

Chess AI Bot

Anuj Bhatia| 3035549029 |

# Basic Information

The Chess engine has been created in JAVA language, utilizes the basics of Object-oriented programming. The program has 3 classes, Chess, UI and Evaluation. The chess class contains the majority of the code for the chess engine, it contains the following methods:

* AlphaBeta: Explained later in section AlphaBeta
* Flipboard: It rotates the global variable Board, that contains the chess pieces to depict the moves by both the humans and the computer from one side only, to reduce the code redundancy.
* makeMove: The makeMove functions is called after the AI or the human has finalized a move the change the board from the initial position to the new position.
* undoMove: The undoMove is to return from the initial position to the position before that, it comes into hand when the computer needs to evaluate the next move to be made by using minimax algorithm.
* movesPossible: The movesPossible function generates a list of moves that are possible given the current state of the board. It in turn calls functions like possibleP, possibleQ, etc. to identify the moves possible for a particular piece and returns it.
* movement: The movement is a temp function which replaces the position of a piece to a new position and validates if that move is valid or not based on the safeKing function.
* safeKing: This function checks the position of the king and verifies if a particular move is made then the king would not be under a check. If the move yield a check, then that move is declared as invalid.

The Evaluation function class is explained later. The UI class has the following description of its functions:

* paintComponent(the interface function): gives the basic UI to the application by providing a background color, the alternate chess board and places the chess pieces as per the Board global variable of the chess class.
* mouseClicked, mouseEntered, mouseExited, mouseDragged, mouseMoved: These functions have just been initialized to ensure no error is there as they are required to be implemented due the using of the interfaces.
* mousePressed: The functions tries and identifies the position the user has selected and piece that corresponds to it.
* mouseReleased: The function tries to identify the position that the user wants that piece to be at and check if it is valid move or not, if it is then it moves the piece and waits for the response from the computer to that step. If it is an invalid move then it return to the original position of the piece and waits for the human to make a move.

# Game Play

The UI gives an option for the computer to take a move first or the user to make a move first. If the you choose computer, the computer will still be black, then please wait for the computer to make its move before making a move. If it is your move then make a valid move by selecting a piece and dragging and dropping it at a position of your choice. Note: the move doesn’t reflect immediately, so as ling as you have made a valid move the piece will be moved and once the computers response is decided the pieces are moved.

# Methodology

The game works as the following:

* if it is the humans turn: the human selects the piece and makes a move, the mouse listener interfaces identifies the mouse pressed and released and then based on which the decides the move the human intends to make. Once the move is decided the code generates the list of possible moves and checks if this move is in the list of the possible moves, if it then it makes the move, else it waits for the user to make another move.
* if it is the computers turn: the computer calls the AlphaBeta function(explained later), which then identifies the best possible move given the current board based on the evaluation function on the possible moves at the depth of 4. It then finally returns a move which is made.

# Alpha Beta Pruning

The AlphaBeta method of the chess class is the function that is used to identify the next move the that computer should make. The function is called when either the computer starts the game or when the human has successfully made a valid move.

The function is a recursive function with the base case as if the depth is zero or the list of possible moves is zero, it returns the move followed by its evaluation value. Hence when the tree reaches the bottom it is evaluation value is found out.

Apart from the base case the function first changes the player to the opposite of the current. It then makes a move and calls itself to identify the value of the evaluation for that moves tree. Based on this estimation made by the this moves tree, it updates the values of the alpha or beta based on the player it is on in the stack, if it is the computer then it updates the beta whereas when it is the choosing for the human it updates the alpha. After updating it checks if the alpha has grown bigger that beta, if so then it returns the move depending on the player.

This keeps on going till the depth of 4, that is currently the maxDepth till which the program looks forward to. If we change this depth the following data is observed:

# Evaluation Function

The evaluation function that determines the rating of the current board position is based on the following 4 factors:

1. Evaluate the piece values: Based on the remaining number of pieces on the board for a particular board, value is assigned to each board. The values are assigned based on the following table.

|  |  |
| --- | --- |
| Pawn | 100 |
| Knight | 300 |
| Queen | 900 |
| Rook | 500 |
| Bishop (if one) | 250 |
| Bishop (if two) | 300 |

The bishops are more valuable if both exist as one moves on the black tile whereas the other on white tile, hence individual existence of them is less valuable.

1. Evaluate the moves available: Based on the number of moves available in a board, points are allotted i.e. if more moves are available then more points are given to a particular board as it reflects the ability to move at different places. Whereas if the number of moves is zero, it means that the board is either in checkmate or stalemate, so if that is the case a high negative value is assigned to show that it is not an ideal situation for the current player.
2. Evaluate the safety: The safety of a particular piece can be easily checked and based on which negative points are assigned to the board if a particular piece maybe under attack. The points for the same are assigned based on the following if a particular piece is not safe:

|  |  |
| --- | --- |
| Pawn | -64 |
| Knight | -300 |
| Queen | -900 |
| Rook | -500 |
| Bishop | -300 |
| King | -200 |

1. Evaluation based on the position tables:

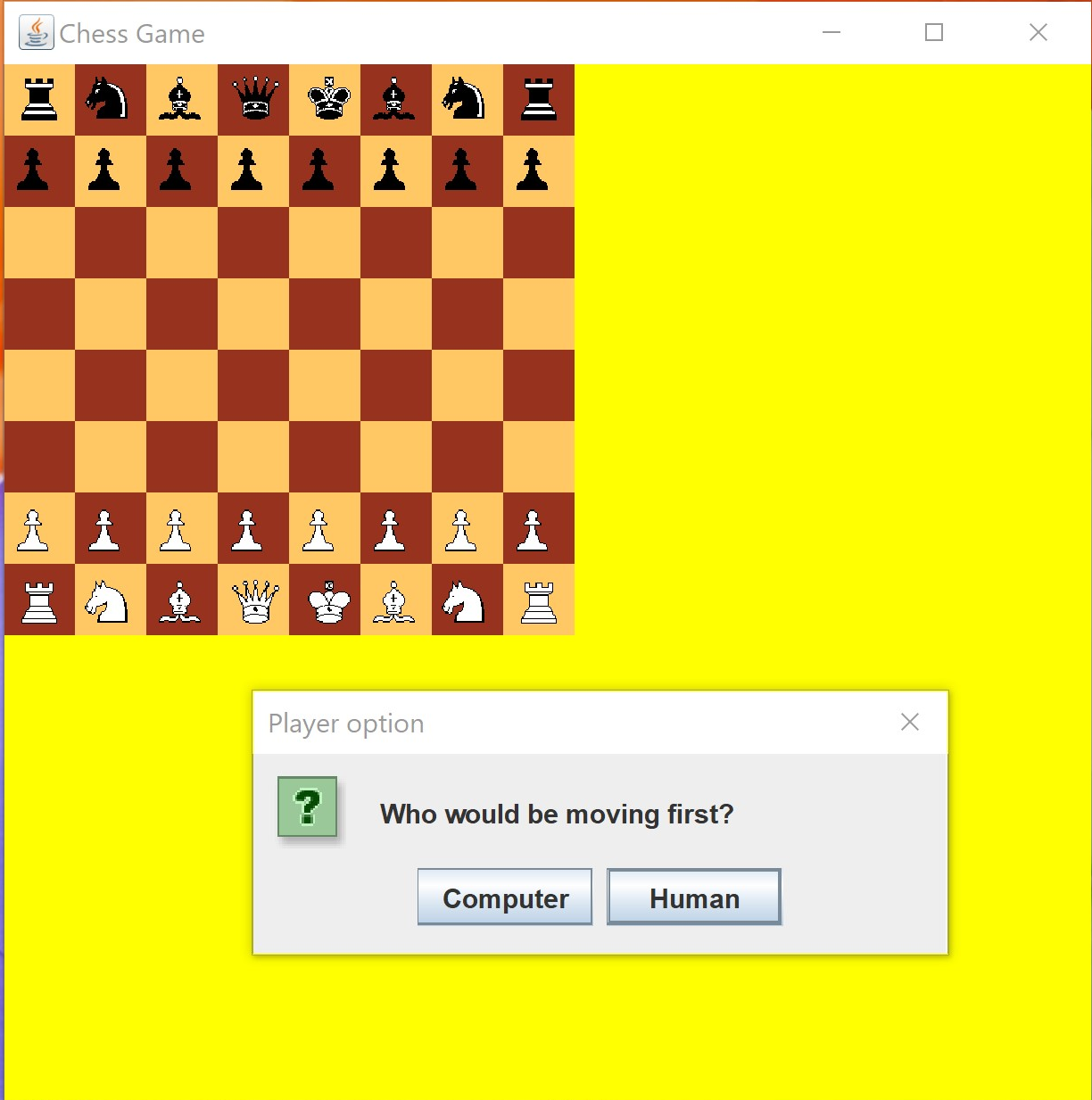
Based on the current position of a piece, the following position tables allot points to it, they have been designed keeping in mind the future possible moves and the way they are advantageous to the next moves or saving the king.

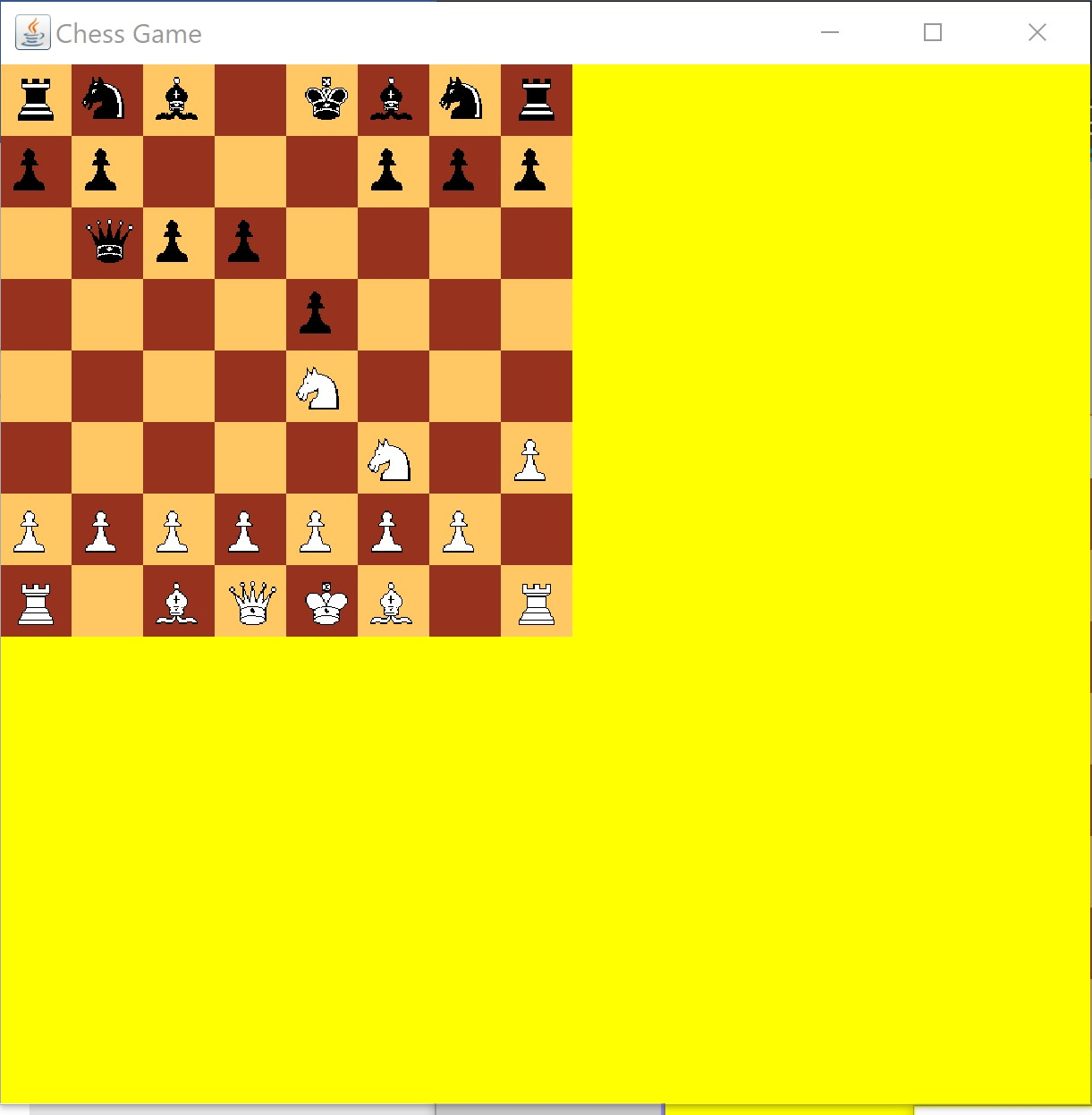
|  |  |
| --- | --- |
| Pawn |  |
| Knight |  |
| Queen |  |
| Rook |  |
| Bishop |  |
| King (middle move) |  |
| King (end move) |  |

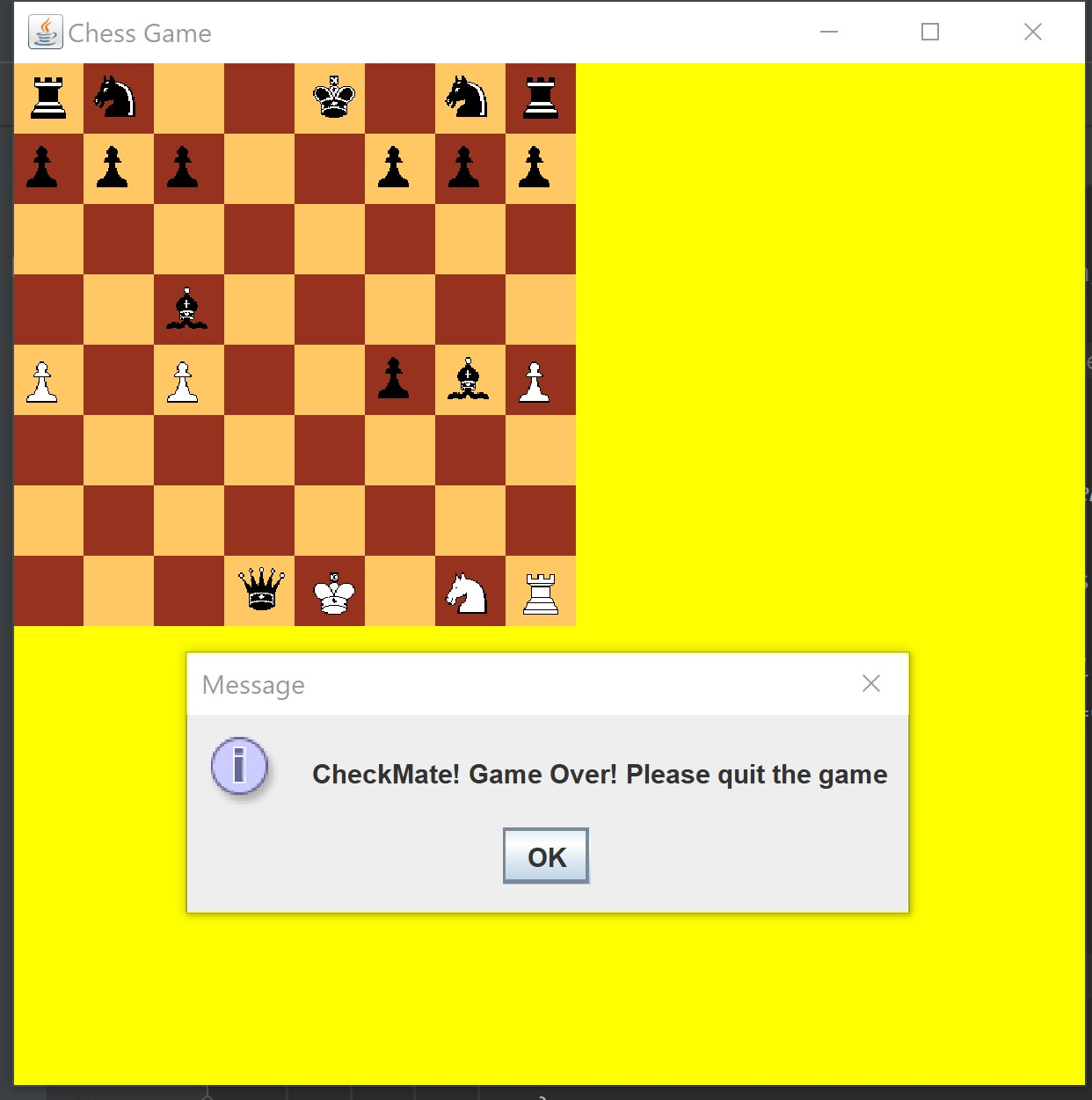
The king end and middle move is decided based on the number of pieces left on the board.

The values that are returned are added for the current player and then subtracted from the perspective of the other play the get the actual value of the board. Also, in the alphabeta function based on which players move it is the algorithm changes the value to positive or negative.

# Sample game play pictures:







NOTES:

1. The terminal at the same of the game would keep on displaying the list of possible moves for the human.
2. It takes a little time for the move to take please as it waits for the computer to decide it and make the move and moves take place concurrently.
3. To run the program:
   1. Extract Chess.zip
   2. Open IntelliJ and locate this file and open it.
   3. Right click on the Chess.java file located in the src and select Run.
   4. The file should run now.