Homework #3: Sudoku meets numbrix

Due Thursday, September 20th at 11:59 p.m.

In this assignment, you will first create two versions of your sudoku solution verifier from Homework 2: one version to demonstrate the strategy design pattern and one version to demonstrate the template method design pattern. You will then add a solution verifier for another puzzle, numbrix. For both parts, you will reflect on the strengths and weaknesses of your designs.

As with Homework 2, your solution to this assignment should achieve as much code reuse as possible and be extensible so that new puzzle verifiers could easily be added in the future. You will also write unit tests to check the correctness of your solution. Overall, your goals for this homework are to:

- Demonstrate mastery of earlier learning goals, especially the concepts of information hiding and polymorphism, software design based on informal specifications, and Java coding and testing practices and style.
- Use inheritance, delegation, and design patterns effectively to achieve design flexibility and code reuse.
- Discuss the relative advantages and disadvantages of alternative design choices.
- Gain experience programming against existing interfaces and implementations and designing an API for reuse.

Part 1: The strategy and template method design patterns

Create two versions of your sudoku solution verifier from Homework 2. One version must demonstrate the strategy design pattern, and the other version must demonstrate the template method design pattern. These two versions must be functionally equivalent. If your existing Homework 2 solution already demonstrates one of these design patterns, you may turn in your existing solution (for the one respective design pattern) without modifying it. Test both versions using appropriate unit tests.

When you are done, discuss the strengths and weaknesses of each design pattern in the context of this problem domain (puzzle solution verifiers). Discuss the similarity of your Homework 2 solution to each of the design patterns and the changes you made (if any) to demonstrate each design pattern. Clearly state and provide arguments for which design pattern you believe is a more appropriate solution to this problem. Turn in your discussion as `discussion.md` or `discussion.pdf` in your Homework 3 directory.

Part 2: Extending your solution to a new puzzle type

Numbrix is a logic-based, number-placement puzzle, developed by Marilyn vos Savant[1], that has appeared in *Parade Magazine* since 2008. The player is given a 9×9 grid partially filled with numbers 1 through 81. The objective is to fill the grid with a sequence of consecutive numbers from 1 to 81, with adjacent numbers following a horizontal or vertical path. Diagonal paths are not allowed.

A sample numbrix puzzle (left) and its solution (right) are[2]:

51		49		47		33		31
59								29
61								27
63								15
77		75		5		9		11

51	50	49	48	47	34	33	32	31
52	53	54	55	46	35	22	23	30
59	58	57	56	45	36	21	24	29
60	67	68	43	44	37	20	25	28
61	66	69	42	41	38	19	26	27
62	65	70	71	40	39	18	17	16
63	64	81	72	3	2	1	14	15
78	79	80	73	4	7	8	13	12
77	76	75	74	5	6	9	10	11

Add a numbrix solution verifier to one of the versions of your Part 1 solution, using either the strategy or template method design pattern. As before, your design should achieve as much code reuse as possible and be extensible for new puzzle types. Test your solution using appropriate unit tests.

In `discussion.md` or `discussion.pdf`, discuss the strengths and weaknesses of your design for extending your solution to a new puzzle type. Explicitly discuss any changes you made to your design to accommodate numbrix and summarize why you extended the version of your solution from Part 1 that you chose, as opposed to the other version.

[1] https://entertainment.howstuffworks.com/puzzles/how-to-play-numbrix2.htm

[2] From https://parade.com/numbrix/.

Evaluation

This assignment is worth 100 points. We will evaluate your solution approximately as follows:

- Mastery of earlier learning goals, especially the concepts of information hiding and polymorphism, software design based on informal specifications, and Java coding, specification, and testing practices and style: 40 points
- Effective use of inheritance, delegation, and design patterns to achieve design flexibility and code reuse: 40 points
- Discussion of design alternatives: 20 points

Hints and advice

- Your overall discussion for both parts should be at most 1–1.5 pages.
- Use precise design terminology (including the exactly correct design pattern names) in your discussions.
- "Code reuse" includes appropriate reuse of test code. You should avoid needlessly duplicating test code when testing similar or equivalent implementations.
- In `src/main/resources`, we have provided a set of valid puzzle instances. For both puzzle types, we use 0 to denote an empty cell.
- We plan to strongly critique your design choices. Please plan adequate time to get feedback on your solution from the course staff and substantially revise your solution before the homework deadline.