Chess

chess uwu. we live in a society.

Team Name: Some Asian Kid
Team members: Johnny Zhang, Nathan Gu, Albert Yu
Revision Date: 5/18/21

Program Purpose:
The purpose of the program is to display a two player chess game that takes input from a text line and selects pieces from left clicking(You can play the game by yourself if you're lonely or just trying to experience how to defend your own moves and how to counter the counter). The program appeals to anyone who plays chess, as it provides entertainment and practice for both chess players.

Instructions:
The two players will play on the same device.
Users will be able to input moves on a text line. First, the white player will select a piece to move and then the piece selected will appear on the top right corner. Upon selecting the piece, an auto generated line of text appears in the text field on the top left corner of the GUI to prompt for a move. Only enter the position you want to move to(eg. c4). For example, typing "e4" will move the piece at its current position to e4.

Target User Profile:
Our target user profile contains a range of people from avid chess players to people who wish to practice playing with another opponent. We want to expand our target user profile as we further improve the program's usefulness and quality by hooking in people who never play or show any interest in chess.

Feature List:
Scuffed chess(will add checks, checkmates, and pins if we have extra time), there's a nice looking chess board GUI and pieces. The classes use mouseListener class to allow the users to select the pieces to move in a much easier way rather than entering old position and new position in a single text field. Because we used a HashMap, we don't need to implement a system to remove the piece after it is taken. Each of the individual classes with the piece names all contain a method called isLegal that checks whether or not the move is legal in terms of chess. We made the GUI resizable so that we can accompany users who run this program on smaller screens to have the chess board well sized, making it easier for them to play. We also have a class to convert the HashMap to a matrix for each of the individual classes to see where the other peices are in order to determine whether or not a move is legal. WE ALSO HAVE CUSTOMIZABLE ASSETS!!!

Instructions:
Download the java files and initialize the package. Also, add the .pngs and .jpg files for the chess pieces and board. Upon completing these tasks, run the program from the board class and you will

see a chess board with the pieces. If you feel the need to adjust the chess board due to a small screen, resize the GUI into a more comfortable area or position; you can also run this on a monitor. Finally, you can move your pieces accordingly based on how your opponent moves in chess. Click on a piece and then enter just the location where you wish to move the piece. You will see what piece you selected on the top left corner. Enter your moves through the text field(Next step White Queen "e4") and have fun :).

Class List:
Board, Piece, ChessPanel, MyPanel, PieceImg, Pawn, Rook, Knight, Bishop, King, Queen, MatBoard

UML:
(will be attached with the readme)

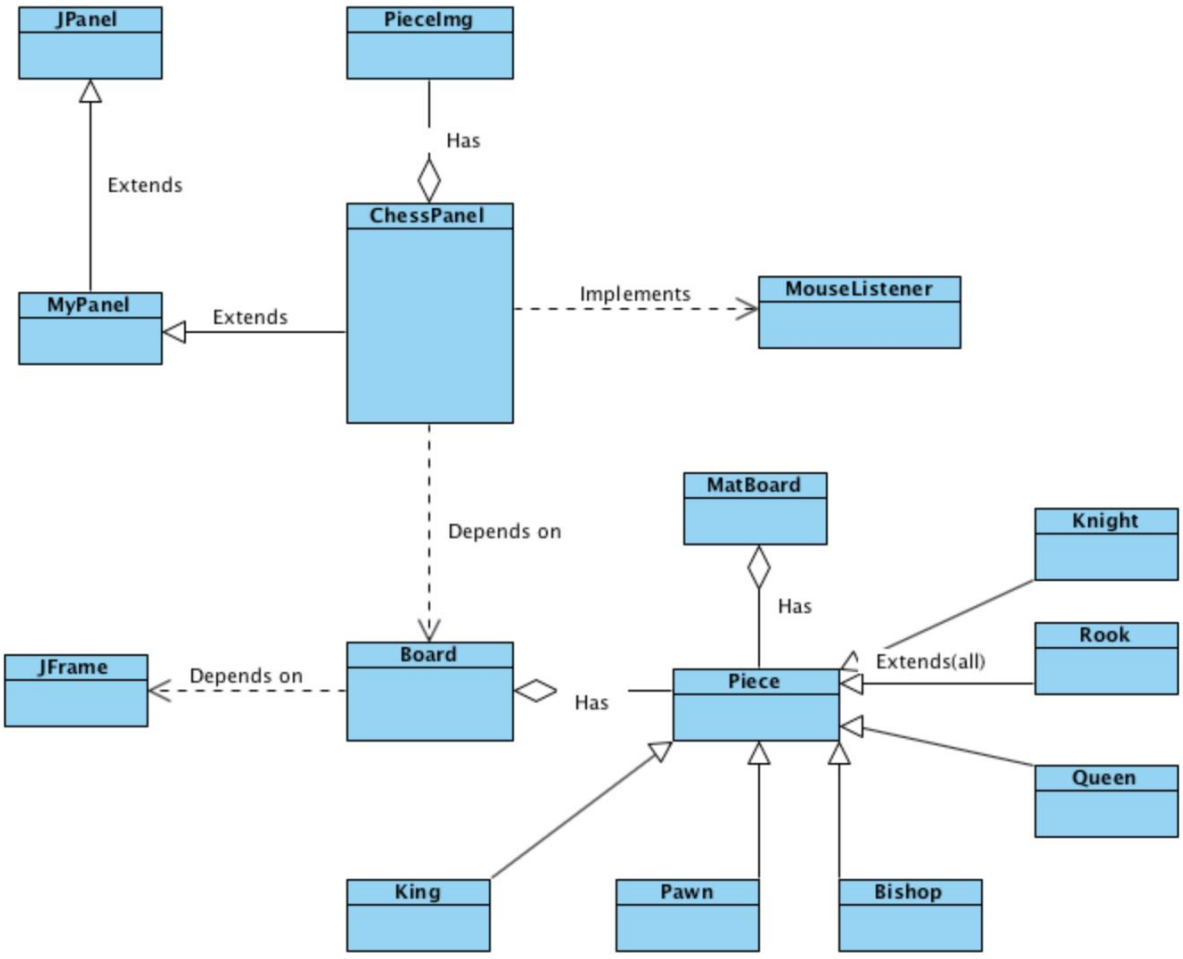
Team Responsibilities:
Try and stay ahead of guidelines as much as possible and make sure to complete your class(don't procrastinate please). Ask for help and help others. Make sure to share the most recent version of the code on GitHub as well so we can update the UML and readme :). Johnny will do the code's "skeleton," Albert will do graphics, and Nathan will do UML, readme and anything else that needs doing. (In the end, Nathan created a "skeleton" of all the classes but the graphics related, while Johnny writes the methods for the individual pieces(king, knight, pawn, etc..). Albert's role remained the same). We will meet at least once a week from now on to check progress and to work on/ practice the presentation.

Known Bugs/Workarounds:
Classes compile with each other now. If you select a piece but don't enter a location and press enter, the piece will delete itself. When you launch the program from the board and the screen size is too small, the chess board will be too big sometimes and the pieces are out of place. Some QOL would be nice, such as highlighting which piece to move, adding a win screen and telling the user that a move is illegal. Also, en passant. God i hate en passant. It's exactly the kind of random move you would come up with on the spot to improve your position. It's also difficult to implement, so we left it out.

Key learnings:
JPanel/JFrame usage, Arrays, Matrix, Hashmaps, MouseListener, Constructors, Conditions, Strings, getting images, Fields, relationship between classes, better UML creation(specifically class level with the different relations between all the classes), and how to collaborate well.

Credit List:
https://codereview.stackexchange.com/questions/71790/ , Albert's father, Nathan's father, red bull.

UML DIAGRAM:


JPanel
PieceImg
Has
Extends
ChessPanel
MyPanel
Extends
Implements
MouseListener
Depends on
MatBoard
Knight
Has
Rook
JFrame
Depends on
Board
Has
Piece
Extends(all)
Queen
King
Pawn
Bishop