```
import javax.swing.*;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class ChessGame extends JFrame {
    private JPanel chessBoard;
    private JButton[][] squares = new JButton[8][8];
    private JButton selectedPiece = null;
    private Color originalSquareColor;

    public ChessGame() {
        setTitle("Chess Game");
        setSize(400, 400);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        chessBoard = new JPanel(new GridLayout(8, 8));
        add(chessBoard);

        initializeChessBoard();
        setupChessPieces();
        addChessPieceListeners();
    }

    private void initializeChessBoard() {
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                squares[i][j] = new JButton();
                if ((i + j) % 2 == 0) {
                    squares[i][j].setBackground(Color.WHITE);
                } else {
                    squares[i][j].setBackground(Color.BLACK);
                }
                chessBoard.add(squares[i][j]);
            }
        }
    }

    private void setupChessPieces() {
        // Place pawns
        for (int i = 0; i < 8; i++) {
            squares[1][i].setIcon(new ImageIcon("white_pawn.png")); // Assuming you have images for the pie
ces
            squares[6][i].setIcon(new ImageIcon("black_pawn.png"));
        }

        // Place rooks, knights, bishops, queens, and kings - Implement this part for other pieces
    }

    private void addChessPieceListeners() {
        for (int i = 0; i < 8; i++) {
            for (int j = 0; j < 8; j++) {
                squares[i][j].addActionListener(new ActionListener() {
                    @Override
                    public void actionPerformed(ActionEvent e) {
```

```
                    JButton clickedSquare = (JButton) e.getSource();
                    if (selectedPiece == null) {
                        // Handle selecting a piece
                        selectedPiece = clickedSquare;
                        originalSquareColor = selectedPiece.getBackground();
                        // Highlight legal moves for the selected piece
                        clickedSquare.setBackground(Color.YELLOW);
                    } else {
                        // Handle moving the selected piece to the clicked square
                        if (isValidMove(selectedPiece, clickedSquare)) {
                            // Update the chessboard with the new position
                            clickedSquare.setIcon(selectedPiece.getIcon());
                            selectedPiece.setIcon(null);
                            // Reset the square colors
                            selectedPiece.setBackground(originalSquareColor);
                            clickedSquare.setBackground(originalSquareColor);
                            selectedPiece = null;
                        } else {
                            // Invalid move, handle accordingly
                            selectedPiece.setBackground(originalSquareColor);
                            selectedPiece = null;
                        }
                    }
                }
            });
        }
    }
}

private boolean isValidMove(JButton source, JButton target) {
    int sourceX = -1;
    int sourceY = -1;
    int targetX = -1;
    int targetY = -1;

    for (int i = 0; i < 8; i++) {
        for (int j = 0; j < 8; j++) {
            if (squares[i][j] == source) {
                sourceX = i;
                sourceY = j;
            }
            if (squares[i][j] == target) {
                targetX = i;
                targetY = j;
            }
        }
    }

    if (sourceX == -1 || sourceY == -1 || targetX == -1 || targetY == -1) {
        return false;
    }

    int deltaX = targetX - sourceX;
    int deltaY = Math.abs(targetY - sourceY);
```

```
        // Check if it’s a move one square forward for a pawn
        if (source.getIcon().toString().contains("pawn")) {
            if (deltaX == 0 && deltaY == 1) {
                // You need to check if the target square is empty
                return target.getIcon() == null;
            }

            // Check if it’s a capture diagonally
            if (deltaX == 1 && deltaY == 1) {
                // You’ll need to check if there is an opponent’s piece on the target square
                return target.getIcon() != null;
            }
        }

        // Add logic for other piece types here

        return false;
    }

    public static void main(String[] args) {
        SwingUtilities.invokeLater(() -> {
            ChessGame game = new ChessGame();
            game.setVisible(true);
        });
    }
}
```