Final Project Report

SE 450: OBJECT-ORIENTED SOFTWARE DEVELOPMENT

Term: 2016-2017 Autumn

Professor Name : Anthony Freund
Student Name : Aziz Alqulaysh
Student ID : 1805201

(SUCCESSES AND FAILURES) OVERVIEW OF MY EXPERIENCE IN GENERAL AND HOW I APPROACHED THE FINAL PROJECT

I use to think that when I have already provided working template it will make it easier for me to code the rest of it and have it done. However, after this project it turns out that this is not always the case; Dealing with already exited source code or programming template is in fact might going to add more complexity to the project and more effort to understand the whole idea of the project also to get familiar with the code that have almost no documentation which of of course made it even worse. I worked hard just to understand the previous code and make sense of it which means more work before even I begin the real work.

To begin with, in this project I started working on how to implement the GUI to get the input from the user. However, soon I realized that I should finish the main functionality first of the project such as how the behavior of the cars going to be and the traffic lights colors and other important functional requirements. Moreover, the Parameter Manager Demo provided by our professor made it easier for me to just focus on most important functions of the project.

So one of first successful steps is when I followed the advice from professor James that we should start small and that to get all the initial code working, we need to keep everything as simple as possible. So I first focused all my energy on making cars to go all the way to the end of the road. Then I started to change the traffic light from being

boolean values that are blue and yellow to allow them to be green, yellow and red. After that, I worked on making cars stops when it is red and go when it is green.

I found it very challenging that I work on the project— the coding part— and in the same time thinking of a good design pattern to solve the problems that I had. I knew that we must use the Observer pattern because one of the classes provided was already extended the Observable (The class Model extends Observable).

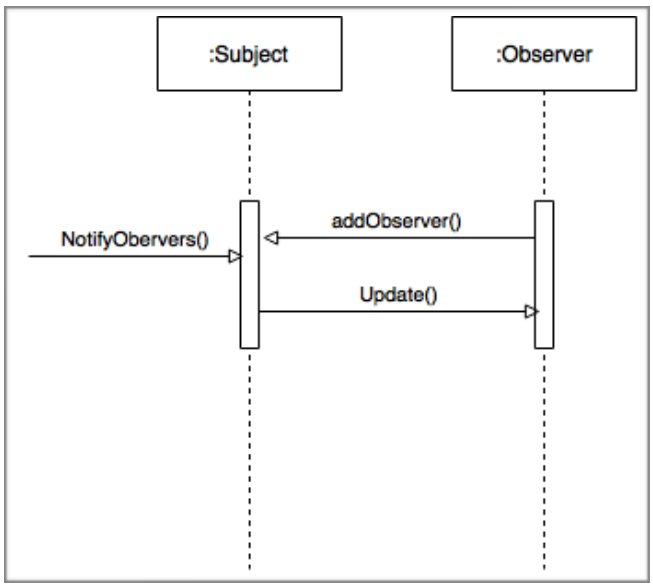
Furthermore, I found also utilizing the Factory pattern in the final project also very beneficial especially after providing the (ParameterManagerDemo) which uses the JSON file. By implementing this pattern it might be very easy in future to make new changes or add new objects without going into a lot of modification.

In addition I found that the project example provided by Professor James very useful it provided me the level of complexity that we should have in the project. Because at first I thought the project is just filling the missing prats of the codes like what I use to do in almost all other courses. But after exploring his example I realized that we needed to add many other interfaces and abstract classes to the project and it will be almost impossible to build the project without adding more classes.

Lastly, I found it also confusing when it came to the UML diagrams since there were a tremendous number of classes and methods and interactions between them so I am not sure if I picked the more important classes because for some reasons I think they are almost all important.

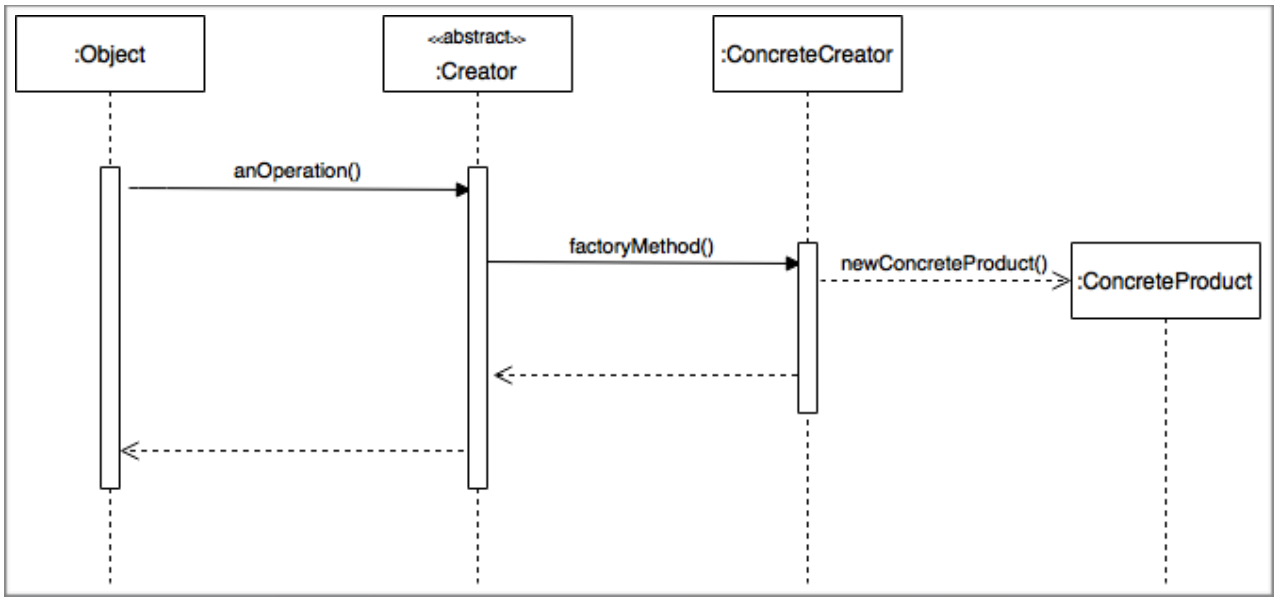
UML SEQUENCE DIAGRAMS:

The Observer Pattern Sequence Diagram:



(Figure 1)

The Factory Pattern Sequence Diagram:



(Figure 2)

DISCUSSING THE PREVIOUS MENTIONED TWO DESIGN PATTERNS BRIEFLY AND HOW THEY ARE APPLIED INTO THE FINAL PROJECT.

The Observer Pattern:

The Observer pattern is definitely one of my favorite pattern especially after utilizing it in final project and had the opportunity to see its real benefits of it. So the observer pattern is a software design pattern in which notifies other responsibility automatically of any state that might occur or change during the running of an the application or the software. It is usually being used to implementing distributed events handling systems. Hence, in this project it was very useful because there were a tremendous number of events that need to be handle and get response at the same time.

In addition, the observer pattern is also a very important part in the most famous model–view– controller (MVC) architectural pattern and its part is the (View). In this project the model was the Model class which is responsible of controlling and managing the data, logic, and rules of the application. The View is the user interface is the PopUp class which allows the user to see data on the screen. Lastly the controller is the UIMenu class which is allows the user to enter data and control the application.

Moreover, The observer pattern is implemented in numerous programming libraries and systems, including almost all GUI toolkits. So it is very useful. Its strength is that it is very powerful when it becomes in handling multiple events at the same time. However, some of its weaknesses is that it might cause memory leaks although this might be avoided by other available solutions.

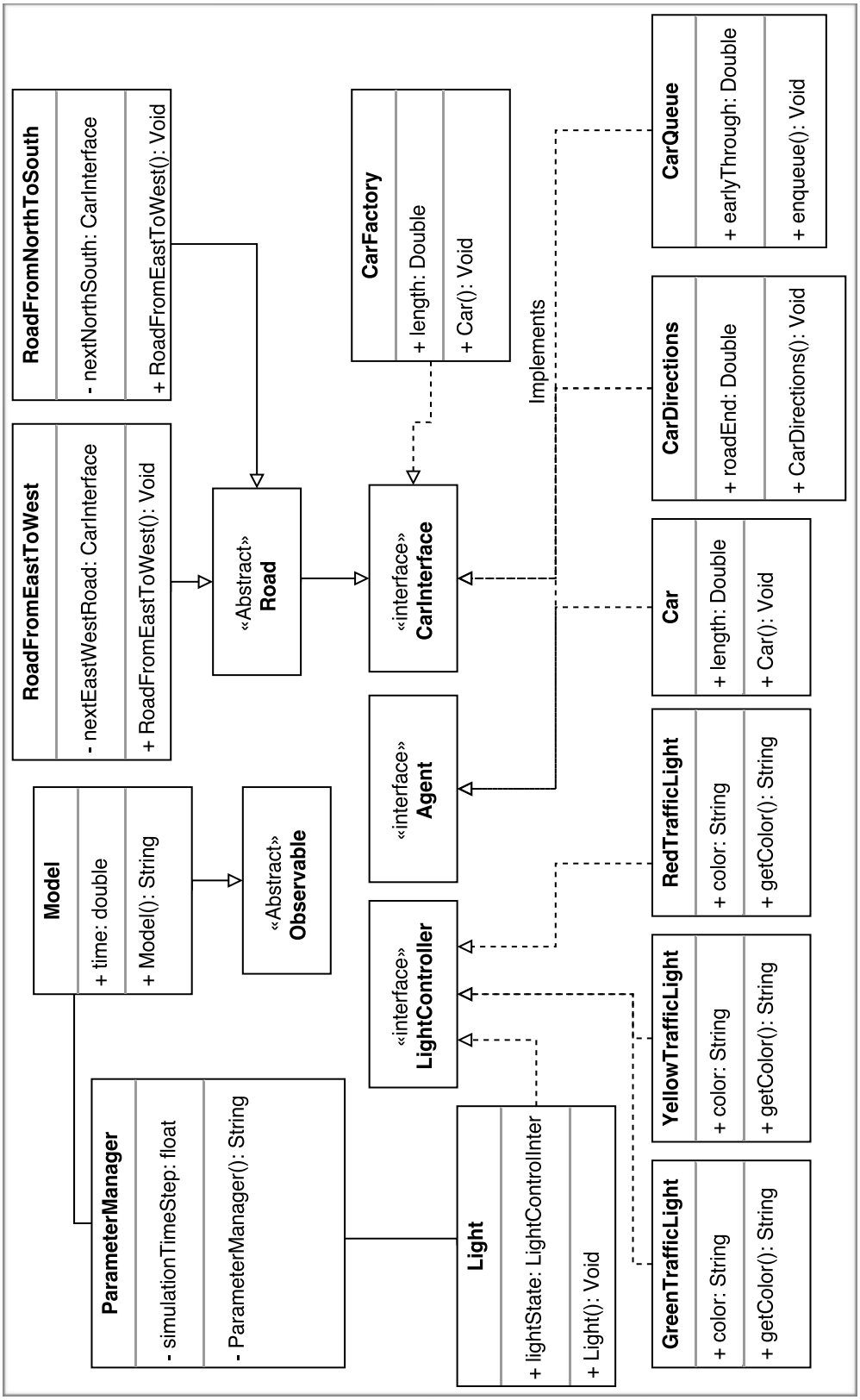
The Factory Pattern:

The Factory pattern was very useful for me in this project because it helped me to overcome the problem of creating objects without specifying a particular class of the object that I am going to be creating. So in my case in this project using the CarFactory class allows to me to create any kind of car different shapes and different feature.So for example, if I want to build a car that have many similar types of objects and in the future I might need to add more details for any car that I might add.

Therefore, the Factory pattern was the perfect choice because also when I am going to add a new class later in the future it will be very easy to do so I'm not going to have to change a lot of things in order to that so my code will be maintainable and flexible. In addition, some of the strengths that this pattern have is that it is is easy to implement and makes the code easy to change and easy to add more new classes or objects without needing to fix a huge amount of code and previous work that I have done.

Lastly, I think that some of weakness is that this design pattern have is that the code might look too abstract and hence that might make it difficult to understand by other people who want to change or add something in code. Also, I believe that it might add some extra work for small projects so this design pattern should only be used with large projects.

UML CLASS DIAGRAM



(Figure 3)

CONCLUSION AND ACKNOWLEDGMENT

Going through this project is definitely one of the biggest challenges here at DePaul University. At first I thought it is would be impossible to have it done just within almost a couple of weeks. However, after reading previous students success and failures comments they really inspired me to do something and not give up. Even though I am not completely satisfied with the overall result of my project but at least I did not give up and I did my best until the last day. Thanks for our professors Anthony Freund and James Riely as well as all my classmates and friends who helped me a lot.