

Sudoku solver

Name: Man Yiu Lam

Date: 10th December 2016

Student Number: 16458032

Introduction

- Aim of my project.** A Sudoku Solver is a program that will solve a given and solvable Sudoku puzzle. The program takes in any Sudoku puzzle and processes the possibilities that are definite, displaying them in the correct boxes. The definition of Sudoku is “a puzzle in which players insert the numbers one to nine into a grid consisting of nine squares subdivided into a further nine smaller squares in such a way that every number appears once in each horizontal line, vertical line, and square.” It was very important for me to use ArrayLists in my project, also making methods saved me a lot of time. **(THERE IS AN AUTOMATICALLY SET SUDOKU IN MY PROJECT(I found this useful to show others that it worked without needing to type in an entire Sudoku puzzle to be solved.) There are few lines where you can change if you want a manual set Sudoku project.**

```

53
54 for(int i=0; i<=8; i++) {           // Automatic System ( No need to enter numbers )
55     for(int j=0; j<=8; j++) {
56         boxes[i][j].setText(String.valueOf(array[i][j]));    // This collects all the information from my ARRAY Above and stores it
57     }
58 }
59
60 //for(int i=0; i<=8; i++) {           //Manually System ( Enter the Sudoku to be solved )
61 //for(int j=0; j<=8; j++) {
62 //    JTextField f1 = boxes[i][j]; String text = f1.getText();    // This collects all the information from my Fieldtext and is stored
63 //    array[i][j] = Integer.parseInt(text);
64 // }
65 //}
66
67 fullArray();

```

Summary

- **Testing and Development.** I have tested this program throughout many stages of development. I would see what needed to be changed in my program. If I had decided to change the Model of my program, I would have saved the code leading up to the “Final Solution”.

I will explain how I proceeded to test my program and the importance of “Stages” in development.
- **Usage of Application.** This solver may be used to test if a Sudoku puzzle does exist or not. It may also be used to give hints, as my program gives all the “POSSIBLE” values in every empty box. It also shows how many boxes are filled and how many are open to being filled.

I feel that this program is capable of solving puzzles extremely fast because it can exclude duplicate numbers. This Program also gives great opportunities for others to progress and create even harder algorithms. *etc. 10 x 10 Rubik's cube solver.*
- **Achievements I wanted.** This project was a key event to improve my programming skills. I learned how to produce and use Java functions in the most efficient way. I also tackled Java GUI building System and improved dramatically in creating objects ETC. Buttons, JField grid and Key Listeners. I was asked to pick a simpler project to program by a lot of demonstrators and other people but I rejected their advice. I picked the hardest program to solve that I found on the example list and started on the first day. I spent almost a day with paper going through the logic of my program, backtracking the key to my program but I felt that I wasn't capable of it yet. I decided to go with an elimination method that I am really proud of.

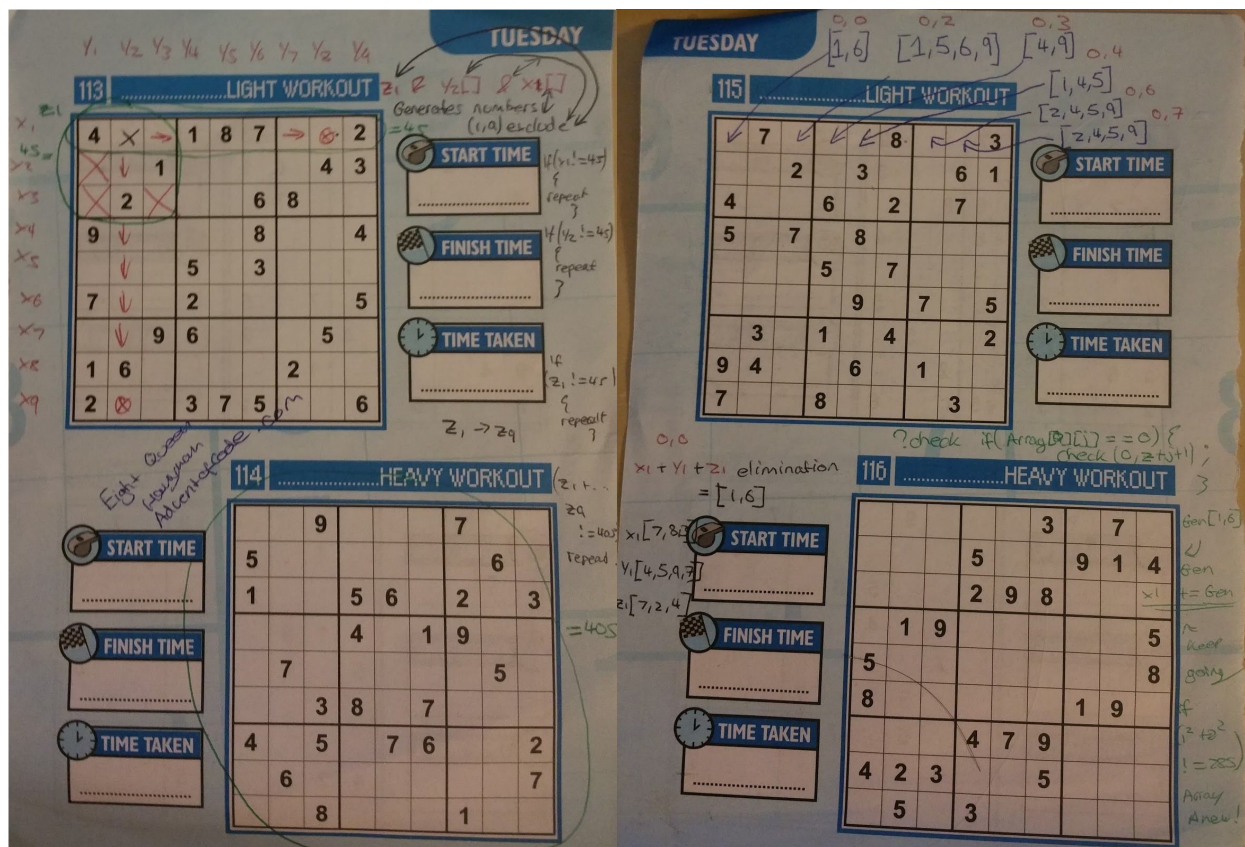
On the fifth day, I finished my Sudoku solver (10/12/16).
- **Shortcuts I took.** I created a lot of methods and classes to slim down the total amount of code that was necessary. I actually took fewer shortcuts than I expected, I made a massive method that checks 81 slots of the entire Sudoku puzzle. I had two of these, One was to return information about the entire puzzle. The other one was for the program to understand what numbers are placed where and then a method called “creating()” produced everything onto screen in blue. I have “generateNumbersGiven()”

That produces all the given Sudoku numbers in black print.

ArrayLists was really important in my program, it allowed me to add elements into ArrayLists and then remove them from [1,9] and then adding the numbers into slots.

Specification

I wanted to program something that was hard to tackle. I wanted to improve my coding skills and the way I solve my problems. The hardest program to code on the list for me was the Sudoku solver, a puzzle game I really enjoy playing. This program is used to solve Sudoku problems, it does not use the "BackTrack" method which I would had tried to use (currently trying to make one + working on other projects). This Sudoku solver uses the possible values for a certain box and only places a value if it is **certain to be there**.



This method runs through the whole 81 boxes where there are empty gaps (as 0s), and places the numbers in slots in a loop. This will happen until 81 slots are filled or the solver cannot solve. A lot of code was required for even the smallest parts.

There may be some complex code in my program, but I might not have felt it was necessary to explain it in detail because it was easier programmed than explained (but I commented every part of my entire program).

This program gives the user the possible solutions for every empty box.

This program requires the user to **understand** how to solve the "Sudoku Puzzle".

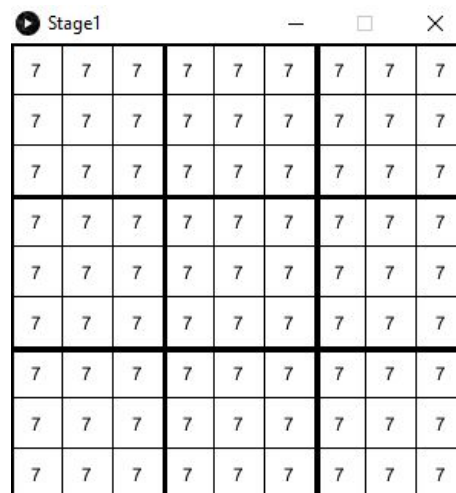
It runs on any Processing version.

Overview of the Code

Testing and Development:

Throughout my project, I had 3 stages and a final stage. These were very important because they were crucial points of developments.

#1 The first stage in my development was to create a Sudoku looking image with the ability to include numbers inside each and every box. This was a really important stage in my development because it helped me to progress into making a drawn out game than in the command prompt. I decided to go with black for the "Set Values" and blue for the solved values. These will easily be told apart. I was harsh on myself on this project... I wanted no help, no library files and ONLY Advice by some demonstrators I knew well. This was the stage where only the basic of the setup is created.



7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7
7	7	7	7	7	7	7	7	7

#2 The second stage in development, I had set the black print to work as the "Set Values" and blue print for the solved values. I also stored every row and column of code into array variables which were done by separate functions. I had tried different fonts but decided to keep the normal one. I have a function which prints the sum of the entire array, and the sum should add up to 2565 when each number was squared and added on. I printed out the Set array and "random number generated" array which gave the false solution. This was so that I knew what was going on and that I wanted to exclude some numbers that was generated and use a BACKTRACK method which I gave up on, even though I spent much effort on it. The code was quite advanced and recursion would have been too much. I set up methods for my program, this was really useful and it slimmed down my code by a lot. I set methods for creating the grids, displaying the numbers and storing variables.

```

64 int allSum = 0;
65 for(int i=0; i<=8; i++) {
66     for(int j=0; j<=8; j++) {
67         allSum = array[i][j];
68         print(allSum+ " ");
69     }
70     println();
71 }
72
73 for(int i=0; i<=8; i++) {
74     y1[i] = array[i][0];
75 } for(int i=0; i<=8; i++) {
76     y2[i] = array[i][1];
77 } for(int i=0; i<=8; i++) {
78     y3[i] = array[i][2];
79 } for(int i=0; i<=8; i++) {
80     y4[i] = array[i][3];
81 } for(int i=0; i<=8; i++) {
82     y5[i] = array[i][4];
83 } for(int i=0; i<=8; i++) {
84     y6[i] = array[i][5];
85 } for(int i=0; i<=8; i++) {
86     y7[i] = array[i][6];
87 } for(int i=0; i<=8; i++) {
88     y8[i] = array[i][7];
89 } for(int i=0; i<=8; i++) {
90     y9[i] = array[i][8];
91 }

```

```

5 3 1 5 7 4 2 9 7
6 1 4 1 9 5 5 4 6
3 9 8 4 4 4 1 6 7
8 5 2 5 6 8 9 8 3
4 5 2 8 6 3 4 9 1
7 4 6 2 2 1 6 6 6
1 6 7 7 5 5 2 8 2
7 9 9 4 1 9 8 2 5
5 7 6 2 8 4 1 7 9

```

The sum of the whole array is = 2610 must equal 2565
 First row squared + sum = 259.0 must equal 285 though
 First column squared + sum = 274.0 must equal 285 though

Stage2

5	3	1	5	7	4	2	9	7
6	1	4	1	9	5	5	4	6
3	9	8	4	4	4	1	6	7
8	5	2	5	6	8	9	8	3
4	5	2	8	6	3	4	9	1
7	4	6	2	2	1	6	6	6
1	6	7	7	5	5	2	8	2
7	9	9	4	1	9	8	2	5
5	7	6	2	8	4	1	7	9

#3 The third stage in my development was almost finished. At this stage, the solver would solve puzzles but I wanted to include buttons and a working typing field grid to work. In stage three, the program already gave a solution set to every empty box in the Sudoku puzzle. These could be used as hints, and this is how the solver's logic work, if there is a definite value for an empty box, the value is set and the loop continues until no values are left or all 81 boxes are filled.

Stage3

9	3	5	7	6	1	2	8	4
2	6	8	3	9	4	7	5	1
4	7	1	5	2	8	6	3	9
6	8	3	2	5	9	1	4	7
7	2	4	8	1	6	5	9	3
1	5	9	4	7	3	8	2	6
5	9	2	6	3	7	4	1	8
8	1	6	9	4	2	3	7	5
3	4	7	1	8	5	9	6	2

Set, and Solution set >>>>>>

```

9 0 0 0 6 1 2 8 0
2 6 8 0 0 4 7 0 0
4 0 0 5 0 8 0 3 9
0 8 0 2 5 0 1 4 0
0 0 4 8 1 0 0 9 3
1 5 9 0 0 3 0 0 6
5 0 2 0 0 7 4 0 8
0 1 0 9 4 0 0 7 5
0 4 7 1 8 0 9 0 0

```

```

1 1R 2C [3, 7]
2 1R 3C [3, 5]
3 1R 4C [3, 7]
4 1R 9C [4]
5 2R 4C [3]
6 2R 5C [3, 9]
7 2R 8C [1, 5]
8 2R 9C [1]
9 3R 2C [7]
10 3R 3C [1]
11 3R 5C [2, 7]
12 3R 7C [6]
13 4R 1C [3, 6, 7]

```

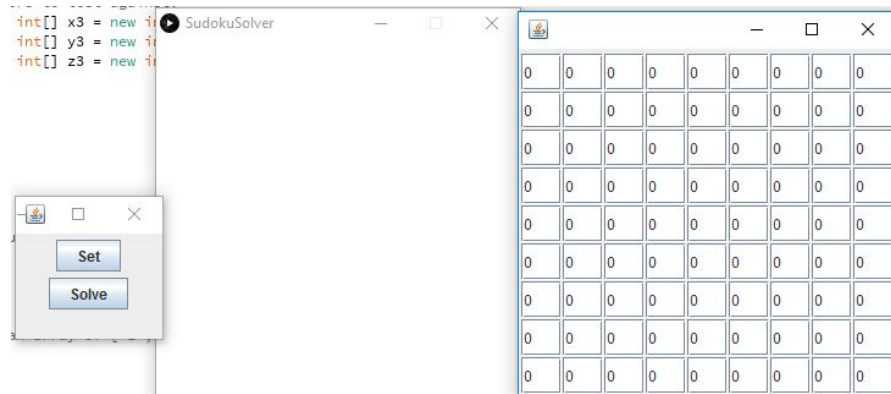
Console Errors

Final Stage

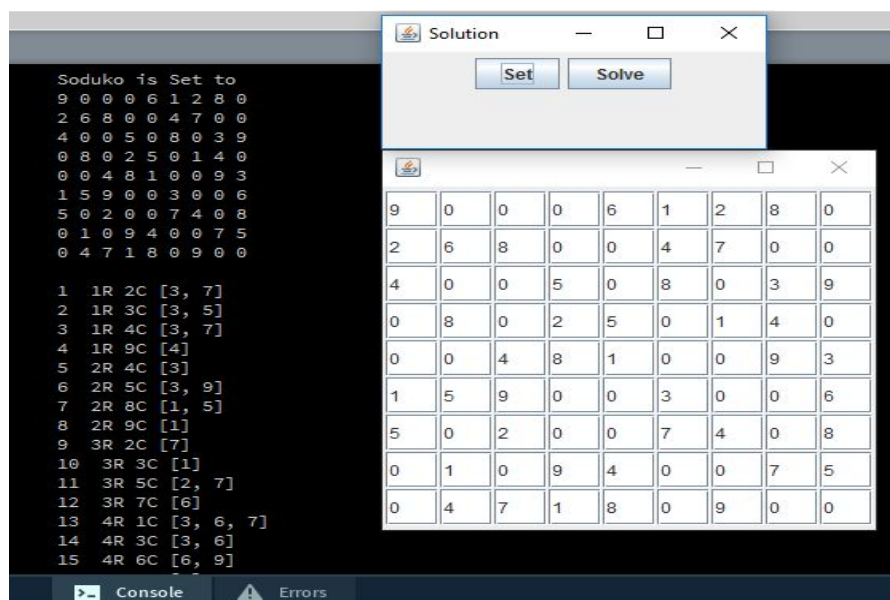
In my final stage of development, I finished creating the buttons and field grid use. I tried my best to figure out backtracking method and by using recursion to solve when there are only boxes left with "Two possible values" in that slot, the system would have produced the first number and try solve for the answer, if that wasn't the value, the other value is produced instead and the program would solve the program by trial and error. This is known as backtracking.

I will include all these stages in a separate folder.

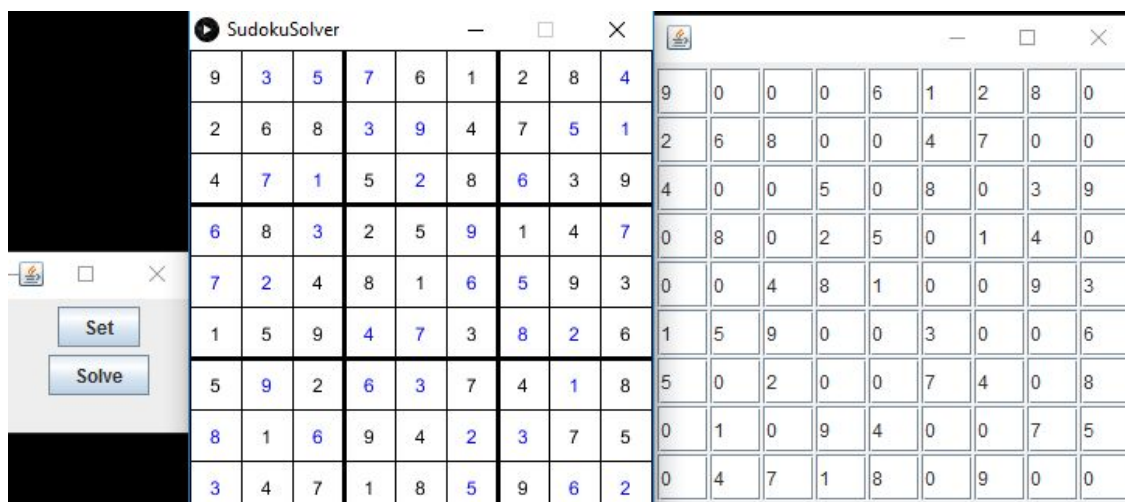
Before Set/Solve >>>



Clicking Set (Below)



Clicking solve (Below)



After Set is clicked (Below)

```

1  1R 2C [3, 7]
2  1R 3C [3, 5]
3  1R 4C [3, 7]
4  1R 9C [4]
5  2R 4C [3]
6  2R 5C [3, 9]
7  2R 8C [1, 5]
8  2R 9C [1]
9  3R 2C [7]
10 3R 3C [1]
11 3R 5C [2, 7]
12 3R 7C [6]
13 4R 1C [3, 6, 7]
14 4R 3C [3, 6]
15 4R 6C [6, 9]
16 4R 9C [7]
17 5R 1C [6, 7]
18 5R 2C [2, 7]
19 5R 6C [6]
20 5R 7C [5]
21 6R 4C [4, 7]
22 6R 5C [7]
23 6R 7C [8]
24 6R 8C [2]
25 7R 2C [3, 9]
26 7R 4C [3, 6]
27 7R 5C [3]
28 7R 8C [1, 6]
29 8R 1C [3, 6, 8]
30 8R 3C [3, 6]
31 8R 6C [2, 6]
32 8R 7C [3, 6]
33 9R 1C [3, 6]
34 9R 6C [2, 5, 6]
35 9R 8C [2, 6]
36 9R 9C [2]
Number of Open boxes is 36
Number of filled boxes is 45

```

After Solve is clicked (Below)

```

18 5R 2C [2, 7]
19 5R 6C [6]
20 5R 7C [5]
21 6R 4C [4, 7]
22 6R 5C [7]
23 6R 7C [8]
24 6R 8C [2]
25 7R 2C [3, 9]
26 7R 4C [3, 6]
27 7R 5C [3]
28 7R 8C [1, 6]
29 8R 1C [3, 6, 8]
30 8R 3C [3, 6]
31 8R 6C [2, 6]
32 8R 7C [3, 6]
33 9R 1C [3, 6]
34 9R 6C [2, 5, 6]
35 9R 8C [2, 6]
36 9R 9C [2]
Number of Open boxes is 36
Number of filled boxes is 45

Solving for the solution
.
..
...

Number of Open boxes is 0
Number of filled boxes is 81

9 3 5 7 6 1 2 8 4
2 6 8 3 9 4 7 5 1
4 7 1 5 2 8 6 3 9
6 8 3 2 5 9 1 4 7
7 2 4 8 1 6 5 9 3
1 5 9 4 7 3 8 2 6
5 9 2 6 3 7 4 1 8
8 1 6 9 4 2 3 7 5
3 4 7 1 8 5 9 6 2

```

Conclusion

I am really proud of the outcome of my project. It was a project that made me think hard, and I was able to solve a lot of complex algorithms. I have programmed everything on Processing which runs off Java. I am a bit disappointed in myself, because I felt that I haven't given this project the best of my abilities, I could had researched and finished off the entire code with a backtracking method also then creating a Sudoku Generator and as an Android app.

I have many things in mind that would improve my project, The backtracking method and to be able to make this into an android app would be some. I would love to make a Sudoku generator too. Also, I would like to program the entire program in Python.

By making this project, I already started another project which is the Eight Queen Algorithm. I have also started a chess game project as well as a Solar System replica project.