AI Assignment 1 - Implementation of AI and Non-AI techniques

Name : Siddhi Rajeshirke
Division : CS-C
Batch : 3
Roll No : 75
PRN : 12110298

Non-AI Technique
Strategy – 2 for blank, 3 for X and 5 for O
Code :

```
import java.util.Scanner;
public class NonAi_TicTacToe {
    private static final int BLANK = 2;
    private static final int X = 3;
    private static final int O = 5;
    private int[] board = new int[10];
    private int turn = 1;
    public NonAi_TicTacToe() {
        for (int i = 1; i < board.length; i++) {
            board[i] = BLANK;
        }
    }
    public int make2() {
        if (board[5] == BLANK) {
            return 5;
        } else {
            for (int i : new int[]{2, 4, 6, 8}) {
                if (board[i] == BLANK) {
                    return i;
                }
            }
        }
        return -1;
    }
    public int possWin(int p) {
        int[][] winPositions = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {1, 4, 7}, {2,
5, 8}, {3, 6, 9}, {1, 5, 9}, {3, 5, 7}};
        for (int[] pos : winPositions) {
            int prod = board[pos[0]] * board[pos[1]] * board[pos[2]];
            if (p == X && prod == X * X * BLANK || p == O && prod == O * O *
BLANK) {
                for (int i = 0; i < 3; i++) {
                    if (board[pos[i]] == BLANK) {
                        return pos[i];
                    }
                }
            }
        }
        return 0;
    }
    public void go(int n) {
        if (turn % 2 == 0) {
            board[n] = O;
        } else {
            board[n] = X;
        }
```

```
        turn++;
    }
    public void printBoard() {
        for (int i = 1; i < board.length; i++) {
            String val = board[i] == BLANK ? " " : (board[i] == X ? "X" : "O");
            System.out.print(val);

            if (i % 3 != 0) {
                System.out.print(" | ");
            } else {
                System.out.println();
                if (i != board.length - 1) {
                    System.out.println("-- -- --");
                }
            }
        }
    }
    public int checkWinner() {
        int[][] winPositions = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}, {1, 4, 7}, {2,
5, 8}, {3, 6, 9}, {1, 5, 9}, {3, 5, 7}};
        for (int[] pos : winPositions) {
            int prod = board[pos[0]] * board[pos[1]] * board[pos[2]];
            if (prod == X * X * X) {
                return X;
            } else if (prod == O * O * O) {
                return O;
            }
        }
        return BLANK;
    }
    public boolean isDraw() {
        return turn > 9 && checkWinner() == BLANK;
    }
    public void play() {
        Scanner scanner = new Scanner(System.in);
        System.out.println("Welcome!");
        System.out.print("Please enter 1 for X or 2 for O - ");
        int choice = scanner.nextInt();
        while (choice != 1 && choice != 2) {
            System.out.print("Invalid entry, enter 1 for X or 2 for O - ");
            choice = scanner.nextInt();
        }
        if (choice == 1) {
            System.out.println("Make the first move for X");
        } else {
            System.out.println("Computer makes the first move for X");
            turn = 2;
        }
        while (turn <= 9) {
            printBoard();
            if (turn % 2 == 0) {
                int pos = possWin(O);
                if (pos == 0) {
                    pos = make2();
                }
                if (pos != -1) {
                    go(pos);
                    System.out.println();
                    System.out.println("Computer move position - " + pos);
                }
            } else {
                System.out.print("Enter your move position - ");
                int pos = scanner.nextInt();
                while (pos < 1 || pos > 9 || board[pos] != BLANK) {
```

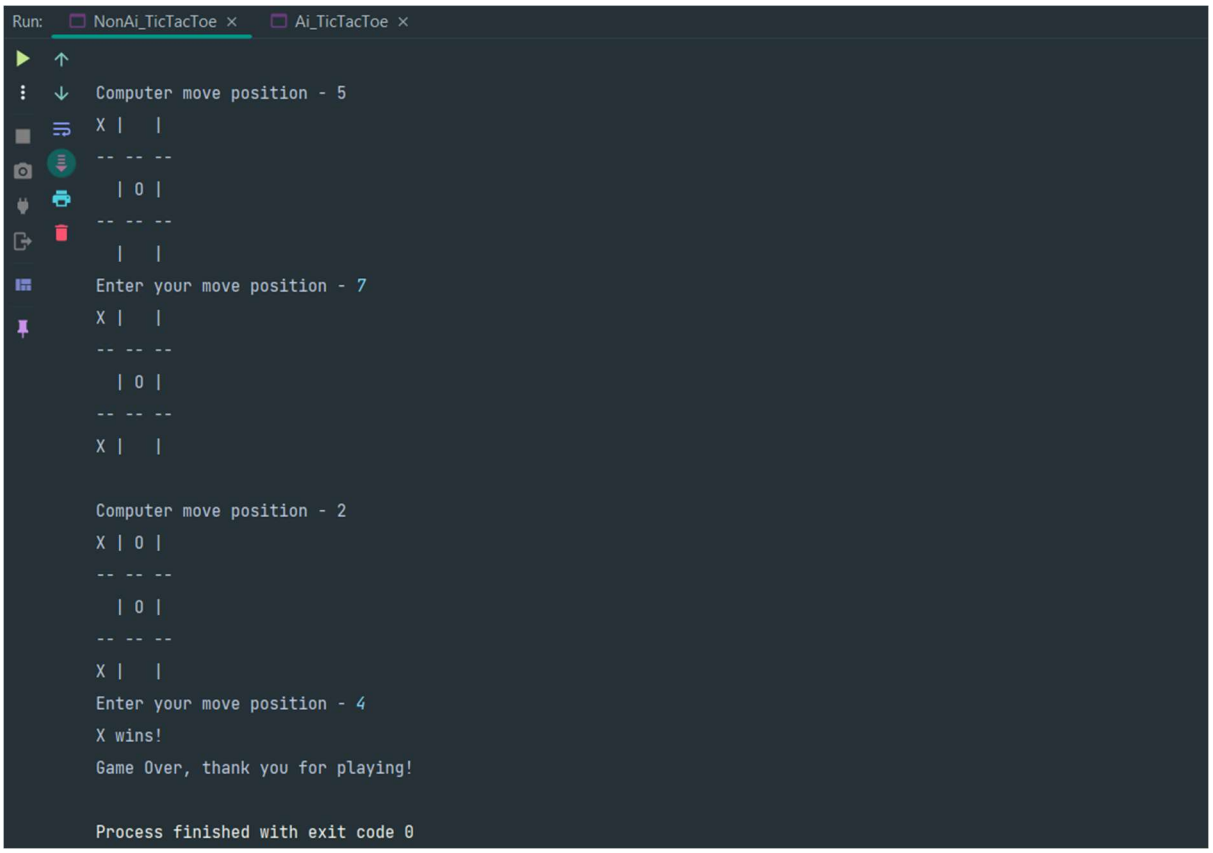
```
                    System.out.print("Invalid move, try again : ");
                    pos = scanner.nextInt();
                }
                go(pos);
            }
            int winner = checkWinner();
            if (winner != BLANK) {
                System.out.println(winner == X ? "X wins!" : "O wins!");
                break;
            }
            if (isDraw()) {
                System.out.println("It's a draw!");
                break;
            }
        }
        System.out.println("Game Over, thank you for playing!");

    }
    public static void main(String[] args) {
        NonAi_TicTacToe game = new NonAi_TicTacToe();
        game.play();
    }
}
```

Output :

```
Run:  NonAi_TicTacToe ×   Ai_TicTacToe ×
C:\Users\siddh\.jdks\openjdk-19.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ ID
Welcome!
Please enter 1 for X or 2 for O - 1
Make the first move for X
  |   |
-- -- --
  |   |
-- -- --
  |   |
Enter your move position - 1
X |   |
-- -- --
  |   |
-- -- --
  |   |

Computer move position - 5
X |   |
-- -- --
  | O |
-- -- --
  |   |
Enter your move position - 7
```



AI Technique
Strategy – Min-Max
Code :

```
import java.util.Scanner;
public class Ai_TicTacToe {
    char[][] board;
    char player;
    char aiPlayer;
    char currPlayer;
    public Ai_TicTacToe(char player) {
        board = new char[3][3];
        this.player = player;
        this.aiPlayer = (player == 'X') ? 'O' : 'X';
        this.currPlayer = player;
        initializeBoard();
    }
    void initializeBoard() {
        for(int i=0; i<3; i++ ){
            for(int j=0; j<3; j++ ){
                board[i][j] = '-';
            }
        }
    }
    void playGame() {
        boolean gameOver = false;
        Scanner sc = new Scanner(System.in);
        System.out.println("Welcome!");
        printBoard();
        while(!gameOver) {
            if(currPlayer == player) {
```

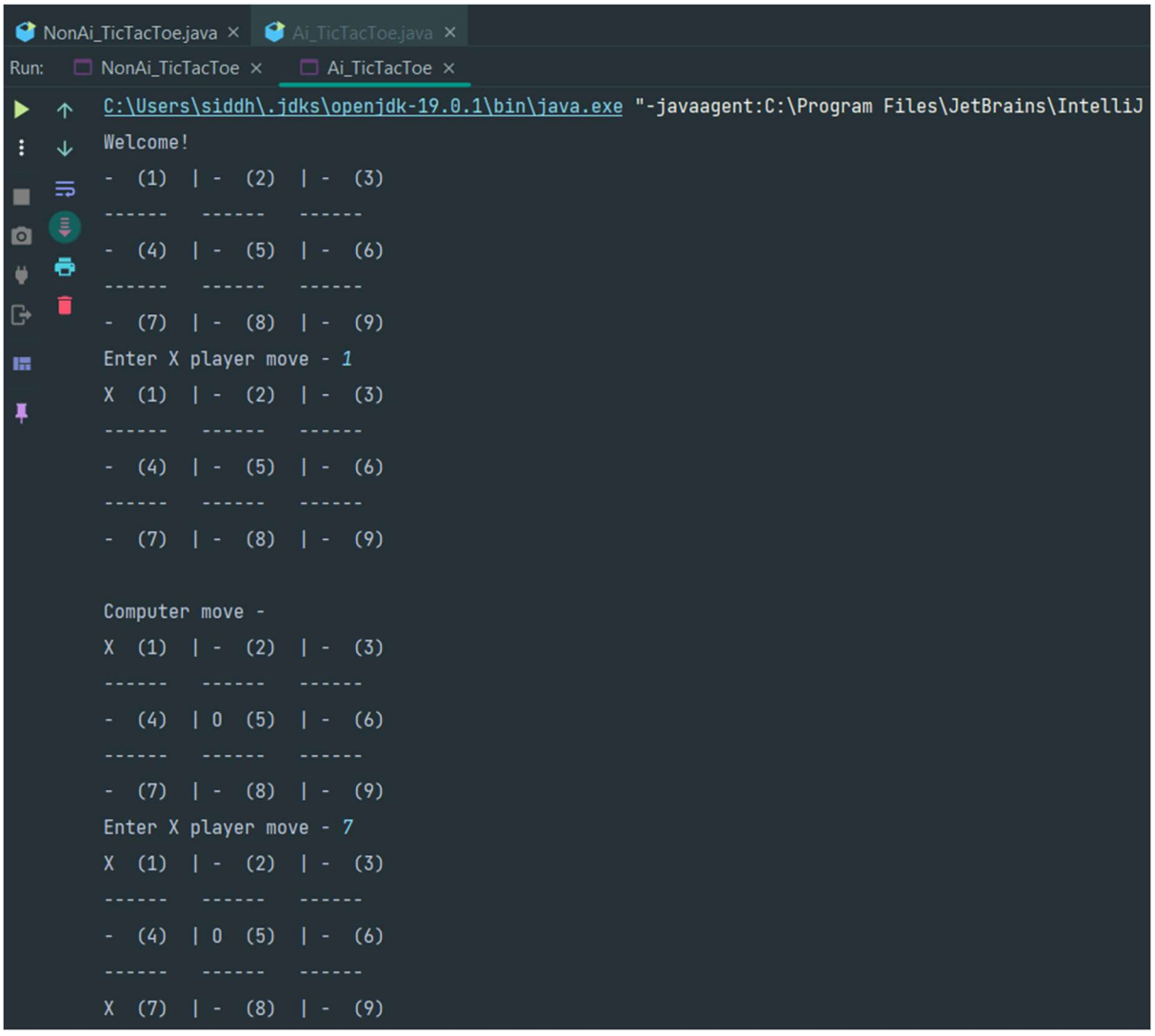
```
                System.out.print("Enter " + currPlayer + " player move - ");
                int move = sc.nextInt();
                int row = (move - 1) / 3;
                int col = (move - 1) % 3;
                if(isValidMove(row, col)) {
                    makeMove(row, col, currPlayer);
                    printBoard();
                    if(checkWin(currPlayer)) {
                        System.out.println("Player " + currPlayer + " wins!");
                        gameOver = true;
                    } else if(boardFull()) {
                        System.out.println("It's a draw!");
                        gameOver = true;
                    } else {
                        currPlayer = aiPlayer;
                    }
                } else {
                    System.out.println("Invalid move, try again.");
                }
            } else {
                System.out.println();
                System.out.println("Computer move - ");
                int[] move = getBestMove();
                makeMove(move[0], move[1], aiPlayer);
                printBoard();
                if(checkWin(aiPlayer)) {
                    System.out.println("Computer wins!");
                    gameOver = true;
                } else if(boardFull()) {
                    System.out.println("It's a draw!");
                    gameOver = true;
                } else {
                    currPlayer = player;
                }
            }
        }
        System.out.println("Game over, thank you for playing!");
    }
    private int[] getBestMove() {
        int[] bestMove = new int[]{-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] == '-') {
                    board[i][j] = aiPlayer;
                    int score = minimax(0, false);
                    board[i][j] = '-';
                    if(score > bestScore) {
                        bestScore = score;
                        bestMove[0] = i;
                        bestMove[1] = j;
                    }
                }
            }
        }
        return bestMove;
    }
    private int minimax(int depth, boolean isMaxPlayer) {
        char opponent = (aiPlayer == 'X') ? 'O' : 'X';
        if(checkWin(aiPlayer)) {
            return 1;
        } else if(checkWin(opponent)) {
            return -1;
        } else if(boardFull()) {
```

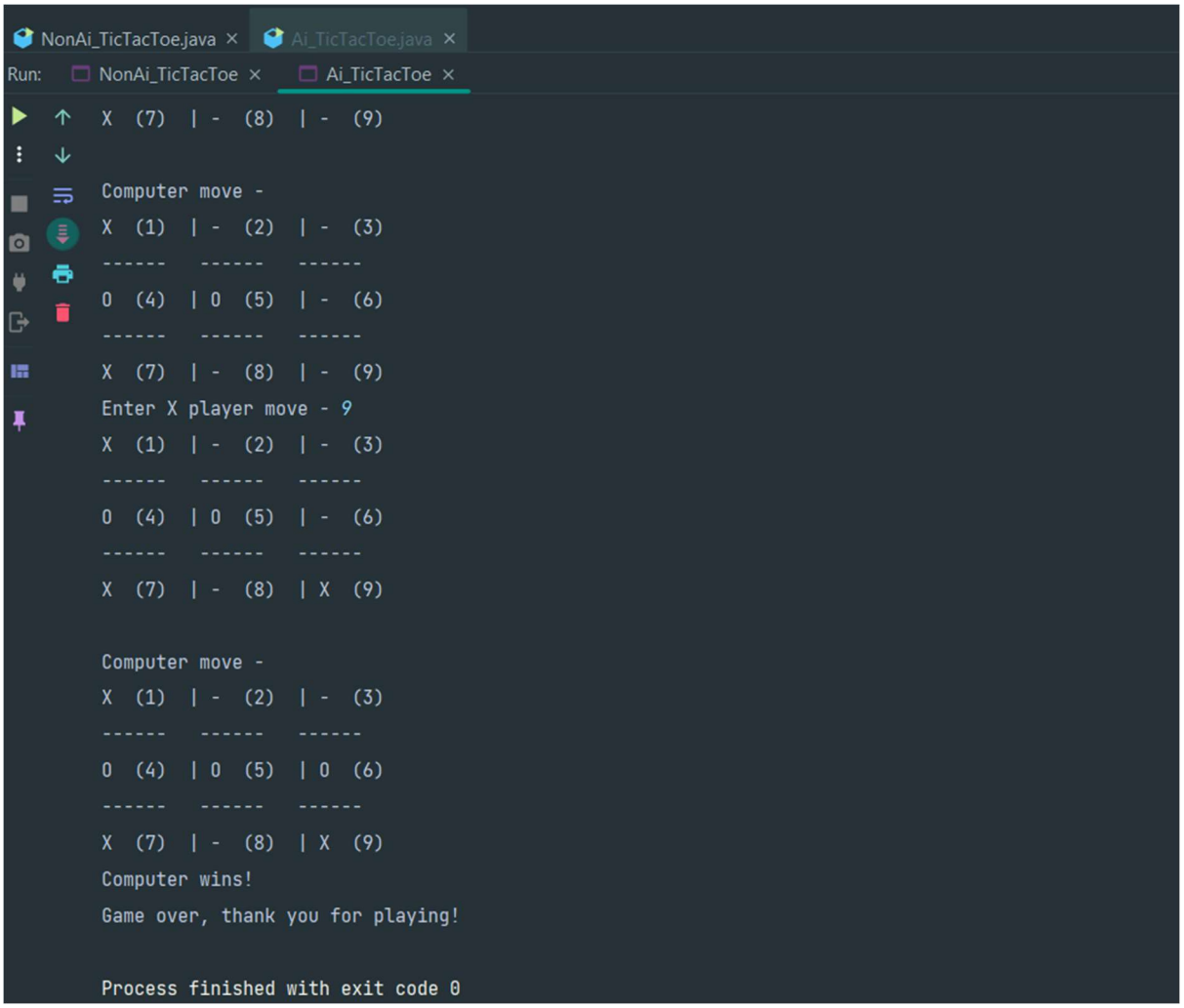
```
            return 0;
        }

        if(isMaxPlayer) {
            int bestScore = Integer.MIN_VALUE;
            for(int i=0; i<3; i++) {
                for (int j=0; j<3; j++) {
                    if(board[i][j] == '-') {
                        board[i][j] = aiPlayer;
                        int score = minimax(depth + 1, false);
                        board[i][j] = '-';
                        bestScore = Math.max(score, bestScore);
                    }
                }
            }
            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] == '-') {
                        board[i][j] = opponent;
                        int score = minimax(depth + 1, true);
                        board[i][j] = '-';
                        bestScore = Math.min(score, bestScore);
                    }
                }
            }
            return bestScore;
        }
    }
    private void makeMove(int row, int col, char player) {
        board[row][col] = player;
    }
    boolean boardFull() {
        for(int i=0; i<3; i++) {
            for(int j=0; j<3; j++) {
                if(board[i][j] == '-') return false;
            }
        }
        return true;
    }
    boolean checkWin(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;
        if(board[0][2] == player && board[1][1] == player && board[2][0] ==
player) return true;
        return false;
    }
    boolean isValidMove(int row, int col) {
        if (row >= 0 && row < 3 && col >= 0 && col < 3 && board[row][col] == '-
') return true;
        return false;
    }
    void printBoard() {
        int pos = 1;
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 3; j++) {
```

```
                System.out.print(board[i][j] + "   (" + pos + ")   ");
                pos++;
                if (j < 2) {
                    System.out.print("| ");
                }
            }
            System.out.println();
            if (i < 2) {
                System.out.println("------    ------    ------");
            }
        }
    }
    public static void main(String[] args) {
        char player = 'X';
        Ai_TicTacToe b = new Ai_TicTacToe(player);
        b.playGame();
    }
}
```

Output :




NonAi_TicTacToe.java
Ai_TicTacToe.java
Run:
NonAi_TicTacToe
Ai_TicTacToe
X (7) | - (8) | - (9)
Computer move -
X (1) | - (2) | - (3)
------ ------ ------
0 (4) | 0 (5) | - (6)
------ ------ ------
X (7) | - (8) | - (9)
Enter X player move - 9
X (1) | - (2) | - (3)
------ ------ ------
0 (4) | 0 (5) | - (6)
------ ------ ------
X (7) | - (8) | X (9)
Computer move -
X (1) | - (2) | - (3)
------ ------ ------
0 (4) | 0 (5) | 0 (6)
------ ------ ------
X (7) | - (8) | X (9)
Computer wins!
Game over, thank you for playing!
Process finished with exit code 0