Sudoku Project Part 4

Due Sunday, 12/05/2021 at 11:59 pm (20 points)

Overview

We are going to continue adding code to `Sudoku.java`. Here, we will write two methods that we will use to determine what values are allowed to go in a particular square.

In case you need it, here are links to the project introduction, part 1, part 2, and part 3.

Getting started

For this part we will continue working on the `Sudoku.java` file. We are going to add some new methods to help us with the logic of determining what numbers can go in what squares.

We will provide starter code on Canvas or the course website. You may use this code if you didn't finish part 2, or if you simply don't feel confident in your part 2 solutions, or even if you simply prefer working with the starter code with the comments. The starter code contains the solutions to part 2, so I can't make it available until after the due date for part 2.

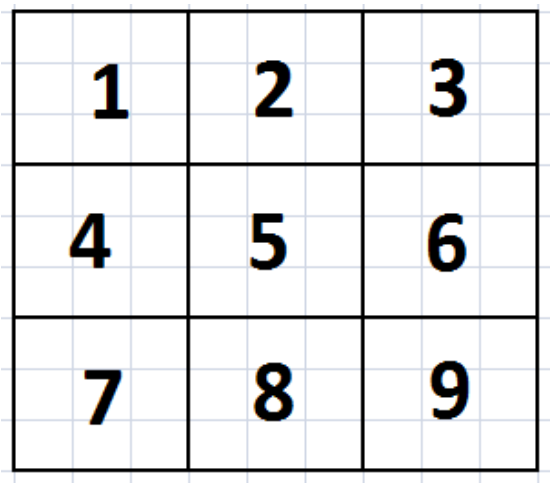
Alternatively, if you want to get a head start on part 3, you are free to simply edit the code that is already on your computer.

You'll be modifying the `Sudoku.java` file. We will provide you with the code for `SudokuGame.java` that you can use to test your output. If you are modifying your own code, make sure your `SudokuGame.java` file matches the starter code.

Let's walk through all of the code components that you need to add to `Sudoku.java`.

`getBoxNumber` method

First, we are going to write a `private` helper method called `getBoxNumber`. This method has two parameters: `int row` and `int col`. The method returns the "box number" to which the square at that particular row and column belongs.



This diagram illustrates what we are trying to calculate:

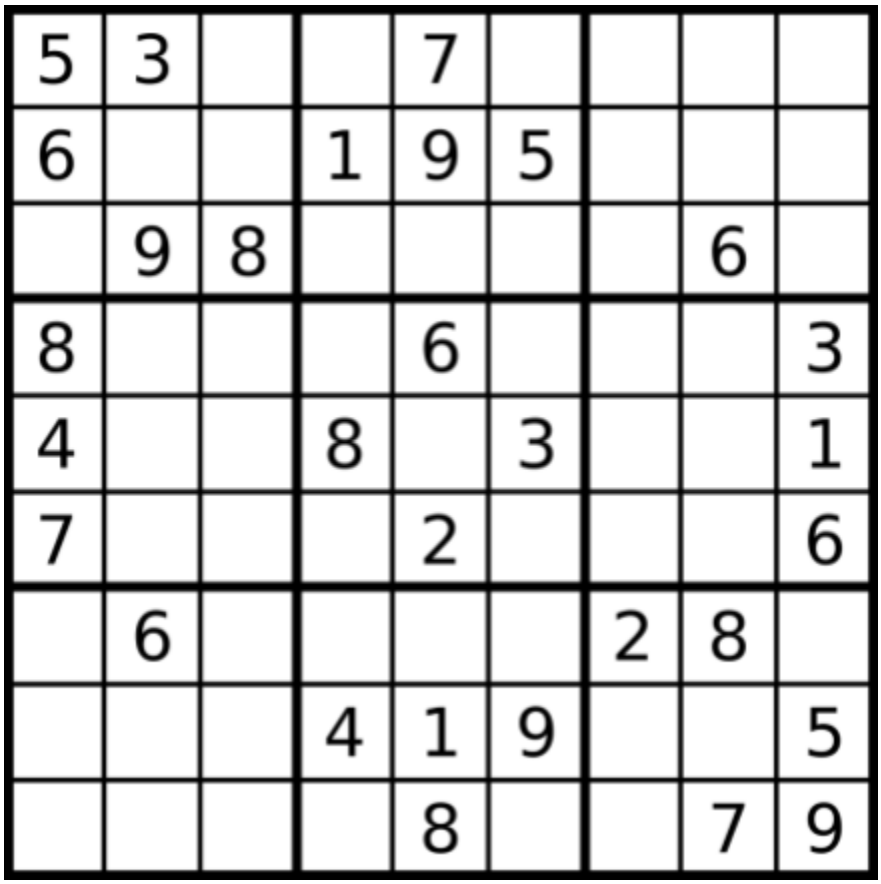
- rows 1-3:
  - rows 1-3, columns 1-3 = box 1
  - rows 1-3, columns 4-6 = box 2
  - rows 1-3, columns 7-9 = box 3
- rows 4-6:
  - rows 4-6, columns 1-3 = box 4
  - rows 4-6, columns 4-6 = box 5
  - rows 4-6, columns 7-9 = box 6
- rows 7-9
  - rows 7-9, columns 1-3 = box 7
  - rows 7-9, columns 4-6 = box 8
  - rows 7-9, columns 7-9 = box 9

Hint: there are a couple of approaches to try to convert a row/column pair into a box number. One way is to use a series of (nested) `if` statements. There is also a formula that makes clever use of the division (/) and modulus (%) operators. **If you use this approach, you will receive one (1) point of extra credit.**

`isValid` method

Next, we will write a method that allows us to determine if we are allowed to place a particular number in a particular sudoku square.

Write a `public` method isValid that takes three parameters: `int row`, `int col`, `int val`. The method has a `boolean` return type. It returns `true` if we are allowed to put the given value at the given `row` and `col`, and `false` otherwise.



In the above example:

- `isValid(3, 1, 2)` would return `true` - there's no conflict
- `isValid(4, 6, 8)` would return `false` - there is already an 8 in row 4
- `isValid(6, 2, 3)` would return `false` - there is already a 3 in column 2
- `isValid(7, 4, 9)` would return `false` - there is already a 9 in that box
- `isValid(7, 9, 4)` would return `true` - there's no conflict

Make use of the `getRowValues`, `getColumnValues`, and `getBoxValues` to determine what values are not allowed at the given `row` and `col`. Then, check if the parameter `val` matches any of the illegal values.

Before calling the `getBoxValues` method, you'll need to determine what "box" the given `row` and `col` are a part of. You'll use your `getBoxNumber` method for this.

Testing your code

Put the following code into your `SudokuGame.java` file (or download it from Canvas or the course website):

```
public class SudokuGame
{
    public static void main(String[] args)
    {
        int[][] exampleGrid = {{5, 3, 0, 0, 7, 0, 0, 0, 0},
                               {6, 0, 0, 1, 9, 5, 0, 0, 0},
                               {0, 9, 8, 0, 0, 0, 0, 6, 0},
                               {8, 0, 0, 0, 6, 0, 0, 0, 3},
                               {4, 0, 0, 8, 0, 3, 0, 0, 1},
                               {7, 0, 0, 0, 2, 0, 0, 0, 6},
                               {0, 6, 0, 0, 0, 0, 2, 8, 0},
                               {0, 0, 0, 4, 1, 9, 0, 0, 5},
                               {0, 0, 0, 0, 8, 0, 0, 7, 9}};


        Sudoku sudoku = new Sudoku(exampleGrid);
        sudoku.printGrid();
        System.out.println();

        System.out.println("Can I put a 2 at row 3, column 1?");
        System.out.println(sudoku.isValid(3, 1, 2));
        System.out.println();

        System.out.println("Can I put an 8 at row 4, column 6?");
        System.out.println(sudoku.isValid(4, 6, 8));
        System.out.println();

        System.out.println("Can I put a 3 at row 6, column 2?");
        System.out.println(sudoku.isValid(6, 2, 3));
        System.out.println();

        System.out.println("Can I put a 9 at row 7, column 4?");
        System.out.println(sudoku.isValid(7, 4, 9));
        System.out.println();;

        System.out.println("Can I put a 4 at row 7 column 9?");
        System.out.println(sudoku.isValid(7, 9, 4));
```

```
        System.out.println();

    }
}
```

Compile and run your code. You should get the following output:

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

Can I put a 2 at row 3, column 1?
true

Can I put an 8 at row 4, column 6?
false

Can I put a 3 at row 6, column 2?
false

Can I put a 9 at row 7, column 4?
false

Can I put a 4 at row 7 column 9?
true
```

If your output is correct, you are ready to submit.

Note: The `SudokuGame.java` file attempts to test your code under the assumption that you have filled in `Sudoku.java`. If you try to compile `SudokuGame.java` without having done the exercises for `Sudoku.java`, then it will not work, because `SudokuGame.java` will be trying to access/test variables and methods that don't yet exist.

Submitting Your code

Once your code compiles without errors and your output is correct, you can submit part 4. Please submit the following files:

- `Sudoku.java`
- `SudokuGame.java`

Please do NOT submit any of the following:

- An image or screenshot of your code - we can't run or test this
- `Sudoku.class` or `SudokuGame.class` - this is just machine code, and it doesn't let us evaluate the code that you actually wrote.
- Files with names other than the ones specified above

If you don't follow these directions, you will temporarily receive a 0. You will have to resubmit and be assessed a late penalty.

Please submit both of your files on Canvas.

Grading rubric

First, note that if your code does not compile, you will automatically receive 0 points. This indicates to us that you did not try to test your code.

Assuming your code compiles, you will be graded as follows:

`getBoxNumber`**:** 5+ points

- **Correct method header:** 1 point
- **Correct method body:** 4 points
- **Clever use of division and modulus:** 1 point (extra credit)

`isValid`**:** 10 points

- **Correct method header:** 1 point
- **Correct method body:** 9 points
    - **Correctly check for row conflicts:** 3 points
    - **Correctly check for column conflicts:** 3 points
    - **Correctly check for box conflicts:** 3 points

Programming Style: 5 points

- You will lose one (1) point per style mistake, but no more than 5 points.