

## HW5: Sudoku Solver and GUI with Swing

-Princy Jain

For this assignment I extended the Sudoku Solver with a GUI and improved the design and project structure. I added the following functional parameters this time:

1. A GUI with MVC design pattern.
2. Sudoku Solver Algorithms with Strategy Pattern.
3. Solver works on a Template method pattern.

At first, I was thinking of applying Command Pattern or Observer Pattern on GUI but I was not able to figure out how to apply Command Pattern and I don't want to use Observer Pattern since we covered it in Assignment 2 already. So, I implemented Model-View-Controller pattern after a lot of research and time.

I am attaching the class diagram of my previous design with the repository.

Other Functionalities:

A Sudoku GUI that replaces the console from the last assignment. The GUI has following buttons:

- Load button: to upload a puzzle from the text file
- Reset button: to resets the grid to blank
- individual buttons for solving the puzzle with respective algorithms (4 buttons for each algorithm)

Implementation:

The application starts with the user selecting puzzle from a text file to initialize an unsolved Sudoku board. On click of Algorithm buttons the flow goes to the solver and it executes the algorithm to solve the board. The updated solved Sudoku board is rendered on the GUI which is the solved puzzle for the user.

The Reset button will set all the cells to blank at any time.

I created a Driver class which has the main method. When we execute the main method, it calls the instance of view and model first to create the layout of GUI. After this, the controller is called which add the action listener to each method with respect to what it actually does.

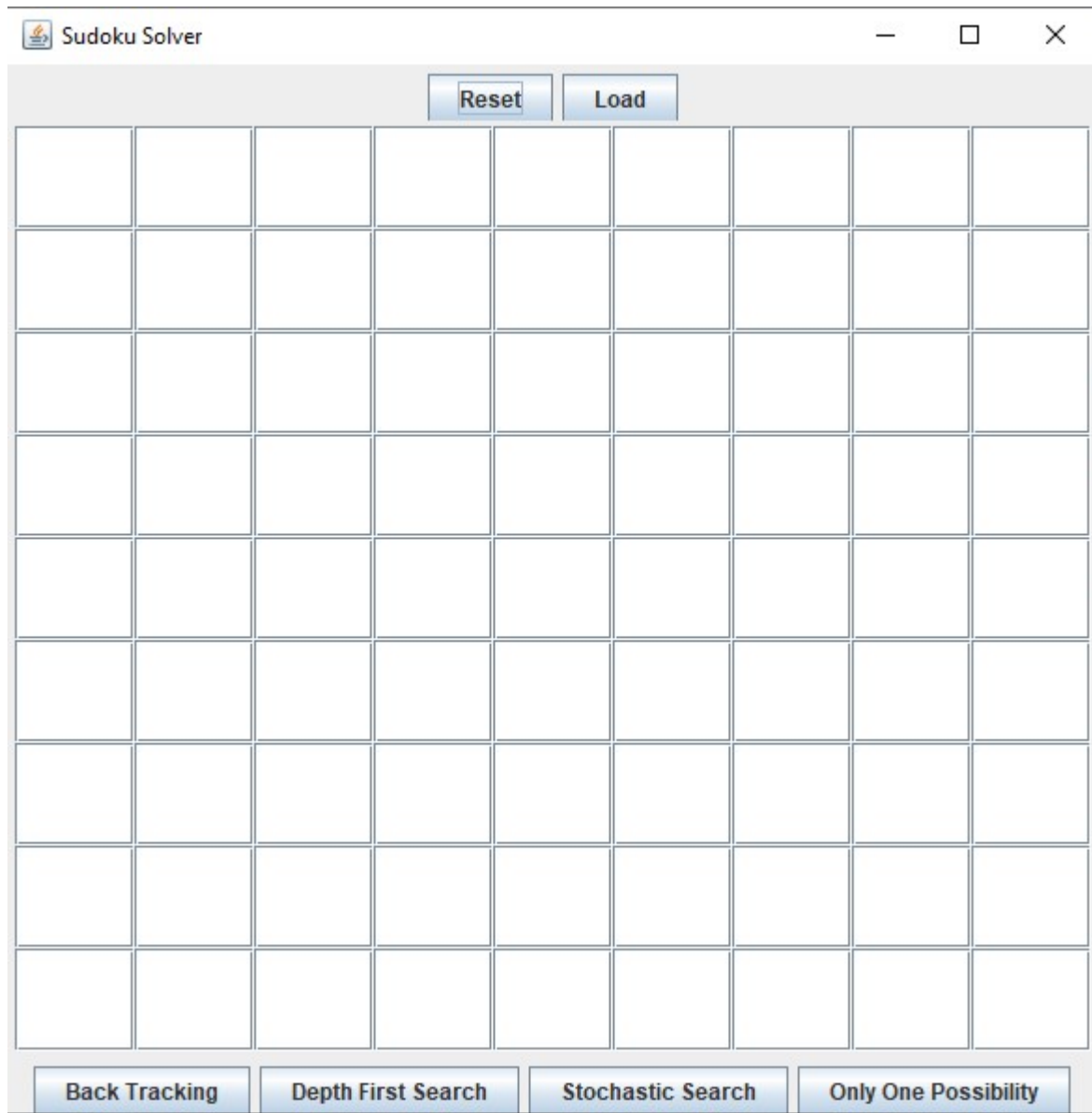
Once the GUI shows up, one can select the file from which they want to load the puzzle. Once we click OK, the unsolved puzzle after all the validation is displayed on the GUI. The text file is read through the SudokuReader class. Once the puzzle is available, one can select the algorithm by which they want to solve the puzzle out of the four options available.

At any point of time, if one wants to reset the grid and load another puzzle, they are free to do so with the help of reset button.

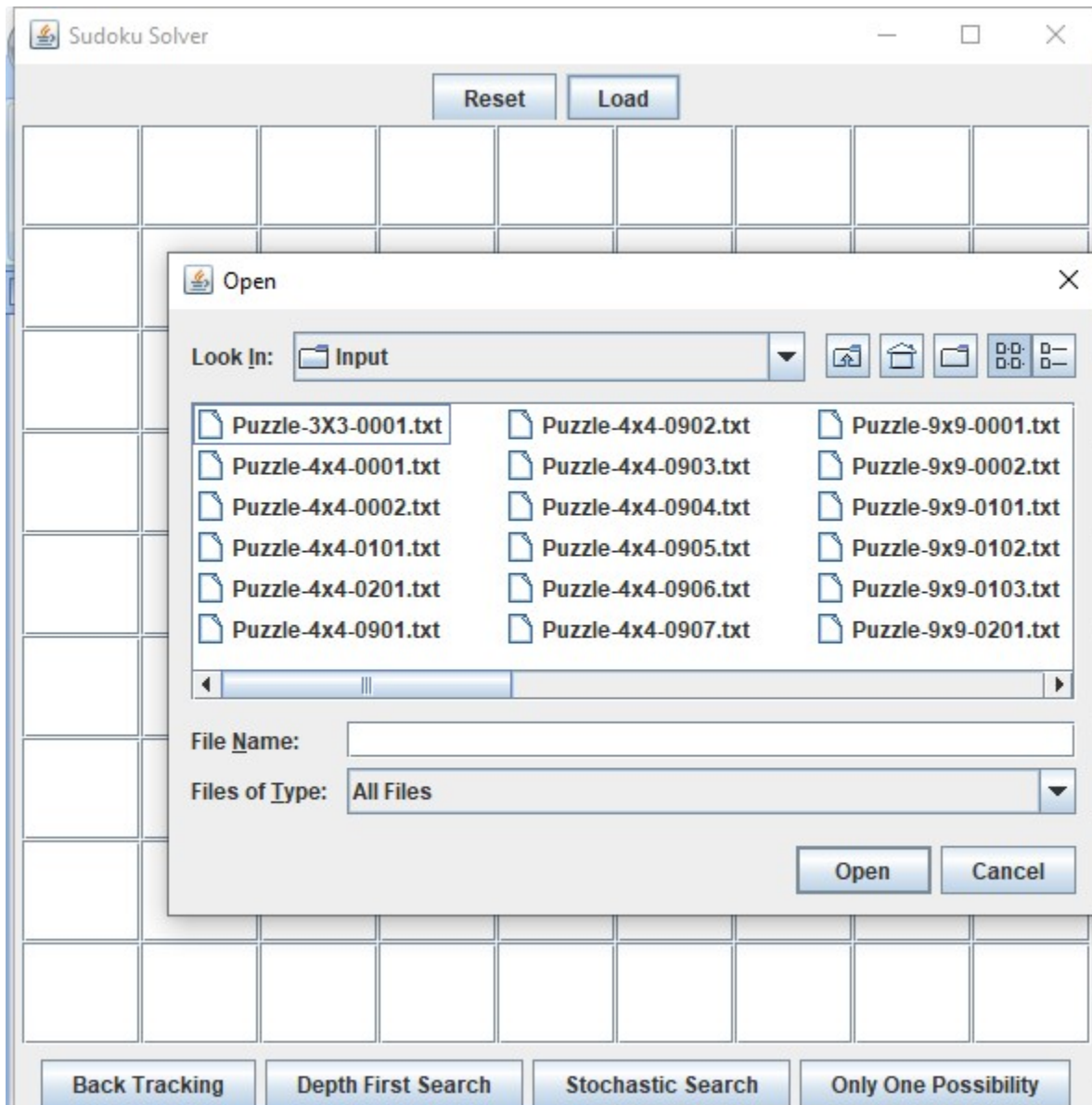
Some of the instances of the GUI is captured and is pasted below:

GUI Screenshots:

Initial View:



Load Button will open File Selector:



Solved Sudoku:

Reset

Load

4	9	2	1	3	6	7	5	8
2	6	3	5	4	8	1	9	7
5	1	8	7	2	9	3	6	4
8	2	5	3	1	7	9	4	6
6	7	4	8	9	5	2	1	3
7	3	1	2	6	4	5	8	9
1	8	6	9	7	3	4	2	5
9	4	7	6	5	1	8	3	2
3	5	9	4	8	2	6	7	1

Back Tracking

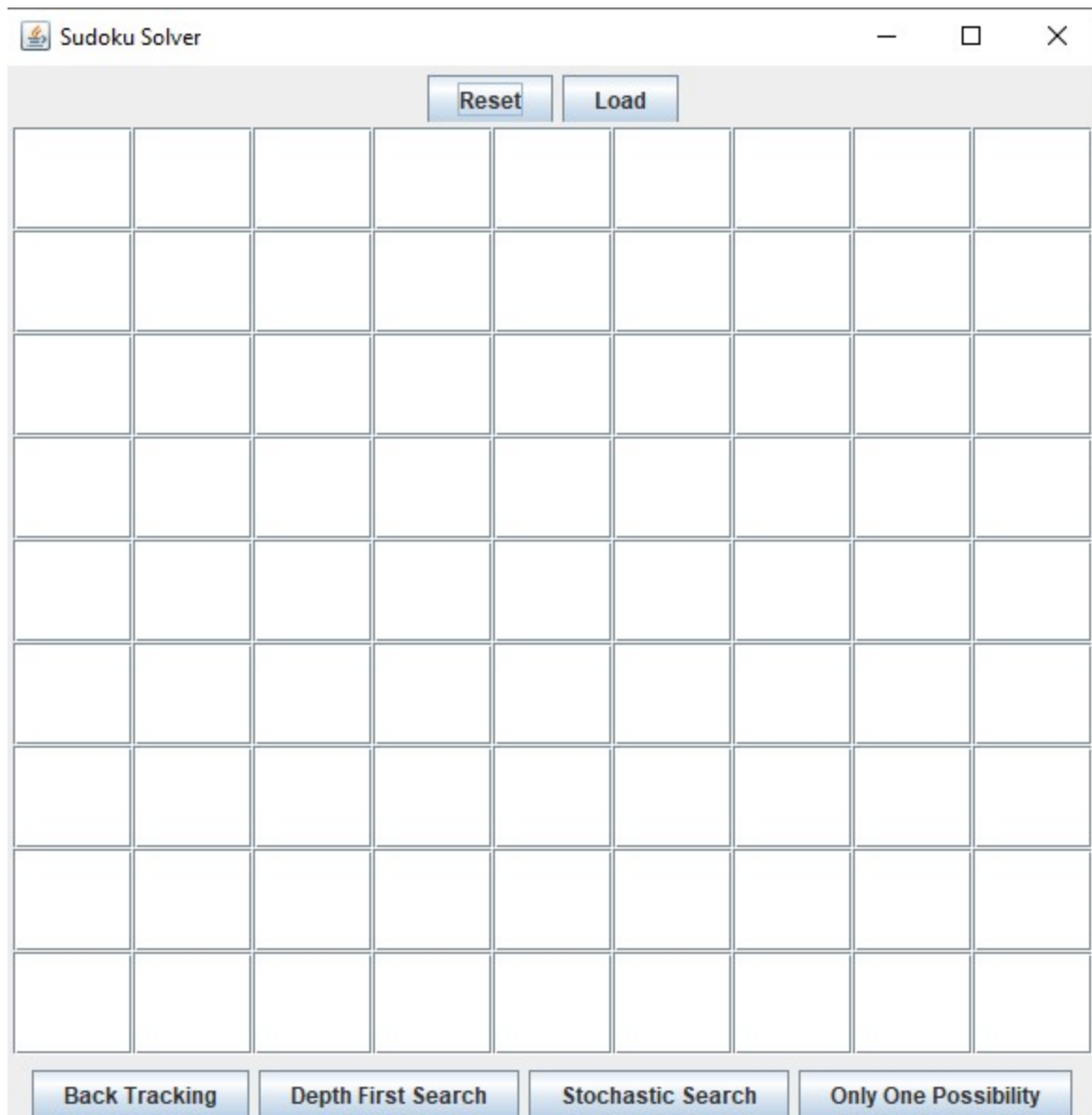
Depth First Search

Stochastic Search

Only One Possibility

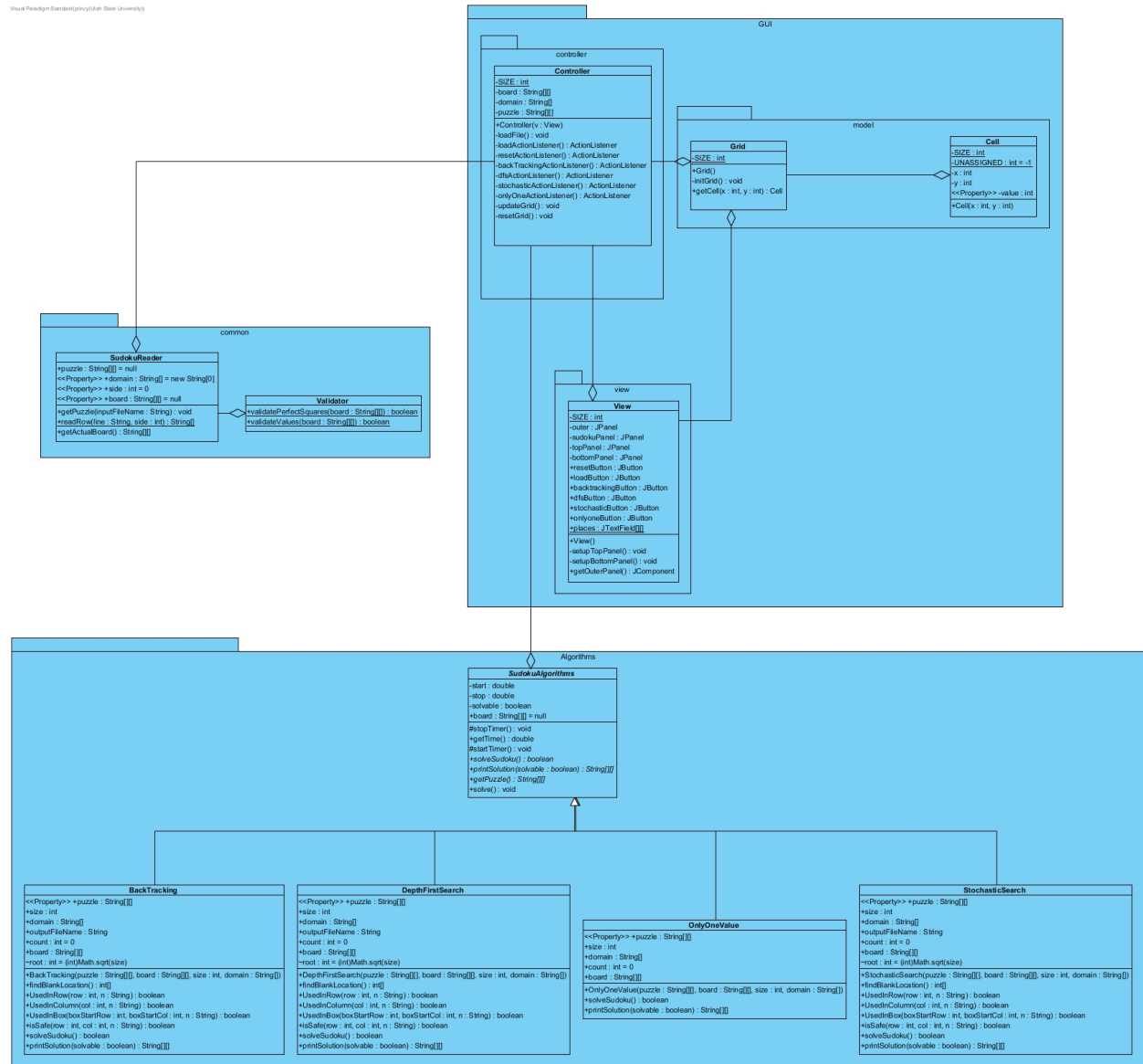
Reset Button:

Reset Button will clear entire grid.



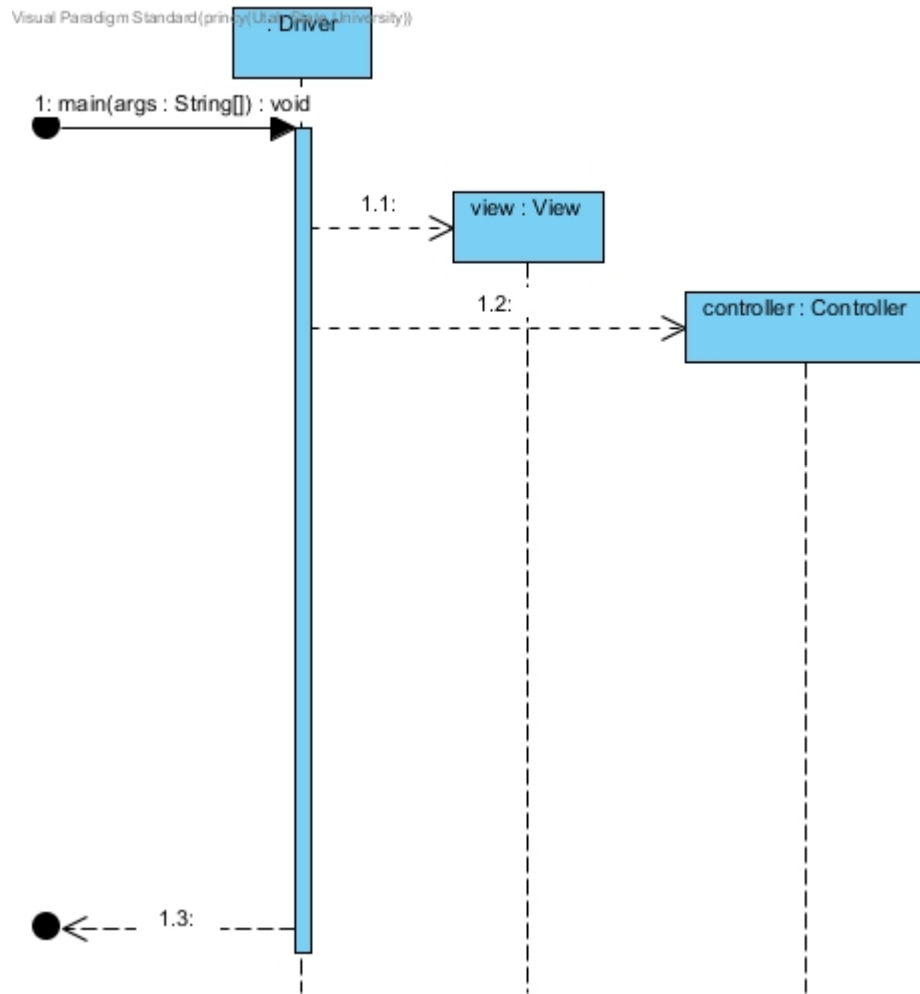
## Class Diagram:

Visual Paradigm Standard (Copyright © 2010, State University)

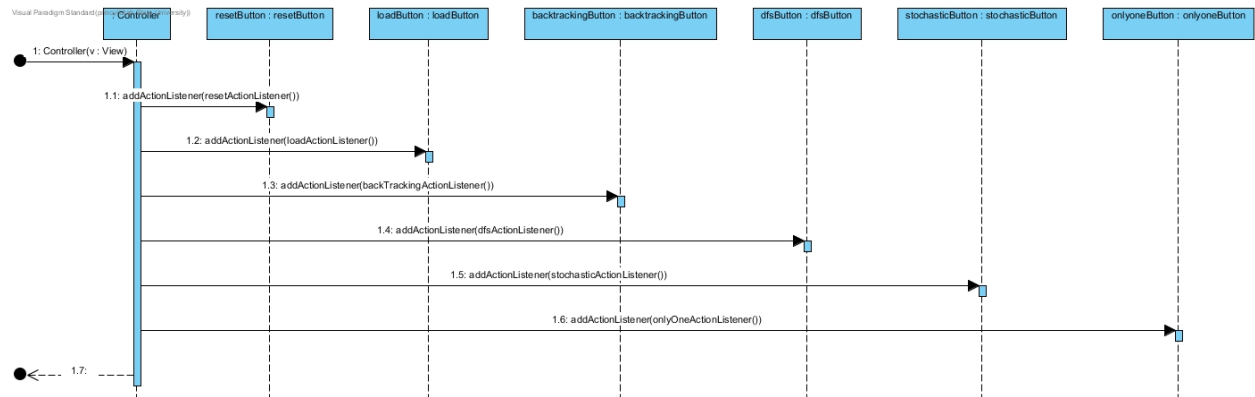


## Interaction Diagram:

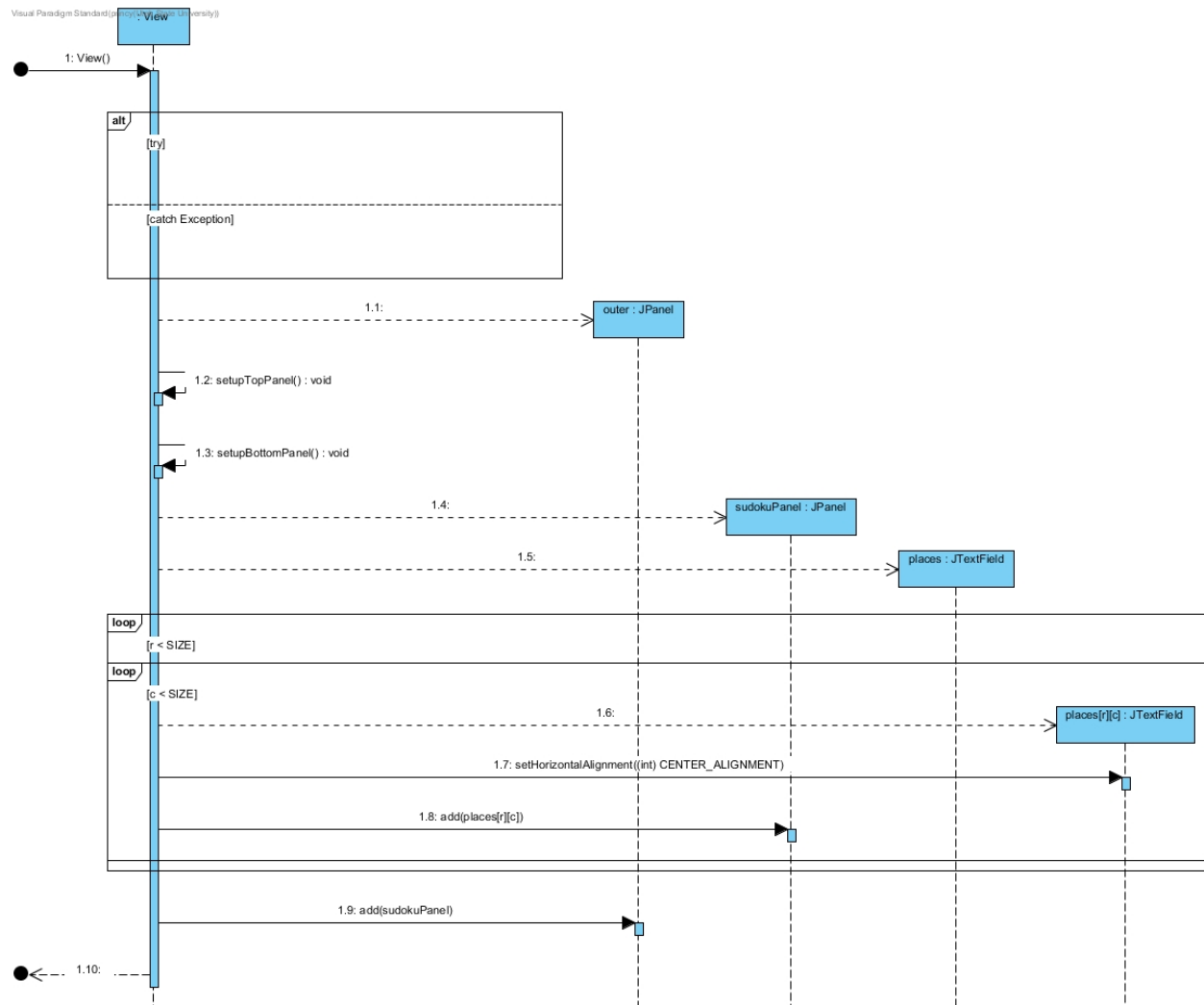
### Main:



### Controller:

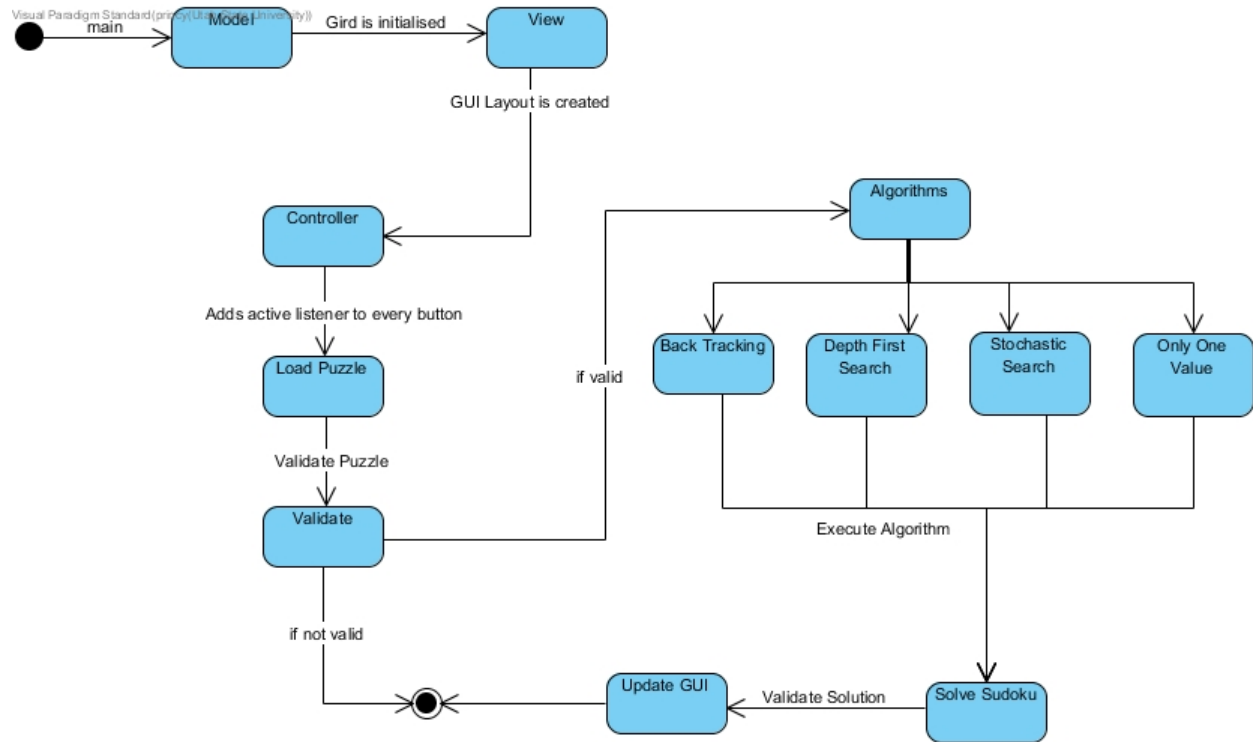


## View:





## State Diagram:



I have completed all the executable unit test cases with a good coverage. The upload




## Algorithm:

100% classes, 92% lines covered in package 'Algorithms'			
Element	Class, %	Method, %	Line, %
BackTracking	100% (1/1)	100% (9/9)	92% (52/56)
DepthFirstSearch	100% (1/1)	100% (9/9)	92% (52/56)
OnlyOneValue	100% (1/1)	100% (4/4)	84% (33/39)
StochasticSearch	100% (1/1)	100% (9/9)	93% (59/63)
SudokuAlgorithms	100% (1/1)	100% (4/4)	100% (15/15)

## Common:

100% classes, 86% lines covered in package 'common'			
Element	Class, %	Method, %	Line, %
SudokuReader	100% (1/1)	100% (6/6)	91% (43/47)
Validator	100% (1/1)	100% (2/2)	71% (10/14)

## GUI:

100% classes, 55% lines covered in package 'GUI'			
Element	Class, %	Method, %	Line, %
 controller	100% (7/7)	58% (14/24)	26% (22/83)
 model	100% (2/2)	100% (6/6)	95% (21/22)
 view	100% (1/1)	100% (5/5)	94% (37/39)

The GUI components are not tested here.

Insights Uncovered:

The biggest learning from this assignment was the implementation of GUI. Before starting with the assignment, I was completely blank on how to create one. So, it took me more time to get familiar with GUI rather than implementing it.

Other learning was the implementation of Model – View – Controller pattern. I had this option of implementing GUI using Observer Pattern but since I have covered it in Assignment 2 already, so thought to go with something else. Hence, MVC came as a fun learning for me.