Team: TEAM 01, Falco Winkler (FW), Daniel Schruhl (DS)

Aufgabenteilung:

- IDL Compiler
- Namensdienst
- mware_lib Library

Quellenangaben:

- Aufgabe 4, 11.06.2017, C. Klauck & H. Schulz:
  http://users.informatik.haw-hamburg.de/ schulz/pub/Verteilte-Systeme/AI5-VSP/Aufgabe4/

Bearbeitungszeitraum:

- 11.06.2017 4 Stunden (DS)

Aktueller Stand:

- IDL Compiler begonnen
- Namensdienst
- mware_lib Library

Änderung des Entwurfs:

- Keine Änderungen

1 Einführung und Ziele

Es soll eine einfache objektorientierte Middleware entworfen werden, die Methodenaufrufe eines entfernten Objektes ermöglicht.
Zur Orientierung gilt hierbei die CORBA Architektur. Genauer soll hier ein ORB zur Verfügung gestellt werden, der es ermöglicht Methoden von entfernten Objekten aufzurufen.
Zur Abstraktion und Beschreibung der Schnittstellen der Objekte soll eine IDL verwendet werden. Diese IDL wird dann zur Erzeugung von Klassen- und Methodenrümpfen verwendet.
Außerdem beinhaltet der ORB einen Namensdienst, der Objektreferenzen in einem Netz mit Namen finden kann.
Die Middleware an sich soll durch eine Library abstrahiert und verwendbar sein.

1.1 Randbedingungen

Der Namensdienst soll auf einem entfernten Rechner unabhängig von der Middleware Library lauffähig sein. Der Port muss zur Laufzeit einstellbar sein.
Der IDL-Compiler soll in einem Package oder einer `.jar` Datei zur Verfügung gestellt werden. Der Compiler soll folgende IDL Typen unterstützen:

- module (keine Schachtelung, 1 Modul pro Datei)
- class (nicht als Parameter oder Returnwert, keine Schachtelung)
- int
- double
- string

Ein Beispiel:

```
module math_ops {
  class Calculator {
   double add(double a, double b);
   string getStr(double a);
 };
};
```

Die Middleware Library soll in einem Package `mware_lib` zusammengefasst werden.
Wenn eine Serverapplikation während eines entfernten Methodenaufrufes eine RuntimeException wirft, soll diese an den Aufrufer weitergeleitet werden.
Es soll möglich sein, dass zwei oder mehrere Klienten die selbe Objektreferenz zeitgleich nutzen wollen. Das soll innerhalb der Middleware nicht zu Deadlocks führen.

1.2 Kontextbegrenzung

Die Implementierung soll in Java vorliegen.
Die Behebung von Deadlocks in den Anwendungen ist nicht Aufgabe der Middleware.

2 Gesamtsystem

2.1 Bausteinsicht

2.2 Laufzeitsicht

3 Subsysteme und Komponenten