3DTicTacToe ReadME File

This project is similar to my connect4 AI, in which I use the same minimax algorithm, but I apply it to the 3DTicTacToe Game

How does it Work:

This project uses bit positions to represent the board. I use a long to store the bit positions of the cpu and of the player it is playing against. From here, I use the bit positions along with bit operations to check if spaces are empty and whether or not a player has won the game.

The minimax algorithm goes to a certain ply depth which can be assigned on the command line. The bestMove function takes in the current game state and determines which move would be best for the AI. The max value function looks at the value of all of its children nodes in the tree and takes the maximum. The minimum value function looks at all the values of its children in the tree and takes the minimum. The max value is called on the nodes in which the opponent is making a move so that the computer can take into account the assumption that the opponent will make the move that is best for the opponent to win. The min value is called on nodes in which the cpu is making the move. The min value is called so that the computer can make contingency plans in case the opponent makes the best decision for itself. When the ply depth is reached and a win state is not reached, an evaluation function is used to evaluate the game state.

The TicTacToe class allows a game to be played.

Alpha Beta Pruning is also implemented as a way of provably shrinking the search space in order to increase the ply depth that can be reached.

Running Application on Terminal:

Reach the directory where the project is in and ensure the specific java files are located in.

```
javac *.java
java TicTacToe.java 5
```

5 represents the ply depth you want the computer to reach. The higher the ply depth the longer the computer will take to return its move. In testing, 5 achieved an approximate 20-30 second return time.