import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

public class Chess {

static JLabel[][] labels = new JLabel[8][8];

static int selectedRow = -1, selectedCol = -1;

static boolean isWhiteTurn = true;

static JPanel board;

public static void main(String[] args) {

JFrame frame = new JFrame("Chess Game");

frame.setSize(600, 600);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

board = new JPanel(new GridLayout(8, 8));

frame.add(board);

String[][] initialBoard = {

{"♜", "♞", "♝", "♛", "♚", "♝", "♞", "♜"},

{"♟", "♟", "♟", "♟", "♟", "♟", "♟", "♟"},

{"", "", "", "", "", "", "", ""},

{"", "", "", "", "", "", "", ""},

{"", "", "", "", "", "", "", ""},

{"", "", "", "", "", "", "", ""},

{"♙", "♙", "♙", "♙", "♙", "♙", "♙", "♙"},

{"♖", "♘", "♗", "♕", "♔", "♗", "♘", "♖"}

};

boolean white = true;

for (int row = 0; row < 8; row++) {

for (int col = 0; col < 8; col++) {

JPanel square = new JPanel(new BorderLayout());

square.setBackground(white ? Color.WHITE : Color.GRAY);

JLabel label = new JLabel(initialBoard[row][col], SwingConstants.CENTER);

label.setFont(new Font("Segoe UI Symbol", Font.PLAIN, 36));

labels[row][col] = label;

final int r = row;

final int c = col;

square.addMouseListener(new MouseAdapter() {

public void mousePressed(MouseEvent e) {

String piece = labels[r][c].getText();

if (selectedRow == -1) {

if (!piece.equals("") && isCorrectTurn(piece)) {

selectedRow = r;

selectedCol = c;

square.setBackground(Color.YELLOW);

}

} else {

if (isLegalMove(selectedRow, selectedCol, r, c)) {

String movedPiece = labels[selectedRow][selectedCol].getText();

labels[r][c].setText(movedPiece);

labels[selectedRow][selectedCol].setText("");

isWhiteTurn = !isWhiteTurn;

}

resetColors();

selectedRow = -1;

selectedCol = -1;

}

}

});

square.add(label);

board.add(square);

white = !white;

}

white = !white;

}

frame.setVisible(true);

}

static boolean isCorrectTurn(String piece) {

return (isWhiteTurn && "♙♖♘♗♕♔".contains(piece)) || (!isWhiteTurn && "♟♜♞♝♛♚".contains(piece));

}

static void resetColors() {

boolean white = true;

Component[] squares = board.getComponents();

for (int i = 0; i < squares.length; i++) {

squares[i].setBackground(white ? Color.WHITE : Color.GRAY);

white = !white;

if ((i + 1) % 8 == 0) white = !white;

}

}

static boolean isLegalMove(int fromRow, int fromCol, int toRow, int toCol) {

String piece = labels[fromRow][fromCol].getText();

String target = labels[toRow][toCol].getText();

if (piece.equals("")) return false;

if (!target.equals("") && isSameSide(piece, target)) return false;

int dRow = toRow - fromRow;

int dCol = toCol - fromCol;

switch (piece) {

case "♙":

if (fromCol == toCol && target.equals("")) {

if (fromRow == 6 && toRow == 4) return true;

if (dRow == -1) return true;

} else if (Math.abs(dCol) == 1 && dRow == -1 && !target.equals("")) {

return true;

}

break;

case "♟":

if (fromCol == toCol && target.equals("")) {

if (fromRow == 1 && toRow == 3) return true;

if (dRow == 1) return true;

} else if (Math.abs(dCol) == 1 && dRow == 1 && !target.equals("")) {

return true;

}

break;

case "♘": case "♞":

return (Math.abs(dRow) == 2 && Math.abs(dCol) == 1) || (Math.abs(dRow) == 1 && Math.abs(dCol) == 2);

case "♖": case "♜":

if (dRow == 0 || dCol == 0) return isPathClear(fromRow, fromCol, toRow, toCol);

break;

case "♗": case "♝":

if (Math.abs(dRow) == Math.abs(dCol)) return isPathClear(fromRow, fromCol, toRow, toCol);

break;

case "♕": case "♛":

if ((dRow == 0 || dCol == 0) || (Math.abs(dRow) == Math.abs(dCol))) return isPathClear(fromRow, fromCol, toRow, toCol);

break;

case "♔": case "♚":

return Math.abs(dRow) <= 1 && Math.abs(dCol) <= 1;

}

return false;

}

static boolean isPathClear(int fromRow, int fromCol, int toRow, int toCol) {

int dRow = Integer.compare(toRow, fromRow);

int dCol = Integer.compare(toCol, fromCol);

int r = fromRow + dRow;

int c = fromCol + dCol;

while (r != toRow || c != toCol) {

if (!labels[r][c].getText().equals("")) return false;

r += dRow;

c += dCol;

}

return true;

}

static boolean isSameSide(String piece1, String piece2) {

return ("♙♖♘♗♕♔".contains(piece1) && "♙♖♘♗♕♔".contains(piece2)) ||

("♟♜♞♝♛♚".contains(piece1) && "♟♜♞♝♛♚".contains(piece2));

}

}