Alexandria University
Faculty of Engineering
Computer and Communication Engineering
Specialized Scientific Programs

CC272
Programming II
Fall 2023 / 2024

Lab 7: Chess game

You are required to design an object-oriented model for a chess game. This includes the game board, board square, pieces, and any other things you find helpful for your design.
This all should be implemented in a package named **`ChessCore`**.
Note that your implementation in **`ChessCore`** must be generic, meaning that it should be reusable easily. **Imagine** that you want the same **`ChessCore`** package to work whether you'll implement the game as a console application, a desktop GUI application, or even a web application.
This does **NOT** mean you'll implement all that, it only means ChessCore is only about modelling and implementing the game logic without it knowing about *how* it will be used.
For example, you shouldn't attempt to do any prints or take any user inputs in **`ChessCore`**. This is the responsibility of the caller (outside ChessCore package).

Following is a basic set of classes you may want to use. You may need to add more classes or modify the given ones to improve the overall design and follow the object oriented principles.

1) **`Piece:`**
- This will be an abstract class representing a chess piece.
- Specific pieces will inherit from it, e.g, Pawn, Bishop, Knight, Rook, Queen, and King.
- The Piece class will need to have a method **`isValidMove`**

2) **`ChessGame:`**
- This is the core class representing the chess game with all the needed state for representing the game.
- It should have **`isValidMove`** method.
- It should also have **`getAllValidMovesFromSquare`** method, which given a specific square, returns all valid moves that can be done from this square. The square must be having a piece that belongs to the player who has the turn to play. Otherwise, there are no valid moves for that square.

3) Implement more classes and/or interfaces as needed to make the design better.

4) You may need to use Java enumerations. You can read that here:
https://docs.oracle.com/javase/tutorial/java/javaOO/enum.html

Dr. Layla Abou-Hadeed

Eng. Ahmed El-Sayed
Eng. Tarek Salah
Eng. Ahmed Ashraf
Eng. Youssef Victor
Eng. Mazen Sallam
Eng. Mostafa Ibrahim
Eng. Hania Yasser
Eng. Omar Hossam

Requirements:

1) You are required to implement the chess logic correctly, including moves like "en passant", castle, and pawn promotion.

To know more about "en passant" move: https://www.youtube.com/watch?v=0I-PAaAMmSA
To know more about "castle" move: https://www.youtube.com/watch?v=4dLZ5bv2tn8

2) The game should be marked as draw if there is insufficient material. See https://support.chess.com/article/128-what-does-insufficient-mating-material-mean

3) You are required to be able to load the game from a text file where each line represents a move. The format of each line is **`SOURCE_SQUARE,DESTINATION_SQUARE`**, for example, **E2,E4**. The promotion move will have a different format, which is **`SOURCE_SQUARE,DESTINATION_SQUARE,PROMOTE_TO`**, note that PROMOTE_TO will be one of the following values: K, B, R, Q representing Knight, Bishop, Rook, Queen, respectively.

4) You should always make sure that the game doesn't get into invalid state.

5) You should communicate any errors via **Exceptions**

6) You are required to write a main method that reads a text file named **"ChessGame.txt"**
When opening the file, **DON'T** use full path, you **MUST** use **ChessGame.txt** exactly, e.g, `new Scanner(new File("ChessGame.txt"))`

7) After reading each line, print exactly the following, **in the same order, and exactly as shown**:

- If the move is castling, print "**Castle**"
- If the move is en-passant, print "**Enpassant**"
- If the move captures a piece, print "**Captured PIECE_NAME**" where PIECE_NAME is one of Pawn, Bishop, Queen, Rook, Knight.
- If the move causes a king to be in check, print "**White in check**" or "**Black in check**" (except if it's a checkmate)
- If the move causes white to win, print "**White Won"**
- If the move causes black to win, print **"Black Won"**
- If the move ends the game with insufficient material, print "**Insufficient Material**"
- If the move causes a stalemate, print "**Stalemate**"
- If the move is invalid because the game has already ended (i.e, white won, black won, stalemate, insufficient material), print "**Game already ended**"
- If the move is invalid while the game is in progress, print "**Invalid move**"

Note that it's possible for a move to satisfy multiple of the above stated conditions. In this case, your prints **MUST** be in the same order as stated above.

- Promotion (all + put new queen mesh shaghala)
- Reading elpromotion
- Reading from file
- Saving to file
- Order of saving to file - LOL

Exceptions done

Required:

1. Implement the ChessCore package per the specified requirements above. The implementation should be in a clean object-oriented design. You will deliver that online through a Google Form that will be available for you.
2. The due time is 18 November 2023 at 10:00 AM
3. A discussion is made with you at your lab time next week on what you have delivered.

What to be delivered:

- On the Google Form, you should deliver a zipped file that contains the .java files.
- Your zip file should be named as id1_id2_groupNumber1_groupNumber2. For example, 4678_4679_G2_G4.

Policies:

- You should work in teams of two.
- Delivering a copy will be severely penalized for both parties, so delivering nothing is so much better than delivering a copy.
- No late submission is allowed