SUDOKU CHECKER

This readme is a copy of the README.md present inside the sources

Description

This is basically a sudoku validator. My approach was mainly focusing on the correctness and extensibility of this game. My aim was to create a validator capable of validate also bigger sudoku than the classical dimension of 9x9. I didn't perform performance tests, but I've done (in my tests) some simple examples of 16x16 and 25x25 sudoku, without seeing a reduction of performances.

Usage

The application can be launched in three different ways (without changing anything in the code):

1) through a terminal (following the specifications), specifying the .bat file and the input file:

   validate.bat <inputfile.txt>

2) through the IDE, just launching the main class SudokuApp (debugging purposes etc...)

3) through a terminal, specifying just the .bat file, the default "puzzleName.txt" present inside the jar will be called by the respective classloader

The first way is the suggested one if there are no specific needs

Build

To build the application, from the root of the proj where the pom is present just do:

mvn clean install

Testing/Reporting

I've tested all the classes reaching an overall coverage of 100% with jacoco. I've just excluded in the coverage analysis the main class and the two classes/enum defined in the 'com.ubs.sudoku.model' package.

To launch the tests and produce the reports you should simply execute:

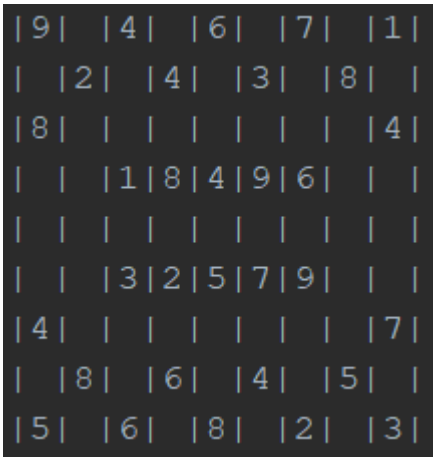
mvn test

Once that is done, inside the target folder will be created another folder called "*jacoco-report*".
Inside that folder there is an index.html page containing the report, with the branch/lines coverage statistics

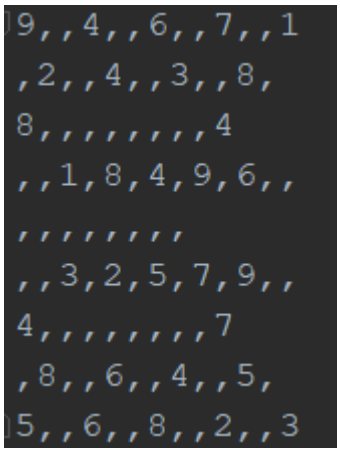
Doubts/Assumptions/Formatting

On the description of the exercise there was a confusion about the file format to use (csv or txt). But asking to who has sent the exercise, the reply was to consider txt files.

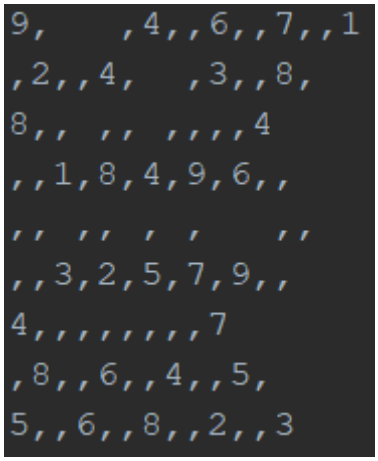
So, having a sudoku of this form:



let's see the representation that is required by the exercise (with ',' as a delimiter):



Considering that the spaces from 0 to n are permitted, we could write this matrix in many equivalent forms, one of this is for example the following:



Algorithm

Basically the check happens in three phases : Phase 1: Parse and validation of the input (external file) to a matrix of strings Phase 2: Transforming the matrix of string in matrix of int, substituting 0 to eventual spaces Phase 3: check the dimensions (rows, columns, submatrices) to see if the sudoku is valid

Logging

Is not a good practice use the stdout, but in this exercise the logging was not required