**NAME – TAVISHI MAHAJAN**

**ROLL NO. 56**

import java.util.\*;

public class TicTacToeAI {

private static final int BOARD\_SIZE = 3;

private static final char EMPTY\_CELL = ' ';

private static final char PLAYER\_X = 'X';

private static final char PLAYER\_O = 'O';

private static char[][] board = new char[BOARD\_SIZE][BOARD\_SIZE];

public static void main(String[] args) {

initializeBoard();

Scanner scanner = new Scanner(System.in);

System.out.println("Tic Tac Toe - Human vs. Computer");

System.out.println("You are X, and the Computer is O.");

printNumberedBoard();

while (!isGameFinished()) {

humanMove(scanner);

printNumberedBoard();

if (isGameFinished()) {

break;

}

aiMove();

printNumberedBoard();

}

scanner.close();

char winner = determineWinner();

if (winner == PLAYER\_X) {

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

} else if (winner == PLAYER\_O) {

System.out.println("Computer wins! Better luck next time.");

} else {

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

}

}

private static void initializeBoard() {

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

board[i][j] = EMPTY\_CELL;

}

}

}

private static void printNumberedBoard() {

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

int cellNumber = 1;

for (int i = 0; i < BOARD\_SIZE; i++) {

System.out.print("| ");

for (int j = 0; j < BOARD\_SIZE; j++) {

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

System.out.print(cellNumber);

} else {

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

}

System.out.print(" | ");

cellNumber++;

}

System.out.println();

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

}

}

private static void humanMove(Scanner scanner) {

System.out.println("Your turn. Enter a cell number (1-9):");

int cellNumber = scanner.nextInt();

if (isValidMove(cellNumber) && board[getRow(cellNumber)][getCol(cellNumber)] == EMPTY\_CELL) {

board[getRow(cellNumber)][getCol(cellNumber)] = PLAYER\_X;

} else {

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

humanMove(scanner);

}

}

private static boolean isValidMove(int cellNumber) {

return cellNumber >= 1 && cellNumber <= 9;

}

private static int getRow(int cellNumber) {

switch (cellNumber) {

case 1:

case 2:

case 3:

return 0;

case 4:

case 5:

case 6:

return 1;

case 7:

case 8:

case 9:

return 2;

}

return 0;

}

private static int getCol(int cellNumber) {

switch (cellNumber) {

case 1:

case 4:

case 7:

return 0;

case 2:

case 5:

case 8:

return 1;

case 3:

case 6:

case 9:

return 2;

}

return 0;

}

private static void aiMove() {

int[] bestMove = findBestMove();

board[bestMove[0]][bestMove[1]] = PLAYER\_O;

}

private static int[] findBestMove() {

int bestScore = Integer.MIN\_VALUE;

int[] bestMove = new int[]{-1, -1};

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

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

board[i][j] = PLAYER\_O;

int score = minimax(board, false);

board[i][j] = EMPTY\_CELL;

if (score > bestScore) {

bestScore = score;

bestMove[0] = i;

bestMove[1] = j;

}

}

}

}

return bestMove;

}

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

if (isGameFinished()) {

return evaluateBoard();

}

if (isMaximizingPlayer) {

int bestScore = Integer.MIN\_VALUE;

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

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

board[i][j] = PLAYER\_O;

bestScore = Math.max(bestScore, minimax(board, false)); // AI

board[i][j] = EMPTY\_CELL;

}

}

}

return bestScore;

} else {

int bestScore = Integer.MAX\_VALUE;

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

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

board[i][j] = PLAYER\_X;

bestScore = Math.min(bestScore, minimax(board, true)); // Player

board[i][j] = EMPTY\_CELL;

}

}

}

return bestScore;

}

}

private static boolean isGameFinished() {

return isBoardFull() || determineWinner() != EMPTY\_CELL;

}

private static boolean isBoardFull() {

for (int i = 0; i < BOARD\_SIZE; i++) {

for (int j = 0; j < BOARD\_SIZE; j++) {

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

return false;

}

}

}

return true;

}

private static char determineWinner() {

for (int i = 0; i < BOARD\_SIZE; i++) {

if (board[i][0] == board[i][1] && board[i][1] == board[i][2] && board[i][0] != EMPTY\_CELL) {

return board[i][0];

}

if (board[0][i] == board[1][i] && board[1][i] == board[2][i] && board[0][i] != EMPTY\_CELL) {

return board[0][i];

}

}

if (board[0][0] == board[1][1] && board[1][1] == board[2][2] && board[0][0] != EMPTY\_CELL) {

return board[0][0];

}

if (board[0][2] == board[1][1] && board[1][1] == board[2][0] && board[0][2] != EMPTY\_CELL) {

return board[0][2];

}

return EMPTY\_CELL;

}

private static int evaluateBoard() {

char winner = determineWinner();

if (winner == PLAYER\_O) {

return 10;

} else if (winner == PLAYER\_X) {

return -10;

} else {

return 0;

}

}

}