# CS 203 HW #6 — Mini Sudoku

**Due Date:** Wednesday, Friday March 1st at 11:00pm

In this homework you will implement a simplified version of Sudoku game in which board size is 4x4. To learn more about Sudoku and play the game, you can go to <https://1sudoku.com/play/sudoku-kids-free/sudoku-4x4/>

### Grading

Your assignment grade for this assignment will be based on:

* Program functionality (90%)
* Code quality (10%)

### Getting Started

* Download the starter code from Moodle.
* Open the BlueJ project and examine the MiniSudoku class' current source code. Examine the constants and instance variables that define the Sudoku game. Most notably, board is a two-dimensional array of int that defines the contents of the playing board

**Specification**

You must complete the following methods in the starter code:

|  |  |
| --- | --- |
| completeRow | Determines if the given row contains numbers 1 thru NUM\_ROWS, not necessarily in that order, if so, return true and false otherwise (15 points) |
| completeCol | Determines if the given column contains numbers 1 thru NUM\_COLS, not necessarily in that order, if so, return true and false otherwise (15 points) |
| completeBlock | Given a block number, determine if the block has all the needed numbers. Return true if it does and false otherwise (15 points) |
| validMove | Determines whether the given location is a legal space for a new value (15 points) |
| CompleteBoard | Determines if the solution is valid and the board is completed. i.e. all rows have each number, all cols have each number, and all blocks have each number (10 points) |
| playGame | Plays the Sudoku game. This contains the while loop (20 points) |

More details on each of these methods is in the comment headers provided in the starter code.

**Additional Enrichment**

1. Expand the game into a 9x9 Sudoku. If you want to try this, save a copy of the completed assignment before you upgrade. Be sure to explain what you've done in the comment header at the top of the file.
2. The starter code creates a random board by adding holes to pre-generated hardcoded full Sudoku. Implement a method to randomly generate a full Sudoku, then add holes as needed.
3. Have three levels of difficulties of the game. Easy, medium and hard. Define your roles and implement them.
4. Use JavaFx libraries to create a GUI for this game.
5. Create a solver for your Sudoku. The solver method will solve the game and find a complete solution to your Sudoku.
6. Think of any other improvements that can be added to the game. If you want to implement this improvement, please document it in the comment header.

**Code Quality (10%)**

A good computer program not only performs correctly, it also is easy to read and understand:

* A comment at the top of the program includes the name of the program, a brief statement of its purpose, your name, and the date that you finished the program.
* The comment header should also include a list of known bugs or deficiencies, or a statement that the program has no known deficiencies.
* Variables have names that indicate the meaning of the values they hold.
* Code is indented consistently to show the program’s structure.
* The body of if and else clauses are enclosed in braces and indented consistently, even if they consist of only a single line of code.
* Opening braces are placed consistently, either at the end of the if-statement or directly under the ‘i’ of if or the ‘e’ of else. Closing braces are in the same column as the ‘i’ of if or the ‘e’ of else.
* Within the code, major tasks are separated by a blank line and prefaced by one or more single-line comments identifying the task.
* Methods are separated by blank lines and prefaced by a multi-line comment describing what they do and what their parameters mean. (See starter code for examples.)
* Very long statements (such as long print statements or complex boolean expressions) are broken across lines and indented to show their structure.
* Now that you are familiar with loops and methods, your code should not contain redundant or repeated sections.
* Your program does not contain extraneous or commented out code. It does not contain out-of-date or irrelevant comments. For example, you should not have comments that say “//you need to implement this method” if you have indeed implemented that method
* ***Make use of the provided constants where appropriate. Don't use a literal when a constant is available.***

**Logistics**

To begin the assignment, download the starter code (MiniSudoku.java) from the course website. Create a new BlueJ project. Then drag and drop MiniSudoku.java into the project.

**Turning in this Assignment**

You are responsible for turning in your homework assignments properly.

* Be sure your name is in the comment header at the top of your .java file.
* Remember to include comments to remind yourself and your reader the functionality of the different parts of your program.
* If you did any of the Additional Enrichment (above), carefully document this functionality in the comment header at the top of your .java file.
* Zip up your entire BlueJ project folder and submit the zip file via Moodle.