1 Program Scope

The program should be able to receive as input a chess move in UCI(Universal Chess Interface) format i.e e2e4, and if the movement is valid, output the board state to the user or inform the user the input isn't valid. For this matter, the standard python library is enough address the problem. For debugging purposes, a graphical interface was also required and implemented in pygame, a graphical framework for games. Also for debugging and testing pourposes, it was used the program *pgn-extract* to convert PGN game notation to UCI notation.

2 Program project

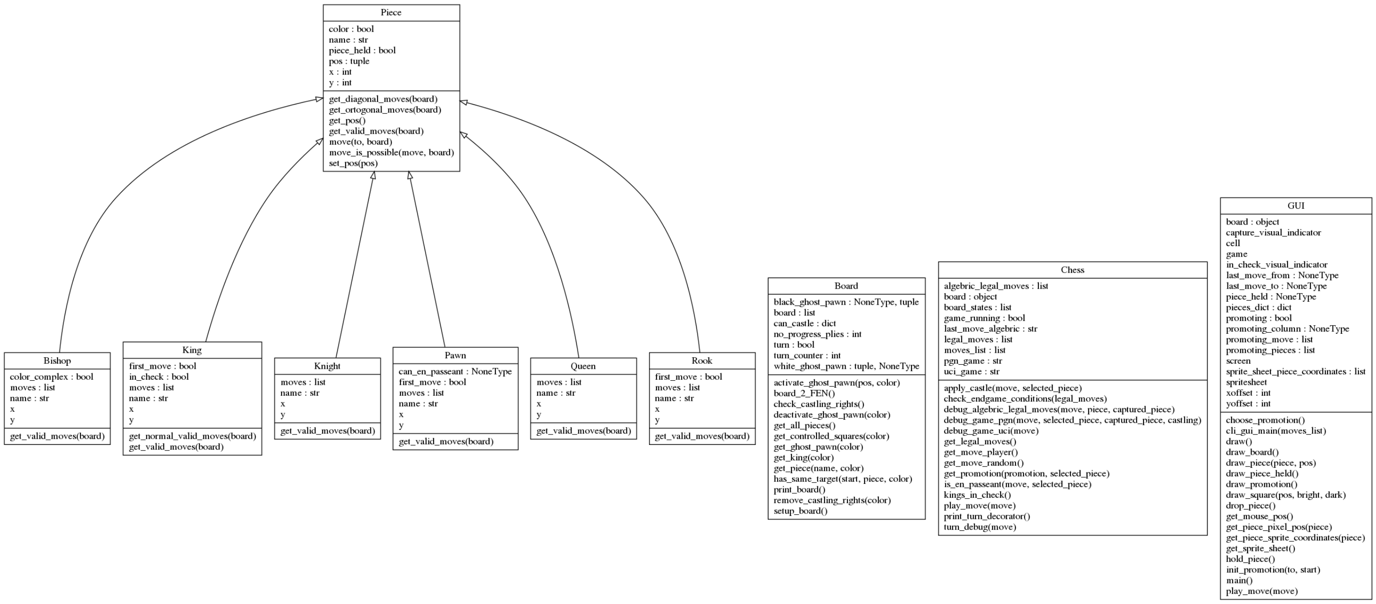
The project is constitued by four modules that contains in itself their respective major classe: The Piece, Board, Chess and GUI.

1. The Pieces module contains the Piece class, that is inherited by all the chess pieces, and specify how to get from each piece their own set of possible moves.

2. The Board module contains the Board class that is used to save all information relative to board state, such as pieces positions, castling rights, number of turns, en passeant possibility, etc.

3. The Chess module contains the Chess class that is used to process the Board information and create legal moves from which the player can chose to play.

4. The GUI module uses the Board and Chess classes to play the game in a graphical interface mode.



3 Testing

Number of plies (half-moves)	Number of possible games
1	20
2	400
3	8092
4	197,281
5	4,865,609
6	119,060,324
...	...
10	69,352,859,712,417

Tabela 1: Shannon's Calculation. Obs: A turn is composed by a white move and a black move. Five plies therefore stands for white playing three times and black two.

For basic operations accuracy, it was used the Shannon Number, which stands for all the possible moves that can be played until a certain ply(half-move). By the limitation of the computer power avaible for our disposal, and considering that the game was not written in a language nor written in a way for fast computation, we could only check the precision of the game until 5 ply, as we can see by the test log:

```
2022-01-22 00:00:36,742 Result of possible games with 1 ply: 20/20 - OK
2022-01-22 00:00:36,742 Elapsed time in 1 ply: 00h00m00s seconds
2022-01-22 00:00:37,312 Result of possible games with 2 ply: 400/400 - OK
2022-01-22 00:00:37,312 Elapsed time in 2 ply: 00h00m00s seconds
2022-01-22 00:00:52,137 Result of possible games with 3 ply: 8902/8902 - OK
2022-01-22 00:00:52,137 Elapsed time in 3 ply: 00h00m14s seconds
2022-01-22 00:07:11,715 Result of possible games with 4 ply: 197281/197281 - OK
2022-01-22 00:07:11,715 Elapsed time in 4 ply: 00h06m19s seconds
2022-01-22 08:45:00,073 Result of possible games with 5 ply: 4865609/4865609 - OK
2022-01-22 08:45:00,073 Elapsed time in 5 ply: 08h37m48s seconds
```

Although this is a good signal that basic operations are working, in 5 plies we cannot test all the complications that might arise during a chess game.

Depth	Captures	E.P	Castles	Promotions	Checks	Dscry Checks	Dbl Checks	Checkmates
1	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0
3	34	0	0	0	12	0	0	0
4	1576	0	0	0	469	0	0	8
5	82,719	258	0	0	27,251	6	0	347

Tabela 2: Number of "special" moves by depth accordingly to https://www.chessprogramming.org/Perft_Results

By this table we can see that we need to concentrate our efforts in testing Castle, Promotions, Discovery Checks and Double Checks.

For this matter, it was needed to create specific tests to check this special moves. For example, at 5 ply, there can't be a game with a promoted pawn case, therefore we need to make a specific test case for that matter.

```
python3 tests/promotionTest.py
python3 chess.py --guitest g2g4 h7h5 g4h5 g7g6  h5h6  h8h7  f2f3  h7g7  h6h7  f7f6
...
PlayedMoves: 1. g4 h5 2. gxh5 g6 3. h6 Rh7 4. f3 Rg7 5. h7 f6
```

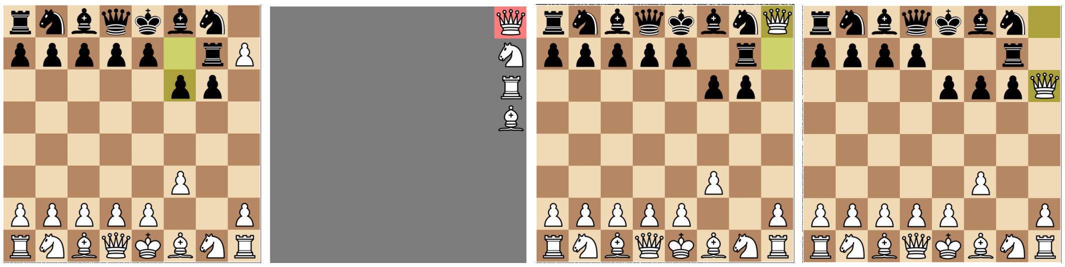


Figura 1: The left-most screenshot is the result of the test, and others are a sequence of screenshots of the user doing the promotion manually, and playing a move after to check if the piece is responsive.

While developing, the GUI and CLI interface could be behaving differently, having that in mind, we can also do the same test with the CLI if we are in doubt:

```
 >> $ python3 tests/promotionTest_cli.py
python3 chess.py -clitest g2g4 h7h5 g4h5 g7g6  h5h6  h8h7  f2f3  h7g7  h6h7  f7f6  h7h8q
e7e6 h8h6
(...)
Black's turn to move!
********************************
8| r | n | b | q | k | b | n |   |
7| p | p | p | p |   |   | r |   |
6|   |   |   |   | p | p | p | Q |
5|   |   |   |   |   |   |   |   |
4|   |   |   |   |   |   |   |   |
3|   |   |   |   |   | P |   |   |
2| P | P | P | P | P |   |   | P |
1| R | N | B | Q | K | B | N | R |
   a   b   c   d   e   f   g   h
********************************
```