CHESS GAME

Roll no.	Name	Registration no.
03	Hasil Haroon	230953019
16	Yashraj Sakunde	230953172
17	Rishit Kumar	230953174

Introdution :-

Chess is a two-player strategy board game played on an 8x8 grid. Each player controls 16 pieces: one king, one queen, two rooks, two knights, two bishops, and eight pawns. The objective is to checkmate the opponent's king, placing it under an inescapable threat of capture. Each piece has specific movement rules: for example, pawns move forward but capture diagonally, while knights move in an "L" shape. Chess involves strategic thinking, tactics, and planning, with players balancing offensive moves, defensive positioning, and long-term strategy to control the board.

Language Framework :-

We used Java as the programming language and Java Swing as the GUI framework. Java is the primary language for the logic and functionality, while Java Swing provides the graphical interface components that enable visual elements like the chessboard, pieces, and interactive features. Java Swing is a part of Java's standard library, designed specifically for building GUI applications across platforms.

Mechanics :-

For the mechanics in our Java Swing chess program, we included:

1. **Piece Movement**: Implemented unique movement rules for each piece:

    Pawn: Moves forward one square (two on first move); captures diagonally. Special moves: *en passant* and promotion. Rook: Moves any number of squares horizontally or vertically. Knight: Moves in an "L" shape: two squares in one direction and one square perpendicular. Bishop: Moves any number of squares diagonally. Queen: Moves any number of squares horizontally, vertically, or diagonally. King: Moves one square in any direction; special move: *castling*.

2. **Special Moves**: Included mechanics for castling, en passant, and pawn promotion.

3. **Check and Checkmate**: Detection of checks and checkmate, ensuring the king cannot move into check.

4**. Stalemate and Draw Conditions**: Recognized stalemates and other draw scenarios, ending the game appropriately.

These mechanics together create a full chess-playing experience in the game.

Game Semantics :-

The **Board** class creates an 8x8 chessboard using a nested loop, where each square alternates color based on position, using two shades of brown for a checkerboard pattern. Each square is drawn with a size of 100x100 pixels, and the colors alternate both horizontally across columns and adjust at the beginning of each row to maintain the pattern.

The **GamePanel** class represents the main game screen of a chess application. It extends JPanel and implements Runnable to support a game loop that updates and repaints the board at 60 FPS, handling piece movement, capture, promotion, and player turn management.

In this Java chess application, piece movements are handled by the GamePanel class. When the player clicks on a piece, the program checks if the piece is selectable by matching its color with the current player's turn. Once selected, the piece moves with the mouse cursor (simulate() method) as it verifies if it can legally move to a square (canMove() method) based on chess rules for each piece type.

If the piece is released over a valid square (validSquare flag is true), the move is finalized, updating the piece's position and reflecting any capture if an opponent piece occupies the square.
The copyPieces method updates the main piece list to maintain the board state and, if necessary, performs additional checks for special moves like castling and promotion.

Here's a condensed technical outline based on the code:

1. **Selection & Movement**: When a piece is selected, it updates its position in response to mouse movement (simulate() method).
2. **Validation**: Each potential move is checked with canMove() for legality, considering factors like piece type, destination square, and board state.
3. **Update on Release**: When the piece is dropped on a valid square, updatePosition() is called, finalizing the move.
4. **Player Switch**: changePlayer() toggles the turn after a valid move

Additional Effects :-

1. Added a separate space to specify which player should play first (black or white).

2. Restricted player from playing a move which is not possible for the piece to do.