Assignment 05: Sudoku Validator

CS101 - Intro To Computer Science

Fall 2023

In this assignment, you will implement the validation functions required to implement a Sudoku game. You will be provided function prototypes, descriptions of the functions, and some unit tests to help validate your implementations.

The files for this assignment are in a zip file called `05_sudoku_validator.zip`. There are five groups of files in the folder

- `SudokuValidator.java`: the base implementation of the assignment - includes comments with the functions you need to implement.
- `SudokuValidatorTest.java`: unit tests - make sure you pass these unit tests before submitting the assignment.
- `.jar` files in the `lib` folder: libraries to enable the use of the unit tests.
- `csv files`: An example text file to run your `main` method on.
- `example output`: Examples of possible command line output.

1. Project Setup

The zip file contains a folder called `05_sudoku_validator`, which in turn contains a VS Code project. Extract the folder from the zip file and open it with VS Code to begin implementing and running your assignment. The folder contains the skeleton code for your submission, as well as further instructions on the various methods you need to implement, in `src/edu/nyu/assignment5/SudokuValidator.java`.

Note that this skeleton code already contains a `Scanner` called `scn`. Use this any time you need to parse user input.

2. Sudoku

Sudoku problems are puzzles that involve filling in a 9×9 grid with the numbers $1, 2, \ldots, 9$ such that in any row, column, or certain 3×3, blocks in the grid each number is used exactly once. The particular 3×3 blocks are the blocks $[3r, 3(r+1)) \times [3c, 3(c+1))$ for $r, c = 0, 1, 2$. If you're not familiar with these blocks or want a less technical description you can check the Sudoku wikepedia page.

The primary task in this assignment is to validate whether a particular Sudoku board contains a valid board and to compute the set of unfilled blocks in the grid. Here, a valid board does not necessarily need to be solveable, but rather it just needs to satisfy the condition of having unique values in each row/column/block. For this assignment we will store each Sudoku board in a `int[][]`, where each entry in this array is one of $0, 1, 2 \ldots, 9$. 0 will denote an unfilled entry in the board.

The skeleton project provided to you contains empty functions for doing different steps of the validation process (like validating individual rows/columns/blocks and functions for validating all rows/columns/blocks). Please follow the instructions found in the comments of these functions for how to implement these functions.

3. Comma Separated Values

We will load different Sudoku puzzles from files using the comma separated values (CSV) format. The CSV format is a format for storing tables of data - exactly like a spreadsheet. In fact, Microsoft Excel can open and save simple spreadsheets using the CSV format.

The format is quite simple: each line of the file is a different row in the table and the columns are denoted by values separated by commas. For example, the following CSV data:

```
"A","B","C","D","E","F","G"
0,1,"hi",,4,,
```

means the following table:

"A"	"B"	"C"	"D"	"E"	"F"	"G"
0	1	"hi"		4		

Note that although this example shows that the CSV format supports empty entries between commas to indicate empty cells, we will not use this for this assignment. In our particular case, every Sudoku puzzle is represented by a 9×9 CSV file of numbers (i.e 9 lines with 9 numbers in each line separated by 8 commas). In our use of the CSV format we will guarnatee that every entry is a number and 0 will represent the empty Sudoku cells. Each entry in each CSV file is therefore a number, which will simplify the work you need to perform when parsing each CSV file into our `int[][]` representation of each puzzle.

The final program must be able to open Sudoku puzzle CSV files specified by the user and then play a game of Sudoku. The main function is implemented for you so you should mostly focus on the validation, printing the boards, and computing the set of remaining moves. After you finish implementing everything, you can try solving a few Sudoku games using your own code for fun.

Every method you need to implement is described in `SudokuValidator.java`. If you correctly implement these methods your code should "pass" the unit tests in

`src/edu/nyu/assignment5/tests/SudokuValidatorTest.java`, but for this assignment we will have additional checks in the autograder so please do not submit the assignment at the last minute.

4. Submission

Submit the following files:

- `SudokuValidator.java`