ChessGame. Informartion.

Board

I made by creating a grid of square panels, adding appropriate borders and labels, and placing the initial pieces in their starting positions. The board is assumed to be an 8x8 grid with columns labeled a-h and rows labeled 1-8. Each square panel is a JPanel with a background color alternating between white and light gray to create a checkerboard pattern.

Pieces

The Piece class is an abstract base class representing a chess piece in a chess game. It provides the common structure and functionality for all chess pieces, such as type, color, and position (row and column). Each specific type of chess piece (e.g., King, Queen, Rook, etc.) extends this class and implements the required abstract methods.

PieceColor

An enumeration (enum) in Java, representing the colors of chess pieces: white (WHITE) and black (BLACK). This provides a simple and convenient way of representing the color of pieces, which simplifies the code and avoids errors related to using strings or numbers to denote color.

PieceType

Is an enumeration representing the different types of chess pieces in a chess game. The enumeration has six constants, each corresponding to a specific type of chess piece (e.g., PAWN: Represents a pawn. KNIGHT: Represents a knight, etc.). These constants are used to define and distinguish the different types of chess pieces in the game.

PiecePanel

Is an abstract class that extends JPanel and represents the visual representation of a chess piece in a chess game. It uses the Kitfox SVG library to draw the chess piece on the panel, allowing for scalable vector graphics (SVG) to be used for the chess piece icons. This abstract class is designed to be extended by specific chess piece classes that will provide the SVG file paths and the necessary implementations for the paintComponent method.

Move

The Move class can be used for searching for possible moves of both a specific piece and all the pieces on the board. By creating Move objects for each potential move, you can analyze and evaluate the legality and quality of these moves in the context of the game. This can be useful for implementing game logic, move validation, and even AI algorithms that aim to find the best move for a given position.

Opening

This is a simple class representing an opening in the game of chess. It has two fields: name, which is a String representing the name of the opening, and moves, which is a List of Strings representing the sequence of moves typically played in that opening. It has a constructor that initializes both fields and getter methods for both fields.

TextFieldDialog

The TextFieldDialog class is a custom dialog window in Java Swing that provides a text field for user input, along with buttons for saving or canceling the input. The class extends the JDialog class and includes a constructor that takes a JFrame object and a string as parameters. The string is used as the default text for the text field. The dialog also has a private instance variable savedText that stores the user input once the "Load" button is clicked. The class includes a public method getSavedText() that returns the savedText variable.

ChessGui

The ChessGui is main class which contains methods for handling user input, such as clicking on a chess piece and moving it to a new position. It also includes methods for updating the user interface to reflect the current state of the game, such as displaying the chess board and highlighting valid moves. Overall, the ChessGui class serves as the central hub of the chess program, coordinating the various components and providing a user-friendly interface for playing the game.

Game Logic and Functionality:

The game logic includes correct movements for all pieces, tracking of en passant, pawn promotion, and castling by monitoring movements of the king and rook. Additionally, animations have been added, and players can select which piece to promote their pawn to. The game also features a move duration chart showing the time taken by each player for their moves. Also, the last move played is displayed to show the opponent how their opponent played. Also, when a chess piece is clicked on, all possible moves for that piece are displayed.

Additional Functionality:

Clock. The game includes a feature where players have a set amount of time to make their moves, and an increment can be added to their time after each move. This feature is represented through clocks that display the remaining time for each player separately. If a player's time runs out, the game ends and the results are displayed. The clocks can be customized through the command-line arguments, with the **`<-h> [seconds]`** flag specifying the total time in seconds and the **`<-i> [seconds]`** flag specifying the increment in seconds. Alternatively, the clocks can be set through the game's general settings.

```
java -cp ./bin <mainclass> <-h> [seconds] <-i> [seconds]
```

CPU. It was implemented to have a CPU player for the black pieces, which randomly selects a valid move from all possible moves in the current position, including castling and en passant capture. Also, the CPU player is capable of randomly selecting the promoted piece (queen, knight, rook, or bishop) when its pawn reaches the opposite end of the board. Additionally, the player can switch back and forth between human and CPU player modes.

PNG_SVG Save. The ChessGui program has additional functionality for exporting the current state of the game in PNG and SVG formats. To export in PNG format, the user can click on a button or select the corresponding option in the menu, which will save the current state of the game as an image in PNG format. Similarly, the user can export the image in SVG format by selecting the appropriate option in the menu.

Promotion. In addition, the program has the ability to choose the type of piece when a pawn is promoted. When a pawn reaches the last row, the user is presented with the choice between promoting to a queen, rook, bishop, or knight.

Opening. The name of the opening position is also displayed in the program and is gradually refined as the game progresses.

FEN. Additionally, a game loading feature was added to ChessGui, which allows users to load a game using FEN notation. The FEN notation correctly sets up the board with the pieces in their correct positions, determines whether castling is possible, identifies whose turn it is to move, and keeps track of the number of half-moves.

For better code comprehension, the entire code has been commented using JavaDoc.

```
This method updates the size of the chess board based on the current size of the frame. It calculates the
minimum size of the board based on the width and height of the frame, subtracting 100 pixels from each
dimension to account for the size of the side panels. It then sets the bounds of the board panel and
animation layer to match the calculated size, and sets the preferred size of the center panel to match the
board size. Finally, it revalidates the center panel to ensure that the changes take effect.

1 usage
private void updateBoardSize() {...}


Determines if the specified color's king is in check.
Params: color – the color of the king to check
Returns: true if the king of the specified color is in check, false otherwise

7 usages
private boolean isInCheck(PieceColor color) {...}


This method sets up the chessboard based on a given FEN (Forsyth-Edwards Notation) string. It parses
the FEN string to create the appropriate pieces and sets their initial positions. It also sets up the castling
and en passant flags, and updates the current player. At the end of the method, it calls createBoard to
update the GUI with the new board state.
Params: fen – the FEN string representing the desired board state

1 usage
private void setBoardFromFEN(String fen) {...}
```