Alejandro Rubio
Alejandro Serrano
Design Document:

The purpose of the assignment is for the program to take in a Sudoku game and to be able to solve it, either in a recursive manner or by elimination.

Two classes need to be implemented. First one is SudokuBacktrackRecursive and the other is SudokuByElimination. Both classes are implementing the "Sudoku" interface and using methods "element", "setElement", "solve" and "verify". Just as well, using the variables **PUZZLE_HEIGHT_WIDTH** and **BOX_HEIGHT_WIDTH.** Nonetheless, there are certain methods which are particular to each class. For instance, SudokuBacktrackRecursive is also using "isValidForRow", "isValidForColumn", "isValidForBox", and "isValidForPosition", which we can assume will be very important during the recursive implementation.

Sets will allow us to have the possibility to remove elements in a more simple matter than arrays since they grow and shrink according to its implementation. Also, sets can't have any duplicates so we will be certain that the options inside a set are not repeated when trying to solve the game. Clearly, this will not be a problem during the recursive implementation since we think the 2D implementation will be the most simple to tackle.

2D array implementation seems like it could make methods such as "isValidForColumn" and "isValidForRow" a lot easier. The reasoning behind this is that we would have access to the rows and columns directly due to the way a 2D array is implemented. We believe that this could be a little more difficult to do when using a 1D array, since we wouldn't have the access to rows and columns directly.

We predict that it will take us around 10 hours to finish this assignment considering that it is far more complex than the previous one. Nonetheless, since the assignment is with a partner maybe the time could be reduced.

We can assume that from "naive" implementation it might mean that at some point the solver won't be able to find any solution to the Sudoku puzzle, which will make it fail. In this case, not taking into account all of the possible options to solve the puzzle. For instance, this

naive implementation won’t work with the harder Sudoku puzzles since at some points trial and error is needed.