Java Programming
Project: Chess
2019

Submitted to Nathalie Dittrich

By:

Konka Yamini(119485)

Afshin Ghasemi (120521)

Abstract

The project implements a Chess with a Graphical User Interface. The Chess game follows the basic rules of chess, and all the chess pieces only move according to valid moves for that piece. Our implementation of Chess is for two human players (no Artificial Intelligence).

Introduction

This project implements a Chess using Java language and a Graphical User Interface (GUI). It is played on an 8x8 checkered board with vision of right (White) and left (Black) coins. The Game design for two components in the same board.

1) ***Graphical User Interface***

For the Graphical User Interface (GUI) we use JFrame, This GUI was very easy to use to design, since we can use the position of clicked and release points.

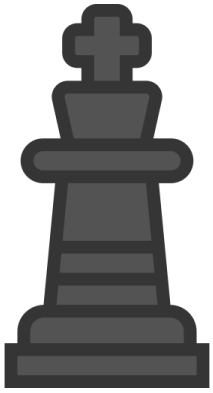
*2) **Playing** Chess*

After the GUI window appears upon execution, the first player clicks on the piece that player would like to move and then clicks on a valid position for this piece (Invalid moves are not allowed).

A player is also not allowed to move his or her coins two times in a row.

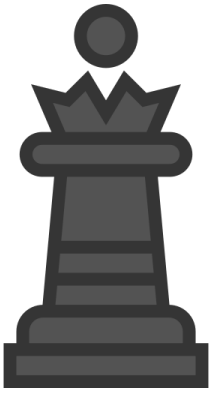
If the king becomes in danger the color of threatening's figure's position will change.

3) The Pieces and the Rules



The KING:

The King is the most important piece. When it is trapped so it cannot move without being captured, then the game is lost. This trap is called checkmate. The King can move one square in any direction. A King can never move into check, or onto a square where it can be captured by an opponent's piece. If a King is not in check, and no other legal move is possible, then the position is said to be in stalemate. A stalemated game is a draw, or a tie.

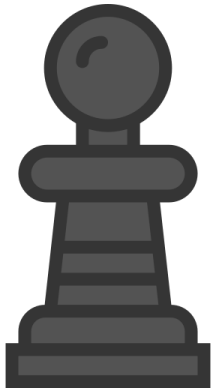


The QUEEN:

The Queen can move to any square in any direction as long as her path is not blocked. Her range and the ability is very high to attack many pieces.

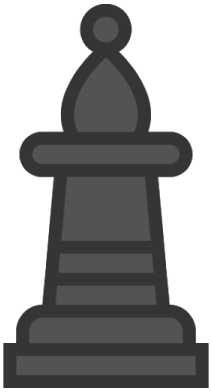
The ROOK:

The Rook can move to any square along its file or row as long as its path is not blocked. Its range is the source of its power.



The BISHOP:

The Bishop can move to any square along its diagonals as long as its path is not blocked. Its range is the source of its power.



The KNIGHT:

The Knight is the only piece that can hop over other pieces in an L-shaped path. This ability makes it particularly powe3rful in the early stage of a game when the board is crowded with pieces.

The PAWN:

The Pawn may move only one square forward if its path is not blocked. However, it may move as an option one or two squares forward on its first move only. It may capture only diagonally

one square. It may not capture forward. It may not move backward. The lowly Pawn usually does not last long, but if it is able to reach the 8th row or rank, then it can promote itself to any other piece except the King. A Pawn thus promoted is replaced by that piece. Therefore, it is possible to have more than one Queen, or two Rooks, Bishops, or Knights on the board at one time.

4) How this chess had been programmed

As we mentioned the program uses JFrame as an interface and we implement all attributes and methods in this main class. Now we explain methods which we used.

getXis: This method will get the position of row(x).

getYxis: This method will get the position of column(y).

setXis: This method will set the position of row(x).

setYxis: This method will set the position of column(y).

getXking_white: This method will get the position of King's row(x).

getYking_white: This method will get the position of King's column(y).

setXking_white: This method will set the position of King's row(x).

setYking_white: This method will set the position of King's column(y).

getXking_black: This method will get the position of King's row(x).

getYking_black: This method will get the position of King's column(y).

setXking_black: This method will set the position of King's row(x).

setYking_black: This method will set the position of King's column(y).

getXknight1_white: This method will get the position of knight1’s row(x).

getYknight1_white: This method will get the position of knight1’s column(y).

setXknight1_white: This method will set the position of knight1’s row(x).

setYknight1_white: This method will set the position of knight1’s column(y).

getXknight1_black: This method will get the position of knight1’s row(x).

getYknight1_black: This method will get the position of knight1’s column(y).

setXknight1_black: This method will set the position of knight1’s row(x).

setYknight1_black: This method will set the position of knight1’s column(y).

getXknight2_white: This method will get the position of knight2’s row(x).

getYknight2_white: This method will get the position of knight2’s column(y).

setXknight2_white: This method will set the position of knight2’s row(x).

setYknight2_white: This method will set the position of knight2’s column(y).

getXknight2_black: This method will get the position of knight2’s row(x).

getYknight2_black: This method will get the position of knight2’s column(y).

setXknight2_black: This method will set the position of knight2’s row(x).

setYknight2_black: This method will set the position of knight2’s column(y).

getPreX: This method will get the previous position of figure’s row(x).

getPreY: This method will get the previous position of figure’s column(y).

setPreX: This method will set the previous position of figure’s row(x).

setPreY: This method will set the previous position of figure’s column(y).

getXturn: This method will get the updated position of figure’s row(x).??

getYturn: This method will get the updated position of figure’s column(y).??

setXturn: This method will set the updated position of figure’s row(x).??

setYturn: This method will set the updated position of figure's column(y).??

initUi: This method will make a checkboard's squares.

initPieces: This method will create pieces and add them to LinkedHashMap.

isIconThere: This method will check whether there is any figure in the selected square or not.

isIconInWay: This method related just to Queen and will??

isBlack: This method will check whether the selected figure is black or not.

checksFigure:

noBackStep: This method will check pawn which doesn't lets them go backward.

check: This method will test the possibility of existence of check in chess and changing the square color.

direction_1: this is for the left parallel direction of the piece, i.e y is same

direction_2: this is for the right parallel direction of the piece, i.e y is same

direction_3: this is for the upward movement, i.e x is same

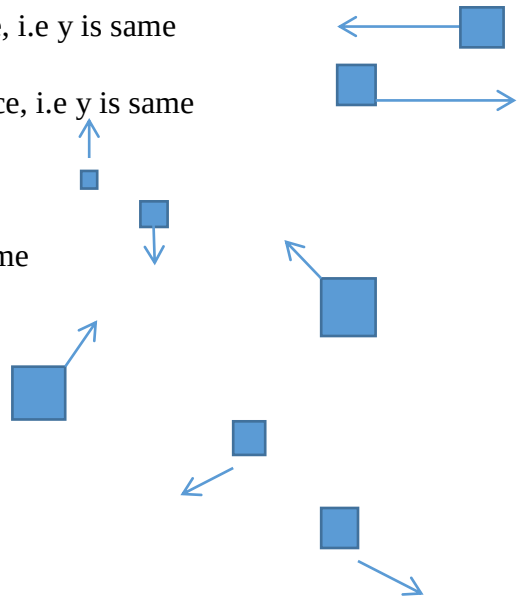
direction_4: this is for the downward movement, i.e x is same

direction_5: this is for the left top corner movement

direction_6: this is for the right top corner movement

direction_7: this is for the left bottom corner movement

direction_8: this is for the right bottom corner movement



pointingToCheck: this method to checking the given square has been threatned by any piece

checkingSquareForKing: for the checkmate condition.

movePiece: This method will do all the possible move for the selected figure according to rules.

CLASSES:

In this project we have totally 6 separate classes for the each individual pieces, they are named as Knight class, Pawns class, Queen class, King class, Rook class and Bishop class.

In these class we have a common method is move the respected piece and with conditions.