

Tic Tac Toe

This summer, I made two versions of a tic tac toe game using java. They are both popup windows that you can run using a terminal window. One version is for two players and the other is for one player against the computer. Both games can tell if someone has three in a row and if there is a draw. The computer can tell if you clicked in the “play again” square by adding a *mouselistener* and saying what is in English: “If these range of coordinates are clicked in after the game is over, all the squares will be empty and the counter is at 0 again”.

Two Player Game

The two player game is much simpler than the one player game. First of all, I added a *mouselistener* to tell where in the popup window someone clicked and I called paint using *repaint* to draw the squares and portions of circles. All the squares start out empty and when you click in a square, an X appears. The next click in an empty square will display an O. The next click in an empty square will show an X and so on and so forth. How this works is that there is a counter that tells if the next turn is X or O's turn. The counter will not increase if you click outside of a square or in already occupied square so players cannot cheat. The counter will stop when all the squares are full or if someone wins. To keep track of where the X and O's are, I used an array for all nine squares. X, O, and an empty square have their own values. For example, X is 2, O is 0, and an empty square is -1. The computer knows when somebody won when all the squares in a row have the value that is not -1. Then the game will stop. The game will also stop when the counter is at 9.

One Player Game

Like the two player game, I added a *mouselistener* and a *repaint*. All the squares start out empty and when you click, an X appears in the empty square. Then the computer will move into an empty square. Like the two player game, an array is used to keep track of the empty squares and the squares with X's and O's. The computer can tell if it needs to block or it has a win. How it tells if there is a win or a block is pretty simple. This is part of the *javascript* is unnecessary in the two player game because both player are able to use their own minds, while the computer has its own strategy. Each option inside the square has its own value to represent what value is inside every square. I figured out a strategy so that it was almost impossible for the computer to lose and match. Like one player, X, O, and an empty square have their own values. The computer can

tell if there if it has a win if the sum of a row's values are equal to -1. The computer then checks if it needs to block, which is if the sum of a row's values are equal to 4. After that the computer will use its own strategy. Like two player, one player uses to values and the counter to figure out if there is a win or a loss.

Strategy

After the first move, there is obviously no blocks or wins, so I have come up with a strategy that is virtually unbeatable using a switch which decides the first and second move. After that, it isn't important where the computer goes or there are a lot of blocks. The strategy is based on where X's first move is.

X first move: Corner

For the first move, if X moved to the corner, the computer will move in the middle to prevent unseen attacks. If X moves in the opposite corner, the computer will force X to block, preventing an inevitable draw. If X does not move in the opposite corner and there isn't a block, the computer will move on the side to prevent X from prevailing.

X first move: Side

If X's first move was in an empty square on the side, the computer will move to the adjacent corner clockwise of X's first move. After X takes his second turn, the computer will either block or move into the middle to prevent double wins for X. The computer will either draw with X, or if the right moves are played, the computer could win.

X first move: Middle

The computer will move to the top right corner if X's first move is in the middle square. X's second move will be a block, unless he moves in the opposing corner of the computer's first move. Then it is crucial that the computer moves in a corner, which is exactly what it will do. The computer will either win, or draw.

Starting the Program:

This is an image of what the game will look like, if run by Linux, using an editor called Kate. To run the program, you must open a terminal window and find the folder you wrote the java file in by typing 'cd (name of the folder)'. The next step is to type 'javac (name of your java program).java' and wait for the computer to respond. If it tells you there is an error, something in your program is not right, so you will have to fix it before running. If all goes well, type 'java (name of your java program WITHOUT .java)' and a popup window like the one on the right should appear. If not, you have made an error in your program.

