Developer Guide of I See Chess

Overview

I See Chess is a modular fully-featured chess game written in Java. Its design is so that the logic, aesthetics, panel layout and choice of user interface may be switched out or rewritten or extended without changing the code in another class and file.

Files:

`Piece.java`

The Piece class is an enum of the pieces on the board -- white pawn, black pawn, white queen, etc. Note that a white pawn and a black pawn are distinct from one another. Piece objets also include methods to naively find their possible legal moves and they load their icons into memory when constructed.

`Board.java`

The Board class represents a chess game at a specific point in time. A Board class contains a 2D array of Piece objects and has methods for manipulating pieces, checking whether the game is in a state of stalemate, whether either king is checked, and promoting pawns. These methods will throw exceptions as necessary in order to signal illegal moves, checkmates and the like. The Board class represents the logic of chess and can be used as a backend to any chess engine in conjunction with the Piece class.

`ChessPanel.java`

The ChessPanel class is an extension of JPanel that draws the chessboard and pieces on the screen. ChessPanel also handles frontend logic, i.e. whether a click indicates a piece is being selected or unselected or moved, etc. ChessPanel also opens dialog boxes upon events occurring, such as checkmates, stalemates and pawn promotions in order to allow the user to make decisions as necessary.

`GraveyardPanel.java`

The GraveyardPanel class is an extension of JPanel that indicates how many of each type of piece is missing from the board at the beginning of the game, similar to a “graveyard”. Note that when pawns are promoted, this means that the graveyard may count pieces down and even show negative numbers.

`DebugPanel.java`

The DebugPanel class is an extension of JPanel that gives helpful real-time debug information about the ChessPanel and to a certain extent, the Board. By default, this is disabled but can be enabled by setting the useDebugPanel flag to true in ISeeChess’s main method.

`ISeeChess.java`

The ISeeChess class is essentially a driver method / class for the entire game. It initializes the Board class, as well as sets up the panels in the correct layout and ensures communication between them.

`ISeeChessTest.java`

The ISeeChessTest class is a JUnit test class. It contains 16 tests that mostly are designed to unit-test the logic of the Board class.

Suggestions for Improvement

Most improvements in this game would be on the front side of the program, since the backend is very well-tested and functional. Possible improvements include:

- Better icons. This can be achieved by replacing the icons in the img folder. Ensure that icons have a transparent background so the background colour of the square is not drawn over.
- A title screen. This can be achieved by modifying ISeeChess.java to open a title screen prior to opening the rest of the game.
- Different modes of play. This can be achieved by modifying or extending ChessPanel to set timers, etc. Since this happens on a user rather than a game-logic level, changes should be made to the logic within the mouseClicked() listener.
- An AI to play against. Again, this should be achieved by creating another class that can output best moves given the state of the board and its history, and then calling it from a modified or extended ChessPanel to move a piece following every user move.
- Different colours on the board, for example, a different colour could indicate a move that captures a piece rather than a a move that does not. This would be done by a modification or an extension to ChessPanel.
- An undo option that returns the board to a previous state. Also would be achieved by a modification or extension to ChessPanel.

More Information

More information can be found in the Javadocs in the doc/javadoc directory.