Artificial Intelligence

Asssignment 1 B

Name : Swati Anil Sonone

Div : CS\_D

Roll no. : 53

Problem Statement : Tic Tac Toe With AI

Code :

import java.util.\*;  
  
public class Tic\_Tac\_Toe\_AI {  
 static Map<Integer, Character> *board* = new HashMap<>();  
 static char *player* = 'X';  
 static char *bot* = 'O';  
  
 public static void main(String[] args) {  
 System.*out*.println("\*\*\*\*\*Tic-Tac-Toe\*\*\*\*\*\*");  
 System.*out*.println("\*\*\*\*\*Lets play the game\*\*\*\*\*\*");  
 *initializeBoard*();  
 System.*out*.println("Computer goes first! Good luck.");  
 System.*out*.println("Positions are as follows:");  
 System.*out*.println("1, 2, 3 ");  
 System.*out*.println("4, 5, 6 ");  
 System.*out*.println("7, 8, 9 ");  
 System.*out*.println();  
  
 while (!*checkForWin*()) {  
 *playerMove*();  
  
 if (*checkForWin*()) {  
 break;  
 }  
 *compMove*();  
 }  
 }  
  
 public static void initializeBoard() {  
 for (int i = 1; i <= 9; i++) {  
 *board*.put(i, ' ');  
 }  
 *printBoard*();  
 }  
  
 public static void printBoard() {  
 System.*out*.println(*board*.get(1) + "|" + *board*.get(2) + "|" + *board*.get(3));  
 System.*out*.println("-+-+-");  
 System.*out*.println(*board*.get(4) + "|" + *board*.get(5) + "|" + *board*.get(6));  
 System.*out*.println("-+-+-");  
 System.*out*.println(*board*.get(7) + "|" + *board*.get(8) + "|" + *board*.get(9));  
 System.*out*.println();  
 }  
 public static boolean spaceIsFree(int position) {  
 return *board*.get(position) == ' ';  
 }  
  
 public static void insertLetter(char letter, int position) {  
 if (*spaceIsFree*(position)) {  
 *board*.put(position, letter);  
 *printBoard*();  
 if (*checkDraw*()) {  
 System.*out*.println("Draw!");  
 System.*exit*(0);  
 }  
 if (*checkForWin*()) {  
 if (letter == 'O') {  
 System.*out*.println("Cpu wins!");  
 } else {  
 System.*out*.println("Player wins!");  
 }  
 System.*exit*(0);  
 }  
 } else {  
 System.*out*.println("Can't insert there!");  
 position = new Scanner(System.*in*).nextInt();  
 *insertLetter*(letter, position);  
 }  
 }  
  
 public static boolean checkForWin() {  
 return (*checkWhichMarkWon*(*bot*) || *checkWhichMarkWon*(*player*));  
 }  
  
 public static boolean checkWhichMarkWon(char mark) {  
 return ((*board*.get(1) == mark && *board*.get(2) == mark && *board*.get(3) == mark) ||  
 (*board*.get(4) == mark && *board*.get(5) == mark && *board*.get(6) == mark) ||  
 (*board*.get(7) == mark && *board*.get(8) == mark && *board*.get(9) == mark) ||  
 (*board*.get(1) == mark && *board*.get(4) == mark && *board*.get(7) == mark) ||  
 (*board*.get(2) == mark && *board*.get(5) == mark && *board*.get(8) == mark) ||  
 (*board*.get(3) == mark && *board*.get(6) == mark && *board*.get(9) == mark) ||  
 (*board*.get(1) == mark && *board*.get(5) == mark && *board*.get(9) == mark) ||  
 (*board*.get(7) == mark && *board*.get(5) == mark && *board*.get(3) == mark));  
 }  
  
 public static boolean checkDraw() {  
 for (char value : *board*.values()) {  
 if (value == ' ') {  
 return false;  
 }  
 }  
 return true;  
 }  
  
 public static void playerMove() {  
 System.*out*.println("Enter your position : ");  
 int position = new Scanner(System.*in*).nextInt();  
 *insertLetter*(*player*, position);  
 }  
  
 public static void compMove() {  
 int bestScore = Integer.*MIN\_VALUE*;  
 int bestMove = 0;  
 for (int key : *board*.keySet()) {  
 if (*board*.get(key) == ' ') {  
 *board*.put(key, *bot*);  
 int score = *minimax*(*board*, 0, false);  
 *board*.put(key, ' ');  
 if (score > bestScore) {  
 bestScore = score;  
 bestMove = key;  
 }  
 }  
 }  
 *insertLetter*(*bot*, bestMove);  
 }  
 public static int minimax(Map<Integer, Character> board, int depth, boolean isMaximizing) {  
 if (*checkWhichMarkWon*(*bot*)) {  
 return 1;  
 } else if (*checkWhichMarkWon*(*player*)) {  
 return -1;  
 } else if (*checkDraw*()) {  
 return 0;  
 }  
  
 if (isMaximizing) {  
 int bestScore = Integer.*MIN\_VALUE*;  
 for (int key : board.keySet()) {  
 if (board.get(key) == ' ') {  
 board.put(key, *bot*);  
 int score = *minimax*(board, depth + 1, false);  
 board.put(key, ' ');  
 bestScore = Math.*max*(bestScore, score);  
 }  
 }  
 return bestScore;  
 } else {  
 int bestScore = Integer.*MAX\_VALUE*;  
 for (int key : board.keySet()) {  
 if (board.get(key) == ' ') {  
 board.put(key, *player*);  
 int score = *minimax*(board, depth + 1, true);  
 board.put(key, ' ');  
 bestScore = Math.*min*(bestScore, score);  
 }  
 }  
 return bestScore;  
 }  
 }  
}

Output



