HW5-State, Proxy, and Other Patterns

Introduction

In this project, we are developing a Sudoku Solver which is open to extension and closed to modification. Our focus is to keep the solver simple, modular, reusable.

Aim

Our aim to is to become familiar with Strategy pattern.

UML Diagrams

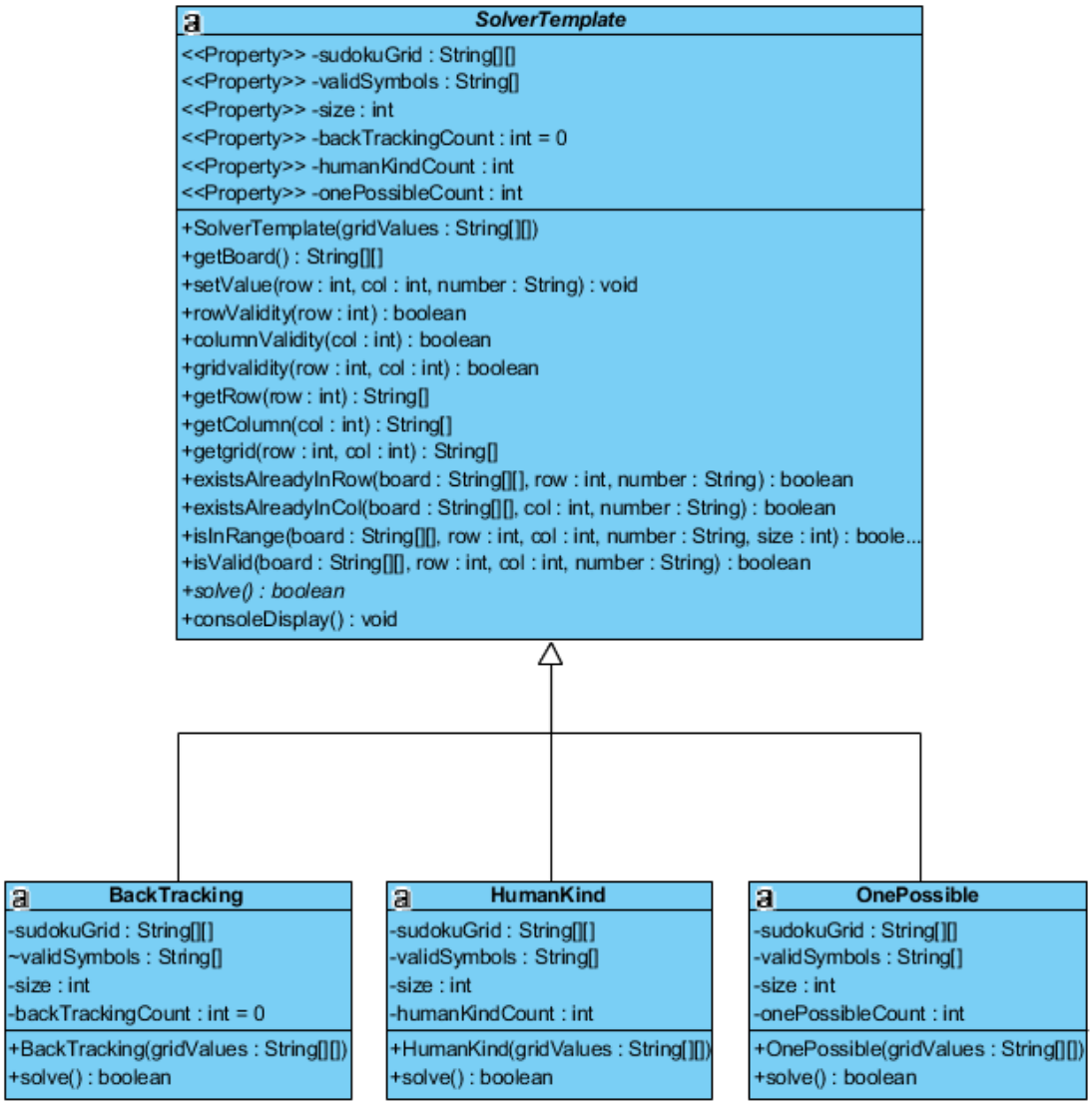

Fig. 1.1 Class Diagram of Strategy pattern

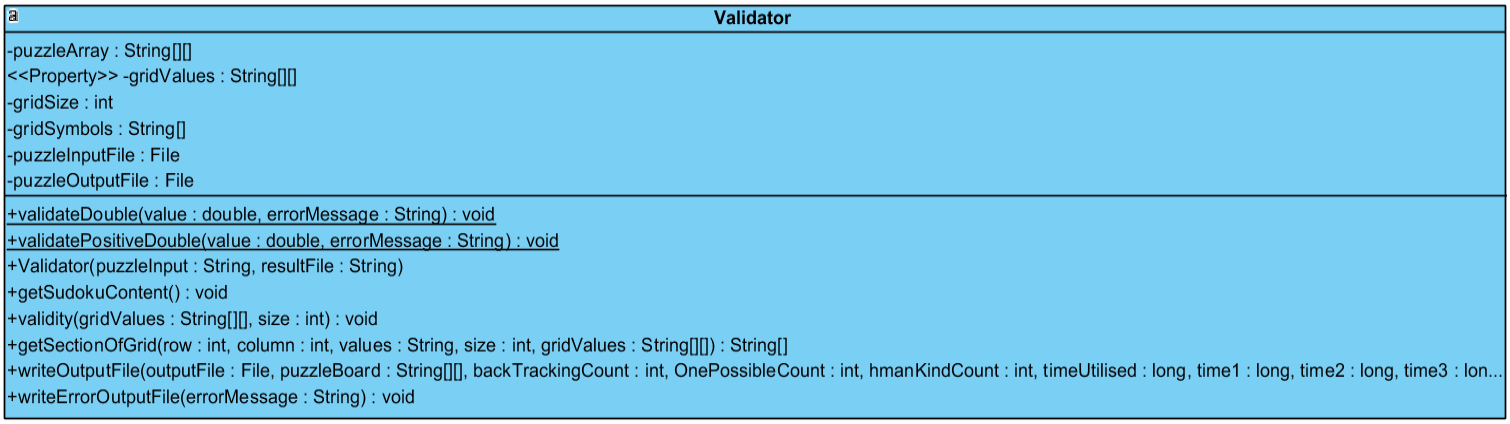

Fig. 1.2 Class Diagram of validator

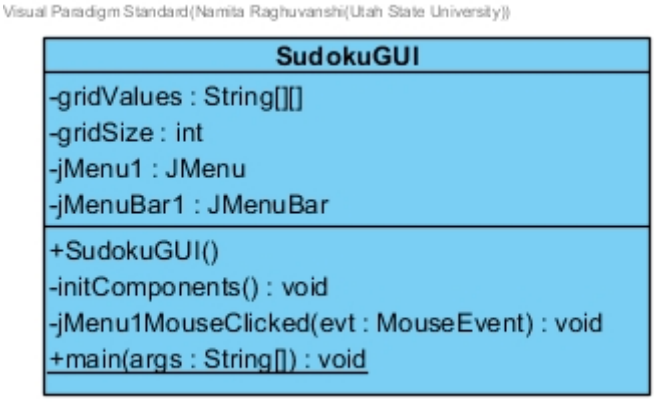

Fig. 1.3 Class Diagram of GUI Component

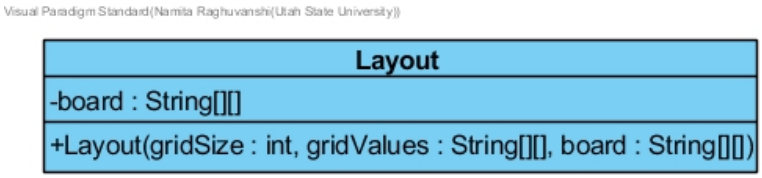

Fig. 1.4 Class Diagram of GUI Component(layout)

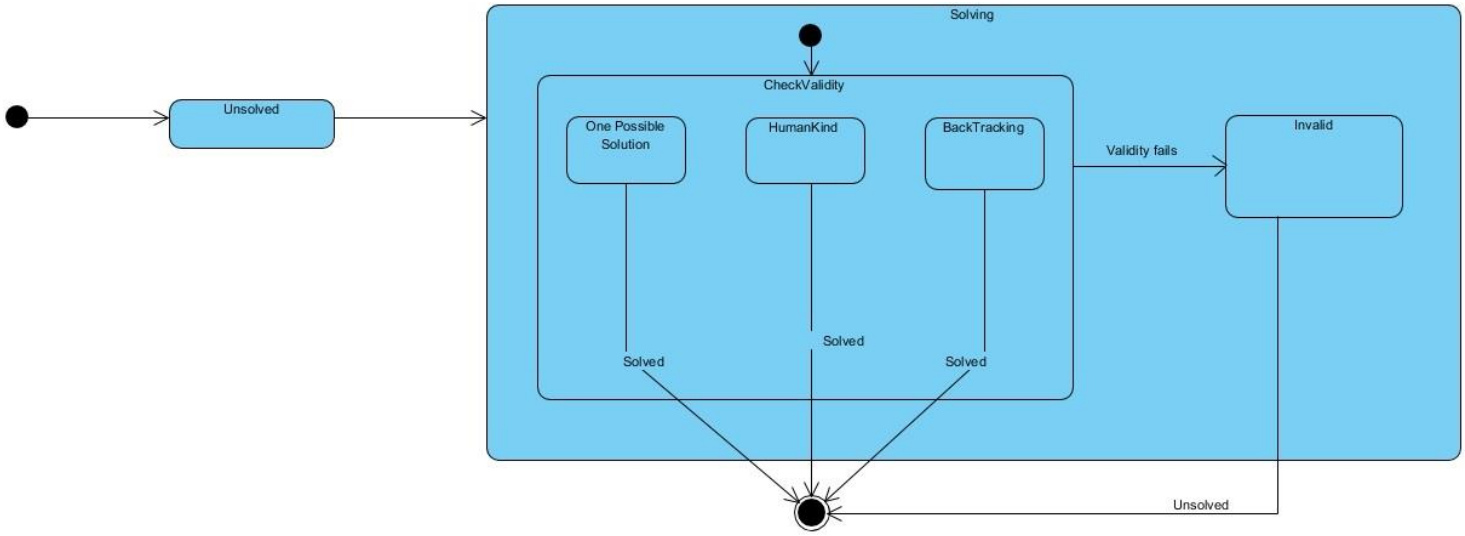

Fig 2.1 State Chart Diagram.

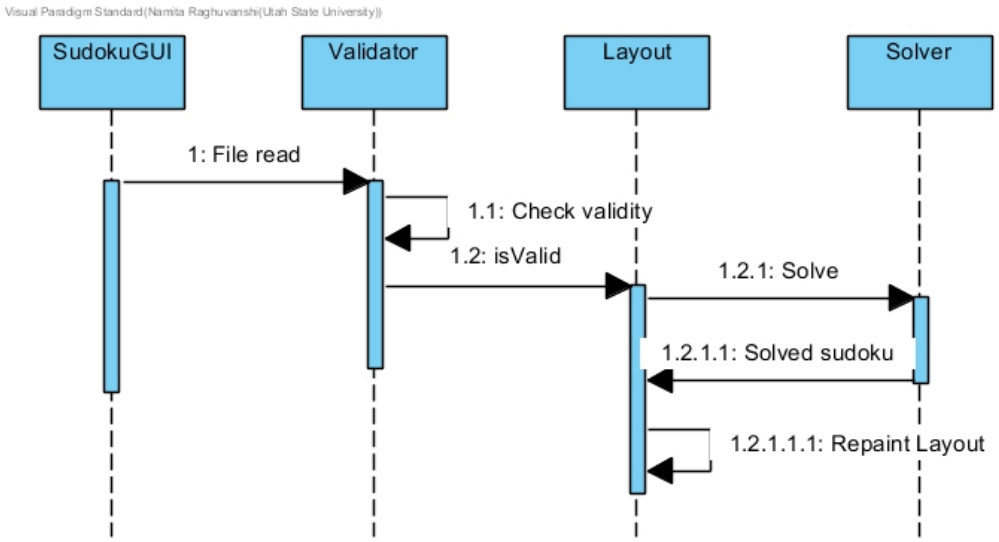

Fig 3.1 Interaction Diagram

Insights Uncovered

-While doing the project I have learned a lot about testing and I also know it now that creating a code with better modularity, encapsulation and abstraction leads to easily testable cases.

- Also I have faced the fact that every time we should try to figure out new methods and ways rather than coding just with the pre-existing knowledge. This way I get insights to many new thins.

-One of my bad practice that I tackled while doing this project is that we should always be open to new ideas and strategy rather than thinking in one direction. Because sometimes the only solution is to change the whole plan and get started again from a new point. I have wasted a lot of time trying to figure out things where all I needed was to think differently about a new solution. I tried applying Observer pattern but I failed then I simply did what I was doing and applied strategy pattern.

-I have also figured out that the naming conventions are not to be taken lightly as it could make the code look messy and also it won't be readable for someone else. Also it is time wasting to debug a code which does not have proper names. In this code, I also used proper scope for variables because that can be code smell as well.

- I also learned that we should always code in such a way that further add-ons and changes should not be a cumbersome task.

- In this project, I made GUI for my old sudoku solver. I applied strategy pattern so that if any other developer wants to add new method, it will be easy in future and reduce ripple effect.