[**Department of Computer Science**](http://www.cs.umd.edu/)

[**CMSC132:**](http://www.cs.umd.edu/class/fall2017/cmsc132) Fall 2017

**Project:** Sudoku

**Due Date:** Thursday Dec 7, 11:59pm (no late deadline)

## Objective

To practice solving and generating sudoku boards.

## Specifications

### ***Sudoku Overview***

For this project you will write a class named Sudoku, that implements the game logic associated with the popular puzzle game sudoku. The goal of the game is to complete a grid with numbers so that every row, column and 3 by 3 grid (differentiated by shading) has every digit from 1 to 9 inclusive. The following is an example of a sudoku matrix:



In addition to the Sudoku class, you will expand an Utilities class to allow the generation of Sudoku boards.

More information about sudoku can be found in the following sites:

* <http://en.wikipedia.org/wiki/Sudoku>
* <http://www.sudokupuzz.com/>
* <http://www.sudoku.com/>
* [Generating Sudoku Puzzles as an Inverse Problem](https://sites.math.washington.edu/~morrow/mcm/team2306.pdf)

For this project the term "move" means entering a value into a square of the sudoku matrix. A legal move is one where the number provided does not violate the sudoku rules.

**This project is a required project for the honors section. Students in the regular sections are welcome to implement it, but they will not receive any credit.**

### ***Grading***

* Tests(65%)
  + Public Tests(25%)
  + Release Tests(40%)
* Sudoku Board Generation(35%)

## Code Distribution

The project's code distribution is available by checking out the project named Sudoku. The distribution includes a data directory with text files representing sudoku games and two packages (sudoku and tests).

1. **data** directory - This directory has text files representing sudoku games. In a text file a period represents an empty square in the sudoku matrix.
2. **sudoku** package - The following classes can be found in this package:
   * **Sudoku** -This is the class you are expected to implement for this project. In the Sudoku.java file you will find a shell for each of the public methods you are expected to implement. The javadoc documentation for this project can be found at: [doc/index.html](http://docs.google.com/doc/index.html). Feel free to add any private data members and private methods you understand you need. You may not add any public data members nor methods.
   * **Utilities** - This class makes use of the Sudoku class you will implement. This class provides supporting methods that allows you to initialize a sudoku game based on the data present in a text file, and to randomly select files. The class is used by the GraphicalUI class.
   * **GraphicalUI** - Represents the GUI (Graphical User Interface) for the sudoku application. Executing the main method of this class will allow you to play sudoku through the GUI. This class relies on the Utility class to select the initial sudoku grid. While using the GUI keep in mind:
     + Once a number representing a valid move has been entered in a square it cannot be changed.
     + A red square means there are no legal moves for that entry.
     + You can select one of the playing modes ("No hints", "Show constrained squares", "Show legal moves", "Show good moves") by selecting the arrow you will see in the right bottom corner.
     + The GUI randomly selects a puzzle from the **data** directory. The selected puzzle is specified at the top of the GUI.
     + The "Show Constrained Squares" option will display those squares which are currently empty, and for which there is only one legal move.
     + The "Show legal moves" option will display the valid moves for a particular square.
     + The "Show good moves" option will display the good moves calculated for a particular square.
     + The "Solve 1 constrained" button will solve the first constrained position on the sudoku board.
     + The "Solve current constrained" button will solve all currently constrained position on the sudoku board.
     + The "Copy board" button will create a new sudoku board which is a duplicate of the current one.
   * **PublicTests.java** - This class represents the set of JUnit public tests. These are the same public tests you will find in the submit server.
   * **StudentTests.java** - File where you will put your own test cases.

## Requirements

* At the top of your Sudoku.java file add a comment indicating what algorithm you used to determine legal moves.
* Make sure you put your own test cases in the StudentTests.java file. Feel free to use the *assertValues* method that you will find in the file.
* For convenience, we have added four release tests (named starting with "testHonors") that apply only to students in the honor section.
* You may wish to implement the Sudoku class using the following data members:
  + **data** - two-dimensional array of integers representing the sudoku cell values. The value 0 will represent an empty (blank) cell.
  + **rowData**- one-dimensional array of **BitSet** objects. Each **BitSet** object provides information regarding the legal moves that has been completed for a row.
  + **columnData**- one-dimensional array of **BitSet** objects. Each **BitSet** object provides information regarding the legal moves that has been completed for a column.
  + **regions** - two-dimensional array of **BitSet** objects. Each **BitSet** object provides information regarding the legal moves that has been completed for each 3 by 3 grid.
* Verify that your project passes the submit server tests (<https://submit.cs.umd.edu/>)

## Honor Section Requirements

* For those students in cmsc132H your project should satisfy all the requirements specified above. In addition, you need to implement a function getAllGoodMoves( ) using an algorithm that is smarter than the standard one of just looking to see what values are prohibited because they occur in the same row, column or square. In particular, on data/puzzle5.txt, the function getAllGoodMoves( ) should be able to determine for the initial board position at least one location that is constrained to have only one good value. You will be able to see how your algorithm is performing by release testing your project and looking at those tests whose named starts with "testHonors".
* You must also expand the Utilities class with a method(s) that generates a particular number of Sudoku boards. The boards will be named with the based name puzzleGen (e.g., puzzleGen.txt). The files can be created in your Eclipse project in the data folder. Your project will be evaluated based on the maximum number of boards you generate. Additional information will be provided in lecture.

## Submission

Submit your project using the "Submit Project" option (available by right clicking on the project folder).

## Academic Integrity

Please make sure you read the academic integrity section of the syllabus so you understand what is permissible in our programming projects. We want to remind you that we check your project against other students' projects and any case of academic dishonesty will be referred to the [University's Office of Student Conduct.](https://osc.umd.edu/)