Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Prof. Dr. Sascha Alda
Robert Hartmann (Dipl.-Inform.)

8. Dezember 2021

Übungen zur Vorlesung Software Engineering I
WS 2021 / 2022

Übungsblatt Nr. 8
(Abgabe bis: Mittwoch, den 15. Dezember 2021, **09:00 Uhr**)

Aufgabe 1 (Übung zu Design Patterns, Einbindung Reise-Anbieter, 20 Punkte)

In der Vorlesung haben sie die Schicht-Architektur einer Software für Reise-Portals kennengelernt (vgl. Kapitel 5, Folien 43-45). Wie zu sehen, ist hier die Schnittstelle eines Reise-Anbieters bereits in der Architektur berücksichtigt (Boundary: `ReiseAnbieter`). Die Control-Klasse `Hotelsuche` soll diese Schnittstelle verwenden.

Bei einer genaueren Analyse während des Objekt-Designs stellt ein Software-Architekt fest, dass die Eingabe- und Ausgabetypen von der Klasse `Hotelsuche` bzw. von dem Interface des Reise-Anbieters *nicht kompatibel* sind. Die Schnittstelle des Reise-Anbieters verfügt über folgende Schnittstelle:

```
+ executeQuery ( d : QueryObject ) : QueryResult
```

Das Reise-Portal verwendet intern die Klassen `SuchAuftrag` zur Beschreibung eines Suchauftrags sowie die Klasse `SuchErgebnis` für die interne Repräsentation eines Suchergebnisses. Die Schnittstelle des Reise-Anbieters wird seit Jahren eingesetzt und besitzt viele Abhängigkeiten von anderen Systemen, so dass die Schnittstelle *nicht* verändert werden darf. Das Software-System des Reise-Anbieters, welches die o.g. Schnittstelle anbietet, ist also ein Legacy-System.

Der Software-Architekt stellt fest, dass sowohl Eingabe- als auch Rückgabeparameter zur Entwicklungszeit *abbildbar* sind und somit zur Laufzeit *transformiert* werden können. Dennoch steht der Software-Architektur vor einem Entwurfsproblem, zu der er keine Lösung kennt, und er bittet sie daher um ihre Unterstützung.

a.)
Welches Entwurfsmuster würden sie für dieses Entwurfsproblem verwenden? Modellieren sie die Lösung zu diesem Entwurfsproblem als UML-Klassendiagramm unter der Verwendung des identifizierten Musters. Modellieren sie die Signaturen der Methoden ihrer identifizierten Klassen (inkl. der Klassen `SuchAuftrag` und `SuchErgebnis`) exakt mit Hilfe der Notationsmöglichkeiten durch die UML. Modellieren sie auch die wichtigsten Attribute (Instanzvariablen). Etwaige Datentransformationen sollten sie als private Methoden exakt mit Eingabe- und Rückgabeparameter modellieren und somit die Transformation hierüber *andeuten*. Die verwendeten Klassen der Schnittstelle des Reise-

Anbieters (`QueryObject` und `QueryResult`) brauchen sie *nicht* zu berücksichtigen. Bei der Bezeichnung der Methoden in dieser Lösung können sie eigene Annahmen machen (Ausnahme: die Methode des Reiseanbieters).

b.)
Entwickeln sie einen beispielhaften Java-Code für die Klasse, welche die notwendigen Datentransformationen durchführt. Auch hier sollten durch private Methoden die Transformationen nur angedeutet werden, eine konkrete Implementierung der Methoden kann ausbleiben.

c.)
Entwickeln sie zu dem Klassendiagramm aus der Aufgabe a) *ein* Sequenz-Diagramm, welches *eine* mögliche Ausprägung der *synchronen* Interaktionen zwischen den beteiligten Objekten zur Laufzeit darstellt. Modellieren sie somit ein konkretes Szenario. Die Erzeugung der notwendigen Objekte vom Typ `SuchAuftrag` bzw. `SuchErgebnis` sollten sie explizit modellieren. Die Objekte aus der Schnittstelle des Reiseanbieters brauchen nicht explizit erzeugt zu werden; sie sollten diese aber Rückgabe-Werte der privaten Methoden zur Datentransformationen andeuten (z.B. `return q : QueryObject`).

Aufgabe 2 (Modellbasierte Refaktorisierung des Verwaltungstool, 10 Punkte)

In der Aufgabe 4-2 haben sie eine erste Version des eines Tools zur Verwaltung von Mitarbeitern entwickelt. Es soll nun die Aufgabe in dieser Woche sein, ihr Tool anhand der neu gewonnenen Kenntnisse aus dem Kapitel 5 zu refaktorisieren (engl.: to *refactor*). Die Refaktorisierung sollte durch ein UML-Paket-Diagramm angedeutet werden, eine Implementierung ist *nicht* notwendig.

a)
Modellieren sie die Software-Architektur anhand eines UML-basierten Paketdiagramms. Orientieren sie sich dabei an die Darstellung aus dem Kapitel 5, Folie 36. Modellieren sie auch die Abhängigkeiten zwischen ihren Paketen. Fügen sie ihre Pakete sinnvoll in eine Schicht-Architektur gemäß folgender Schicht-Aufteilung („Layer Architecture“, vgl. Kapitel 5, Folie 44) ein: UI-Layer, AL-Layer und IN-Schicht (Infrastruktur). Strukturieren sie ihr Software-System zudem anhand der Vorgaben des MVC-Musters (Kapitel 5)! Definieren sie dazu je *ein* Package für folgende Klassen in ihrem System und fügen sie diese sinnvoll in die oben genannten drei Schichten ein:

- Model-Klassen (zentrale `Container`-Klassen zur Verwaltung von Entities sowie die Klasse(n) zur Abspeicherung bzw. zum Laden von Entities)
- Entity-Klassen (langlebige Objekte; hier: die `Employee`)
- View-Klassen (Klassen für die Ausgabe)
- Controller-Klassen (Klasse(n) zum Starten des Programms zur Entgegennahme von Befehlen (z.B. load, exit) aus der Console sowie zum Ausführen von Befehlen)
- Exception-Klassen
- Util-Klassen (für zukünftige Hilfsmethoden und –Algorithmen)