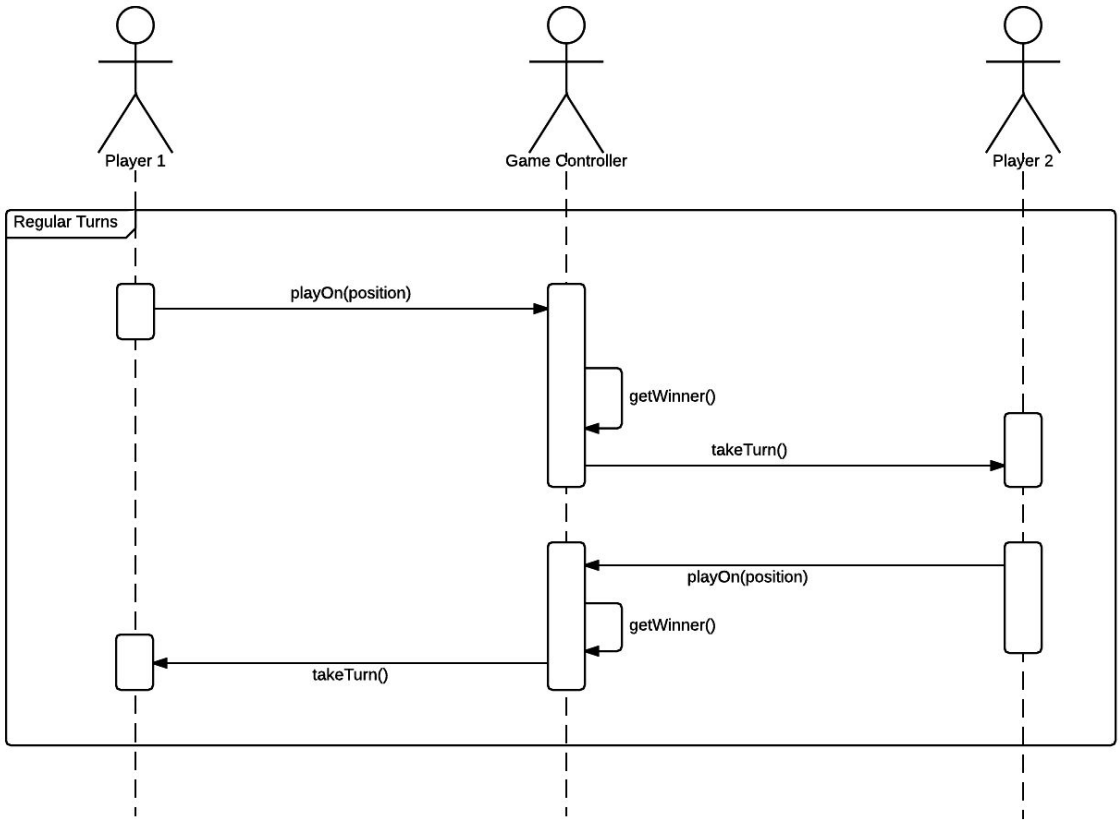
The AI is hosted by the AIPlayer, and is isolated in its own AI module. See the section on the heuristic for more details about the AI.

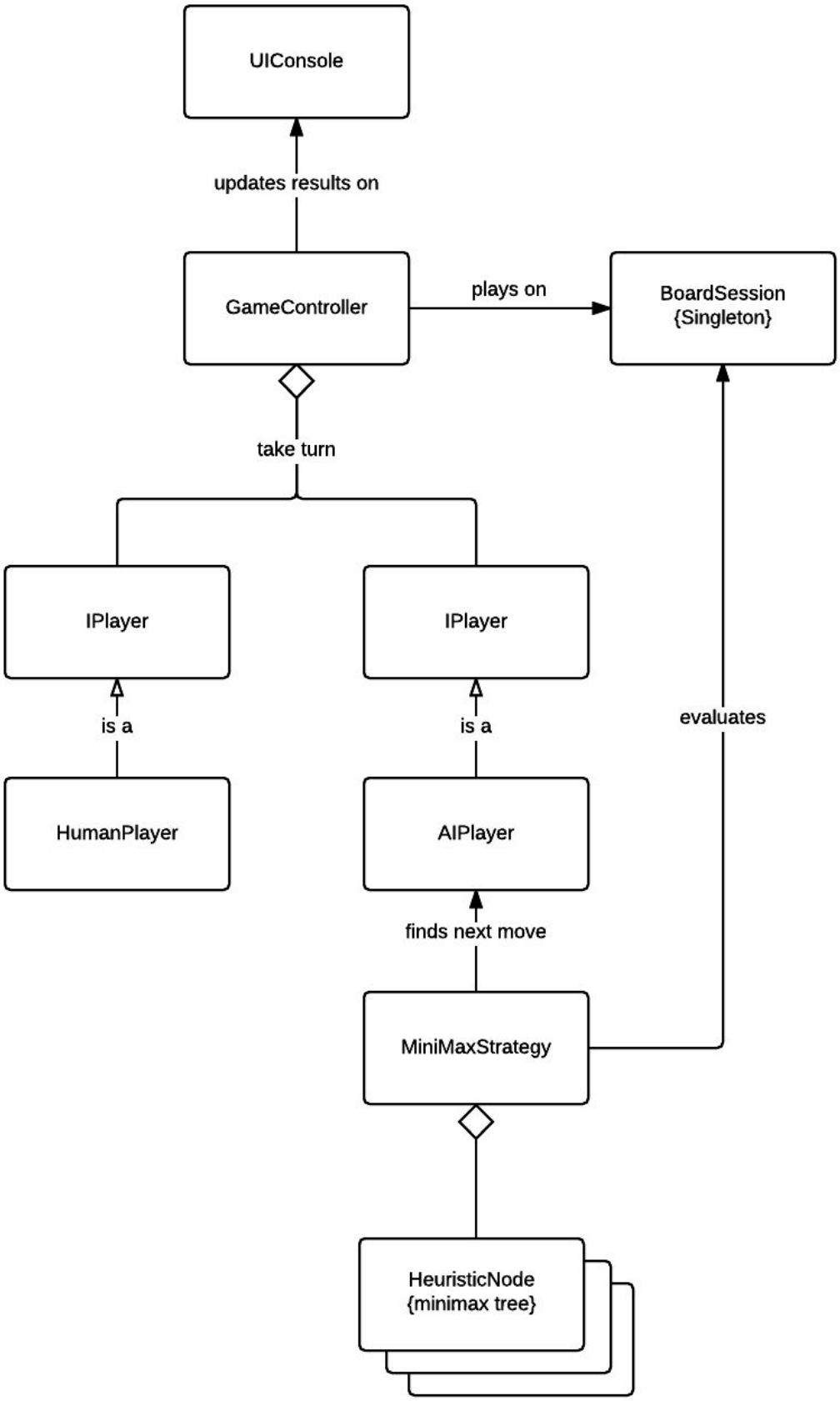
## GameController:

The game controller is in charge of the game. Its responsibilities are:

- Set up the game (the board, the players…)
- Coordinate player's actions
- Validate the player's decisions
- Determine the winner
- Send the results to the UI

## Architecture:




UIConsole
updates results on
GameController
plays on
BoardSession
{Singleton}
take turn
IPlayer
IPlayer
is a
is a
evaluates
HumanPlayer
AIPlayer
finds next move
MiniMaxStrategy
HeuristicNode
{minimax tree}

# The Heuristic

## Minimax

The minimax tree is set at a depth of 2.
The heuristic used by the A.I. is such as:

- Any goal state encountered by max is worth +1.000
- Any goal state encountered by min is worth -1.000
- When a goal state is encountered, its score is immediately returned
- If no goal state is encountered, then every leaf is evaluated depending on its static weight score (see below)

## Static Weight Scoring

Every position on the board has manually been given 2 static scores such as:

- Possible win score: number of polarized ladder a given position can participate in. Any position can participate in at least 1 polarized ladder, but no more than 10.
- Possible counter score: number of polarized ladder a position can counter. Only position at the center of the board might counter an opponent's ladder

The total static weight score of a state is the sum of all the AI Player's positions win and counter score.
For example, if AI plays 'X' and only occupies positions {(1,7) (2,6)} then its score would be:
f(X) = (2+0) + (3+1)
= 6

## Possible Improvements

A better heuristic would be to evaluate the weight of each position dynamically. That is that the initial static weight would be reduced by the amount of possibilities that have been countered by the opponent.
The score could also be refined by evaluating how close each substate is from winning. A substate is a subset of 7 positions forming a polarized ladder and its counter polarization (see README.md file). Each state has a total of 50 substates (there are 50 distinct ladders on a board).