# Design Overview for *Tic Toc Toe Game Program*

Name: Nguyen Mai Lan Vo

Student ID: 641465

# Summary of Program

This is a Java program for a Tic Tac Toe game that allows the user to play against an AI that uses the minimax algorithm to determine its moves.

The program initializes as 3x3,4x4 and 5x5 character array that represents the game board, where 'X' is used to represent the human player, 'O' represents the AI player, and ' ' represents an empty space. The program then runs a game loop that alternates between the human player's turn and the AI's turn until either the human player wins, the AI wins, or the game ends in a tie.

During the human player's turn, the program prompts the user for their desired move (a row and column), and then checks if the space is empty before placing the player's symbol on the board.

During the AI's turn, the program uses the minimax algorithm to determine the best move to make based on the current state of the board. The algorithm works by recursively exploring all possible moves and determining the score of each move based on the resulting board state. The AI then makes the move that yields the highest score.

The program also includes several helper methods, including a method for initializing the board, a method for printing the board to the console, and methods for checking if the board is full and if there is a winner.

Describe each of the record types or (data-oriented) classes and enumerated data types you will create using the following table (one per type). Repeat the tables if you need to define more.

# Required Data Types

Describe each of the record types or (data-oriented) classes and enumerated data types you will create using the following table (one per type). Repeat the tables if you need to define more.

Table 1: Player details

|  |  |  |
| --- | --- | --- |
| Property | Type | Notes |
| EMPTY\_SPACE | char | A char variable Define before play |
| HUMAN\_PLAYER | char | A char variable Define human player |
| AI\_PLAYER | char | A char variable Define AI player |

Table 2 : Symbol (inside the TicTacToe details table above)

|  |  |
| --- | --- |
| Value | Notes |
| X | assigned to the HUMAN\_PLAYER in the game |
| O | assigned to the AI\_PLAYER in the game |

# Overview of Program Structure

The program is a console-based implementation of the popular game, Tic Tac Toe, which is played between a human player and the computer AI. The game is played on a 3x3 ,4x4 and 5x5 grid, where each player takes turns to place their symbol on an empty cell. The first player to get three of their symbols in a row, column, or diagonal wins the game. If no player achieves this, and the grid is fully filled with symbols, then the game is a draw.

The implementation consists of a single class, TicTacToe, which contains the main method and several helper methods. The main method initializes the game, including the game board and printing initial messages. It then enters into an infinite loop, which continues until the game is over. Within this loop, it alternates between the human player and the AI player, accepting input from the human player and selecting the best move for the AI player.

The program structure includes several constants for the game symbols and an array for the game board. The board is represented as a 2-dimensional char array, with the game symbols and the empty space represented by specific characters. The main method also includes calls to several helper methods, including initializing the board, printing the board, checking for a win or draw, and selecting the best move for the AI player.

The implementation of the AI player uses the minimax algorithm, a common technique in game theory for selecting the best move for a player. The minimax algorithm evaluates all possible moves by both players and selects the one that maximizes the AI player's chance of winning and minimizes the human player's chance of winning. The algorithm is implemented recursively, and the implementation includes two functions: minimax and getBestMove. The minimax function recursively evaluates the game board and returns a score for the current state, while the getBestMove function selects the best move for the AI player by evaluating all possible moves and selecting the one with the highest score.

|  |  |  |  |
| --- | --- | --- | --- |
| Class | Name | Return type | Description |
| TicTacToe3x3 | initializeBoard() | void | Method to init the board |
| TicTacToe3x3 | main() | void | Main method that initializes the game and runs the game loop |
| TicTacToe3x3 | getBestMove() | int | Method to get the player's choice and return the corresponding position on the board |
| TicTacToe3x3 | minimax() | int | Minimax algorithm for determining the best move for the AI |
| TicTacToe3x3 | printBoard() | void | Method to print the game board |
| TicTacToe3x3 | isBoardFull() | void | Method to check if the game is board full |
| TicTacToe3x3 | checkWin() | boolean | Method to check if the game is a win |
| TicTacToeCustom | initializeBoard() | void | Initializes the game board with empty spaces. |
| TicTacToeCustom | printBoard() | void | Prints the current state of the game board. |
| TicTacToeCustom | isBoardFull() | boolean | Checks if the game board is full |
| TicTacToeCustom | hasPlayerWon() | boolean | Checks if a player has won the game |
| TicTacToeCustom | isValidMove() | boolean | Checks if a move is valid on the game board |
| TicTacToeCustom | makeMove() | void | Makes a move on the game board. |
| TicTacToeCustom | makeAIMove() | void | Makes a move for the AI player. |
| Main | main | void | Start game Tictactoe. |