|  |
| --- |
| Fontys ICT |
| pattern |
| Design patterns |

|  |
| --- |
| Jan-Niklas Schneider, Georgiana Manolache  9-13-2016 |

Table of Contents

[1 Introduction 3](#_Toc461572470)

[2 Observer Pattern 3](#_Toc461572471)

[3 Implementation 3](#_Toc461572472)

[3.1 Explanation of classes 3](#_Toc461572473)

[3.1.1 Subject 3](#_Toc461572474)

[3.1.2 Subject 3](#_Toc461572475)

[3.2 Features 4](#_Toc461572476)

[4 Design choices 4](#_Toc461572477)

[5 Graphical user interface 4](#_Toc461572478)

[6 Unit tests 4](#_Toc461572479)

[7 References 4](#_Toc461572480)

# Introduction

The goal of this document is to give an overview over the design pattern observer pattern by giving an example implementation which leverages real weather data gathered through the *OpenWeatherMapApi*. Furthermore, reusability, extensibility, and maintainability of this pattern are elaborated. Also, the implementation, its unit test and graphical user interface (GUI) are reviewed.

# Observer Pattern

The observer pattern is a software design pattern which

# Implementation

Figure 1: Class diagram

The figure above depicts a class diagram of the implementation of a WPF application that showcases weather data in real time.

In addition to the implementation of the strategy pattern, the Model–view–viewmodel architectural pattern (MVVM) has been utilized. Despite the influence in the application it has minor implication to the strategy pattern. Thus, it is shown only partially in the class diagram to retain a full overview of the present classes but leaving focus on the design pattern. This also holds for the next subchapter which explains classes of the pattern itself but leaving out MVVM or the *MainWindow* class.

## Explanation of classes

This subchapter gives a descriptive explanation of the observer patterns implementation, such as methods, properties or fields.

### ISubject

*ISubject* holds the methods of the Observer pattern Subject interface: Attach, Detach, Notify.

|  |  |  |
| --- | --- | --- |
| **ISubject** | | |
| Type | Definition | Explanation |
| Method | + Attach(IObserver observer): void |  |
| Method | + Detach(IObserver observer): void |  |
| Method | + Notify(): void |  |

### Subject

*Subject* is the abstract class which implements all the methods from *ISubject*.

|  |  |  |
| --- | --- | --- |
| **Subject** | | |
| Type | Definition | Explanation |
| Method | + Attach(IObserver observer): void |  |
| Method | + Detach(IObserver observer): void |  |
| Method | + Notify(): void |  |

### WeatherSubject

The *WeatherSubject* implements all methods form abstract class *Subject*.

|  |  |  |
| --- | --- | --- |
| **WeatherSubject** | | |
| Type | Definition | Explanation |
| Method | + ReadRequest(List<Request> requests, Request request) : Request | The interface method which strategies derive from |

### WeatherAlertSubject

The *WeatherAlertSubject* implements all methods form abstract class *Subject*. This is the second subject which is implemented in a pull request method.

|  |  |  |
| --- | --- | --- |
| **WeatherAlertSubject** | | |
| Type | Definition | Explanation |
| Method | + ReadRequest(List<Request> requests, Request request) : Request | The interface method which strategies derive from |

### IObserver

|  |  |  |
| --- | --- | --- |
| **IObserver** | | |
| Type | Definition | Explanation |
| Method | + Update(object data): void | The interface method which strategies derive from |

## Features

# Design choices

.

# Graphical user interface

Figure 2: User Interface of the application

# Unit tests

For each implemented strategy unit tests have been defined to assert correct behavior.

# References

MSDN Microsoft. (2016, September). *Observer Design Pattern*. Retrieved from MSDN Microsoft: https://msdn.microsoft.com/en-us/library/ee850490(v=vs.110).aspx