HW4 – Leveraging Reuse with Adapters, Facades and Template Methods

Estimated time: 12-20 hours

Objectives

- Become more familiar with UML modeling
- Become familiar with applying the Adapters, Facades, and Template Methods patterns appropriating
- Improve unit testing skills to reduce complexity and improve performance

Overview

In this assignment, you will build a Sudoku puzzle solve and in the process make use of appropriate patterns like the Template Method Pattern. For information about the Sudoku game, look at https://en.wikipedia.org/wiki/Sudoku and other online resources for learning the game and it's rules.

Functional Requirements

1. The solver must be able to access puzzles of the following sizes: 4x4, 9x9, 16x16, and 25x25.
2. The solver must be able to be able to solve all solvable puzzles and report unsolvable puzzles as such.
3. The solver must out the solution to a specific output file.
4. The solver must allow the user to specify an input file containing the puzzle's initial state. The input file will be a text file containing the grid size n followed by, a full set of n symbols and then n rows of n symbols or blanks (dashes). Here are some examples:

```
4
1 2 3 4
4 2 - 1
- - - 2
3 - 2 -
- 4 - 3
```

```
16
1 2 3 4 5 6 7 8 9 A B C D E F G
4 9 - 1 3 6 7 - 8 - - - - D - -
- 6 3 5 - - - 9 - - A - - - - -
5 - - - 2 9 3 6 4 - - - B - - -
```

```
- 2 - 3 1 - - 4 - - - - - - - - -
- 7 4 - - - 2 1 - - - F - - - -
- - 1 - 6 4 - 8 - - - - - - 2 -
1 8 6 9 - - - 2 5 - - - - - - -
- 4 - - 5 1 8 3 - D - - 2 - - -
3 - 9 4 8 - - 7 - - - - - - - -
4 9 - 1 3 6 7 - 8 - - - - D - -
- 6 3 5 - - - 9 - - A - - - - -
5 - - - 2 9 3 6 4 - - - B - - -
- 2 - 3 1 - - 4 - - - - - - - -
- 7 4 - - - 2 1 - - - F - - - -
- - 1 - 6 4 - 8 - - - - - - 2 -
1 8 6 9 - - - 2 5 - - - - - - -
```

5. The solver must be able to save a solution into an output text file in the format, but with all of the blanks filled in. For example, the first puzzle show about has a solution and it is shown below. The second puzzle is not solvable. The solver should simply report that and not output a solution.

```
4
1 2 3 4
4 2 3 1
1 3 4 2
3 1 2 4
2 4 1 3
```

Instructions

To build this system, you will need to do the following:

1. Research different techniques or tips for solving Sudoku puzzles. We'll call this the cell-solution algorithms. Try to find at least two, but be open to finding five or more.
2. Model Sudoku puzzles from a structural perspective. Note that how you model the puzzles will have a direct impact on your cell-solution algorithms.
3. Design the rest of your system.
   3.1. Look for **good** opportunities to apply the adapter, façade, or template method patterns. Do not use the pattern without good justification.
   3.2. Also look for good opportunities to apply other patterns. Again, the emphasis will be on **appropriate justified application**, and **not blind use**. Hint: the template method and strategy patterns often work well together.
4. Implement and the solver based on your design
5. Test all non-GUI components using thorough executable test cases.
6. Test the system against the functional requirements

Submission Instructions

Zip up your entire solution, including test cases and sample input files, in an archive file called CS5700_hw4_<*fullname*>.zip, where fullname is your first and last names. Then, submit the zip file to the Canvas system.

Grading Criteria

Criteria	Max Points
A clear and concise designs consisting of UML class diagrams for the structural components of system, plus interaction diagrams that several key scenarios (e.g., the loading and solving a solvable puzzle and the loading and report on an unsolvable puzzle).	20
A working implementation that can solve puzzles with an easy or medium difficulty level. The implementation must follow the design and have good encapsulations, abstractions, inheritance, and aggregation, and appropriate use patterns study in class to date.	40
Thorough executable unit test cases for core application logic	30
Reasonable systems test using ad hoc methods	10

Optional Extra Credit

Criteria	Max Points
Implementation of more than 3 cell-solution techniques and that can solve puzzle with hard difficulty level.	10
An GUI that allows the user to select an input file, see the initial puzzle, view the solution, and save the solution to an output file	20
An extension to the GUI that shows an animation of the solving process and allows the user to control the speed of the solving	5