Solution/Design (Waterfall Model)

Requirements & Analysis: To start, I had a look at the requirements. It must have UI containing X and O (which must be .jpg), as well as a place for information (Whose turn / Congratulations / Draw). So, I needed to use a language that had suitable UI. Other than that I want it to work quick and look clean. It doesn't need to have insane features, but just be a good functional piece of software.

Design: I chose javax.Swing for my GUI integration as I know Java well and this was not a monumental task. JButton is feasible technology, and Swing frames work well with grids. The main data structure I will use will be an array that holds whether the square is an 'X' or 'O'. For example, 1 will be in the top left corner of the board, and 9 will be in the bottom right. Then I can go through and check if array indexes/spots (for example) 123, 147, or 159 are filled by either Xs or Os. If they get they have 3 'in a row' (in a board sense, not the actual array), then they win.

Coding: Starting to code, I am writing the backend in Java, the language used with Swing. I am using Eclipse IDE as this is what I am most comfortable with when coding in Java. I initially just did the logic of the tic tac toe game, printing to the console. Then it was time to add the GUI and integrate. I had originally made the board with just text on the buttons. I slowly added features, such as whose turn, the alert on win/draw, and finally the images instead of text on the buttons.

Testing: I have begun to write JUnit Tests on basic logical parts. Once those have been written and approved I move on to the GUI integration. Once it basically works I finally move on to higher level tests, such as how it actually plays and functions. Now I can actually play the game to test and try many cases. Total acceptance testing, style, and look are all tested and the code is tweaked.

Maintenance: After every level of testing, I made changes to ensure functionality. I ran into many bugs along the way. Specifically, my conditions around win/draw cases had quite a bit of problems to begin. Also, the conditions when the game reset (turn, finding the winner again) proved to be difficult. After a final round of testing (and playing!) it was an acceptable program that ran well.