

Sam Polyakov
Project 5
3/30/2023
CS231 B

Sudoku Solver

Abstract:

This week, we created a java program that created a sudoku board and solved it. We used java classes and objects, array lists, arrays, methods, loops, abstract classes, inheritance, linked lists, iterators, and stacks to build all of the classes. In the end, my program created a sudoku board of a given size, added a given number of locked cells, and solved the board. For my extensions, I created the ability to change the size of the board, the ability to use command line arguments, I drew lines on the displayed board, and I created tests.

Results/Exporation:

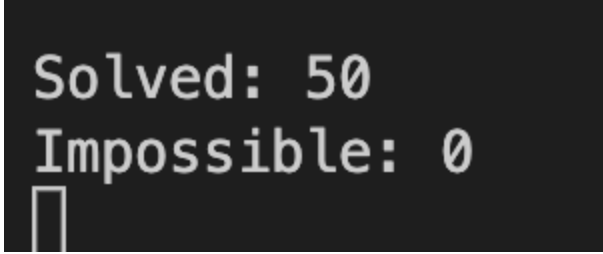
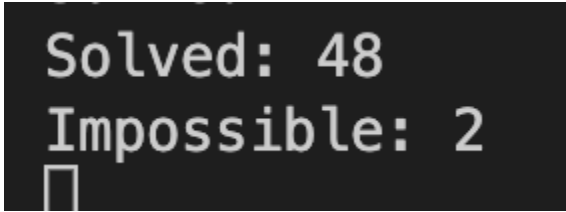
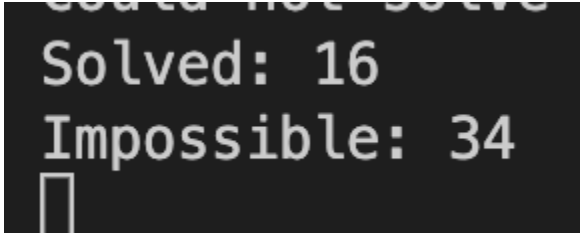
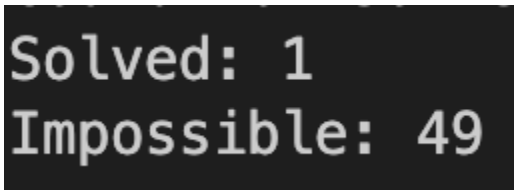
Board1 solved in display window	Board 2 solved in terminal																																																																																																												
<div><div><div></div><div></div><div></div></div><div>Sudoku</div><table><tr><td>1</td><td>2</td><td>3</td><td>4</td><td>6</td><td>5</td><td>7</td><td>8</td><td>9</td></tr><tr><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1</td><td>2</td><td>3</td></tr><tr><td>8</td><td>7</td><td>9</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td></tr><tr><td>2</td><td>1</td><td>4</td><td>3</td><td>5</td><td>6</td><td>8</td><td>9</td><td>7</td></tr><tr><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>1</td><td>2</td><td>3</td><td>4</td></tr><tr><td>3</td><td>9</td><td>8</td><td>2</td><td>4</td><td>7</td><td>5</td><td>6</td><td>1</td></tr><tr><td>9</td><td>4</td><td>2</td><td>6</td><td>1</td><td>8</td><td>3</td><td>7</td><td>5</td></tr><tr><td>6</td><td>3</td><td>1</td><td>5</td><td>7</td><td>2</td><td>9</td><td>4</td><td>8</td></tr><tr><td>7</td><td>8</td><td>5</td><td>9</td><td>3</td><td>4</td><td>6</td><td>1</td><td>2</td></tr></table><div>Hurray!</div></div>	1	2	3	4	6	5	7	8	9	4	5	6	7	8	9	1	2	3	8	7	9	1	2	3	4	5	6	2	1	4	3	5	6	8	9	7	5	6	7	8	9	1	2	3	4	3	9	8	2	4	7	5	6	1	9	4	2	6	1	8	3	7	5	6	3	1	5	7	2	9	4	8	7	8	5	9	3	4	6	1	2	<div>Solved!</div> <table><tr><td>1 2 3</td><td>4 5 6</td><td>7 8 9</td></tr><tr><td>4 5 6</td><td>7 8 9</td><td>1 2 3</td></tr><tr><td>7 8 9</td><td>1 2 3</td><td>4 5 6</td></tr><tr><td>2 1 4</td><td>3 6 5</td><td>8 9 7</td></tr><tr><td>3 6 5</td><td>8 9 7</td><td>2 1 4</td></tr><tr><td>8 9 7</td><td>2 1 4</td><td>3 6 5</td></tr><tr><td>5 3 1</td><td>6 4 2</td><td>9 7 8</td></tr><tr><td>6 4 2</td><td>9 7 8</td><td>5 3 1</td></tr><tr><td>9 7 8</td><td>5 3 1</td><td>6 4 2</td></tr></table>	1 2 3	4 5 6	7 8 9	4 5 6	7 8 9	1 2 3	7 8 9	1 2 3	4 5 6	2 1 4	3 6 5	8 9 7	3 6 5	8 9 7	2 1 4	8 9 7	2 1 4	3 6 5	5 3 1	6 4 2	9 7 8	6 4 2	9 7 8	5 3 1	9 7 8	5 3 1	6 4 2
1	2	3	4	6	5	7	8	9																																																																																																					
4	5	6	7	8	9	1	2	3																																																																																																					
8	7	9	1	2	3	4	5	6																																																																																																					
2	1	4	3	5	6	8	9	7																																																																																																					
5	6	7	8	9	1	2	3	4																																																																																																					
3	9	8	2	4	7	5	6	1																																																																																																					
9	4	2	6	1	8	3	7	5																																																																																																					
6	3	1	5	7	2	9	4	8																																																																																																					
7	8	5	9	3	4	6	1	2																																																																																																					
1 2 3	4 5 6	7 8 9																																																																																																											
4 5 6	7 8 9	1 2 3																																																																																																											
7 8 9	1 2 3	4 5 6																																																																																																											
2 1 4	3 6 5	8 9 7																																																																																																											
3 6 5	8 9 7	2 1 4																																																																																																											
8 9 7	2 1 4	3 6 5																																																																																																											
5 3 1	6 4 2	9 7 8																																																																																																											
6 4 2	9 7 8	5 3 1																																																																																																											
9 7 8	5 3 1	6 4 2																																																																																																											

Board 3 no solution in display window	Board 3 no solution in terminal																																																																																										
<table><tr><td>0</td><td>5</td><td>6</td><td>1</td><td>0</td><td>0</td><td>3</td><td>0</td><td>9</td></tr><tr><td>7</td><td>2</td><td>0</td><td>5</td><td>4</td><td>0</td><td>8</td><td>0</td><td>6</td></tr><tr><td>0</td><td>0</td><td>0</td><td>0</td><td>7</td><td>8</td><td>0</td><td>2</td><td>1</td></tr><tr><td>9</td><td>7</td><td>5</td><td>0</td><td>6</td><td>0</td><td>2</td><td>0</td><td>0</td></tr><tr><td>6</td><td>0</td><td>0</td><td>9</td><td>1</td><td>2</td><td>0</td><td>7</td><td>0</td></tr><tr><td>4</td><td>3</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>8</td></tr><tr><td>0</td><td>4</td><td>3</td><td>7</td><td>0</td><td>9</td><td>0</td><td>0</td><td>0</td></tr><tr><td>8</td><td>0</td><td>1</td><td>0</td><td>0</td><td>0</td><td>0</td><td>0</td><td>7</td></tr><tr><td>0</td><td>0</td><td>0</td><td>3</td><td>0</td><td>4</td><td>0</td><td>6</td><td>2</td></tr><tr><td colspan="3"></td><td colspan="6">No solution!</td></tr></table>	0	5	6	1	0	0	3	0	9	7	2	0	5	4	0	8	0	6	0	0	0	0	7	8	0	2	1	9	7	5	0	6	0	2	0	0	6	0	0	9	1	2	0	7	0	4	3	0	0	0	0	0	0	8	0	4	3	7	0	9	0	0	0	8	0	1	0	0	0	0	0	7	0	0	0	3	0	4	0	6	2				No solution!						<pre>Could not solve 0 5 6 1 0 0 3 0 9 7 2 0 5 4 0 8 0 6 0 0 0 0 7 8 0 2 1 9 7 5 0 6 0 2 0 0 6 0 0 9 1 2 0 7 0 4 3 0 0 0 0 0 0 8 0 4 3 7 0 9 0 0 0 8 0 1 0 0 0 0 0 7 0 0 0 3 0 4 0 6 2</pre>
0	5	6	1	0	0	3	0	9																																																																																			
7	2	0	5	4	0	8	0	6																																																																																			
0	0	0	0	7	8	0	2	1																																																																																			
9	7	5	0	6	0	2	0	0																																																																																			
6	0	0	9	1	2	0	7	0																																																																																			
4	3	0	0	0	0	0	0	8																																																																																			
0	4	3	7	0	9	0	0	0																																																																																			
8	0	1	0	0	0	0	0	7																																																																																			
0	0	0	3	0	4	0	6	2																																																																																			
			No solution!																																																																																								

These results definitely make sense. When I had no or very few locked values, the board solved perfectly. When I started adding lots of locked values, the board could no longer solve. One thing that I noticed was that if there were no or very few locked values, the board solved very quickly. Once you started adding more(10+) locked values, the board took significantly longer to solve. If you added a very significant amount of values, the board would realize it was not solvable very quickly. This makes sense as once you add more preset values, the board is more likely to need to go back into the stack.

Exploration:

Unsurprisingly, I found that the more cells were locked, the less likely the boards were to be solved. The drop off in solved boards between 10 and 20 locked cells was somewhat surprisingly small, but definitely believable as there is still a relatively small percentage of the cells on the board that are pre-filled in.

10 cells locked	
20 cells locked	
30 cells locked	
40 cells locked	

Reflection:

This week, we focused on using stacks. Stacks are used for “first in, last out” , like creating a stack of sudoku cells and adding and removing cells from the stack as you solve, with the cell at the (0,0) coording being at the bottom of the stack as it was the first cell checked. We used this to do depth first search, which was relatively efficient to use for this project. We used this so the solve method could traverse through the sudoku board, and backtrack if there was no valid value for a certain cell. One way this could potentially be improved is instead of going to the next cell in the grid, to instead go to the cell that has the least number of possible values.

Extensions:

This week, I did a few extensions. First, I added the ability to change the size of the board. I did this by modifying all of the methods to be able to take in a size variable other than 9. One issue I faced with this was that values above 9 would be drawn as different symbols instead of numbers as the ascii values did not represent values above 9. I solved this by drawing a 1 if the value was between 10 and 19, and a 2 if the value was between 20-25. I then drew the second digit of the value slightly to the right of the 10 or 20. This can be seen in the draw method below. I did not allow board sizes above 25x25 because after this they would become too large and difficult to manage.

```
public void draw(Graphics g, int x, int y, int scale){
    // draws each cell
    if(getValue() <= 9){
        char toDraw = (char) ((int) '0' + getValue());
        g.setColor(isLocked()? Color.BLUE : Color.RED);
        g.drawChars(new char[] {toDraw}, 0, 1, x, y);
    }
    else if(getValue() > 9 && getValue() <= 19){
        char toDraw1 = (char) ((int) '1');
        char toDraw2 = (char) ((int) '0' + getValue()-10);
        g.setColor(isLocked()? Color.BLUE : Color.RED);
        g.drawChars(new char[] {toDraw1}, 0, 1, x, y);
        g.drawChars(new char[] {toDraw2}, 0, 1, x+8, y);
    }
    else{
        char toDraw1 = (char) ((int) '2');
        char toDraw2 = (char) ((int) '0' + getValue()-20);
        g.setColor(isLocked()? Color.BLUE : Color.RED);
        g.drawChars(new char[] {toDraw1}, 0, 1, x, y);
        g.drawChars(new char[] {toDraw2}, 0, 1, x+8, y);
    }
}
```

16x16 Board Solved

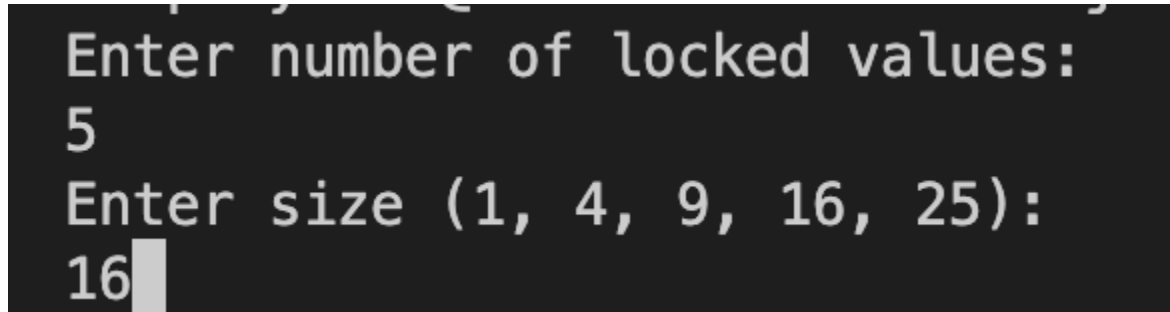
Sudoku															
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
5	6	7	8	1	2	3	4	13	14	15	16	9	10	11	12
9	10	11	12	13	14	15	16	1	2	3	4	5	6	7	8
13	14	15	16	9	10	11	12	5	6	7	8	1	2	3	4
2	1	4	3	6	5	8	7	10	9	12	11	14	13	16	15
6	5	8	7	2	1	4	3	14	13	16	15	10	9	12	11
10	9	12	11	14	13	16	15	2	1	4	3	6	5	8	7
14	13	16	15	10	9	12	11	6	5	8	7	2	1	4	3
3	4	1	2	7	8	5	6	11	12	9	10	15	16	13	14
7	8	5	6	3	4	1	2	15	16	13	14	11	12	9	10
11	12	9	10	15	16	13	14	3	4	1	2	7	8	5	6
15	16	13	14	11	12	9	10	7	8	5	6	3	4	1	2
4	3	2	1	8	7	6	5	12	11	10	9	16	15	14	13
8	7	6	5	4	3	2	1	16	15	14	13	12	11	10	9
12	11	10	9	16	15	14	13	4	3	2	1	8	7	6	5
16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1

Solve with symbols

Sudoku															
1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	@
5	6	7	8	1	2	3	4	=	>	?	@	9	:	;	<
9	:	;	<	=	>	?	@	1	2	3	4	5	6	7	8
=	>	?	@	9	:	;	<	5	6	7	8	1	2	3	4
2	1	4	3	6	5	8	7	:	9	<	;	>	=	@	?
6	5	8	7	2	1	4	3	>	=	@	?	:	9	<	;
:	9	<	;	>	=	@	?	2	1	4	3	6	5	8	7
>	=	@	?	:	9	<	;	6	5	8	7	2	1	4	3
3	4	1	2	7	8	5	6	;	<	9	:	?	@	=	>
7	8	5	6	3	4	1	2	?	@	=	>	;	<	9	:
;	<	9	:	?	@	=	>	3	4	1	2	7	8	5	6
?	@	=	>	;	<	9	:	7	8	5	6	3	4	1	2
4	3	2	1	8	7	6	5	<	;	:	9	@	?	>	=
8	7	6	5	4	3	2	1	@	?	>	=	<	;	:	9
<	;	:	9	@	?	>	=	4	3	2	1	8	7	6	5
@	?	>	=	<	;	:	9	8	7	6	5	4	3	2	1

Another extension I did was adding lines to separate the subgrids within the sudoku grid. I did this by using the java graphics line method. It was somewhat difficult to figure out where to draw the lines as I changed the size of the grids, but I figured out an equation that did it. The before and after can be seen above.

Another extension I added was the ability to use command line arguments. I made it so that you can use command line arguments to enter the number of locked cells and to change the size of the grid.



For my final test, I created high quality tests for LinkedList, Cell, Board, and Sudoku. These tests tested all of the methods within each class.

Cell Tests

```
sampolyakov@MacBook-Pro-83 Project5 % java -ea CellTests
0 == 0
5 == 5
10 == 10
true == true
false == false
2 == 2
0 == 0
3 == 3
0 == 0
4 == 4
-1 == -1
false == false
true == true
```

LinkedList Tests

```
● sampolyakov@MacBook-Pro-83 Project5 % java -ea LinkedListTests
    != null
    5 == 5
    6 == 6
    0 == 0
    true == true
    true == true
    false == false
    true == true
    false == false
    false == false
    false == false
    0 == 0
    3 == 3
    4 == 4
    true == true
    false == false
    0 == 0
    1 == 1
    0 == 0
    4 == 4
    7 == 7
    0 == 0
    1 == 1
    2 == 2
    3 == 3
    4 == 4
    5 == 5
    Stack: 4 == 1
    Stack: 4 == 1
    Stack: 3 == 2
    Queue: 1 == 1
    Queue: 1 == 1
    Queue: 2 == 2
    Stack: 1 == 5
    1 == 3
    Done testing
```

Board Tests:

```
Board 1:
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0

Cell at row 0, col 0: Row: 0
Col: 0
Val: 0
Cell at row 4, col 5: Row: 4
Col: 5
Val: 0

Board 2:
0 0 0 3 0 1 0 0 0
0 0 0 0 0 0 0 0 0
9 0 0 0 0 0 0 0 0

0 0 0 0 0 0 9 0 0
0 0 6 0 0 0 0 0 7
2 0 0 0 0 0 0 0 0

0 0 0 0 5 0 0 0 0
0 0 0 0 7 0 8 0 0
0 0 0 0 0 0 0 0 0

Is cell at row 0, col 0 locked? false
Is cell at row 2, col 0 locked? true
```

```
Board 2:
0 0 0 3 0 1 0 0 0
0 0 0 0 0 0 0 0 0
9 0 0 0 0 0 0 0 0

0 0 0 0 0 0 9 0 0
0 0 6 0 0 0 0 0 7
2 0 0 0 0 0 0 0 0

0 0 0 0 5 0 0 0 0
0 0 0 0 7 0 8 0 0
0 0 0 0 0 0 0 0 0

Number of locked cells on Board 2: 10

Board 2:
0 0 0 3 0 1 0 0 0
0 0 0 0 0 0 0 0 0
9 0 0 0 0 0 0 0 0

0 0 0 0 0 0 9 0 0
0 0 6 0 0 0 0 0 7
2 0 0 0 0 0 0 0 0

0 0 0 0 5 0 0 0 0
0 0 0 0 7 0 8 0 0
0 0 0 0 0 0 0 0 0

After changing cell at row 0, col 0:

2 0 0 3 0 1 0 0 0
0 0 0 0 0 0 0 0 0
9 0 0 0 0 0 0 0 0

0 0 0 0 0 0 9 0 0
0 0 6 0 0 0 0 0 7
2 0 0 0 0 0 0 0 0

0 0 0 0 5 0 0 0 0
0 0 0 0 7 0 8 0 0
0 0 0 0 0 0 0 0 0
```

```
Board 2:
0 0 0 3 0 1 0 0 0
0 0 0 0 0 0 0 0 0
9 0 0 0 0 0 0 0 0

0 0 0 0 0 0 9 0 0
0 0 6 0 0 0 0 0 7
2 0 0 0 0 0 0 0 0

0 0 0 0 5 0 0 0 0
0 0 0 0 7 0 8 0 0
0 0 0 0 0 0 0 0 0

Value at cell row 0, col 1: 0

Board 2:
0 0 0 3 0 1 0 0 0
0 0 0 0 0 0 0 0 0
9 0 0 0 0 0 0 0 0

0 0 0 0 0 0 9 0 0
0 0 6 0 0 0 0 0 7
2 0 0 0 0 0 0 0 0

0 0 0 0 5 0 0 0 0
0 0 0 0 7 0 8 0 0
0 0 0 0 0 0 0 0 0

Number of rows on Board 1: 9
Number of columns on Board 1: 9
```

```
Board 4:
1 4 7 0 0 0 0 0 0
2 5 8 0 0 0 0 0 0
3 6 9 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

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

0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0

0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0
```

Sudoku Tests

```
● sampolyakov@MacBook-Pro-83 Project5 % java -ea SudokuTests
This cell is locked.
Value can't be: 9
3 != 9
This cell is locked.
This cell is locked.
Warning: the fonts "Times" and "Times" are not available fo
es" font to remove this warning.
3 != 0
6 != 0
```

References:

This week, I worked with Dave Boku and got help from Max Bender at office hours. I also used ChatGPT to help myself understand some concepts.