**Aim: Implement A star Algorithm for any game search problem**

import java.util.Scanner;

public class TicTacToe {

static Scanner in = new Scanner(System.in);

static String board[][] = new String[3][3];

static int moveCount;

static String humanPlayer;

static String aiPlayer;

static int moveI,moveJ; //board indexes of current AI moves - used in printBoard()

public static void init(){

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

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

board[i][j] = " ";

}

}

public static void selectSymbol(){

String input = "";

System.out.println("Choose X|O : ");

input = in.next();

while(!input.equalsIgnoreCase("x") && !input.equalsIgnoreCase("o")){

System.out.println("\nInvalid Input! (Only X or O)\n");

System.out.println("Choose X|O : ");

input = in.next();

}

humanPlayer = (input.equalsIgnoreCase("x")) ? "X" : "O";

aiPlayer = (humanPlayer.equals("X")) ? "O" : "X";

}

public static void printBoard()

System.out.println("\n\n\n\n\n\n\n\n\n\n" );

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

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

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

}

System.out.println(" |\n");

//System.out.println("\n\t\t--\t\t--\t\t--\n");

}

System.out.println();

if(checkWin(board)=="nothing")

{

if(moveCount!=0)

{

System.out.println("AI Player marked "+aiPlayer+" at position "+(moveI+1)+"-"+(moveJ+1));

System.out.println("Your Move (Use Numpad) : ");

}

else

{

System.out.println("Your Move (Use Numpad) : ");

}

}

}

public static boolean validMove(int boardNumber) //Check if position on board is empty

{

int indexes[] = getBoardIndexes(boardNumber);

if(!board[indexes[0]][indexes[1]].equals(" "))

return false;

return true;

}

public static void inputMove(){ //input move from user (1-9) - Numpad

int humanPlayerInput = 0;

do{

try{

humanPlayerInput = in.nextInt();

if((humanPlayerInput < 1 || humanPlayerInput > 9))

System.out.println("Invalid Input (Only 1-9)");

else if(!validMove(humanPlayerInput))

System.out.println("Invalid Move! Place Already Marked!");

}catch(Exception e)

{

System.out.println("Invalid Input (Only 1-9)");

in.next();

}

}

while(humanPlayerInput < 1 || humanPlayerInput > 9 || !validMove(humanPlayerInput));

int indexes[] = getBoardIndexes(humanPlayerInput);

board[indexes[0]][indexes[1]] = humanPlayer;

}

public static int[] getBoardIndexes(int boardNumber) //Convert 1-9 to board indexes

{

int []indexes = new int[2];

switch(boardNumber){

case 7:

indexes[0] = 0;

indexes[1] = 0;

return indexes;

case 8:

indexes[0] = 0;

indexes[1] = 1;

return indexes;

case 9:

indexes[0] = 0;

indexes[1] = 2;

return indexes;

case 4:

indexes[0] = 1;

indexes[1] = 0;

return indexes;

case 5:

indexes[0] = 1;

indexes[1] = 1;

return indexes;

case 6:

indexes[0] = 1;

indexes[1] = 2;

return indexes;

case 1:

indexes[0] = 2;

indexes[1] = 0;

return indexes;

case 2:

indexes[0] = 2;

indexes[1] = 1;

return indexes;

case 3:

indexes[0] = 2;

indexes[1] = 2;

return indexes;

}

return indexes;

}

public static boolean isHumanTurn(){ //Check if its human's turn

if(moveCount % 2 == 0)

return true;

return false;

}

public static void findBestMove(){ //Find best move for AI

int bestValue = -10000;

moveI = -1;

moveJ = -1;

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

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

if(board[i][j] == " ")

{

board[i][j] = aiPlayer; //make move

int val = getBoardValue(board,0,false); //send new board to getBoardValue

board[i][j] = " "; //undo move

if(val > bestValue)

{

bestValue = val;

moveI = i;

moveJ = j;

}

}

board[moveI][moveJ] = aiPlayer;

}

public static int getBoardValue(String newBoard[][],int depth,boolean isMax){ //Calculates values of every possible board

if(checkWin(newBoard) == "draw")

return 0 + depth; //draw in max possible moves

if(checkWin(newBoard) == aiPlayer)

return 10 - depth; // win in min possible moves

if(checkWin(newBoard) == humanPlayer)

return -10 + depth; // lose in max possible moves

if(isMax) //if score is to be maximized : in case of AI Player

{

int maxValue = -100;

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

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

if(newBoard[i][j] == " "){

newBoard[i][j] = aiPlayer; //make move

int value = getBoardValue(newBoard, depth+1, !isMax);

newBoard[i][j] = " "; //undo move

if(value > maxValue)

maxValue = value;

}

return maxValue;

}

else //if score is to be minimized : in case of Human Player

{

int minValue = 10000;

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

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

if(newBoard[i][j] == " "){

newBoard[i][j] = humanPlayer; //make move

int value = getBoardValue(newBoard, depth+1, !isMax);

newBoard[i][j] = " "; //undo move

if(value < minValue)

minValue = value;

}

return minValue;

}

}

public static String checkWin(String newBoard[][]){ //Accepts a board and Checks if User/AI has won, or draw

//returns win or draw or nothing

for(int i=0;i<3;i++) //checking rows & cols

{

if(newBoard[i][0].equals(newBoard[i][1]) && newBoard[i][1].equals(newBoard[i][2]))

{

if(!newBoard[i][0].equals(" "))

if(newBoard[i][0] == humanPlayer)

return humanPlayer;

else

return aiPlayer;

}

if(newBoard[0][i].equals(newBoard[1][i]) && newBoard[1][i].equals(newBoard[2][i]))

{

if(!newBoard[0][i].equals(" "))

if(newBoard[0][i] == humanPlayer)

return humanPlayer;

else

return aiPlayer;

}

}

if((newBoard[0][0].equals(newBoard[1][1]) && newBoard[1][1].equals(newBoard[2][2])) || (newBoard[0][2].equals(newBoard[1][1]) && newBoard[1][1].equals(newBoard[2][0])))

{

if(!newBoard[1][1].equals(" "))

if(newBoard[1][1] == humanPlayer)

return humanPlayer;

else return aiPlayer;

}

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

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

if(newBoard[i][j] == " ")

return "nothing";

return "draw";

}

public static void aiMove(){ //calls findBestMove()

findBestMove();

}

public static void play(){ //keep playing while value returned from checkWin() is "nothing"

moveCount = 0;

printBoard();

while(checkWin(board) == "nothing")

{

if(isHumanTurn())

inputMove();

else

aiMove();

printBoard();

moveCount++;

}

if(checkWin(board)==humanPlayer)

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

else if(checkWin(board) == aiPlayer)

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

else System.out.println("Draw!");

}

public static void main(String args[])

{

init();

selectSymbol();

play();

}

}

**OUTPUT:**

Choose X|O :

O

| | | |

| | | |

| | | |

Your Move (Use Numpad) :

1

| | | |

| | | |

| O | | |

Your Move (Use Numpad) :

| | | |

| | X | |

| O | | |

AI Player marked X at position 2-2

Your Move (Use Numpad) :

4

| | | |

| O | X | |

| O | | |

AI Player marked X at position 2-2

Your Move (Use Numpad) :

| X | | |

| O | X | |

| O | | |

AI Player marked X at position 1-1

Your Move (Use Numpad) :

6

| X | | |

| O | X | O |

| O | | |

AI Player marked X at position 1-1

Your Move (Use Numpad) :

| X | | |

| O | X | O |

| O | | X |

AI Player wins!







