Sudoku Resolver

If you do not know what sudoku is, start by heading to this link

Setup

1. Create a new repository for this project
2. Create a new file `script.php` at the project root repository
3. Create a src/ folder
4. Init composer for this project

   a) Activate autoloading, and require vendor/autoload.php in your script.php file (give `Sudoku` as a name for the main namespace under src/)
5. Create a class `SudokuCase` in a file src/SudokuCase.php (namespace `Sudoku`)

   This will be used to represent a case in the grid (`Case` being a reserved name, we had to name it differently, thus SudokuCase)

   a) Declare the following private properties:

   - `row` : (int) The row the case belongs to (from 0 to 8 included)
   - `col` : (int) The column the case belongs to (from 0 to 8 included)
   - `region` : (int) The region the case belongs to (from 0 to 8 included)
   - `value` : (int) The number (if any) in the case. If it is a missing number (= `?` , see below) set the value to `null`

   b) Declare a constructor that takes `row` , `col` , `region` and `value` (optional, default to `null`) as parameters and set the respective private properties.
6. Create a class `Sudoku` in a file src/*Sudoku.php* (namespace `Sudoku`)

   a) Declare the following private properties:

   - `cases` : array The array containing 81 SudokuCase objects representing the case in the grid.

   b) Create a method `loadFromFile(string $filePath): void`

   This method will load a text file containing the grid configuration of the sudoku we want to solve. (you can see an example of the source file in examples/level1.txt)

   In this txt file, you will see numbers and `?` signs separated by ' ' (spaces). `?` represent the numbers to solve. Each line is separated by a new-line \n character.

   What you have to do in this method is:

   - Read the file content (see file_get_contents for more info)

    - For each number or `?`, instanciate a new `SudokuCase` and push it to the `cases` array (private property)
        - make sure to set the right `row`, `col`, `region` and `value` (if any) for each case (e.g. respectively for the 1st 3 cases : <0,0,0>, <0,1,0>, <0,2,0> ...)

   c) Create 3 methods : `getCasesForRow(int $row): array`, `getCasesForCol(int $col): array`, `getCasesForRegion(int $region): array`. These methods will return an array of `SudokuCases` respectively for the given row, col or region. (you can have a look at array_filter)

   d) Create a method `displayGrid(): void`

   This will echo every cases of your grid in the terminal. Make it beautiful so the result looks like a grid. You can even use Colors by echo-ing special codes before your text (see https://misc.flogisoft.com/bash/tip_colors_and_formatting). e.g. use different colors for numbers and `?` signs so you easily see what cases are missing and need to be solved.

Testing

1. In your `script.php` file, instanciate a new `Sudoku\Sudoku` object.

2. You will have to get the filepath of the level.txt to test from the command line arguments (remember $argc and $argv).
   You can try out your script by launching the following command in your term : `php script.php < examples/level1.txt`

3. Once your `Sudoku` object is instantiated, you will have to call the method `loadFromFile` and give it the level filepath as an argument. It should fill up our cases array in our Sudoku Object.
4. Once done, use the method `displayGrid` to make sure your data has been correctly stored and restitued.

Level 1 resolver

Alright we are all setup to start coding our very first algorithm for simple grid resolution.

The idea to solve our first grid is as following :

- foreach row, col, or region, see if there is only one case missing (= `?`). If it is the case, deduce from the other numbers what the missing number is, and set the case value.
- Repeat this step for each row, col and region as long as the entire grid is not solved

Todo

1. In the `Sudoku` class, create a `isSolved(): bool` method.

   a) This method will return `true | false` wether the grid is solved or not.

   A grid is considered solved once every case of the grid has a value set.

2. In the `Sudoku` class, create a `solve(): void` method.

   This method will contain a main loop which will test every row, col and region and try to deduce the value of missing cases.

   Note: if there is more than 1 missing case in the row, col or region, don't bother and go on to the next one. When you solve a case, this will unlock new solving possibilities.

   You may have to call your methods declared earlier: `getCasesForRow(int $row): array` , `getCasesForCol(int $col): array` , `getCasesForRegion(int $region): array` to test every cases.

   You exit the main loop once the Sudoku is solved (`isSolved()` returns `true`).

3. Test your solution by calling your `solve()` method from `script.php` .

4. You may then visualise the solution of the grid calling the `displayGrid()` method.