Discussion of TicTacToe Java Application:

Computer Science II Honors Option

The basis of the assignment was to develop a 5x5 version of the classic pen-and-paper game TicTacToe, also known as Crosses and Noughts. The game was meant to be played by one human player against the computer. The computational intelligence was designed to be completely defensive algorithm, always forcing a draw. The game begins with a splash screen asking if the human or computer will move first and then progresses turn by turn until one player has 5 pieces in alignment or all spaces on the board are filled without a winner. The player then has the option to play again with the starting player alternating.

The application was spread out into 4 classes: Game, GUI, AI, and Main. The Main class was very basic, simply containing the main method, working as the insertion point. The Main class was unnecessary but made it simple to quickly find the main method. The Game class contained all methods relating to the overall progression of the game, such as testing the winner, switching the player, and setting the board values. The GUI class, as expected, handles all the user interface elements, with methods such as initializing the buttons and labels, and also processes all button clicks. The AI class contains all methods relating to the computer's move selection, such as checking for a winning row.

There were some final variables declared, such as ROWS = 5 and COLS = 5, and some GUI elements declared as final, such as the JPanels and JFrame. Otherwise, only local variables were used to store currently relevant information. The most important variables were the 2-dimensional integer array (int[][] board) that stored all row, column, and player values; the public final integers ROWS and COLS; and the public integer player that kept track of the current player. A Cell object was originally going to be used in place of a 2D array, and was originally implemented, but it added some unnecessary complexity, I felt. The object oriented design could use a bit of reworking, and was planned, but fell short of the deadline.

The computational intelligence algorithm was the main challenge of the project. Especially with having no experience in AI, the algorithm was initially very difficult. On a 3x3 board, the AI could simply be a handful of if-else, but 5x5 presumed a greater challenge. After completely rewriting the AI code at least 5 times, the final algorithm was implemented. Basically, the AI checks through specific 'chunks' of the int[][] board: row, column, left diagonal, and right diagonal. It tests for 4 X pieces, as human is always X, in the specific alignment and if there is an O piece already blocking the alignment. If the test returns true, the AI then finds the empty cell and claims it. If the test returns false, the AI simply uses the Random class to generate a random row and random column value and places it's piece there.

The major challenges I faced was determining how to write the AI. I did a lot of research and followed many code videos online, but there was never a very good explanation, nor was there ever a 5x5 version. Many proper AI algorithms were suggested such as minimax and negamax, but those were much too far out of range of my conceptual understanding. Eventually, through trial and error and many rewrites the obvious solution hit me.

The other major challenge was object oriented design. It was difficult trying to determine what I could consolidate into objects. It seems almost that object oriented design was completely neglected. There were plans to refactor the code utilizing the object oriented paradigm, but, again, the deadline cut it short. Also, I feel as though I am not familiar enough with creating my own objects, as the labs and homework typically tell us what objects to create. This was my first solo project and my object oriented practice should evolve over time.