**CS211 Sudoku Solver Assignment**

Martin Zokov

User id: mvz

This submission is my work, except where clearly indicated.

I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree. I have read the sections on unfair practice in the Students’ Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.

I understand and agree to abide by the University’s regulations governing these issues.

**Contents**

1. Classes

1a. Diagram

1b. Discussion of design choices

1. Algorithms

2a. Finding candidates

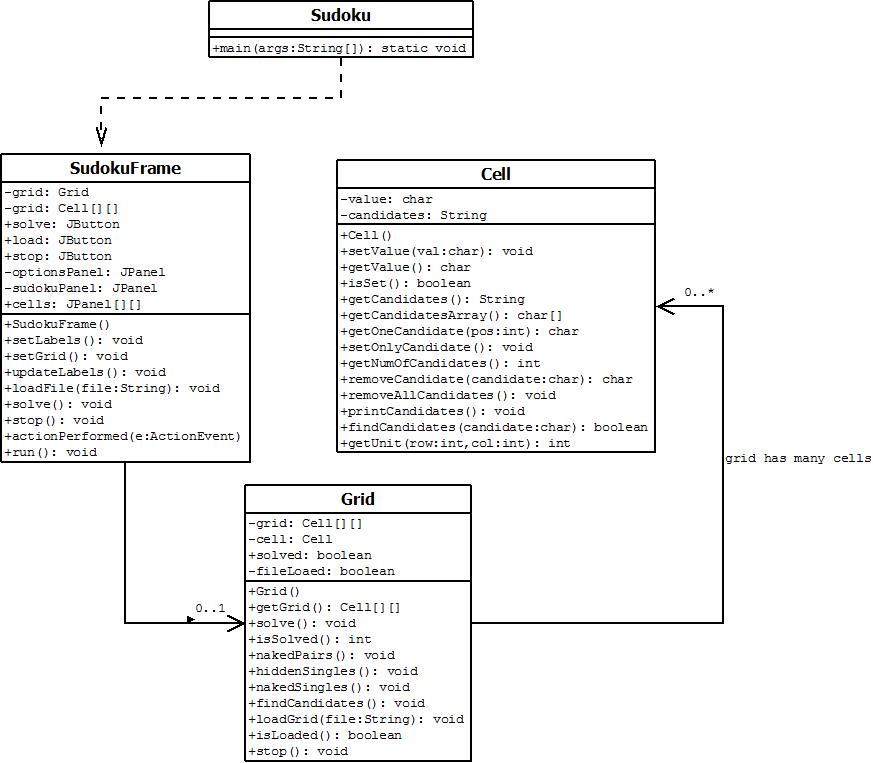
2b. Single Candidates

2c. Hidden Candidates

2d. Naked Pairs

1. Testing
2. **Classes**

1a. A class diagram of the implemented design for the application:



1b. Discussion of design choices

**Cell class**

The Cell class was a choice I made because it is easier to keep track of every single cell, its value and the possible candidates for that cell. A 2D array of chars, strings or integers would not be enough to easily manipulate the grid in an efficient way.  
The value of a cell is stored as a single char variable, because it takes less memory, better suited for holding only one number and it eliminates the possibility of additional characters added to the value of a cell.  
The candidates are stored in a string. First I considered a char array, but the string has more flexible possibilities for manipulation and is basically an array of chars as well. The String is initialized with the numbers from 1 to 9 and then numbers are removed from specific position and replaced with a space character.

**Grid class**

The actual grid is represented by a 9x9 2D array of Cell objects. When the grid is initialized, a Cell is created for each of the array positions. When a file is loaded the cell objects have their value set to the ones loaded. The actual solving algorithms are implemented through methods in this class.

**SudokuFrame class**

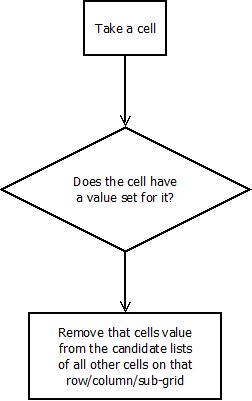
This class created the GUI for the program and starts up the Grid class and initiates it. The class implements Runnable to create a new thread in which the solver algorithms are run.

1. **Algorithms**

In order to solve the Sudoku puzzles I implemented four algorithms. One for finding the candidates of each empty cell, one which checks for a cell with only one candidate, one to check a row/column/sub-grid for only one candidate and a naked pairs implementation.

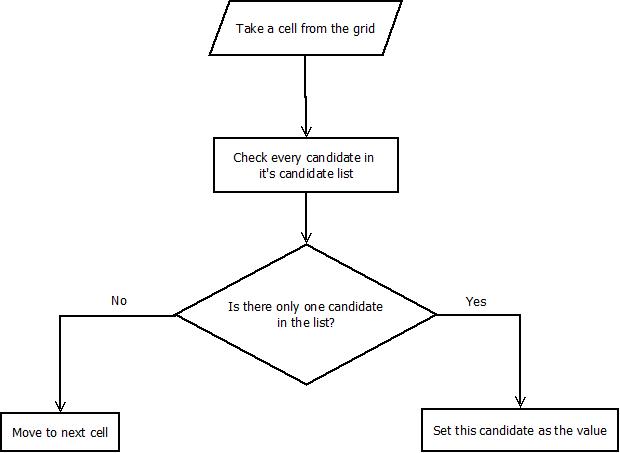
2.a Finding candidates

The algorithm goes through every cell on the grid. Compares a cell to all the cells on its row/column/sub-grid and if a value is set for it, the algorithm removes it from the candidates list of all other cells. If the cell already has a value, removes all of the candidates for that cell.  
The flowchart representation of this algorithm is:



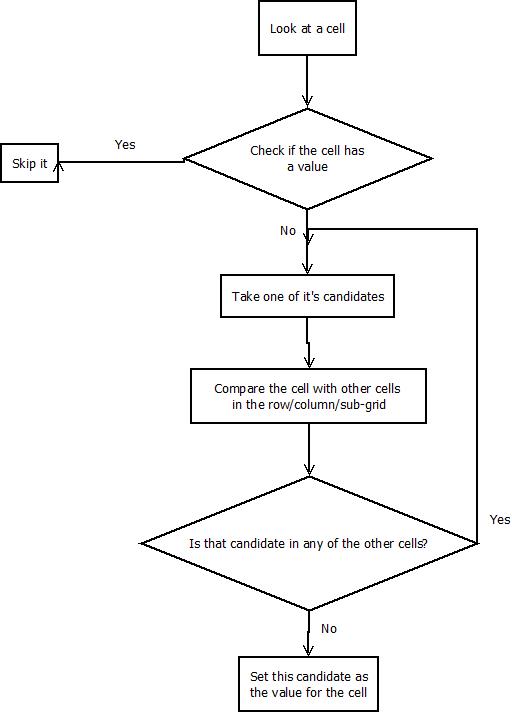
2b. Single candidates

It is a basic algorithm which goes through every cell in the grid. The Cell class has a method which checks if there is only one candidate in the string with candidates. If that is the case, the candidate is set as the cell’s value.



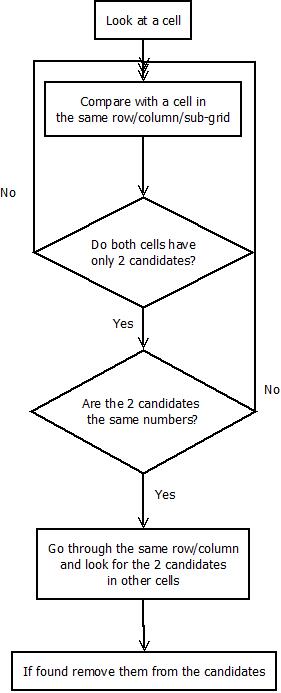
2c. Hidden singles

An algorithm that is a bit more complex. The logic of this method looks at a single candidate at a time and checks a whole row/column/sub-grid for the number of occurrences of that candidate. If a candidate is found only once on a row/column or in one sub-grid, then this candidate must be in the cell that it is found in.



2d. Naked pairs

This algorithm looks for two cells in the same unit (row/column/sub-grid) and compares the number of candidates they have as well as the candidates. If they have the same candidates, then the two number must be in these two cells, which means that these candidates can be removed from all other cells in the same unit.



1. Testing

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| ID | Description | Input | Expected output | Pass/Fail | Comments |
| T1 | Solving a puzzle | File “book55.sud” | Solved puzzle | Pass | Puzzle is solved without mistakes |
| T2 | Solving a puzzle | File “book56.sud” | Solved puzzle | Fail | The algorithms start solving and the values for some cells but not all |
| T3 | Solving a puzzle | File “book57.sud” | Solved puzzle | Pass | Puzzle is solved without mistakes |
| T4 | Solving a puzzle | File “book58.sud” | Solved puzzle | Pass | Puzzle is solved without mistakes |
| T5 | Solving a puzzle | File “book62.sud” | Solved puzzle | Fail | The algorithms start solving and the values for some cells but not all |
| T6 | Solving a puzzle | File “book63.sud” | Solved puzzle | Fail | Only one cell is solved |
| T7 | Solving a puzzle | File “book64.sud” | Solved puzzle | Fail | Partially solved the puzzle with a few missing cells |
| T8 | Solving a puzzle | File “book65.sud” | Solved puzzle | Fail | Partially solved the puzzle with a few missing cells |
| T9 | Solving a puzzle | File “book67.sud” | Solved puzzle | Fail | Some cells are solved with only few left blank |
| T10 | Solving a puzzle | File “book68.sud” | Solved puzzle | Pass | All cells are solved |
| T11 | Solving a puzzle | File “book69.sud” | Solved puzzle | Fail | Only one solved cell is found for the puzzle |
| T12 | Solving a puzzle | File “book70.sud” | Solved puzzle | Fail | Partially solved the puzzle with a few missing cells |
| T13 | Solving a puzzle | File “guardian.sud” | Solved puzzle | Pass | All cells solved correctly |
| T14 | Solving a puzzle | File “newone.sud” | Solved puzzle | Pass | All cells solved correctly |
| T15 | Solving apuzzle | File ”prize352.sud” | Solved puzzle | Pass | All cells solved correctly |
| T16 | Solving a puzzle | File “simple349.sud” | Solved puzzle | Pass | All cells solved correctly |
| T17 | Solving a puzzle | File “simple351.sud” | Solved puzzle | Pass | All cells solved correctly |
| T18 | Solving a puzzle | File “web.sud” | Solved puzzle | Pass | All cells solved correctly |
| T19 | Loading a file after a solved puzzle | A file different than the one currently loaded | Replace grid values | Fail | GUI bug happens when a new file is loaded and doesn’t display values properly |
| T20 | Pausing the solving of a puzzle | Pause button press | Algorithm should stop solving and wait for another press of the “Solve” button before continuing | Pass | Solver algorithm pauses |

Only 10 out of the 18 given Sudoku puzzles are solved because the implemented algorithms are not enough to finish the other puzzles.