Team: <08>, <(Porep, Lars), (Niebsch, Oliver)>

Aufgabenaufteilung:

1. Aufgabe 3, komplett

Quellenangaben:

Begründung für Codeübernahme:

Bearbeitungszeitraum:
(Gemeinsame Bearbeitungszeit)
03.06.2015 - 6 Stunden
04.06.2015 - 1 Stunde
09.06.2015 - 10 Stunden

Aktueller Stand: Alles ist implementiert und getestet.

Skizze: *siehe nächste Seite*

Entwurf:

Es soll eine Middleware in Java entwickelt werden, über die entfernte Methodenaufrufe auf Objekten ermöglich wird. Der Client soll dabei nicht merken, dass sich das Objekt auf einem entfernten Server befindet.

Zu entwickeln sind folgende Komponenten:

- Middleware: Instanzen der Middleware werden unabhängig voneinander jeweils auf den Clients und den Servern verwendet, indem sie als Komponente eingebunden werden.
- Namensservice: Global agierender Dienst für die Namensauflösung von Objekt-IDs zu Position (IP, Port) des Rechners im Netzwerk, der dieses Objekt zur Verfügung stellt.

Der Aufbau der Objekte, die entfernt zur Verfügung gestellt werden sollen, sind in den accessor_* Packages vorgegeben.

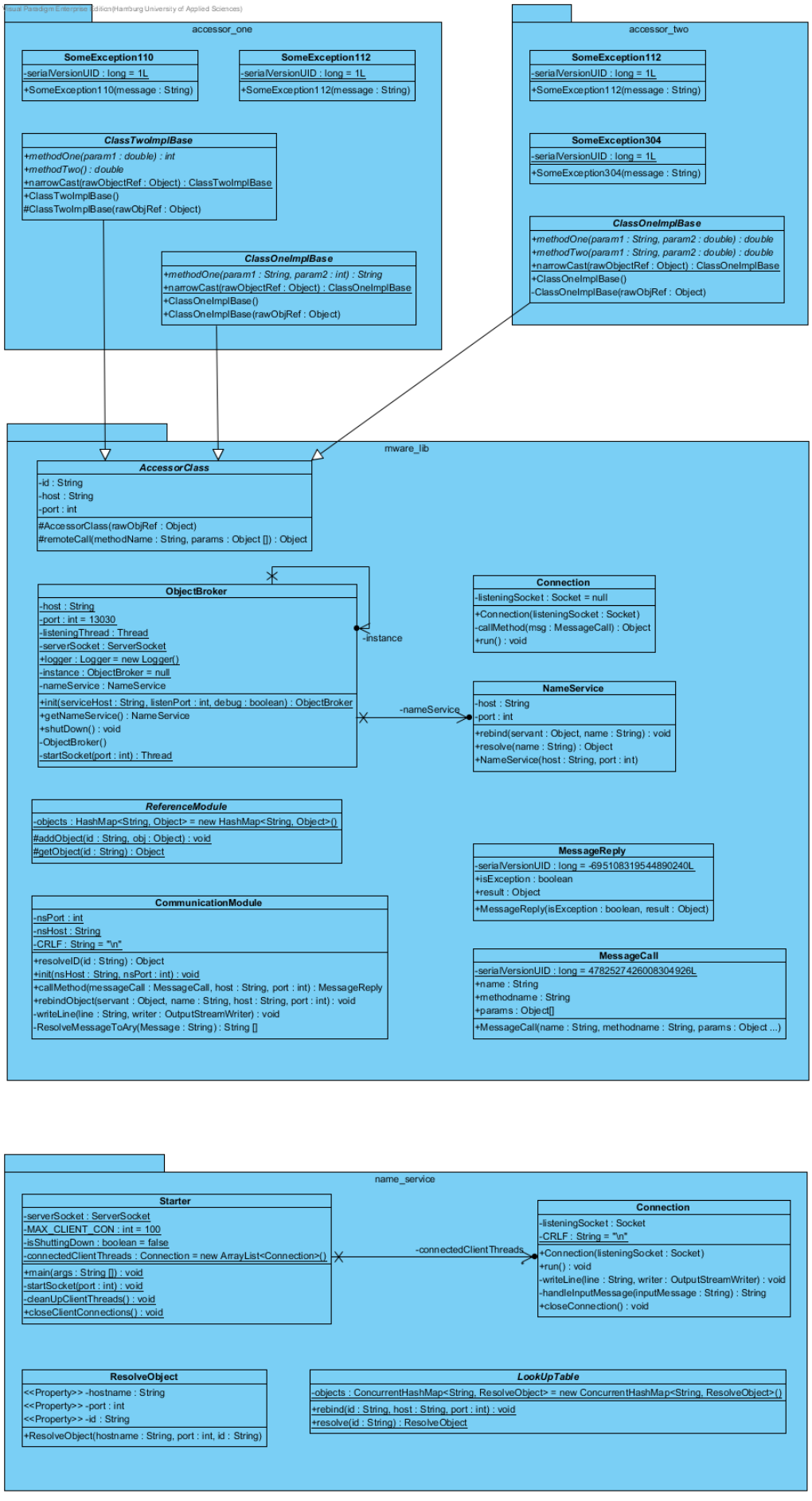
Der interne Aufbau von Middleware und Nameservice erfolgt gemäß dem Klassendiagramm auf der nächsten Seite und orientiert sich an der CORBA-Architektur.

Das mware-lib Package besteht aus den Komponenten:

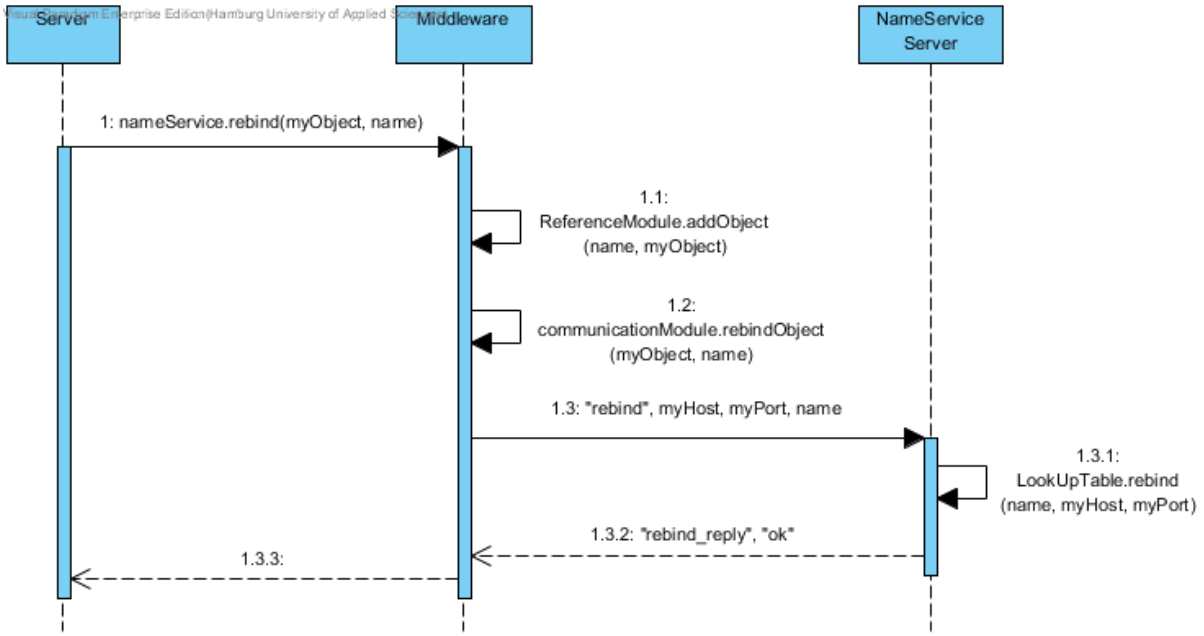
- Accessor-Class: die Oberklassen für alle Stub-Objekte der Clients
- CommunicationModule: übernimmt die Kommunikation per TCP
- ReferenceModule: interne LookUp-Tabelle zum Abspeichern von Objekt-Referenzen zu IDs mit denen diese beim Namensserver angemeldet wurden
- ObjectBroker: vordefinierte Schnittstelle der Middleware für Client und Server-Applikationen
- NameService: Repräsentant des Namensservers zur Verwendung für Client und Server
- Connection: Empfangsmodul für TCP-Verbindungen anderer Middlewares.
- MessageCall & MessageReply: Objekte, die zur Kommunikation von Middleware zu Middleware verschickt werden

Der name_service (der eigentliche NamensserviceServer) besteht intern aus:

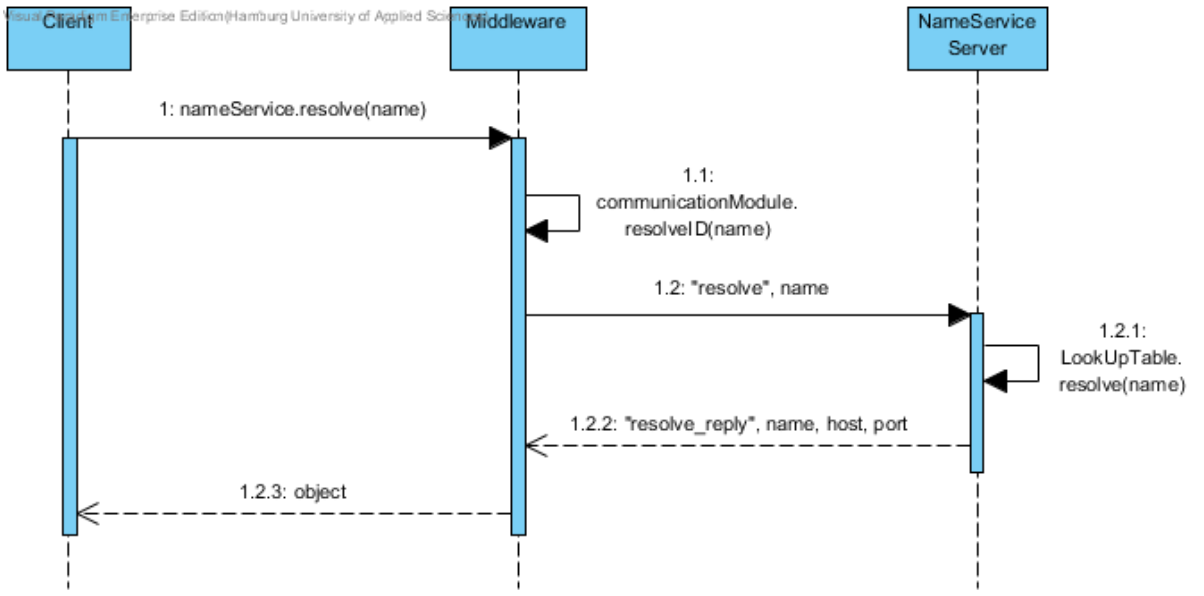
- Starter: Startet den Server auf einem, als Parameter übergebenen, Port
- Connection: Thread, der die konkrete Kommunikation mit dem CommunicationModule übernimmt, sobald eine Anfrage kommt. So kann der NameServiceServer mehrere Anfragen gleichzeitig beantworten bzw. annehmen
- ResolveObject: Datencontainer zur Speicherung von angemeldeten Objekten
- LookUpTable: Struktur zur internen Speicherung von ResolveObjects


Visual Paradigm Enterprise Edition(Hamburg University of Applied Sciences)
accessor_one
SomeException110
-serialVersionUID : long = 1L
+SomeException110(message : String)
SomeException112
-serialVersionUID : long = 1L
+SomeException112(message : String)
ClassTwoImplBase
+methodOne(param1 : double) : int
+methodTwo() : double
+narrowCast(rawObjectRef : Object) : ClassTwoImplBase
+ClassTwoImplBase()
#ClassTwoImplBase(rawObjRef : Object)
ClassOneImplBase
+methodOne(param1 : String, param2 : int) : String
+narrowCast(rawObjectRef : Object) : ClassOneImplBase
+ClassOneImplBase()
+ClassOneImplBase(rawObjRef : Object)
accessor_two
SomeException112
-serialVersionUID : long = 1L
+SomeException112(message : String)
SomeException304
-serialVersionUID : long = 1L
+SomeException304(message : String)
ClassOneImplBase
+methodOne(param1 : String, param2 : double) : double
+methodTwo(param1 : String, param2 : double) : double
+narrowCast(rawObjectRef : Object) : ClassOneImplBase
+ClassOneImplBase()
-ClassOneImplBase(rawObjRef : Object)
mware_lib
AccessorClass
-id : String
-host : String
-port : int
#AccessorClass(rawObjRef : Object)
#remoteCall(methodName : String, params : Object []) : Object
ObjectBroker
-host : String
-port : int = 13030
-listeningThread : Thread
-serverSocket : ServerSocket
+logger : Logger = new Logger()
-instance : ObjectBroker = null
-nameService : NameService
+init(serviceHost : String, listenPort : int, debug : boolean) : ObjectBroker
+getNameService() : NameService
+shutDown() : void
-ObjectBroker()
-startSocket(port : int) : Thread
-instance
Connection
-listeningSocket : Socket = null
+Connection(listeningSocket : Socket)
-callMethod(msg : MessageCall) : Object
+run() : void
-nameService
NameService
-host : String
-port : int
+rebind(servant : Object, name : String) : void
+resolve(name : String) : Object
+NameService(host : String, port : int)
ReferenceModule
-objects : HashMap<String, Object> = new HashMap<String, Object>()
#addObject(id : String, obj : Object) : void
#getObject(id : String) : Object
MessageReply
-serialVersionUID : long = -6951083195448902401
+isException : boolean
+result : Object
+MessageReply(isException : boolean, result : Object)
CommunicationModule
-nsPort : int
-nsHost : String
-CRLF : String = "\n"
+resolveID(id : String) : Object
+init(nsHost : String, nsPort : int) : void
+callMethod(messageCall : MessageCall, host : String, port : int) : MessageReply
+rebindObject(servant : Object, name : String, host : String, port : int) : void
-writeLine(line : String, writer : OutputStreamWriter) : void
-ResolveMessageToAry(Message : String) : String []
MessageCall
-serialVersionUID : long = 4782527426008304926L
+name : String
+methodname : String
+params : Object[]
+MessageCall(name : String, methodname : String, params : Object ...)
name_service
Starter
-serverSocket : ServerSocket
-MAX_CLIENT_CON : int = 100
-isShuttingDown : boolean = false
-connectedClientThreads : Connection = new ArrayList<Connection>()
+main(args : String []) : void
-startSocket(port : int) : void
-cleanUpClientThreads() : void
+closeClientConnections() : void
-connectedClientThreads
Connection
-listeningSocket : Socket
-CRLF : String = "\n"
+Connection(listeningSocket : Socket)
+run() : void
-writeLine(line : String, writer : OutputStreamWriter) : void
-handleInputMessage(inputMessage : String) : String
+closeConnection() : void
ResolveObject
<<Property>> -hostname : String
<<Property>> -port : int
<<Property>> -id : String
+ResolveObject(hostname : String, port : int, id : String)
LookUpTable
-objects : ConcurrentHashMap<String, ResolveObject> = new ConcurrentHashMap<String, ResolveObject>()
+rebind(id : String, host : String, port : int) : void
+resolve(id : String) : ResolveObject

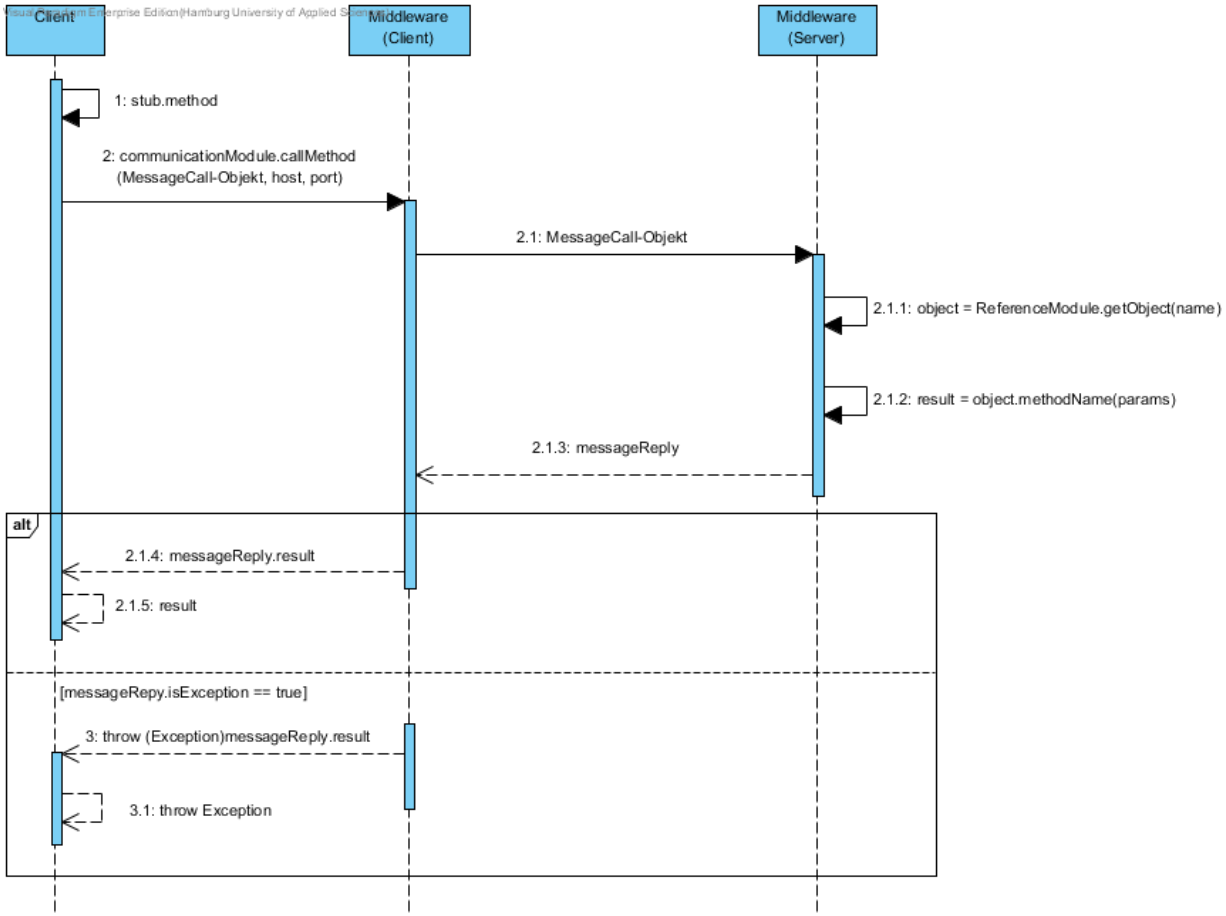
Das Bereitstellen (rebind) eines Objektes von einem Server beim Namensservice geschieht nach folgendem Ablauf:



Das Auflösen (resolve) einer ObjektID in ein Stellvertreterobjekt für den Client geschieht mit Hilfe des Namensservice nach folgendem Ablauf:



Das Aufrufen einer Methode auf einem entfernten Objekt geschieht nach folgendem Ablauf:



Hinweise:

- Alle Verbindungen zwischen den Kommunikationsmodulen untereinander, sowie mit dem Nameserver erfolgen über TCP. Anfragen sind erst nach aufgebauter Verbindung (Handshake) möglich.
- Nachrichten von und zum NameServer sind Komma separierte Strings, wobei der erste Wert den Typ der Nachricht angibt. Das Ende einer Nachricht wird mit einem \n markiert.
- Nachrichten zwischen Middlewares sind Instanzen der Klassen „MessageCall“ und „MessageReply“. Um das zu realisieren, sind ObjectStreams zu verwenden.
- Für nebenläufige Kommunikation und Berechnungen (auf Seite des Servers) sind Threads zu verwenden

Schnittstellen:

Zur Nutzen der Dienste der Middleware gibt es vorgegebene Schnittstellen:

```
public class ObjectBroker { //- Front-End der Middleware -
  public static ObjectBroker init(String serviceHost,
                                  int listenPort, boolean debug) { ... }
    // Das hier zurückgelieferte Objekt soll der zentrale Einstiegspunkt
    // der Middleware aus Applikationssicht sein.
    // Parameter: Host und Port, bei dem die Dienste (hier: Namensdienst)
    // kontaktiert werden sollen. Mit debug sollen Test-
    // ausgaben der Middleware ein- oder ausgeschaltet werden
    // können.

    public NameService getNameService() {...}
      // Liefert den Namensdienst (Stellvetreterobjekt).

    public void shutDown() {...}
      // Beendet die Benutzung der Middleware in dieser Anwendung.
}

public abstract class NameService { //- Schnittstelle zum Namensdienst -
  public abstract void rebind(Object servant, String name);
    // Meldet ein Objekt (servant) beim Namensdienst an.
    // Eine eventuell schon vorhandene Objektreferenz gleichen Namens
    // soll überschrieben werden.

  public abstract Object resolve(String name);
    // Liefert eine generische Objektreferenz zu einem Namen. (vgl. unten)
}
```