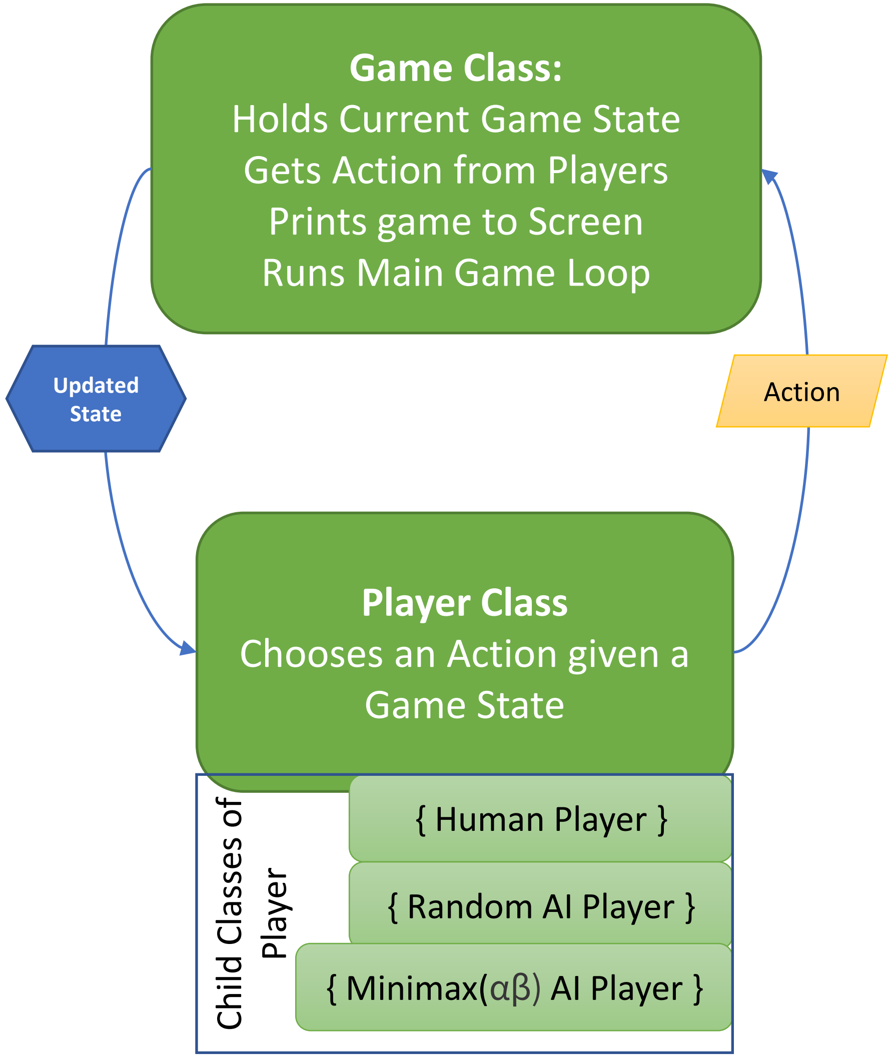
Chess

Console Based

Minimax AI

Simplified Overview – State-based Gameplay



Game Class:
Holds Current Game State
Gets Action from Players
Prints game to Screen
Runs Main Game Loop

Updated State

Action

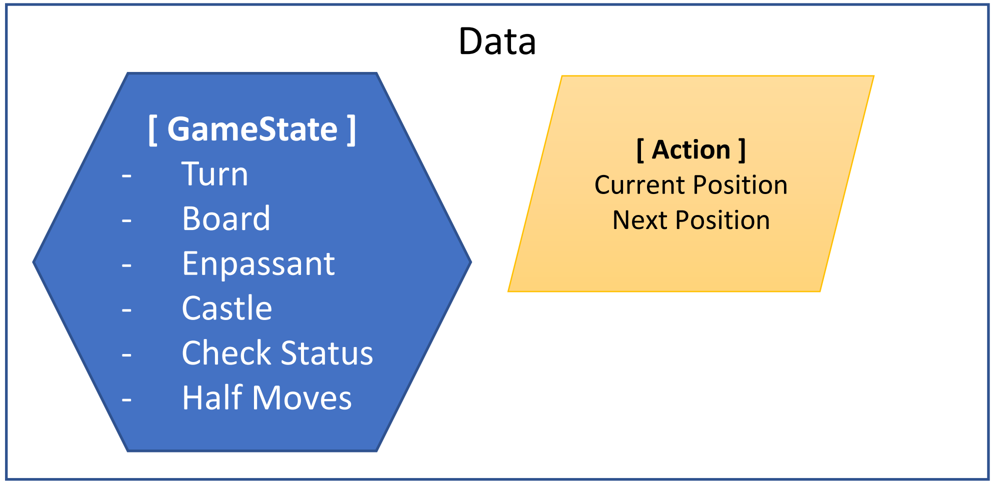
Player Class
Chooses an Action given a Game State

Child Classes of Player

{ Human Player }

{ Random AI Player }

{ Minimax(αβ) AI Player }



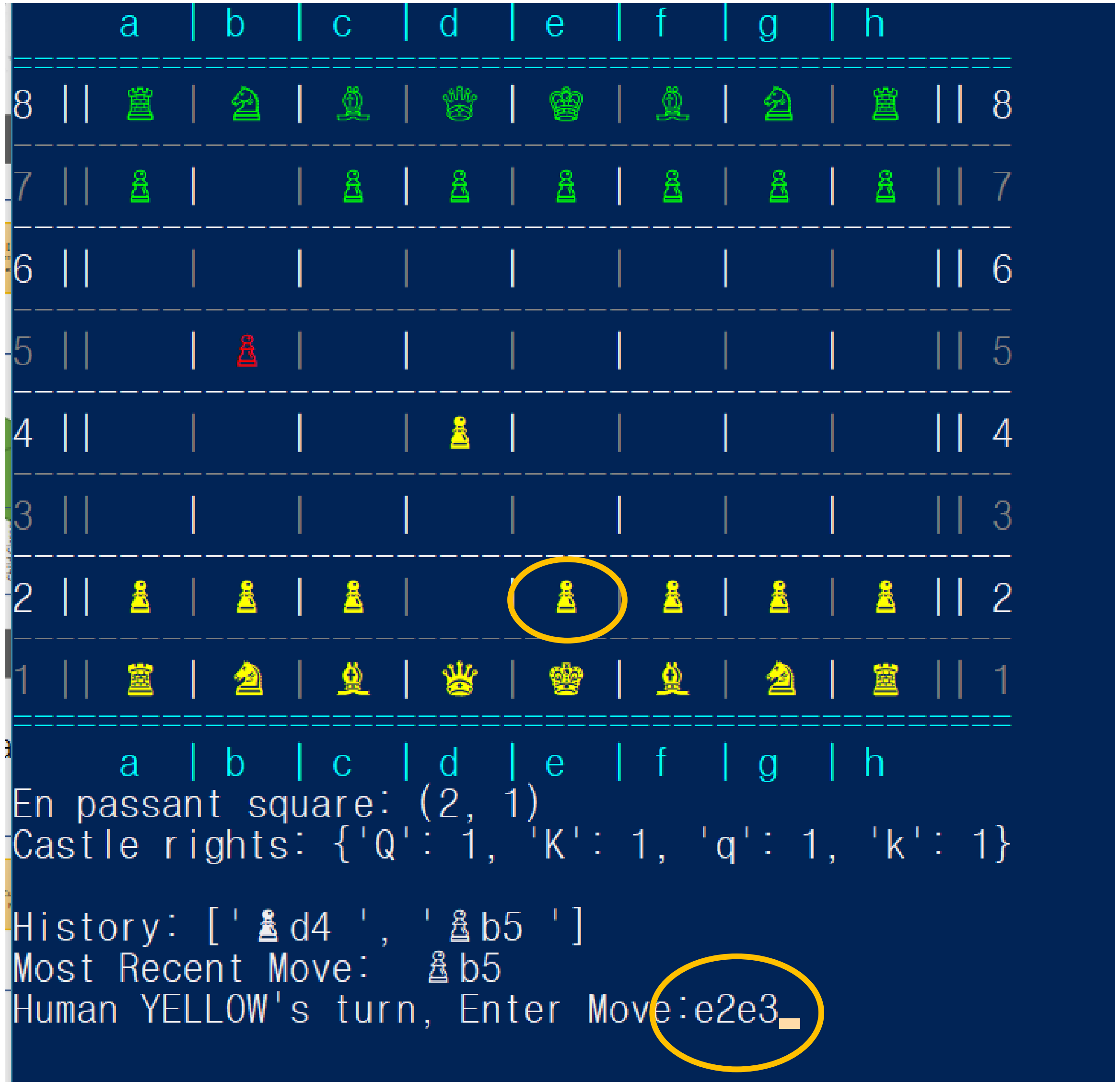
Data

[GameState]

- Turn
- Board
- Enpassant
- Castle
- Check Status
- Half Moves

[Action]
Current Position
Next Position

Console Interface



```
       a  | b  | c  | d  | e  | f  | g  | h
==================================================
8 || ♖ | ♘ | ♗ | ♕ | ♔ | ♗ | ♘ | ♖ || 8
--------------------------------------------------
7 || ♙ |    | ♙ | ♙ | ♙ | ♙ | ♙ | ♙ || 7
--------------------------------------------------
6 ||    |    |    |    |    |    |    |    || 6
--------------------------------------------------
5 ||    | ♙ |    |    |    |    |    |    || 5
--------------------------------------------------
4 ||    |    |    | ♙ |    |    |    |    || 4
--------------------------------------------------
3 ||    |    |    |    |    |    |    |    || 3
--------------------------------------------------
2 || ♙ | ♙ | ♙ |    | ♙ | ♙ | ♙ | ♙ || 2
--------------------------------------------------
1 || ♖ | ♘ | ♗ | ♕ | ♔ | ♗ | ♘ | ♖ || 1
==================================================
       a  | b  | c  | d  | e  | f  | g  | h
En passant square: (2, 1)
Castle rights: {'Q': 1, 'K': 1, 'q': 1, 'k': 1}

History: ['♙d4 ', '♙b5 ']
Most Recent Move:  ♙b5
Human YELLOW's turn, Enter Move:e2e3
```

Move is made by typing in:
[Source Square][Dest Square]

Ex: e2e3 moves the pawn in e2 to square e3

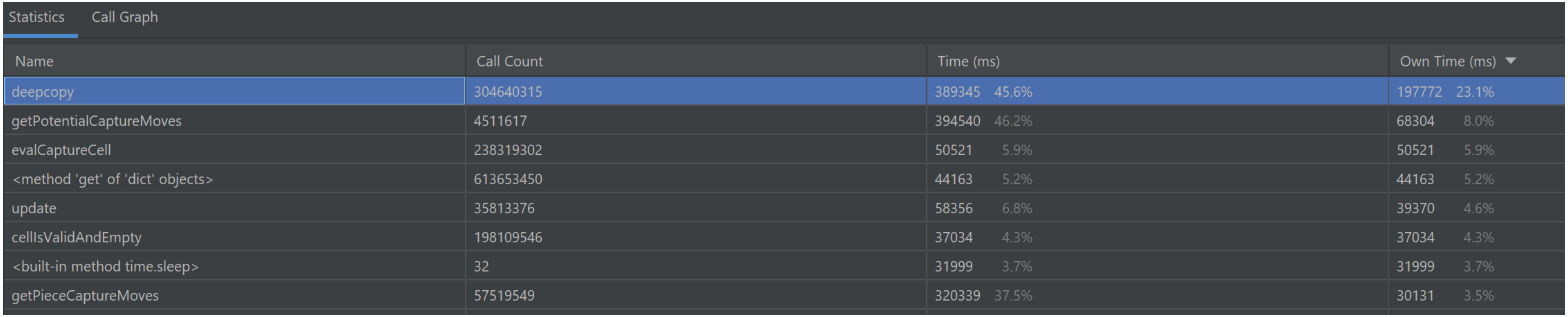
Minimax with αβ pruning

- Goal of Minimax AI Agent is to minimize the opponent's value
  In other words, the computer is doing its best to make you lose

- How to Value a given state of the game – Hardcoded (vs TD Learning)
    - Material Score:
        - Based on pieces (King 1000, Queen 9, Rook 5, Bishop/Knight 3, Pawn 1)
    - Mobility Score: higher if more moves are possible
    - Check Value: bonus if CHECK move
    - Ending utility: 10000 for checkmate, -10000 for losing via checkmate

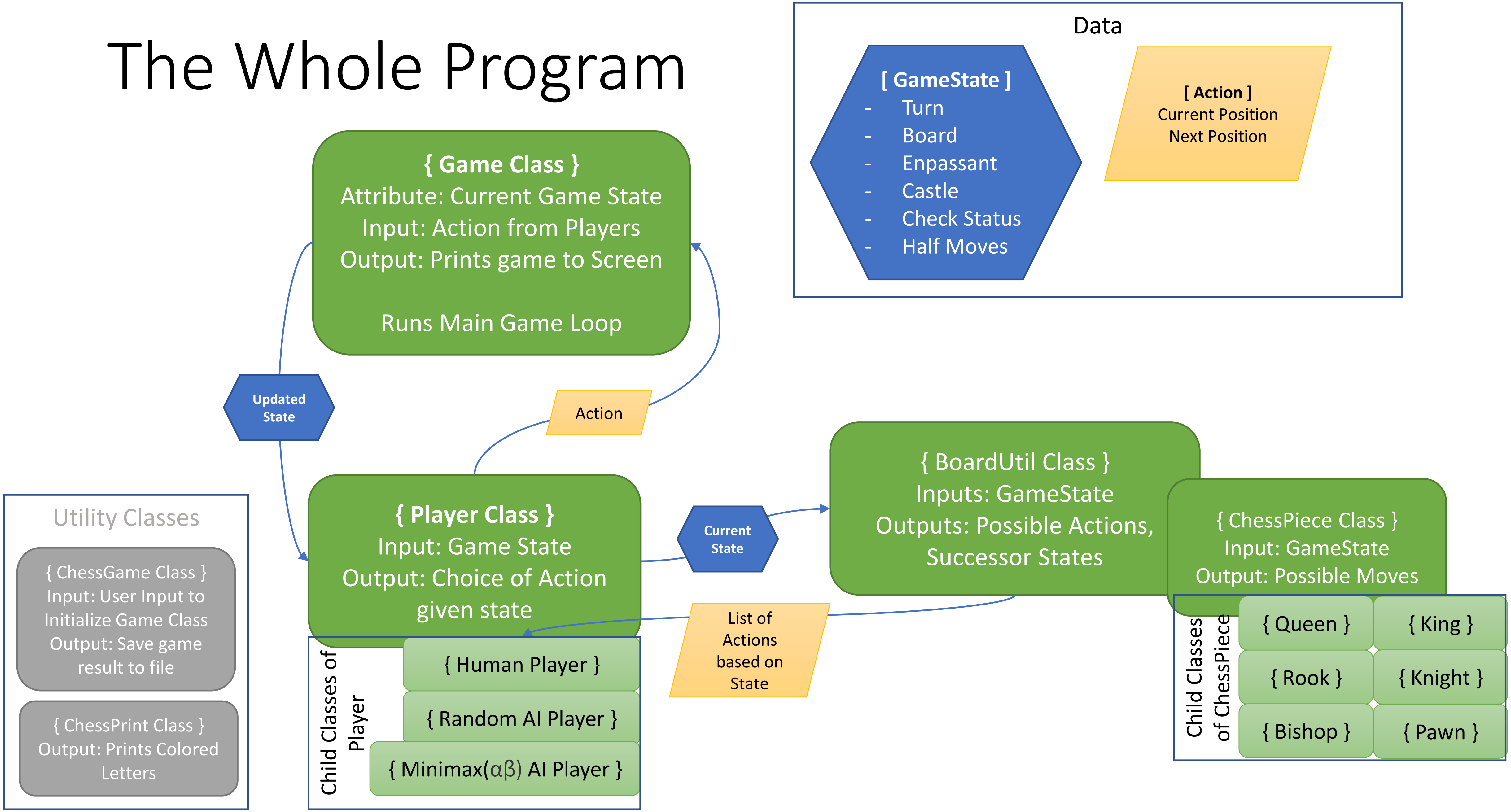
Technical Challenges: AI Takes too long!

- Approach: Code profiling
  - Deepcopy() is very slow!



Name	Call Count	Time (ms)	Own Time (ms) ▼
deepcopy	304640315	389345 45.6%	197772 23.1%
getPotentialCaptureMoves	4511617	394540 46.2%	68304 8.0%
evalCaptureCell	238319302	50521 5.9%	50521 5.9%
<method 'get' of 'dict' objects>	613653450	44163 5.2%	44163 5.2%
update	35813376	58356 6.8%	39370 4.6%
cellIsValidAndEmpty	198109546	37034 4.3%	37034 4.3%
<built-in method time.sleep>	32	31999 3.7%	31999 3.7%
getPieceCaptureMoves	57519549	320339 37.5%	30131 3.5%

The Whole Program



Data

[GameState]

- Turn
- Board
- Enpassant
- Castle
- Check Status
- Half Moves

[Action]
Current Position
Next Position

{ Game Class }
Attribute: Current Game State
Input: Action from Players
Output: Prints game to Screen

Runs Main Game Loop

Updated State

Action

{ Player Class }
Input: Game State
Output: Choice of Action given state

Current State

{ BoardUtil Class }
Inputs: GameState
Outputs: Possible Actions, Successor States

{ ChessPiece Class }
Input: GameState
Output: Possible Moves

Child Classes of ChessPiece

{ Queen } { King }
{ Rook } { Knight }
{ Bishop } { Pawn }

List of Actions based on State

Child Classes of Player

{ Human Player }
{ Random AI Player }
{ Minimax(αβ) AI Player }

Utility Classes

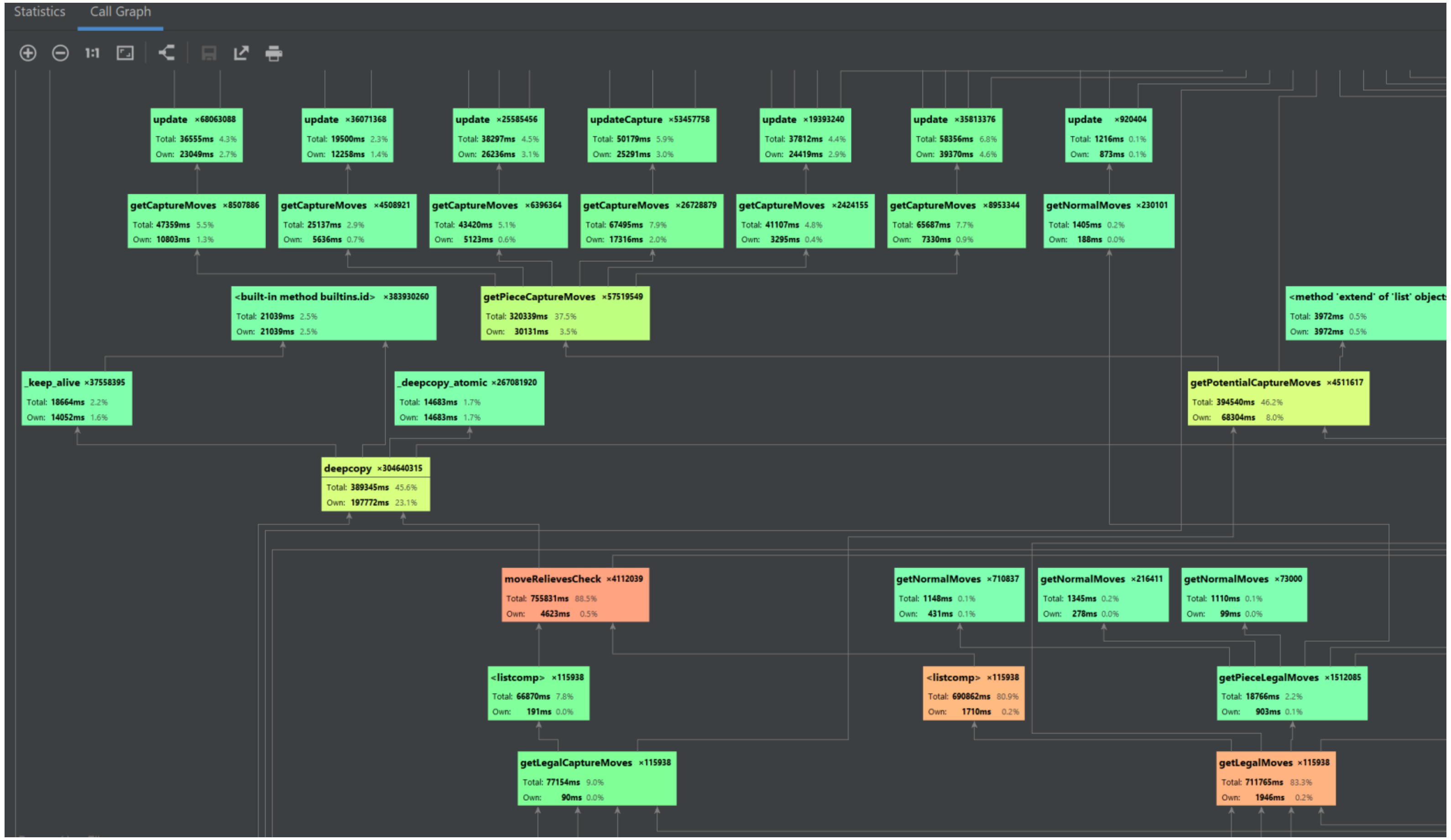
{ ChessGame Class }
Input: User Input to Initialize Game Class
Output: Save game result to file

{ ChessPrint Class }
Output: Prints Colored Letters

Technical Challenges: AI Takes too long!

- Gained more insights from Inspection of call graph

- BUT, not enough time to fully optimize


Statistics
Call Graph
1:1
update ×68063088
Total: 36555ms 4.3%
Own: 23049ms 2.7%
update ×36071368
Total: 19500ms 2.3%
Own: 12258ms 1.4%
update ×25585456
Total: 38297ms 4.5%
Own: 26236ms 3.1%
updateCapture ×53457758
Total: 50179ms 5.9%
Own: 25291ms 3.0%
update ×19393240
Total: 37812ms 4.4%
Own: 24419ms 2.9%
update ×35813376
Total: 58356ms 6.8%
Own: 39370ms 4.6%
update ×920404
Total: 1216ms 0.1%
Own: 873ms 0.1%
getCaptureMoves ×8507886
Total: 47359ms 5.5%
Own: 10803ms 1.3%
getCaptureMoves ×4508921
Total: 25137ms 2.9%
Own: 5636ms 0.7%
getCaptureMoves ×6396364
Total: 43420ms 5.1%
Own: 5123ms 0.6%
getCaptureMoves ×26728879
Total: 67495ms 7.9%
Own: 17316ms 2.0%
getCaptureMoves ×2424155
Total: 41107ms 4.8%
Own: 3295ms 0.4%
getCaptureMoves ×8953344
Total: 65687ms 7.7%
Own: 7330ms 0.9%
getNormalMoves ×230101
Total: 1405ms 0.2%
Own: 188ms 0.0%
<built-in method builtins.id> ×383930260
Total: 21039ms 2.5%
Own: 21039ms 2.5%
getPieceCaptureMoves ×57519549
Total: 320339ms 37.5%
Own: 30131ms 3.5%
<method 'extend' of 'list' object
Total: 3972ms 0.5%
Own: 3972ms 0.5%
_keep_alive ×37558395
Total: 18664ms 2.2%
Own: 14052ms 1.6%
_deepcopy_atomic ×267081920
Total: 14683ms 1.7%
Own: 14683ms 1.7%
getPotentialCaptureMoves ×4511617
Total: 394540ms 46.2%
Own: 68304ms 8.0%
deepcopy ×304640315
Total: 389345ms 45.6%
Own: 197772ms 23.1%
moveRelievesCheck ×4112039
Total: 755831ms 88.5%
Own: 4623ms 0.5%
getNormalMoves ×710837
Total: 1148ms 0.1%
Own: 431ms 0.1%
getNormalMoves ×216411
Total: 1345ms 0.2%
Own: 278ms 0.0%
getNormalMoves ×73000
Total: 1110ms 0.1%
Own: 99ms 0.0%
<listcomp> ×115938
Total: 66870ms 7.8%
Own: 191ms 0.0%
<listcomp> ×115938
Total: 690862ms 80.9%
Own: 1710ms 0.2%
getPieceLegalMoves ×1512085
Total: 18766ms 2.2%
Own: 903ms 0.1%
getLegalCaptureMoves ×115938
Total: 77154ms 9.0%
Own: 90ms 0.0%
getLegalMoves ×115938
Total: 711765ms 83.3%
Own: 1946ms 0.2%