Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences

Vorlesung Objektorientierte
Komponenten-Architekturen
(OOKA)
SS 2022

Prof. Dr. Sascha Alda

2. Mai 2022

Übungen zur Vorlesung
Objektorientierte Komponenten-Architekturen

SS 2022

Übungsblatt Nr. 3
(Abgabe bis: Montag, den 9.5.2022, **12:00 Uhr**)

Aufgabe 1 (Modellierung der Laufzeitumgebung mit dem 4-Sichten-Modell, Integration eines CRs):

In den letzten Wochen haben sie ein Komponentenmodell sowie eine Laufzeitumgebung für das Deployment von komponentenbasierten Applikationen entwickelt. Es soll nun ihre Aufgabe sein, diese Laufzeitumgebung anhand des 4-Sichten Modells nach Starke (vgl. Kapitel 3, Abschnitt 2) zu (*re*-)modellieren. Fokussieren sie sich dabei auf die Bausteinsicht des Systems. Diese sollte durch Komponentendiagramme (Kapitel 3, Abschnitt 2) sowie, ergänzend, durch Klassendiagramme modelliert werden. Bitte dazu auch noch mal die dazu empfohlene Quelle (Rupp, 2012) beachten!

Für die Laufzeitumgebung selbst sind noch zwei Change Requests (*CRs*) dazugekommen, die es gilt zu beachten:

CR1: Die aktuelle Konfiguration einer Laufzeitumgebung (welche Komponenten wurden in welcher Reihenfolge eingesetzt) soll stets persistent auf einem externen Server abgespeichert werden, so dass bei einem Absturz der lokalen Laufzeitumgebung der Zustand (= die Konfiguration) der Laufzeitumgebung wiederhergestellt werden kann. Der Zustand *innerhalb einer* Komponente braucht *nicht* gespeichert zu werden.

CR2: Erweitern sie ihre Laufzeitumgebung um einen horizontalen Logging-Dienst, der von ihrer Laufzeitumgebung für alle kompatiblen Komponenten angeboten werden soll. Möchte eine Komponente diesen Dienst verwenden, so muss sie eine typ-konforme Variable in der Start-Class (die Klasse, wo die @Start-Methode vorhanden ist) definieren und diese mit einer neuen, selber zu implementierenden Annotation versehen:

```
import ihrLZUpackage.Logger;
import ihrLZUpackage.Inject;

@Inject
private Logger mylog; // Logger kann ein Interface oder eine Klasse
sein
```

```
// Eine entsprechende setter-Methode kann implementiert bzw. durch
das Komponentenmodell vorgegeben werden.
```

Mit dieser Annotation signalisieren sie der Laufzeitumgebung, dass zur Laufzeit „bitte“ eine konkrete Referenz von Interface-Typ Logger *injiziert* werden soll. In welchem Zustand bzw. bei welchem Zustandsübergang wäre diese Injection notwendig bzw. sinnvoll? Den genauen Typ der Annotation sollte sie natürlich über eine eigene Package-Bezeichnung klar definieren (vgl. obiges Beispiel).

Eine beliebige Methode aus einer Komponente kann dann über die Methode:

```
+ sendLog( str : String ) : void
```

diesen Logging-Dienst wie folgt verwenden:

```
public String foo() {
      ...
      mylog.sendLog( "Meldung aus Component: Prozess gestartet" );
      ....
}
```

Die konkrete Implementierung sollte dann auf einer Console folgendes ausgeben:

```
++++ LOG: Meldung aus Component: Prozess gestartet ([TimeStamp])
```

Andere Klassen in einer Komponente (Regular Class) können diesen Dienst über eine Factory verwenden:

```
import ihrLZUpackage.LoggerFactory;
import ihrLZUpackage.Logger;

public String foo() {
      ...
      Logger myLog = LoggerFactory.createLogger();
      myLog.sendLog( "Meldung aus Component: Prozess gestartet" );
      ....
```

Diese CRs müssen sie bei der Modellierung der Bausteinsicht für ihre Laufzeitumgebung mitberücksichtigen. Des Weiteren soll es ihre Aufgabe sein, diese CRs in ihre Laufzeitumgebung als Erweiterung zu integrieren. Eine Auslagerung der relevanten Konfigurationsdaten auf einen externen Server ist nicht notwendig, das Abspeichern kann in einem lokalen File oder unter Verwendung eines lokalen Datenbanksystems prototypisch erfolgen. Im 4-Sichten-Modell sollen sie hingegen eine externe Datenbank berücksichtigen und entsprechend modellieren.