Tony Diep
u0934661
01/28/2017
2420
Program Document Analysis
Milad Bazzazi
I pledge that the work done here was my own and that I have learned how to write this program, such that I could throw it out and restart and finish it in a timely manner. I am not turning in any work that I cannot understand, describe, or recreate. I further acknowledge that I
contributed substantially to all code handed in and vouch for it's authenticity. (Tony Diep)

The purpose of this project is to create a Sudoku solver using recursion and to introduce the idea of a Set data structure. We also reinforce the ideas of the simple array along with the loop patterns since they both work complementary to each other. Recursion is an useful mechanism used for efficiency and conciseness.

There are only two classes to be created, along with a JUnit Test Case class to also be created. The first class is the “Sudoku” class in that it represents a single 9x9 Sudoku puzzle grid. It has some methods that allows us to follow the rules of Sudoku. This includes checking whether each row and column are all filled with numbers, checking if those rows and columns all have unique numbers (in other words, no row or column should have two or more of the same numbers), and inserting some numbers to solve the Sudoku puzzle. The other class we must create is the “Main” class. That class only sets up a Sudoku puzzle and allows the user to choose which method is selected to solve the puzzle (either by “elimination” or “brute force”). The JUnit Test case class is a special class that we must also create to proofread our program so that our solver is at least adequate to solve the majority of Sudoku puzzles.

The Set class is a class that represents the data structure called a "set", which in some sense is like an array but there a couple of differences. A set is a data structure
that contains all unique elements and therefore cannot contain duplicate elements. The elements are stored in a scattered, unordered manner in the set, which implies that the set does not have indexing as opposed to an array. The set is useful in this project because in Sudoku, every row and column must all have unique numbers ranging from 0 to 9. No two
of the exact same numbers can be at the same row and/or column.

There is a relationship between the 1D array implementation of the puzzle and the 2D matrix implementation. Every 9th index at the 1D array is a row equivalent to the end of each row in the 2D array. Every index represented by the modulus operator is a column equivalent to a column in the 2D array.

Given that the last assignment took more than 20 hours to complete, I believe this assignment won't take as long. I believe it will take somewhere between 12 and 15 hours
for me to complete the base part of the program. If I wanted to, I could also work on getting the GUI aspect built, but it may depend how many days I have left until the assignment
due date.

The recursive method solves more puzzles. It works for all puzzles, although it may take a long time because it's essentially a bunch of trial and errors since we are covering every possible number combination. The brute force method, on the
other hand, would have to check for every row, column, and box to see if they are already filled before we can insert our next guess. That method doesn't really check for all possible successful combinations to complete the puzzle since it only checks what regions of the puzzle are already filled versus not filled.

Tony Diep
u0934661
01/28/2017
2420
Program Document Analysis
Milad Bazzazi
I pledge that the work done here was my own and that I have learned how to write this program, such that I could throw it out and restart and finish it in a timely manner. I am not turning in any work that I cannot understand, describe, or recreate. I further acknowledge that I
contributed substantially to all code handed in and vouch for it's authenticity. (Tony Diep)