Universität Basel

Software Architecture

Exercise – Box and Arrow Diagrams

BSc

Ingo Arnold

Exercise Opening
Overview

An essential element of software architecture is to abstract from details that are not "architecture-relevant". See also the article Design vs. Architecture by Paul Clements of the SEI ([Clemens] – available on the file server).

As a prerequisite for performing the *observer exercise*, please study the observer design pattern.

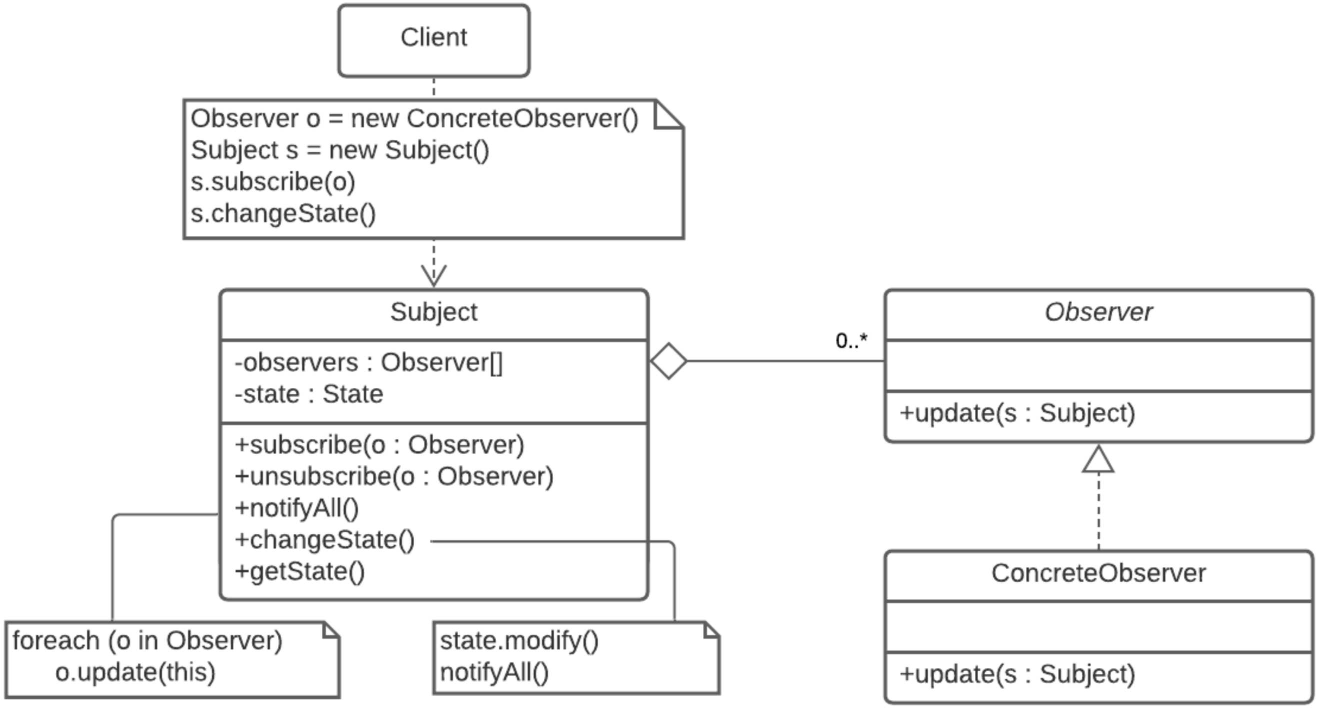
Exercise Agenda

- Observer
- Demilitarized Zone

Observer
Overview

The representation of software architecture as interacting components allows freedom to be left for design decisions that are not classified as architecture-relevant.

Examine this with the concrete example of the *observer design pattern*.



Client

```
Observer o = new ConcreteObserver()
Subject s = new Subject()
s.subscribe(o)
s.changeState()
```

Subject

-observers : Observer[]
-state : State

+subscribe(o : Observer)
+unsubscribe(o : Observer)
+notifyAll()
+changeState()
+getState()

Observer

+update(s : Subject)

0..*

ConcreteObserver

+update(s : Subject)

```
foreach (o in Observer)
    o.update(this)
```

```
state.modify()
notifyAll()
```

Universität Basel

Observer
Overview

The four boxes in the *observer pattern* diagram represent four architecture components. The arrows between them represent different relationships between these components.

Answer the following questions for the above example:

a) What properties of the overall system does the *observer architecture* ensure?

b) From which design decisions (non-architecture decisions) does the *observer pattern* abstract and to which of the individual components are they left? Specify such non-architecture design decisions together with the components to which they are left.

Observer

Overall System Properties

a) what properties of the overall system does the observer architecture ensure?

Observer
Open Design Decisions

b) From which design decisions (non-architecture decisions) does the observer pattern abstract and to which of the individual components are they left? Specify such non-architecture design decisions together with the components to which they are left?

Exercise Agenda

- Observer
- Demilitarized Zone

Demilitarized Zone
Overview

Box-and-arrow diagrams can represent various structures of abstractions and dependencies, not only software components but also, for example, subnetworks in IT networks.

An important relationship between such subnets is the ability to open a connection from one to another. This requires that devices can be addressed and that no firewall blocks the connection.

A common IT network configuration pattern distinguishes between an *intranet* that is shielded from the outside (i.e. an internal network area that cannot be accessed by devices from outside), a so-called *demilitarized zone* (DMZ) in which servers are located whose services are also to be used from the public Internet (e.g., by customers), and the globally open *Internet*.

Illustrate the situation described above in a simple box-and-arrow diagram that shows from which network areas components can open connections to components in which other network areas.

Demilitarized Zone

DMZ in a box-and-arrow diagram

How is this represented by a box-and-arrow diagram?

Bibliography

Lecture

[Clemens]
Clemens, Paul; *What is the difference between Architecture and Design?* 1.3-BSc_SWA_WHAT_Concepts_Exercise_Paul Clements_Design vs Architecture

Questions