COS 161 – Algorithms in Programming

Project 02 – Chess

Objectives

The objective of this assignment is to become familiar with inheritance and extending classes. You will be creating a program that allows the user to play chess.

General Instructions for all Assignments

For each assignment you will write a Java program and test it. Start your programs with a comment block such as:

```
/*
   NAME: <your name>
   COS 161, Fall 2021, Prof. <instructor name>
   Project ##
   File Name: CLASS_NAME.java
*/
```

Your programs should be neatly formatted, and follow the indenting, formatting, spacing, and commenting practices taught in class.

Read each assignment carefully to see what is required. If you do not understand something in the assignment, ask a tutor or the instructor. You will get partial credit if you finish only part of an assignment or it is not working correctly. Turn it in and explain what you completed and what issues it has. It is usually better to turn in an imperfect assignment on the due date, rather than falling behind in the course.

Part 1 (5 points) Look at Provided Files

For this project, we are going to attempt to recreate the age-old game of Chess. If you are unfamiliar with the game, check out the following Wikipedia article: https://en.wikipedia.org/wiki/Chess

The first part of this project just requires you to look through the provided files (Chess.java is the main client program). Make sure you have

a basic understanding of what each part does. Run the program and add a capture of the output to your submission document.

Part 2 (15 Points) Create Array of Pieces and Populate the Board

Create an array of ChessPieces called pieces (you can use what is in my comment in Main of Chess.java). Fill this array with new ChessPiece objects (ignore the King class for now). Set each of the pieces to have the correct initial rank, file, and type values. Use the provided initialized board to help with what piece goes where.

Modify the drawBoard method to draw each piece in the pieces array. Make sure it uses the values in each piece instead of hardcoded values. Look at initializeBoard to give you an idea of what to do. Once drawBoard also draws the pieces, completely remove the inittializeBoard method, and replace its call inside the Main method with a displayBoard() call.

Part 3 (40 Points) Extend ChessPiece

Next, you must extend the ChessPiece class for each unique chess piece. That means you will have a separate child class for King (incomplete example provided), Queen, Bishop, Knight, Rook and Pawn.

You must override the isValidMove() method for each piece class you extend from ChessPiece. Make sure you conform to the restrictions of movement for each piece in the chess rules. The Wikipedia article can help with this.

You can skip special moves like castling, a pawn capturing en passant, and promotion of pawns.

It is highly recommended that you develop an algorithm for searching all pieces in the array for particular rank and file attributes. Doing this efficiently will be necessary in some capacity for every piece, so spend some time really thinking through a good way to tackle it.

Important to remember, there is movement that is valid for a piece that is invalid because it requires it to move through another piece (excluding the knight of course). However, it is valid to move to a space and capture

a piece (if it is of the opposite color).

Part 4 (20 Points)

Next, you must create a method like the one in Project 01 that allow two players to play chess. The method will be called takeTurn, which must do the following:

- Take a parameter of which player's turn it is (boolean for black or white).
- Take in rank and file for the piece that is to be moved.
- Take in the rank and file of where the piece is being moved to.
- Call isValidMove() and then update the game board if it is. Otherwise prompt the user to try again (this can be handled in Main if that is easier)
- Make sure if a piece is killed, the isAlive property is changed.

Once takeTurn is complete, add logic to the Main method in Chess.java that will alternate between players until one of them wins. For this project, the win condition can be simplified into just checking if both kings are still alive for this project.

Once one king is destroyed, add a println statement saying which color wins.

Extra Credit (10 points) Special Moves

Add functionality for Castling and Promotion (for the rook and pawn respectively). Once again check the Wikipedia page for explanations of what these are.

Extra Extra Credit (10 points) En Passant

Add the ability for a pawn to capture en passant.

What to Turn In

Turn in a solution document as a single pdf file. This solution document should contain the following (make sure formatting is clear and use comments!):

- The takeTurn() method and isValidMove() method from your Knight class.
- At least two screenshots of console input of game being played.
- At least three screenshots of the game board DrawingPanel.
- A screenshot of the game being won by wither player.
- If you completed the extra credit(s), include 3 additional screenshots, one from the console output and two from the DrawingPanel. Make sure these are clearly marked "Extra Credit".

Once you have the solution document ready, submit it as an attachment on Brightspace. Be sure to also attach all the .java files you created or edited in this project. Do not zip or compress the files, submissions with archives of any type will be ignored.