**ASSIGNMENT-1 A**

**Name: Swapnil Devman Pawar**

**PRN: 12110917**

**Div.: TY\_CS\_D**

**Batch: 1**

**Roll No.: 52**

**Subject: Artificial Intelligence**

**Problem Statement:** Implementing any two-player game by implementing the Non-AI technique. (tic-tac-toe)

**Code:**

package Assignments;

import java.util.Scanner;

public class ass2\_AI {

    public static void main(String[] args) {

        char[] positions = { '1', '2', '3', '4', '5', '6', '7', '8', '9' };

        char humanPlayer = 'X';

        char aiPlayer = 'O';

        Scanner sc = new Scanner(System.in);

        while (true) {

            // Human's turn

            printPositions(positions);

            System.out.print("Enter your move (1-9): ");

            int move = sc.nextInt();

            move -= 1;

            if (isValidMove(positions, move)) {

                positions[move] = humanPlayer;

            } else {

                System.out.println("Invalid move. Try again!!!\n");

                continue;

            }

            if (checkWin(positions, humanPlayer)) {

                printPositions(positions);

                System.out.println("\nYou win!");

                break;

            }

            // Check for draw

            if (ispositionsFull(positions)) {

                printPositions(positions);

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

                break;

            }

            // AI's Turn

            int bestMove = findBestMove(positions, aiPlayer);

            positions[bestMove] = aiPlayer;

            if (checkWin(positions, aiPlayer)) {

                printPositions(positions);

                System.out.println("\nAI wins!");

                break;

            }

        }

        sc.close();

    }

    public static void printPositions(char[] positions) {

        System.out.println();

        System.out.println(" " + positions[0] + " | " + positions[1] + " | " + positions[2]);

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

        System.out.println(" " + positions[3] + " | " + positions[4] + " | " + positions[5]);

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

        System.out.println(" " + positions[6] + " | " + positions[7] + " | " + positions[8]);

    }

    public static boolean isValidMove(char[] positions, int move) {

        return move >= 0 && move < 9

                && (positions[move] == '1' || positions[move] == '2' || positions[move] == '3' || positions[move] == '4'

                        || positions[move] == '5' || positions[move] == '6' || positions[move] == '7'

                        || positions[move] == '8' || positions[move] == '9');

    }

    public static boolean ispositionsFull(char[] positions) {

        for (int i = 0; i < 9; i++) {

            if (positions[i] == '1' || positions[i] == '2' || positions[i] == '3' || positions[i] == '4'

                    || positions[i] == '5' || positions[i] == '6' || positions[i] == '7'

                    || positions[i] == '8' || positions[i] == '9') {

                return false;

            }

        }

        return true;

    }

    public static boolean checkWin(char[] positions, char player) {

        // Check rows columns and diagonals

        return (positions[0] == player && positions[1] == player && positions[2] == player) ||

                (positions[3] == player && positions[4] == player && positions[5] == player) ||

                (positions[6] == player && positions[7] == player && positions[8] == player) ||

                (positions[0] == player && positions[3] == player && positions[6] == player) ||

                (positions[1] == player && positions[4] == player && positions[7] == player) ||

                (positions[2] == player && positions[5] == player && positions[8] == player) ||

                (positions[0] == player && positions[4] == player && positions[8] == player) ||

                (positions[2] == player && positions[4] == player && positions[6] == player);

    }

    public static int findBestMove(char[] positions, char player) {

        int bestMove = -1;

        int bestScore = Integer.MIN\_VALUE;

        for (int i = 0; i < 9; i++) {

            if (positions[i] == '1' || positions[i] == '2' || positions[i] == '3' || positions[i] == '4'

                    || positions[i] == '5' || positions[i] == '6' || positions[i] == '7'

                    || positions[i] == '8' || positions[i] == '9') {

                char j = positions[i];

                positions[i] = player;

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

                positions[i] = j;

                if (score > bestScore) {

                    bestScore = score;

                    bestMove = i;

                }

            }

        }

        return bestMove;

    }

    public static int minimax(char[] positions, int depth, boolean isMaximizing) {

        if (checkWin(positions, 'O')) {

            return 1;

        }

        if (checkWin(positions, 'X')) {

            return -1;

        }

        if (ispositionsFull(positions)) {

            return 0;

        }

        if (isMaximizing) {

            int bestScore = Integer.MIN\_VALUE;

            for (int i = 0; i < 9; i++) {

                if (positions[i] == ' ') {

                    positions[i] = 'O';

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

                    positions[i] = ' ';

                    bestScore = Math.max(bestScore, score);

                }

            }

            return bestScore;

        } else {

            int bestScore = Integer.MAX\_VALUE;

            for (int i = 0; i < 9; i++) {

                if (positions[i] == ' ') {

                    positions[i] = 'X';

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

                    positions[i] = ' ';

                    bestScore = Math.min(bestScore, score);

                }

            }

            return bestScore;

        }

    }

}

**Result:**

1 | 2 | 3

---|---|---

4 | 5 | 6

---|---|---

7 | 8 | 9

Enter your move (1-9): 10

Invalid move. Try again!!!

1 | 2 | 3

---|---|---

4 | 5 | 6

---|---|---

7 | 8 | 9

Enter your move (1-9): 5

O | 2 | 3

---|---|---

4 | X | 6

---|---|---

7 | 8 | 9

Enter your move (1-9): 4

O | O | 3

---|---|---

X | X | 6

---|---|---

7 | 8 | 9

Enter your move (1-9): 6

O | O | 3

---|---|---

X | X | X

---|---|---

7 | 8 | 9

You win!