COEN 275: Object Oriented Analysis, Design, and Programming
Winter 2017
Individual Assignment 1
Due: February 12th, @11:59 PM

Assignment Overview:
The objective of this assignment is to create a program to read a file containing chess moves, and display those moves to the user in a GUI. The GUI should provide the user with controls to perform or undo one move at a time.

Move File:
A file containing all the moves of a chess game will be provided. Your program may not modify this file. The provided file will contain one pair of moves per line. The exception to this rule is when White wins the game before Black's turn, in which case only one move will appear on the line. The pair of moves describe White's action first, then Black's action. Each move consists of a start coordinate, and an end coordinate for the moving piece separated by a dash.

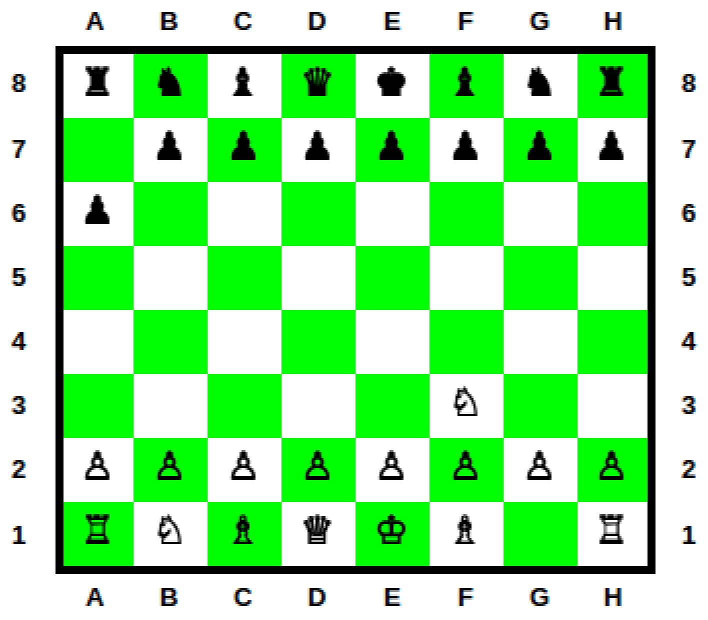
All coordinates contained in the file will conform to the convention of chess piece coordinates shown below:

	A	B	C	D	E	F	G	H	
8	A8	B8	C8	D8	E8	F8	G8	H8	8
7	A7	B7	C7	D7	E7	F7	G7	H7	7
6	A6	B6	C6	D6	E6	F6	G6	H6	6
5	A5	B5	C5	D5	E5	F5	G5	H5	5
4	A4	B4	C4	D4	E4	F4	G4	H4	4
3	A3	B3	C3	D3	E3	F3	G3	H3	3
2	A2	B2	C2	D2	E2	F2	G2	H2	2
1	A1	B1	C1	D1	E1	F1	G1	H1	1
	A	B	C	D	E	F	G	H	

Example:
The following line from the moves list:

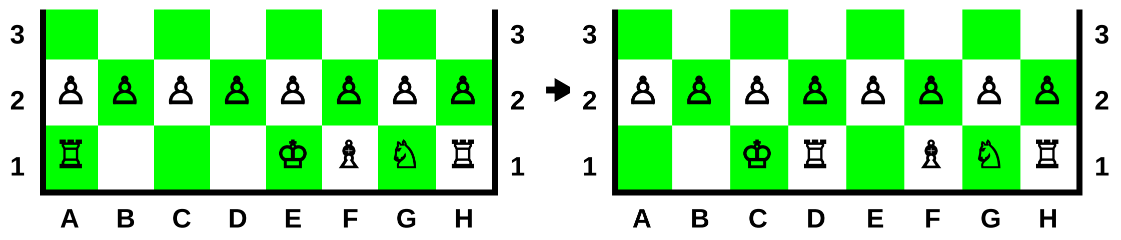
G1-F3 A7-A6

corresponds to the following actions:



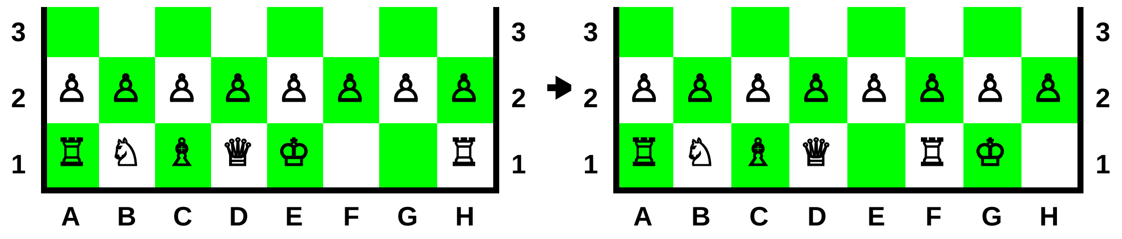
Special Cases:
Chess contains a couple of special cases that the program must address. Each of these special cases will have their own designation ***which will replace*** their coordinate pair described above.

Queen Side Castling: O-O-O
If there are no pieces between the king and rook on the Queen's side of the board, then a special move called Castling may occur. In this move, the Rook AND the King move towards each other until the Rook ends up to the right of the King, as shown below:



This move will be designated with the special coordinate "O-O-O" (Letter O not Number 0)

<u>King Side Castling: O-O</u>
Similarly Castling can occur on the King's side with the Rook on the King's left.



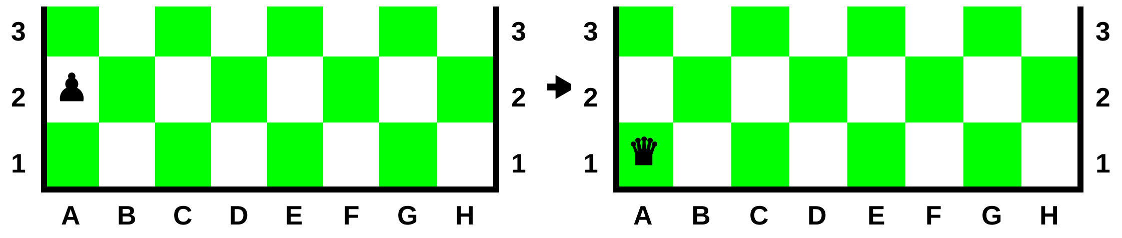
To distinguish a King's side Castle from a Queen's side Castle, only two O's will be used: O-O

<u>Pawn Promotion: XY-XY(P)</u>
If a Pawn manages to move from its side of the board to the opposite edge of the board, it is eligible for a promotion to another chess piece (Queen, Knight, Bishop, or Rook). The designation of a promotion will be the coordinates of the move, as usual, followed by a character in parenthesis designating which piece the pawn will promote to.

Example:
The move

A2-A1(Q)

Implies a Black Pawn being promoted to a Queen.

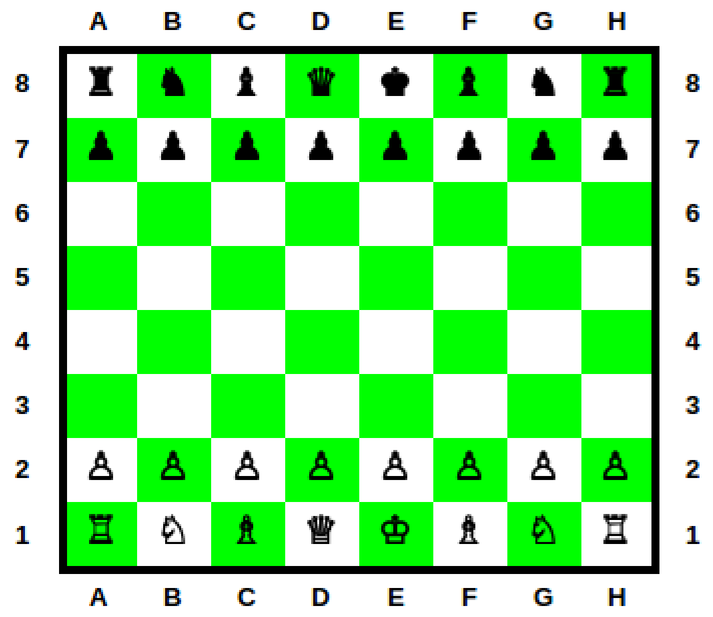


The following characters will be used to represent promotions:

Q = Queen Promotion
N = Knight Promotion
B = Bishop Promotion
R = Rook Promotion

GUI Minimum Requirements:
The GUI must display the current state of the chessboard. Images for each Piece will be provided. The chessboard must be set with White on the bottom, and Black on top. The Checkered pattern should be placed such that the square the queen starts on's color matches

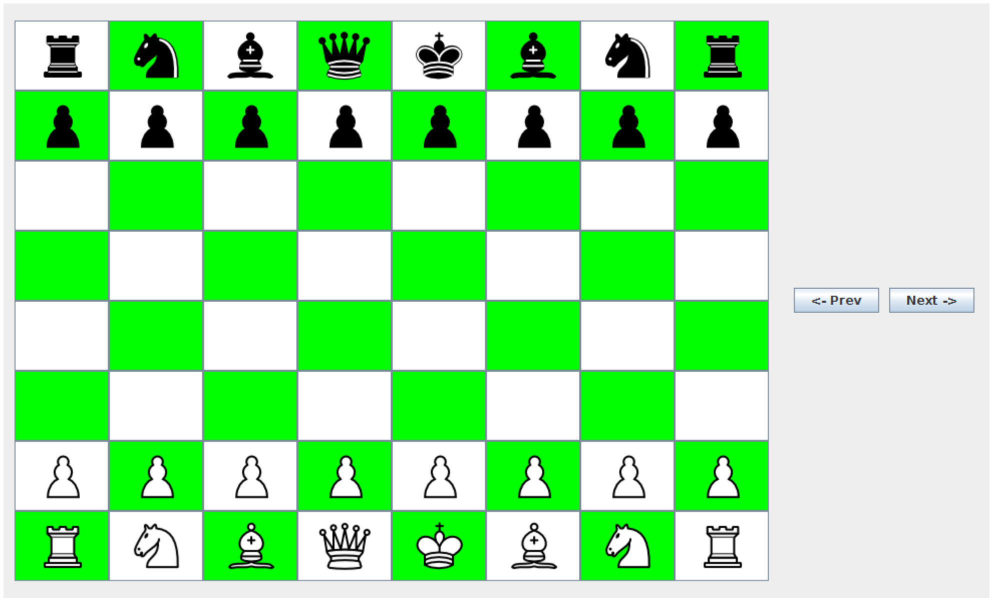
her own. White Queen on light colored Square. Black Queen on dark colored square. You may color the chessboard however you like. The starting board should resemble the board shown below



In addition, the GUI must contain two buttons, clearly marked, that allows the user to progress to the next move, or go back to the previous move.

Your GUI ***may not*** contain a dedicated “Refresh” button. The trigger for refreshing the GUI must be hidden from the user.

At minimum your gui should look like this:



Code Requirements:
Your program must be able to load the movelist form a provided. Hard coding the location of the file is acceptable.

Your code must adhere to proper Java Style. You are welcome to use a Java Style you are comfortable with, as long as you are ***consistent*** in your code. If you do not have a Java Style, a good starting point is the Google Java Style Guide. Ample commenting is welcome and encouraged. As a general rule of thumb, someone reading your code should be able to determine the function of any method within 30 seconds of reading it.

Submission Guidelines:
Your submission must contain the following files:

- Java Source Files
- Combined Source File
- README.txt
- Any other files required to run your program not listed above

Combined Source Files:
You must include with your submission a file that contains the contents of all your source files concatenated together. The file should be named firstInitialLastName.txt

README.txt File:
You must also include with your submission a README file. This file is used to help debug setup related issues. The README file should use the following format:

Name: <Your first and last name>
JDK Used: <JDK Provider> <Version>
IDE Used: <IDE and Version>
Main File: <Filename of file that should be run>
Load File: <Instructions on loading in the move file, including the filename of where you hardcoded the location if applicable.>
Other Instructions: <Optional special instructions specific to your program>

All files EXCEPT the Combined Source file should be compressed into a .zip file. This file should be named firstInitialLastName.zip. There will be two Assignments created in Camino, one for your .zip file, one for your Combined Source file. Submit both the .zip file and the Combined Source File to their respective DropBoxes.

Grade Breakdown:

Functionality	60%
GUI	30%
Style	10%

Miscellaneous:

- Your program does not have the check the legality of the move based off of the rules of Chess.
- You may assume that the input file conforms to the definition laid out. Ie. The input file will not contain malformed data.
- Hint: An easy way to create the chessboard is by using buttons. Which can have a background color and an icon.