**ASSIGNMENT 1**

**Name:** Kapil Wavale

Div.: D

**Batch:** B1

**Roll. No.: 79**

**Prn :** 1211021

Code:

import java.util.Scanner;

public class TicTacToeAI {

    private static final char EMPTY = ' ';

    private static final char PLAYER = 'X';

    private static final char AI = 'O';

    public static void main(String[] args) {

        char[][] board = new char[3][3];

        initializeBoard(board);

        printBoard(board);

        while (!isGameOver(board)) {

            playerMove(board);

            if (isGameOver(board)) break;

            aiMove(board);

            if (isGameOver(board)) break;

            printBoard(board);

        }

        printBoard(board);

        if (isWinning(board, PLAYER)) {

            System.out.println("Congratulations! You win!");

        } else if (isWinning(board, AI)) {

            System.out.println("AI wins!");

        } else {

            System.out.println("It's a draw!");

        }

    }

    private static void initializeBoard(char[][] board) {

        for (int i = 0; i < 3; i++) {

            for (int j = 0; j < 3; j++) {

                board[i][j] = EMPTY;

            }

        }

    }

    private static void printBoard(char[][] board) {

        System.out.println("-------------");

        for (int i = 0; i < 3; i++) {

            System.out.print("| ");

            for (int j = 0; j < 3; j++) {

                System.out.print(board[i][j] + " | ");

            }

            System.out.println();

            System.out.println("-------------");

        }

    }

    private static boolean isGameOver(char[][] board) {

        return isWinning(board, PLAYER) || isWinning(board, AI) || isBoardFull(board);

    }

    private static boolean isWinning(char[][] board, char player) {

        for (int i = 0; i < 3; i++) {

            if (board[i][0] == player && board[i][1] == player && board[i][2] == player) {

                return true;

            }

            if (board[0][i] == player && board[1][i] == player && board[2][i] == player) {

                return true;

            }

        }

        if (board[0][0] == player && board[1][1] == player && board[2][2] == player) {

            return true;

        }

        return board[0][2] == player && board[1][1] == player && board[2][0] == player;

    }

    private static boolean isBoardFull(char[][] board) {

        for (int i = 0; i < 3; i++) {

            for (int j = 0; j < 3; j++) {

                if (board[i][j] == EMPTY) {

                    return false;

                }

            }

        }

        return true;

    }

    private static void playerMove(char[][] board) {

        Scanner scanner = new Scanner(System.in);

        int row, col;

        do {

            System.out.print("Enter row (0-2): ");

            row = scanner.nextInt();

            System.out.print("Enter column (0-2): ");

            col = scanner.nextInt();

        } while (!isValidMove(board, row, col));

        board[row][col] = PLAYER;

    }

    private static boolean isValidMove(char[][] board, int row, int col) {

        if (row < 0 || row > 2 || col < 0 || col > 2) {

            return false;

        }

        return board[row][col] == EMPTY;

    }

    private static void aiMove(char[][] board) {

        int[] bestMove = findBestMove(board);

        board[bestMove[0]][bestMove[1]] = AI;

        System.out.println("AI moves to row " + bestMove[0] + ", column " + bestMove[1]);

    }

    private static int[] findBestMove(char[][] board) {

        int[] bestMove = {-1, -1};

        int bestScore = Integer.MIN\_VALUE;

        for (int i = 0; i < 3; i++) {

            for (int j = 0; j < 3; j++) {

                if (board[i][j] == EMPTY) {

                    board[i][j] = AI;

                    int score = minimax(board, 0, false);

                    board[i][j] = EMPTY;

                    if (score > bestScore) {

                        bestScore = score;

                        bestMove[0] = i;

                        bestMove[1] = j;

                    }

                }

            }

        }

        return bestMove;

    }

    private static int minimax(char[][] board, int depth, boolean isMaximizing) {

        if (isWinning(board, PLAYER)) {

            return -10 + depth;

        } else if (isWinning(board, AI)) {

            return 10 - depth;

        } else if (isBoardFull(board)) {

            return 0;

        }

        if (isMaximizing) {

            int bestScore = Integer.MIN\_VALUE;

            for (int i = 0; i < 3; i++) {

                for (int j = 0; j < 3; j++) {

                    if (board[i][j] == EMPTY) {

                        board[i][j] = AI;

                        int score = minimax(board, depth + 1, false);

                        board[i][j] = EMPTY;

                        bestScore = Math.max(bestScore, score);

                    }

                }

            }

            return bestScore;

        } else {

            int bestScore = Integer.MAX\_VALUE;

            for (int i = 0; i < 3; i++) {

                for (int j = 0; j < 3; j++) {

                    if (board[i][j] == EMPTY) {

                        board[i][j] = PLAYER;

                        int score = minimax(board, depth + 1, true);

                        board[i][j] = EMPTY;

                        bestScore = Math.min(bestScore, score);

                    }

                }

            }

            return bestScore;

}

}

}

Output:

