Computing and Algorithms I
Project 5

CS-101 **Summer, 2013**

For your fifth project, you will check to see if a completed or partially completed Sudoku puzzle is correct.

1 Introduction

A Sudoku puzzle is a 9×9 grid, with some of the elements of the grid filled in with digits from one to nine (see Figure 1). To *solve* a Sudoku puzzle means to fill in the blank entries in the grid with digits, so that each digit appears exactly once in each row, column and 3×3 subgrid. A solved Sudoku appears below, using a standard text representation. For more information, see the introductory section to Wikipedia article on Sudoku (`http://en.wikipedia.org/wiki/Sudoku`).

Figure 1: A Sudoku and Its Solution

```
5 3   |   7   |                  5 3 4 | 6 7 8 | 9 1 2
6     | 1 9 5 |                  6 7 2 | 1 9 5 | 3 4 8
  9 8 |       |   6              1 9 8 | 3 4 2 | 5 6 7
-------+-------+------           -------+-------+------
8     |   6   |     3            8 5 9 | 7 6 1 | 4 2 3
4     | 8   3 |     1            4 2 6 | 8 5 3 | 7 9 1
7     |   2   |     6            7 1 3 | 9 2 4 | 8 5 6
-------+-------+------           -------+-------+------
  6   |       | 2 8              9 6 1 | 5 3 7 | 2 8 4
      | 4 1 9 |     5            2 8 7 | 4 1 9 | 6 3 5
      |   8   |   7 9            3 4 5 | 2 8 6 | 1 7 9
```

2 Requirements

You will write a correct, modularized and conforming program to read one or more Sudoku puzzles (or solutions) from a file, and to determine if it is correct. A puzzle is *correct* if

- It is a solution, or
- It is incomplete, does not violate any of the requirements for a Sudoku.

Both of the diagrams in Figure 1 are correct.

For each puzzle in the input, you will print it to output, and follow it by a line stating whether it is a solution, is not a solution but is correct, or is incorrect.

Input to the program will come from a file in the current directory. The name of the file will be given on the command line, and need not be verified. Each puzzle will consist of nine lines, each line will consist of a String of 9 digits which may or may not have white space before or after the String of digits. The digit zero will be used to represent a blank entry. Each puzzle will be separated from the next by an empty line or a line containing only blanks. A sample input is given in Figure 2. Note that there are two puzzles in this sample input. You may assume the input conforms to our format specifications.

Figure 2: Sample Sudoku Input

```
           530070000
600195000
    098000060
800006003
    400803001
700020006
060000280
           000419005
000080079

630070200
600194000
    095000060
800006113
    480803901
700021036
060040280
          200419705
010080973
```

Output from the program will consist of a title ("Sudoku Checker" or something similar), identification information (who designed and wrote this program), followed by each puzzle and the statement of its correctness. Each puzzle will be formatted as in Figure 1, using blanks, vertical bars, plus and minus signs.

3 Procedure

For this project, you will do the following:

1. Create a test file containing at least three puzzles, one a solution, one a partial solution, and one an incorrect puzzle.

2. Design a program satisfying the above requirements.
3. Write code which satisfies the design and conforms to the requirements.
4. Test your code sufficiently to establish that the code satisfies the requirements.
5. Run your program on your test file.

4 Design

Your design documentation must include the following:

1. A UML class diagram for any class you create for your solution.
2. A UML interaction diagram for all classes you create for your program.
3. An algorithm for each method.
4. A data table for each method and class.

5 Deliverables

Your code, just the .java files, are to be submitted on Blackboard before class begins on Thursday of Week 9 (September 13). All your .java files will be zipped into one file, name it P5.zip, and submit that file only (P5.zip) on Blackboard.

You will turn in the design, java source files, input files, and output files from running your program in one stapled package at the beginning of class on Thursday of week 9.

6 Notes

1. The best way to store the Sudoku puzzle contents is to use a two dimensional array. (A Sudoku puzzle class to create Sudoku puzzles with a two dimensional instance variable, and any other useful variables, is how I would suggest you organize your solution).
2. This is an original assignment from Professor Cater which I have modified to correspond to my course requirements. Professor Cater's assignment was inspired from a project I created for CS203.