Implementation of a Brute Force Sudoku Solver

The typical Sudoku puzzle that one finds in newspapers or pastime books consists of a square grid of size 9, containing preset values in some of its 81 cells. The grid is subdivided into 9 3x3 boxes. The objective of the puzzle is to fill each blank cell so that each column, each row, and each box contains all the digits from 1 to 9. A Sudoku puzzle can be easy for humans, or difficult and time-consuming.

The Sudoku problem can be stated in a general form for squares of size **(w x h)**, for **w** and **h** > 0. The example puzzle below illustrates a puzzle of size 6 or (3 x 2). Some of the **(w x h)^2** cells in the grid contain preset values; in the example, we show the values of these cells and an underscore for the empty cells. A solution to a Sudoku puzzle contains all the integers from 1 to **(w x h)** in each row, each column, and each box. A Sudoku puzzle of size (**w** x **h**) consists of **w** rows of boxes, and each row consists of **h** boxes. Each box has width **w** and height **h**.

A sample Sudoku instance of size (3 x 2):

```
+ - - - + - - - +
| 1 _ 3 | 4 _ _ |
| 4 _ 6 | _ _ 3 |
+ - - - + - - - +
| 2 _ 1 | _ 6 _ |
| 5 _ 4 | 2 _ _ |
+ - - - + - - - +
| 3 _ 2 | _ 4 _ |
| 6 _ 5 | _ _ 2 |
+ - - - + - - - +
```

For this project you will develop a Sudoku solver, a program that receives a Sudoku instance and solves it. Your program will use an enumeration approach as follows. Focus on the empty cells. Generate all the possible assignments of values to the empty cells, checking each assignment as you go for compliance with the Sudoku rules. As soon your program finds one assignment that satisfies all the rules, it can terminate and report its solution.

The enumeration approach should be relatively easy to code, but you may have

already noticed that it will spend time trying some pretty silly assignments. Things to think about: How many assignments for a Sudoku instance are possible using the enumeration approach? Can you write a Sudoku instance for which there is no solution?

Your programming project

You will develop a solver for the Sudoku problem, a program that receives a Sudoku instance as input and computes a (any one) solution for it. If no solution exists, your program will report this.

I am providing Solver and SodokuBoard interfaces which you will implement with BruteForceSolver and RegularSudokuBoard classes (I have provided the stubs). I am also providing utilities class Timer.java, Move.java, as well as the Main class that you will execute. I am also providing some input files that you may use as test cases. In addition to writing the BruteForceSolver and RegularSudokuBoard classes, you will also need to write the RegularSudokuParser class which will read the input.

Your program will read from the command line the name of a file, and then read the input instance from that file. I recommend that you use the Scanner class to read in the input.

An input file contains optional comment line(s) that start with the character 'c', followed by a sequence of **2+(w x h)^2** integers, from **0** to **(w x h),** in the following order:

- **w**, an integer > 0
- **h**, an integer > 0
- The **(w x h)^2** values in Sudoku grid in row-major order. A **0** indicates an empty cell.

The content of a file that specifies the example Sudoku instance follows:

```
c Specification of the sample Sudoku instance with size 6
3
2
103400
406003
201060
504200
302040
605002
```

To run the provided input files using Eclipse, edit the Run Configuration by selecting the Arguments tab and entering the path to the input, like src/inputs/puz1. After your own testing convinces you that your solver is correct, run and time your solver on all of the required input set. Preserve the runtimes since you will need them again to compare with future projects.

Submission instructions

By the deadline, submit your working solver, evidence of its correctness on the required input set, and a meaningful table with its runtimes.

This material is based upon work supported by the National Science Foundation under Grant No. 1140753.