

CS21120: Data Structures and Algorithm Analysis
Assignment 1 – Sudoku

dac46@aber.ac.uk
25/10/13

Description Of Solver Algorithms

Main loop:

1. Do 10 times:
 2. Update pencil marks for each cell
 3. Solve Pointing Pairs for Rows and Columns
 4. Update pencil marks for each cell
 5. Solve Naked Pairs for Rows, Columns and Squares
6. Do 10 times:
 7. Update pencil marks for each cell
 8. Solve Hidden Singles for Rows, Columns and Squares
 9. Solve Naked Singles for Rows, Columns and Squares

Pointing Pairs:

1. For every block:
 2. For every possible number in a block (1-9):
 3. Get the number of times that number occurs in the block
 4. Check if all those occurrences are all in a different block together
 5. Remove all occurrences of the number from the pencil marks in the original block EXCEPT those that also appear in the second block

Naked Pairs:

1. For every block:
 2. Get every cell that has 2 pencil marks in
 3. For every possible pair of those:
 4. Check if they have the same set of pencil marks
 5. If they do:
 6. For each pencil mark they have the same:
 7. Remove all occurrences of the number from the pencil marks in the original block EXCEPT the two cells in the pair

Hidden Singles:

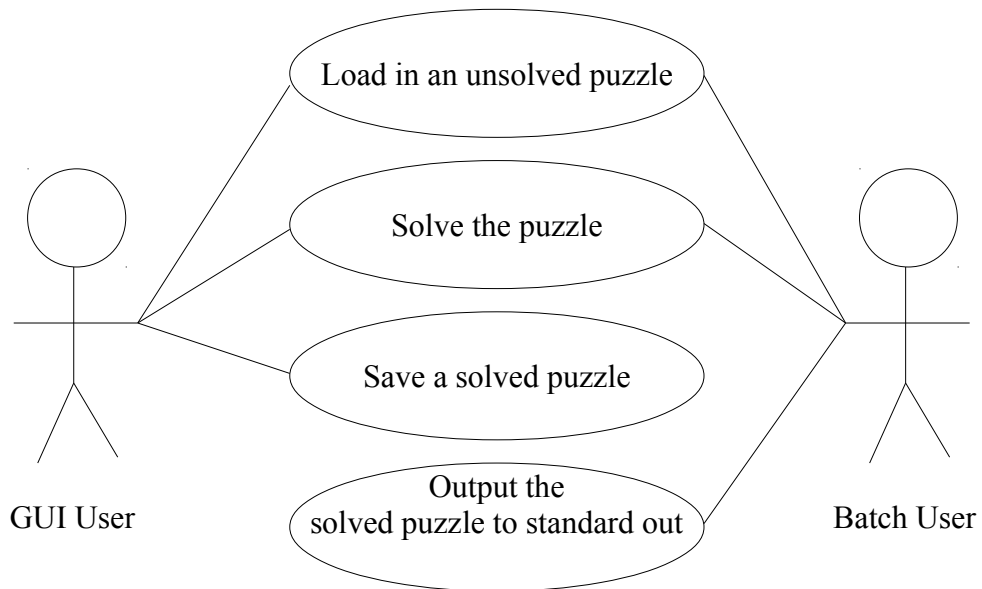
1. For every possible number(0-9):
 2. For every block:
 3. Get all occurrences of the number in the block
 4. If there is only one occurrence:
 5. Set the value of that square to be the number

Naked Singles:

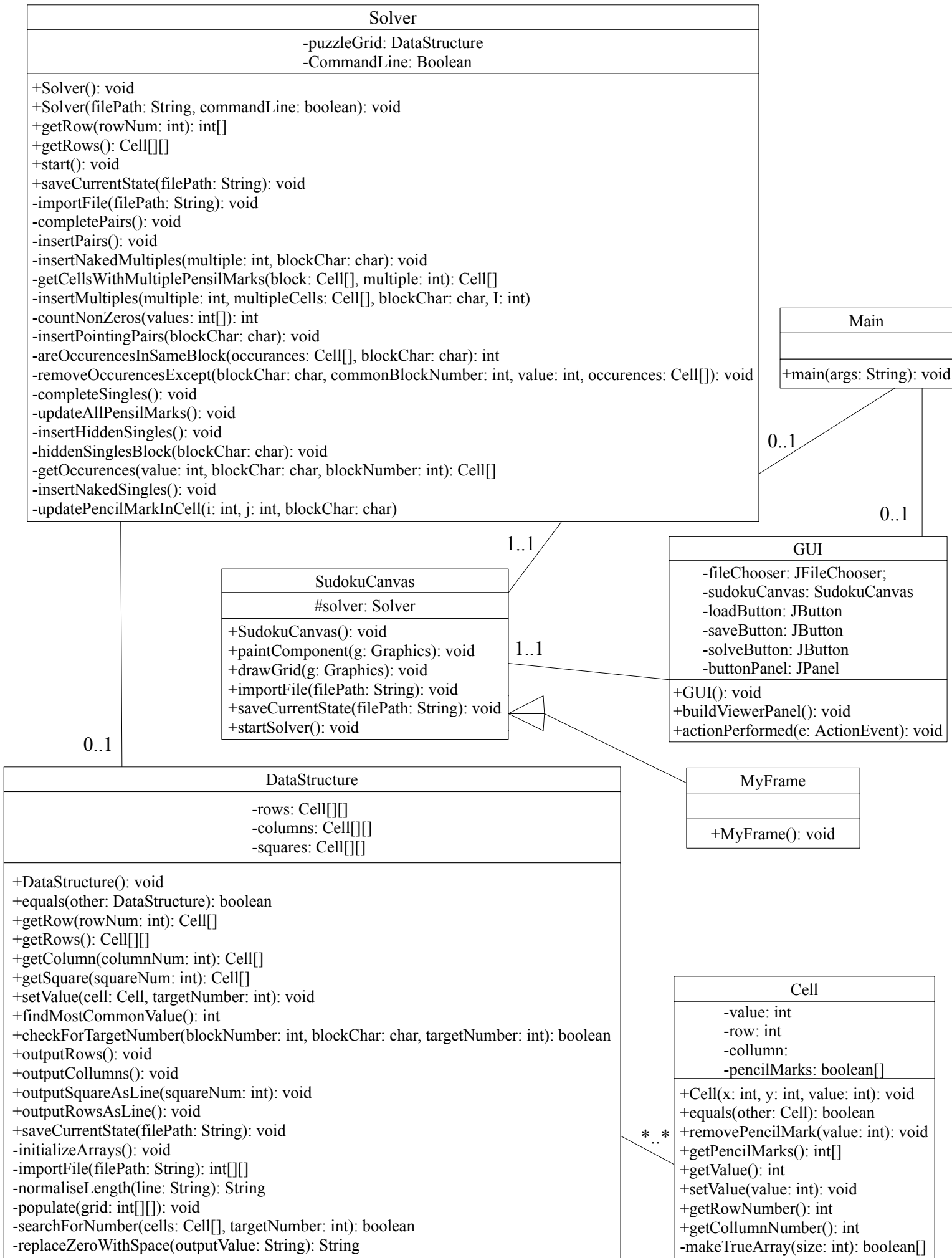
1. For every cell in the grid:
 2. If there is only one pencil mark:
 3. Set the value of the square to be that number

UML Diagrams

Use Case Diagram:



Class Diagram:



Design Decisions

I decided to use 3 2D arrays for my data structure, which all contain the same data, just in a different order. This means that it is very easy to simply traverse each row, each column or each square.

I decide to have each Cell object hold each cell, which would hold the row and column it's in, as well as it's value and the pencil marks for that cell.

For the GUI I decided to use the 3 most distinct colours, Red(255,0,0) for the solved numbers, Green(0,255,0) for the square colour, and Blue(0,0,255) for the pencil marks. It doesn't look amazingly user friendly, but it's good for debugging.

Testing

Test Plan:

I will use both Black Box and unit testing (via JUnit) to test that my code works, is robust and is maintainable.

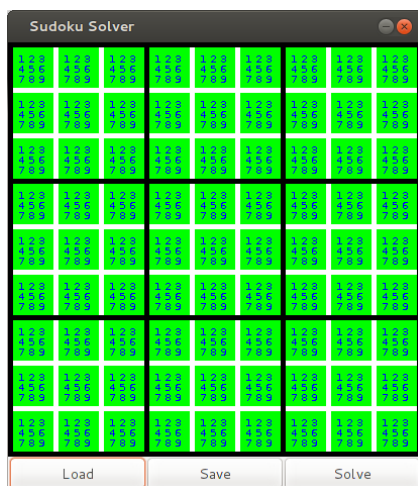
Black Box:

Test No	Test Description	Input	Expected Output	Actual Output	Pass/Fail	Evidence
1	Start program	Start the program	Program Starts	Program Starts	Pass	1
2	Load in correct data via gui	Click Load Choose web.sud	Loaded puzzle is displayed on screen	Loaded puzzle is displayed on screen	Pass	2
3	Load in wrong data via gui	Click Load Try to choose Main.class	Unable to find the file in the file finder	Unable to find the file in the file finder	Pass	3
4	Solve puzzle: book70.sud	Load the puzzle Solve it	Solve puzzle displayed	Solve puzzle displayed	Pass	4
5	Solve puzzle: web.sud	Load web.sud Solve it	Solve puzzle displayed	Solve puzzle displayed	Pass	5
6	Solve puzzle: book58.sud	Load book58.sud Solve it	Solve puzzle displayed	Solve puzzle displayed	Pass	6
7	Solve puzzle: book62.sud	Load book62.sud Solve it	Solve puzzle displayed	Only partially solved	Fail	7
8	Save the solved sudoku	Load web.sud Solve it Save it to web1.sud	Outputted file represents solved grid correctly	File created, but empty, after intensive debugging, I cannot solve the problem	Fail	8
9	Load in single	Run "java	Solved sudoku is		Pass	9

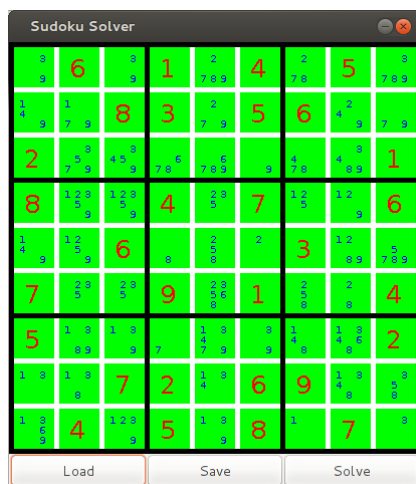
	correct data via command line	Main ../data/web.su d”	outputted to standard out as a line			
10	Load in single incorrect data via command line	Run “java Main ../Main.class”	Program ends		Pass	10
11	Load in multiple correct data via command line	Run “java Main ../data/*”	Solved sudokus outputted as lines		Pass	11

Evidence:

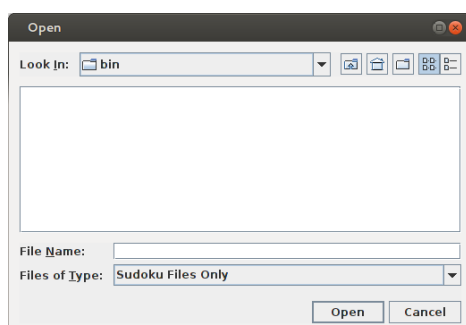
1.



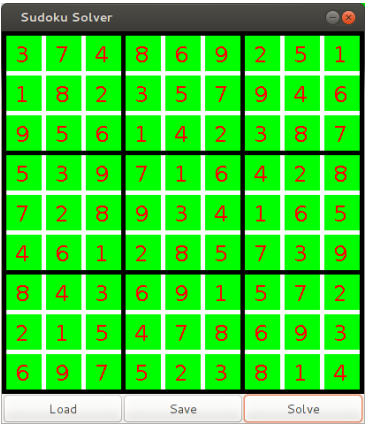
2.



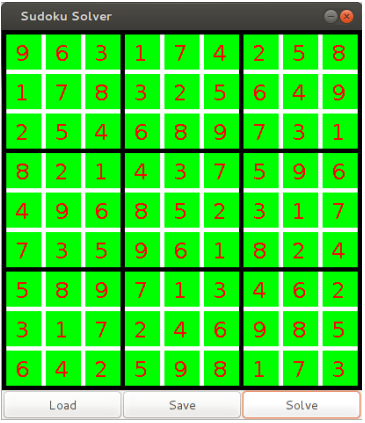
3.



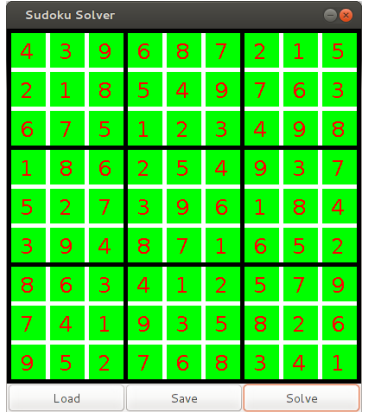
4.



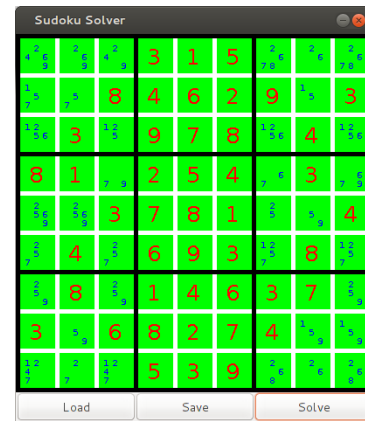
5.



6.



7.



8.

```
daniel@daniel-laptop: ~/Git/Sudoku-Solver/data
"web1.sud" 1L, 1C
1,0-1 All
```

9.

```
daniel@daniel-laptop: ~/Git/Sudoku-Solver/bin
daniel@14:57
/home/daniel/Git/Sudoku-Solver/bin>>java Main ../data/web.sud
963174258178325649254689731821437596496852317735961824589713462317246985642598173

daniel@14:57
/home/daniel/Git/Sudoku-Solver/bin>>

Unit tests inside the default package which all pass

timings/improvements

e to solve 16/18 set puzzles correctly as well as displaying a reasonably good
e user. Also it has the added bonus of being able to be run in batch from the
wall I think, those two were my main objectives for the program, so it was a
```

10.

```
daniel@daniel-laptop: ~/Git/Sudoku-Solver/bin
daniel@14:59
/home/daniel/Git/Sudoku-Solver/bin>>java Main ./Main.class

daniel@14:59
/home/daniel/Git/Sudoku-Solver/bin>>

Unit tests inside the default package which all pass

timings/improvements

e to solve 16/18 set puzzles correctly as well as displaying a reasonably good
e user. Also it has the added bonus of being able to be run in batch from the
```

11.

```
daniel@daniel-laptop: ~/Git/Sudoku-Solver/bin
daniel@15:00
/home/daniel/Git/Sudoku-Solver/bin>>java Main ../data/*
74 2 8 1 45 4 3 8 5 2 4 2 8 3 7 6 5 4 9 6 1 6 97
596317284138426975724985136613279548972854613485163729859732461347691852261548397
295346187387291654416578932974825316621439578538617249753964821162783495849152763
734528196965317824812964375586273419327149568149856237273685941498731652651492783
178629435624573198953841276762418359391752684845936712439287561516394827287165943
617925483543871962298346571154689327376452819982713654465237198829164735731598246
37486925118235784695614287539716426728934165461285739843691572215478693697523814
418896527572814036369725418634289175186473269927561384246397851851642793793158642
215469873749238165638175942194526387852317496376984521923641758567893214481752639
273591486891462357456837921927643815184259763365718294548976132632184579719325648
267935184589714263314826759875492316491673825623158947146389572732541698958267431
834972561162458739597613248318749652649125387725836194451387926276594813983261475
963174258178325649254689731821437596496852317735961824589713462317246985642598173

daniel@15:00
/home/daniel/Git/Sudoku-Solver/bin>>
```

JUnit:

There are a set of JUnit tests inside the default package which all pass

Conclusion/Shortcomings/Improvements

My program is able to solve 16/18 set puzzles correctly as well as displaying a reasonably good user interface to the user. Also it has the added bonus of being able to be run in batch from the command line. Overall I think those two were my main objectives for the program, so it was a success.

The program will not save correctly for an unknown reasons, the advisors figure it out either.

The program only decides it's finished after it has looped 10 times, which is a really inefficient way of doing it, if I had more time I would have implemented something like the following:

```
while(startStateOfGrid != currentStateOfGrid){  
    startStateOfGrid = currentStateOfGrid;  
    solveTheSudoku;  
}
```