Sudoku

1 Overview

Sudoku is a popular puzzle game in which the player is presented with a nine by nine grid of numbers, partially populated. Each square can be filled with a number one through nine, and the puzzle is considered solved when the puzzle is completely filled out and the following rules are unbroken. Each row must have each digit exactly once. Each column must have each digit exactly once. Each of the nine, three by three sub-squares must have each digit exactly once. The example screen shot is invalid because of the 7 value in the bottom right. is repeat both in its row, and its three by three sub square. We will be building an object representing a Sudoku puzzle, capable of accepting values and determining if the puzzle is in a valid state, or completely solved.

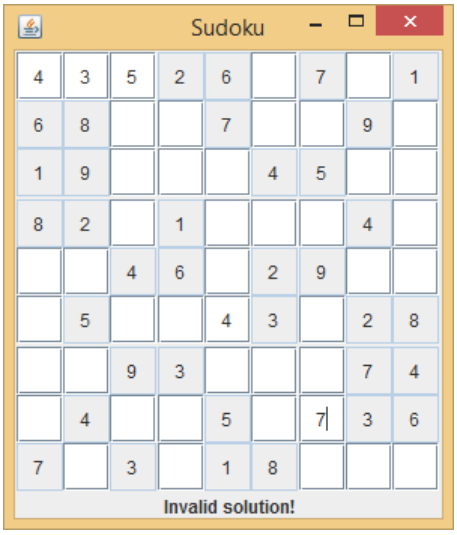

Fig. 1: Sudoku being played with a Swing application.

2 Learning Outcomes

By the end of this project students should be able to:

- Use iteration to solve computational problems;
- Use conditional statements;
- Write a class to a per-defined public contract;
- Instantiate arrays and/or lists;
- work effectively with a partner using pair-programming;
- write an effective report that describes the students' problem solving process.

3 Pre-Lab Instructions

Do this part before you come to lab:

This project will require you to use conditional statements, iteration and arrays.

- Read Big Java chapters 4 5 and 6
- Write psudocode that would create an array 16 items long and populate it with numbers 10 through 25 in order.
- Write psudocode for a function that would verify that a 16 item long array contained the numbers 10 through 25 in order.

4 Lab Instructions

Do this part in lab:

Step 1

In this lab we will create a class following the following class specification:

Sudoku
Sudoku( ) Sudoku(starting_configuration : String) getSquare(row : int, col : int) : char setSquare(row : int, col : int, value : char) : void isValid( ) : boolean isSolved( ) : boolean

Tab. 1: UML Class Diagram

Start by implementing class with stubs for each of the methods, either doing nothing or returning a placeholder value. Once you have stubs for all of the methods, copy the SudokuPlayer.class file provided into the same folder and running the SudokuPlayer class. If your Sudoku object worked it should bring up a window. If it crashes, you are likely not correctly implementing the contract. Ensure you can open the player before moving on to step 2.

Step 2

Now that we know we have the proper public contract, we will provide the class's implementation. First decide what internal state the Sudoku class will

require and record your thought process for the planning stage of your report. Now, implement each method.

Sudoku( )

For our basic constructor initialize your state to represent an empty Sudoku puzzle.

Sudoku(starting_configuration : String)

Our second constructor takes a string representing the initial configuration of the puzzle. This string will contain one character per grid square, each row ending with a character return. Blank spaces are represented with a space. Example is below:

```
String start = "";
start += "   26 7 1\n";
start += "68  7  9 \n";
start += "19   45  \n";
start += "82 1   4 \n";
start += "  46 29  \n";
start += " 5   3 28\n";
start += "  93   74\n";
start += " 4  5  36\n";
start += "7 3 18   \n";
```

getSquare(row : int, col : int) : char

Should take a row and column number and return the digit in that square. Blank spaces should be represented with a space.

setSquare(row : int, col : int, value : char) : void

Takes a row, column and digit, and sets the puzzle to store value at the given location.

isValid( ) : boolean

Returns true if all three Sudoku rules are observed. No duplicate numbers in each row, in each column, or in each sub-square. **This method and methods it calls are REQUIRED to be shorter than 25 lines of code each.** Break the problem down into multiple private methods to make this possible while still writing clear code.

isSolved( ) : boolean

Returns true if the puzzle has no more blank spaces and is still valid.

Once you have completed these tasks, you should be able to run your code with the SudokuPlayer and play a complete game of Sudoku, in which the player accurately tells you when you have violated any Sudoku rules, and congratulates you when you win.

5 Lab Report

Each pair of students will write a single lab report together and each student will turn in that same lab report on BBLearn. Submissions from each student on a pair should be identical.

Your lab report should begin with a preamble that contains:

- The lab assignment number and name
- Your name(s)
- The date
- The lab section

It should then be followed by four numbered sections:

1. Problem Statement

In this section you should describe the problem in **your** own words. The problem statement should answer questions like:

- What are the important features of the problem?
- What are the problem requirements?

This section should also include a reasonably complete list of requirements in the assignment. Following your description of the problem, include a bulleted list of specific features to implement. If there are any specific funtions, classes or numeric requirements given to you, they should be represented in this bulleted list.

2. Planning

In the second section you should describe what planning you did in order to solve the problem. You should include planning artifacts like sketches, diagrams, or pseudocode you may have used. You should also describe your planning process. List the specific data structures or techniques you plan on using, and why.

3. Implementation and Testing

In the third section you should describe how you implemented your plan. As directed by the lab instructor you should (as appropriate) include:

- a copy of your source code (Submitted in BBLearn as a .java files)
- a screen shot of your running application / solution
- results from testing

4. Reflection

In the last section you should reflect on the project. Consider different things you could have done to make your solution better. This might include code organization improvements, design improvements, etc.

You should also ask yourself what were the key insights or features of your solution? Were there alternative approaches or techniques you could have employed? How would these alternatives have impacted a different solution?

5. Partner Rating

Every assignment you are required to rate your partner with a score -1, 0 or +1. This should be submitted in the comment section of the BBLearn submission, and not in the report document. If you don't want to give your partner a negative rating making sure not to use a dash before listing the number! You do not have to tell your partner the rating you assign them. A rating of 1 indicates that your partner was particularly helpful or contributed exceptional effort. A rating of 0 indcates that your partner met the class expectations of them. Rating your partner at -1 means that they refused contribute to the project, failed to put in a resonable effort or actively blocked you from participating. If a student recieves three ratings of -1 they must attend a mandatory meeting with the instructor to dicuss the situation, and recieving additional -1 ratings beyond that, the student risks losing a letter grade, or even failing the course.

Colophon

This project was developed by Richard Lester and Dr. James Dean Palmer of Northern Arizona University.