Hasan Ünalan, Sebastian Seifert

Übungsblatt Nr.1

Aufgabe 1

Was sind die Aufgaben von Ports?

Ports sind Klassen die einen Typen bzw. eine Rollenangabe besitzen. Sie definieren Kommunikationsschnittstellen, die den strukturierten Classifier abkapseln. Der Port übernimmt die Aufgabe der Kommunikation. Dabei findet sowohl die Kommunikation von außen zum Classifier als auch umgekehrt über den Port statt.

Ports können auch weitere Aufgaben übernehmen wie Filterung, Caching oder Protokollüberwachung.

Wie könnte man deren Aufgabe für eine Komponente in Java implementieren?

Implementiert könnte die Aufgabe der Ports durch ein Interface. Die Interfaces geben die Methoden vor und implementiert wird eine Klasse, die das Port darstellt. Dieser dient zur Kommunikation zwischen internen und externen Klassen. Implementiert wird das Interface in dieser Klasse. Durch diesen Port geschehen alle Operationen, die von innen nach außen bzw. von außen nach innen geschehen.

Wie können benötigte bzw. angebotene Interfaces injiziert bzw. entnommen werden?

Mit der Realisierung des Ports durch ein Proxy-Pattern können die Interfaces von der Proxy-Klasse injiziert bzw. entnommen werden.

FA0

Es sollte das Proxy Pattern verwendet werden. In dieser wird ein Interface definiert welches die nach außen/innen zugängliche Methoden definiert. Dann wird diese sowohl in der sich innen befindende Klasse implementiert als auch in der Proxy Klasse (in unserem Fall Port). Von außen werden dann aufrufe exklusiv über die Proxy Klasse getätigt und von der Proxy Klasse aus werden die Methoden in der tatsächlichen Klasse indirekt aufgerufen.[1] (s. Abb. 1)

[1] Design Patterns - Proxy Pattern - Tutorialspoint 2020.

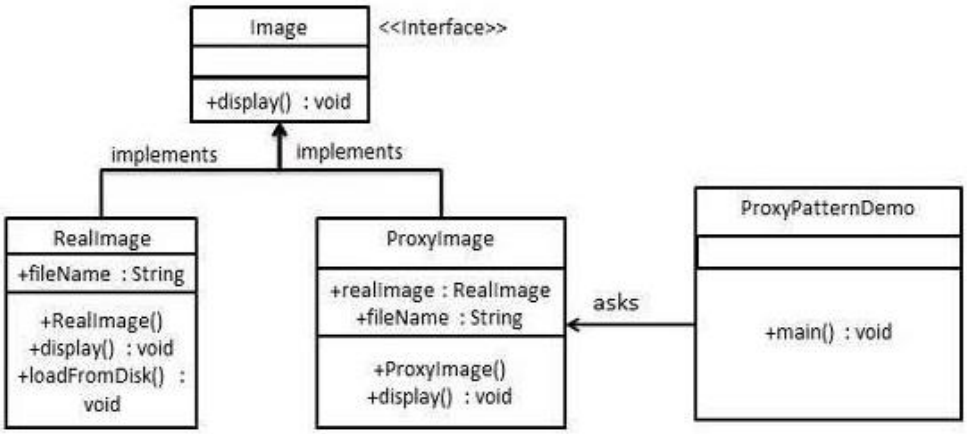

Abb. 1: Proxypattern[2]

FA1

Es wird erst das Interface HotelSuche aufgerufen mit einer Referenz auf HotelSuchePort. Von Dort aus wird das Logging und Caching Interface aufgerufen, wobei bei diesen die Reihenfolge nicht entscheidend ist.

FA2

Caching ist nicht hinreichend definiert. Es besitzt nur eine Methode, um Suchbefehle und die entsprechende Liste im Cache abzuspeichern aber keine, um sie auszulesen. Ohne die Möglichkeit den Cache auszulesen macht dieser auch keinen Sinn zu verwenden.

FA6

Die HotelRetrievalExtended ist nur eine „dummy“ Klasse die technisch gesehen eine Kopie von HotelRetrieval ist. Der Nutzer muss sich beim Zugreifen auf den HotelSuchePort durch eine Navigation die Implementierung des Interfaces HotelSuche eine der beiden Klassen aussuchen. Dabei kann theoretisch in der Klasse HotelRetrievalExtended eine erweiterte Suche in die Methode implementiert werden.

[2] Design Patterns - Proxy Pattern - Tutorialspoint 2020.

Hasan Ünalan, Sebastian Seifert

Code um die .jar zu testen:

```
import org.hbrs.ooka.ws2020.uebung1.Hotel;
import org.hbrs.ooka.ws2020.uebung1.ext.hotelsuche.HotelSuche;
import org.hbrs.ooka.ws2020.uebung1.ext.hotelsuche.HotelSuchePort;

public class ClientTest {

    public static void main(String[] args) {

        HotelSuche hsPort = new HotelSuchePort();
        hsPort.openSession();
        Hotel[] hotels = hsPort.getHotelByName("");
        // TODO remove testing
        for (int i = 0; i < hotels.length; i++) {
            System.out.println(hotels[i].getName() + "," +
hotels[i].getLocation());
        }
        hotels = hsPort.getHotelByName("");
        hsPort.closeSession();
    }
}
```

(Als seperates Projekt und mit der .jar importiert)

Github Link (Hasan Ünalan): https://github.com/Pai7CE/OOKA

Literaturverzeichnis

Design Patterns - Proxy Pattern - Tutorialspoint (2020). Available online at https://www.tutorialspoint.com/design_pattern/proxy_pattern.htm, updated on 10/17/2020, checked on 11/12/2020.