Day 1, Thursday 1/3/19

I created the Block class that controls the methods of a single cell. It is a 3X3 array list of Strings and each item represents a number in the cell. I created a contains method which returns true if the Block already has the number and false otherwise. Then I created the add and remove method which will allow the user to edit if their insertion follows the rules of the game as given by the contains method. I then created a toString() which formats the array into a block of a Sudoku puzzle. I did a soft test on the Block class and everything seems to be working. I will probably run a more extensive test on Saturday.

Day 2, Friday 1/4/19

I created the Grid class which is a collection of 9 blocks and represents the entire Sudoku puzzle. It is a 3X3 array list of Blocks and has the same methods as the Blocks class. The main difference between the Blocks and Grid Class is that the contains method for the Grid class checks if the number being placed is already in the row or column the user is trying to place it in. Add and Remove methods call on their parent methods with the condition that the insertion returns false for contains. Testing revealed that my contains method is flawed and will need to be worked on. On Sunday, I will begin the Sudoku class which will serve as the interface for the user.

Day 3, Saturday 1/5/19

I have been running tests on Grid and Block class and the contains method for Grid class is the only issue. It ignores the Sudoku rules and allows the player to insert a number into the same column or row as a pre existing number.

Day 4, Sunday 1/6/19

I have fixed the contains method for Grid class. All that’s left for this project is the interface portion. I have been reading through Mr.K’s terminal demo and have been using some of the code he has- I will make sure to cite him in the READ.ME and in the code. The Sudoku puzzle can now constantly update and you can add and remove numbers in but the coordinates of the terminal don’t correspond with the indices of the Grid class so I’m going to have to figure out how to fix that. I’m considering changing the toString() of the Grid and Block class so that it fits more smoothly in the coordinate system of the terminal but only if I have to. I have a convert method in the Sudoku class and it will read in the file submitted, and will use that information to create a Sudoku puzzle. The link for where I’m getting the Sudoku puzzles will be in the READ.ME.

Day 5, Monday 1/7/19

I have fixed the coordinate problem from day 4 by creating an array for x and y coordinates. Then by using the indexOf method, I could figure out which index of the arraylist you’re on and can therefore convert the coordinate system of the terminal to my puzzle. I think I have met all of my goals for the minimum viable product and I will probably now work on creating a good menu, a more user friendly interface, and picking Sudoku puzzles for the user to complete. Once I’m satisfied with that which will probably be Wednesday, I’m going to move on in creating the generator method.

Day 6, Tuesday 1/8/19

The menu is up and you can select your difficulties easy, medium, hard by pressing E, M and H respectively. Then it will randomly select a file from the difficulty chosen and add the elements to the interface puzzle. I chose the Sudoku text files and added them to github. Tomorrow I will work more on designs and begin the generator method.

Day 7 Wednesday 1/9/19

I added instructions to the menu and fixed a bug in readPuzzle(). I created a solver function and I can only assume that it works because it has been running for hours without printing out a puzzle. I probably shouldn’t have it start from scratch if it sees that the puzzle is unique and other than that everything’s ready for the demo.