



3. April 2022

Übungen zur Vorlesung Objektorientierte Komponenten-Architekturen

SS 2022

Übungsblatt Nr. 1

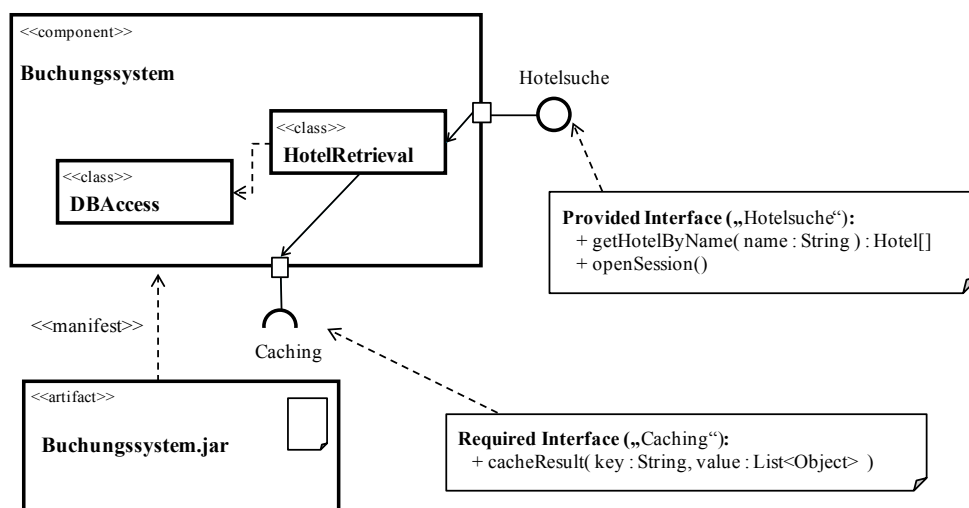
(Abgabe bis: Erster „Aufschlag einer Lösung in der Übung am 4.4.2022, finale Abgabe bis Montag, den 11. April 2022, 12:00 Uhr)

Aufgabe 1 (Source Code zu Komponenten):

Die Java-basierte Umsetzung des Komponentendiagramms (vgl. Folie 26 bzw. 27) entspricht *einer* Möglichkeit der Abbildung des Modells auf einen äquivalenten Source Code. In dieser Aufgabe sollten sie den Source Code optimieren, in dem sie das *Port*-Konzept aus diesem Diagrammtyp anwenden und exemplarisch implementieren. Beantworten sie folgende Fragen:

- Welche Aufgabe haben Ports im Kontext eines Komponentendiagramms? Recherchieren sie dieses Konzept anhand der Quelle von (Rupp, 2012), Kapitel 9.1.3 und 9.4.3 (siehe Auszug in LEA, Ordner Literatur zu Kapitel 1)
- Wie könnte man deren Aufgabe für eine Komponente in Java implementieren? (Siehe dazu auch die Anforderung FA0)
- Wie können benötigte bzw. angebotene Interfaces injiziert bzw. entnommen werden?

Rupp et al.: *UML2 Glasklar. Praxiswissen für die UML-Modellierung*. 5. Auflage. Hanser, 2012. (in der Bibliothek mehrfach vorhanden).



Sie haben nun die Aufgabe, das obige Modell eines Buchungssystems auf Grundlage eines Port-Konzepts zu implementieren! Verwenden sie dazu die Sprache Java nebst einem JDK und einer JRE (ab 1.8). Verwenden sie auch eine entsprechende IDE ihrer Wahl (meine Empfehlung: IntelliJ). Auf dem GitHub-Repository der Vorlesung finden sie dazu eine Klasse `DBAccess` (als .java Datei), die einen Datenbankzugriff auf eine interne Datenbank der Hochschule realisiert, mit der Rohdaten über Hotels abgefragt werden können. Diese Klasse können sie hier runterladen bzw. das Repo in ihre IDE clonen:

<https://github.com/aldaGit/codesOOKA>

Für den Datenbank-Zugriff brauchen sie eine gültige VPN-Verbindung! *Der notwendige Datenbanktreiber wird über Maven in das Projekt hinzugefügt.* Betten sie die Klasse in die Komponente **Buchungssystem** gemäß des UML Komponentendiagramms (siehe Abbildung oben) ein. Diese Klasse können sie von einer Kommandozeile aus oder aber in einer IDE starten. Es wird dabei ein Mini-Tutorial ausgegeben, wie man diese verwendet (Ausgabe auf der Konsole). Realisieren sie nun folgende funktionalen Anforderungen. Die dazu gehörigen Fragen bitte kurz in einem Text-Dokument adressieren und beantworten:

FA0: Implementieren sie das Port-Konzept nach den Vorgaben bzw. Überlegungen gemäß (Rupp, 2012). Welches Design Pattern sollte hier verwendet werden, um die notwendige Delegation zwischen internen und externen Verhalten zu realisieren?

FA1: Offensichtlich ist die Ausgabe der Klasse `DBAccess` nicht sonderlich objektorientiert! Ihre Aufgabe soll es daher sein, die Ausgabe so zu transformieren, dass sie über die öffentliche Schnittstelle der Komponente **Buchungssystem** nur Objekte vom Typ `Hotel` zurückliefern. Die Attribute der Klasse `Hotel` sind entsprechend selber zu bestimmen, die Klasse selber ist in einem separaten Subsystem zu verlagern. Muss das Interface `Hotelsuche` ggf. noch um weitere Methoden erweitert werden? Beachten sie dazu auch die Tutorial-Ausgabe der Klasse `DBAccess`! Gibt es eine dedizierte Reihenfolge beim Aufruf der Methoden des Interfaces?

FA2: Die Komponente `Buchungssystem` benötigt ferner eine Referenz vom Typ `Caching`, mit der die interne Klasse `HotelRetrieval` die Ergebnisse in einem Cache zwischenspeichern kann. Von außerhalb der Komponente muss also eine entsprechende Referenz erzeugt werden und über den Port injiziert werden. Ist die Schnittstelle `Caching` hinreichend modelliert oder fehlen auch hier Methoden? Implementieren sie die Implementierung eines konkreten Cache *rudimentär*.

FA3: Überlegen sie auch einen Mechanismus, damit `HotelRetrieval` stets zumindest scheinbar ohne Probleme (z.B. keine `NullPointerException`) auf den Cache zugreifen kann, auch wenn *keine* konkrete Referenz gesetzt ist. Ein etwaiges Fehlerhandling darf dabei *nicht* von der Klasse `HotelRetrieval` übernommen werden.

FA4: Realisieren sie zudem eine Logging-Funktionalität, mit der die Zugriffe auf das Interface Hotelsuche geloggt werden. Eine Ausgabe sollte wie folgt sein:

01.10.22 21:22: Zugriff auf Buchungssystem über Methode getHotelByName. Suchwort: Berg

Auch das Logging ist eine Querschnittsfunktionalität, die *nicht* in der Klasse HotelRetrieval enthalten sein soll.

FA5: Ihre gesamten Entwicklungen sollen dann als „deploybare“ Komponente im Format .jar exportiert werden. Testen sie ihre Entwicklungen hinreichend mit einem externen Client (nicht Teil der deploybaren Komponente!).

- Hinweis: bitte an dieser Stelle noch *keinen* Microservice erzeugen!

FA6: Eine häufige Forderung an Komponentenmodellen ist die Navigation über vorhandene Schnittstellen einer Komponente (vgl. Kapitel 2, Folie 23). Wie könnte so ein *einfacher* Navigationsmechanismus aussehen? Erweitern sie das Port-Konzept ihrer Anwendung entsprechend, in dem sie erlauben, dass ein Client bei einem Port über alle verfügbaren „Provided Interfaces“ navigieren kann, um dann in einem nächsten Schritt ein passendes Interface zu wählen. Als Beispiel sollten sie für den „Suche-Port“ zwei Schnittstellen anbieten, der eine einfache und eine erweiterte Suche ermöglicht. Die eigentliche Auswahl auf Client-Seite brauchen sie nicht zu implementieren, sollten aber über Herausforderungen für eine Implementierung kurz diskutieren.

R1: Modellieren sie die resultierenden Klassen und die Abhängigkeiten ihrer gesamten Software als ein Klassendiagramm nach der UML.

Bitte laden sie als Ergebnis die Source Codes, das .jar File sowie kurze Antworten zu den obigen Fragen (wenn vorhanden) auf LEA hoch. Als Alternative können sie den Source Code auf ein eigenes öffentliches GitHub-Repository bereitstellen und den Link dazu auf LEA hochladen (z.B. in einer readme-Datei). Die Antworten zu den Fragen sollten als PDF bereitgestellt werden.

Allgemeine Hinweise:

Zur Einrichtung einer VPN-Verbindung sei folgende Seite empfohlen:

<https://ux-2s18.inf.h-brs.de/faq/vpn>

Für die Modellierung mit UML empfehle ich folgende Tools:

- UMLet 14.3

Gutes und übersichtliches Tool zur Modellierung, installierbar für Windows / MAC / Linux. Unterstützt keine Sequenzdiagramme (Anpassung möglich). Bedienung etwas gewöhnungsbedürftig zu Beginn, dann aber sehr effektiv ;-)

Quelle für Download: <http://www.umlet.com/>

- Draw.io

Schlankes browser-basiertes Tool, keine Installation auf ihrem Rechner notwendig! Abspeicherung der Dokumente in verschiedenen Formen möglich (Lokal, Cloud). Läuft nativ ohne Plugin auf allen gängigen Browsern.

Quelle: <https://www.draw.io/>

Für die anstehenden Entwicklungen empfehle ich das Tool IntelliJ. Hier empfiehlt sich der Download der Ultimate-Version, die als registrierter Student kostenlos bezogen werden kann! Mit dieser Ultimate-Version können u.a. auch Java EE Anwendungen entwickelt werden. Auch für moderne Entwicklungsstandards wie Microservices, Docker, Maven usw. bietet IntelliJ einen guten Support. Unübertroffen ist die Auto Completion Funktion, welches IntelliJ recht populär gemacht hat.

<https://www.jetbrains.com/idea/>

Sollten sie noch Probleme haben mit der IDE oder auch mit der Integration von GitHub, so verweise ich gerne auf eine eigene Tutorien-Reihe.

Teil 1: Installation IntelliJ und Entwicklung eines Java-Projekts mit JUnit5 –

<https://www.youtube.com/watch?v=TNtRpkdW64s>

Teil 2: Clone eines GitHub-Repository mit IntelliJ –

<https://www.youtube.com/watch?v=5nr4c3pwu3g>

(Hinweis: in diesem Tutorium wird ein anderes GitHub-Repro verwendet; bitte den o.g. Link zu dem OOKA-Repro verwenden)

Teil 3: Push von Source Code auf ein GitHub-Repository mit IntelliJ

<https://www.youtube.com/watch?v=PbGiYUR9q0A>