Hinweise für die Übungsaufgaben zu Middleware – Cloud Computing

- Jeder Teilnehmer erhält ein Projektverzeichnis unter

  `/proj/i4mw/<loginname>`

  Dieses kann als Workspace für die Übungsaufgabe dienen.

- Die Lösungen sollen in Gruppen mit bis zu 3 Teilnehmern erstellt werden. Jede Gruppe bekommt ein eigenes Projekt-Repository, auf das unter folgender Adresse zugegriffen werden kann:

  `https://www4.cs.fau.de:8088/i4mw/<gruppe>`

- Zur Erstellung des SVN-Repository muss von einem Übungspartner aus der Gruppe das Programm

  `/proj/i4mw/bin/mwgroups`

  aufgerufen werden. Die Zugangsdaten für das Repository werden daraufhin innerhalb von 48 Stunden per E-Mail mitgeteilt.

- Zum Abgabezeitpunkt sollte die finale Version der Lösung in das Repository eingecheckt sein. Die eigentliche **Abgabe der Aufgaben erfolgt durch Präsentation** der eigenen Lösung gegenüber einem Übungsleiter.

- Sollte eine Präsentation der eigenen Implementierung am Abgabetermin nicht möglich sein, ist dies **im Voraus** entweder per E-Mail (siehe unten) oder persönlich einem Übungsleiter mitzuteilen.

- Hilfestellungen und Vorgaben zu den Übungsaufgaben finden sich im *Pub-Verzeichnis* unter

  `/proj/i4mw/pub/<aufgabe>`

- Die *Java Coding Guidelines* sollten befolgt werden:
  - Klassennamen beginnen mit einem Großbuchstaben
  - Methodennamen beginnen mit einem Kleinbuchstaben
  - Variablennamen beginnen mit einem Kleinbuchstaben
  - Konstante (`static final`) Variablen bestehen nur aus Großbuchstaben
  - Namen von Variablen beschreiben deren Zweck

- Wenn eine bestimmte Programmstruktur in der Aufgabenstellung verlangt wird, muss die Lösung diese einhalten. Die Lösung soll insbesondere
  - Namen von Klassen und Packages
  - Sichtbarkeit von Variablen und Methoden
  - Namen, Parametertypen und Rückgabewerte von Methoden

  wie in der Aufgabenstellung beschrieben verwenden.

- Der Code muss lesbar und nachvollziehbar sein. Falls komplizierte Teile vorkommen, sollten diese mit Kommentaren erläutert werden.

- Die Lösungen müssen eigenständig erstellt worden sein. Verstöße werden geahndet.

Bei Problemen mit der Aufgabenstellung könnt ihr euch jederzeit an uns wenden:

`mw@i4.informatik.uni-erlangen.de`

1 Übungsaufgabe #1: Web-Services

In der ersten Aufgabe sollen Freundschaftsbeziehungen in einem sozialen Netzwerk analysiert und das Ergebnis per Web-Service verfügbar gemacht werden. Von jedem Nutzer des Netzwerks sind dabei folgende Daten bekannt:

- *ID*: eine Zeichenkette, die den Nutzer eindeutig kennzeichnet
- *Name*: eine Zeichenkette mit dem (Klar-)Namen des Nutzers
- *Freunde*: eine Liste mit den *ID*s von Freunden des Nutzers

Bereitgestellt werden diese Daten von einem bereits vorhandenen Web-Service („`MWFacebookService`“), der unter anderem folgende Methoden anbietet:

```
public interface MWFacebookServiceInterface {
    public String[] searchIDs(String name);
    public String getName(String id) throws MWUnknownIDException;
    public String[] getFriends(String id) throws MWUnknownIDException;
}
```

Die Methode `searchIDs()` liefert die IDs von allen Nutzern, deren Namen die Zeichenkette `name` enthält. Mit `getName()` lässt sich der Name zu einer ID erfragen; existiert kein Nutzer mit der übergebenen ID, wird eine `MWUnknownIDException` geworfen. Ein Aufruf von `getFriends()` stellt die IDs der Freunde eines Nutzers bereit.

1.1 Client

In der ersten Teilaufgabe soll ein Web-Service-Client implementiert werden, der es ermöglicht, den vorhandenen Web-Service zu nutzen. Hierfür ist es zunächst erforderlich die auf der Client-Seite zur Kommunikation mit dem entfernten Web-Service benötigten Klassen zu erzeugen. Dies erfolgt unter Zuhilfenahme der vom Web-Service bei einer Registry veröffentlichten WSDL-Beschreibung des Diensts.

1.1.1 Registry-Zugriff (für alle)

Die im Rahmen dieser Aufgabe genutzte Registry existiert bereits, ihre aktuelle Adresse wird unter `/proj/i4mw/pub/aufgabe1/ws-registry.adr` bekannt gegeben. Jegliche Interaktion mit dieser Registry soll in einer Klasse `MWRegistryAccess` (im Package `mw`) gekapselt werden, die zunächst mindestens folgende Methoden umfasst:

```
public class MWRegistryAccess {
    public void openConnection(String queryManagerURL, String lifeCycleManagerURL);
    public void closeConnection();
    public void listWSDLs(String serviceName);
}
```

Ein Aufruf von `openConnection()` öffnet eine Verbindung zur Registry unter Verwendung der übergebenen Query-Manager- und Life-Cycle-Manager-URLs, `closeConnection()` schließt diese Verbindung wieder. Die Methode `listWSDLs()` gibt die URLs aller für einen Dienst `serviceName` registrierten WSDL-Beschreibungen sowie die Namen der den Dienst anbietenden Organisationen aus, zum Beispiel durch einen Kommandozeilenaufruf „`java -cp <classpath> mw.MWRegistryAccess LIST MWFacebookService`“.

Aufgabe:

→ Implementierung der Klasse `MWRegistryAccess`

Hinweise:

- Die URL der WSDL ist in der *AccessURI* eines Service-Bindings des Diensts abgelegt.
- Die zur Kommunikation mit der Registry benötigten Bibliotheken liegen in `/proj/i4mw/pub/aufgabe1/`.
- Um korrekt auf die Registry zugreifen zu können, muss `MWRegistryAccess` „`en_US`“ als *Locale* verwenden.

1.1.2 Web-Service-Client (für alle)

Unter Verwendung des `MWRegistryAccess`-Tools lässt sich nun die URL der WSDL-Beschreibung für den `MWFacebookService` ermitteln. Im Anschluss daran können mittels `wsimport` die zur Kommunikation mit dem Dienst benötigten Klassen in einem Subpackage `mw.facebookclient` generiert werden. Mit ihrer Hilfe ist ein Web-Service-Client `MWClient` zu implementieren, der zunächst mindestens folgende Methoden anbietet:

```
public class MWClient {
    public void searchIDs(String name);
    public void getFriends(String id);
}
```

Beide Methoden sollen sich über die Kommandozeile aufrufen lassen (vgl. 1.1.1) und auf die gleichnamigen Methoden des `MWFacebookService` zurückgreifen. Zusätzlich zu den vom Web-Service zurückgegeben IDs sind in beiden Fällen auch die zu den IDs gehörigen Namen (→ `getName()`) auf dem Bildschirm auszugeben.

Aufgaben:

→ Generierung der für die Nutzung des `MWFacebookService` benötigten Klassen mittels `wsimport`
→ Implementierung der Klasse `MWClient`

1.2 Web-Service

In dieser Teilaufgabe soll ein eigener Web-Service `MWPathService` entwickelt werden, der basierend auf den `MWFacebookService`-Daten die kürzeste Verbindung zwischen zwei Nutzern des sozialen Netzwerks ermittelt; z. B. „Woher ‚kennen' sich A und Z?" → „A ist befreundet mit B, B ist befreundet mit C,..., Y ist befreundet mit Z." Zur Berechnung greift der `MWPathService` auf den `MWFacebookService` als Datenbasis zurück.

1.2.1 Pfad-Dienst (für alle)

Der zu erstellende `MWPathService` bietet die Methode `calculatePath()` an, die die kürzeste Verbindung zwischen den durch `startID` und `endID` eindeutig gekennzeichneten Nutzern ermittelt und in Form einer ID-Liste zurückgibt. Kann keine Verbindung ermittelt werden (z. B. weil eine der übergebenen IDs ungültig ist), wird eine `MWNoPathException` geworfen.

```
public interface MWPathServiceInterface {
    public String[] calculatePath(String startID, String endID) throws MWNoPathException;
}
```

Implementiert werden soll der `MWPathService` in einer Klasse `MWPathServiceImpl` im Subpackage `mw.path`. Dabei sollen JAX-WS-Annotationen zum Einsatz kommen (siehe Tafelübung). Als Hilfestellung liegt im Pub-Verzeichnis die Klasse `MWDijkstra` bereit, die den Dijkstra-Algorithmus über folgende statische Methode anbietet:

```
public class MWDijkstra {
    public static String[] getShortestPath(MWFacebookServiceInterface facebook,
                                           Collection<String> ids, String startID, String endID);
}
```

Im Parameter `ids` erwartet `getShortestPath()` eine Menge von Nutzer-IDs, die bei der Berechnung berücksichtigt werden soll; je weniger IDs sie enthält, desto schneller terminiert der Algorithmus. Vor dem Aufruf von `getShortestPath()` ist es also notwendig, eine Vorverarbeitung durchzuführen, um `ids` zusammenzustellen. Hierbei soll folgendermaßen vorgegangen werden (Pseudo-Code):

```
for(int i = 1; true; i++) {
    StartFreundeskreis := alle Nutzer, die von startID in i Schritten erreichbar sind;
    EndFreundeskreis   := alle Nutzer, die von endID   in i Schritten erreichbar sind;
    if(StartFreundeskreis ueberschneidet sich mit EndFreundeskreis) {
        ids := StartFreundeskreis vereinigt mit EndFreundeskreis;
        break;
    }
}
```

Ausgehend von `startID` und `endID` werden (unter Verwendung der Methode `getFriends()` des `MWFacebook-Service`) die erweiterten Freundeskreise der beiden Nutzer so lange schrittweise vergrößert, bis sie sich überschneiden. Ist dies der Fall, soll `ids` die Vereinigung der beiden erweiterten Freundeskreise bilden; im Regelfall handelt es sich hierbei um weniger als 10.000 Knoten.

Aufgabe:

→ Implementierung der Klasse `MWPathServiceImpl`

Hinweise:

- Der Dijkstra-Algorithmus funktioniert nur korrekt, wenn `ids` **keine Duplikate** enthält.
- Die Aufbereitung der Daten kann bei weiter auseinander liegenden IDs (etwa ab einer Distanz von mehr als sieben Schritten) ein paar Minuten dauern. Zum Testen sollten daher zunächst Start- und End-IDs gewählt werden, von denen (z. B. durch Ausprobieren) bekannt ist, dass sie näher beieinander liegen.

1.2.2 Erweiterungen: Registry-Zugriff und Web-Service-Client (für alle)

Zur Veröffentlichung des `MWPathService` in der Registry soll erneut das `MWRegistryAccess`-Tool aus Aufgabe 1.1.1 zum Einsatz kommen. Hierzu muss es um mindestens diese beiden Methoden erweitert werden:

```
public void authenticate(String userName, String password);
public void registerService(String orgName, String serviceName, String wsdlURL);
```

Ein Aufruf von `authenticate()` setzt Credentials für die Verbindung zur Registry. Mittels `registerService()` wird die URL zur WSDL-Beschreibung eines Diensts `serviceName` registriert, der von einer Organization `orgName` angeboten wird. Der eigene Pfad-Dienst kann somit nun in die Registry eingetragen werden.

Als letztes ist der Web-Service-Client aus Aufgabe 1.1.2 so zu erweitern, dass er auch den Zugriff auf die `calculatePath()`-Methode des `MWPathService` anbietet. Hierzu sind (analog zu 1.1.2) zunächst die benötigten Klassen für den Zugriff auf den Pfad-Dienst in einem Subpackage `mw.pathclient` zu generieren. Anschließend ist der `MWClient` so zu erweitern, dass er die IDs sowie die dazugehörigen Namen des berechneten Pfads ausgibt.

Aufgaben:

→ Erweiterung der Klasse `MWRegistryAccess`
→ Registrierung des eigenen `MWPathService`
→ Generierung der für die Nutzung des Pfad-Diensts benötigten Klassen mittels `wsimport`
→ Erweiterung der Klasse `MWClient`
→ Was ist die kürzeste Verbindung zwischen den Nutzern mit den IDs 1694452301 und 100000859170147?

Hinweise:

- Als Nutzername für die Registry sowie als Organisationsname für den `MWPathService` ist jeweils der eigene Gruppenname (z. B. „*gruppe0*“) zu verwenden. Das Passwort ist eine leere Zeichenkette.
- Jede Übungsgruppe soll nur *eine* Organisation in der Registry eintragen, die auch nur *eine einzige* Referenz auf ihren `MWPathService` anbietet. Um dies sicherzustellen, kann die Methode `registerService()` zum Beispiel so implementiert werden, dass sie a) die Organisation nur anlegt, falls diese noch nicht existiert und b) zunächst ein eventuell bestehender Service-Eintrag gelöscht, bevor der neue Eintrag registriert wird.

1.2.3 Verbesserter Pfad-Dienst (optional für 5,0 ECTS)

Die aktuelle Implementierung benötigt eine gewisse Zeit, um die kürzeste Verbindung zwischen den Nutzern mit den IDs 1694452301 und 100000859170147 zu berechnen. Um den Pfad-Dienst aus Aufgabe 1.2.1 verbessern zu können, ist es zunächst erforderlich, sich klar zu machen, welcher der Ausführungsschritte (Senden der Anfrage an den Pfad-Dienst, Zusammenstellung der zu berücksichtigenden IDs, Ausführung des Dijkstra-Algorithmus, Zurücksenden der Antwort) jeweils wie viel Zeit in Anspruch nimmt. Von besonderem Interesse ist hierbei auch die Anzahl der Fernaufrufe am `MWFacebookService`, die eine einzige Ausführung von `calculatePath()` nach sich zieht, da jeder dieser Aufrufe zusätzliche Kommunikation mit dem externen Dienst bedeutet.

Um die Anzahl der zu sendenden und zu empfangenden Nachrichten zu reduzieren, bieten viele Web-Service-APIs die Möglichkeit mehrere Aufrufe in einer einzigen Anfrage zu bündeln („Batching“). Der `MWFacebookService` verfügt hierzu über eine Methode `getFriendsBatch()`, die für jede im Parameter `ids` enthaltene ID lokal die Methode `getFriends()` aufruft und die Ergebnisse gesammelt in einem zweidimensionalen Array (in `result[0]` steht das Ergebnis zum `getFriends()`-Aufruf von `ids[0]` usw.) zurückgibt.

```
public String[][] getFriendsBatch(String[] ids) throws MWUnknownIDException;
```

Mit Hilfe dieser Methode lässt sich für den Ausführungsschritt der Zusammenstellung der IDs für den Dijsktra-Algorithmus die Anzahl der Fernaufrufe am `MWFacebookService` drastisch reduzieren, zum Beispiel indem alle `getFriends()`-Aufrufe für den `StartFreundeskreis` im Schritt i (siehe 1.2.1, `EndFreundeskreis` analog) gebündelt werden. Dies führt in der Konsequenz zu einer erheblichen Effizienzsteigerung des `MWPathService`.

Aufgaben:

→ Analyse der `calculatePath()`-Ausführung im aktuellen Pfad-Dienst
→ Reimplementierung des Pfad-Diensts unter Verwendung von `getFriendsBatch()` (Ursprüngliche Implementierung aus Aufgabe 1.2.1 bitte aufheben!)
→ Analyse der `calculatePath()`-Ausführung im verbesserten Pfad-Dienst (Wie groß ist der Zeitgewinn?)

Hinweis:

- Eine Reimplementierung der `MWDijkstra.getShortestPath()`-Methode ist nicht erforderlich.

Abgabe: am 4.11.2015 in der Rechnerübung