**Team**: 05, Eugen Deutsch, Ralf von der Reith

**Aufgabenaufteilung**:

1. Ralf:
2. Eugen:

**Quellenangaben**: -/-

**Bearbeitungszeitraum**:

**Aktueller Stand**:

* Der Entwurf ist fertig

**Änderungen des Entwurfs**: -

# TODO/Anmerkungen

- Method call Abbildung: Application Object obj ist recht merkwürdig benannt.

- Der RequestHandler enthält den NameService (mware\_lib)? Das ist rein von der Lokalität her unschön,

weil ich als Außenstehender von einem Handler nicht erwarten würde, dass dieser noch einen NameService

mit der Map enthält. Andersherum wäre das schon passender.

Vom RequestHandler selbst würde ich erwarten, dass er sich wirklich nur um eingehende Daten und ausgehende

Daten kümmert und nicht um Berechnungen, um die konkreten Anfragen zu erfüllen.

Aus dem Grund finde ich es auch etwas unpassend, dass der RequestHandler Reflection benutzt, um die Methode

aufzurufen. Wenn man es dabei belässt, dass der RequestHandler den NameService enthält, so würde ich ihn

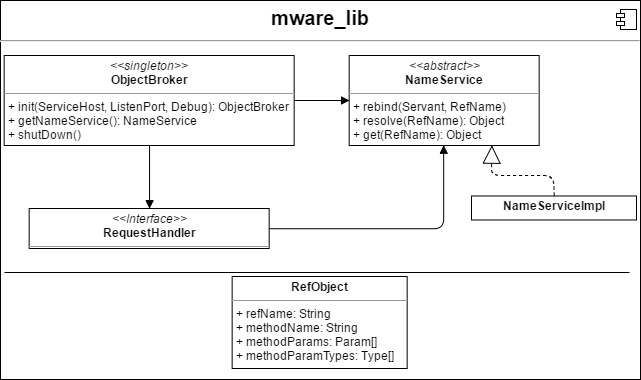
das rmiObj an diesen weiterschicken lassen, woraufhin der NameService dem rmiObj das konkrete Objekt aus seiner Map gibt und das rmiObj regelt dann den Reflection Methodenaufruf. rmiObj impliziert eigentlich schon, dass hier der Methodenaufruf stattfindet.

- NameService hast du mal klein geschrieben (Service), aber bei NameServiceImpl wiederrum Ordnungsgemäß in PascalCase. (Selbes bei refName und refObj heißt es bei rebind, wohingegen es bei resolve objRef heißt)

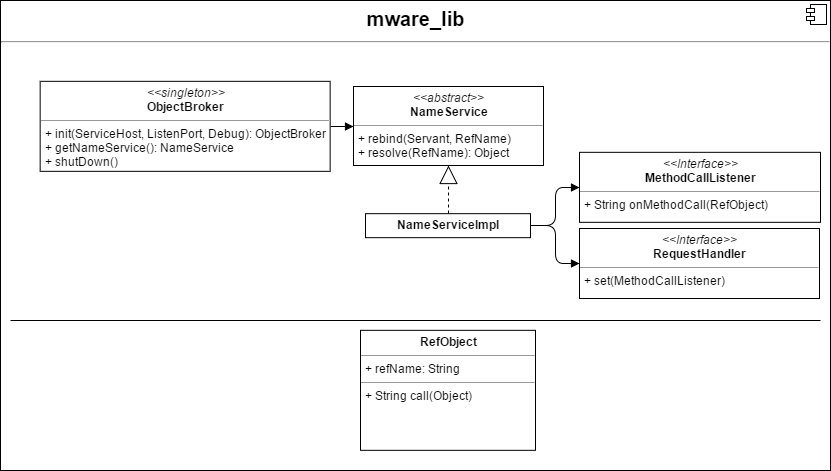
- Laut deiner anderen Notizen (Doku-part-ralf) gibt es einen Listener, der die Request entgegennimmt und daraus jeweils einen RequestHandler

erstellt. Davon ist im Sequenzdiagramm nichts mehr zu sehen. Ist er nun drin, oder nicht? Außerdem steht, er habe eine NameServiceConnection für die Verbindung zum NameServer.

# A



# B



Ablauf

## Rebind

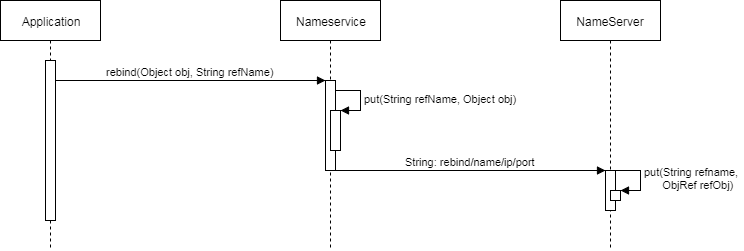


Abbildung 2 Rebind

1. Es wird rebind mit dem zu registrierenden Objekt und einem zugehörigen Namen als Referenz aufgerufen.
2. Der NameService speichert sich das Objekt unter dem Namen und schickt einen String, der wie folgt aufgebaut ist, los: rebind/refName/ip/port
3. Der NameServer erhält die Nachricht und baut den String entsprechend auseinander.
4. Der NameServer speichert sich unter dem refName ein refObj. Das refObj ist ein String in Form von: refName/ip/port.

## Resolve

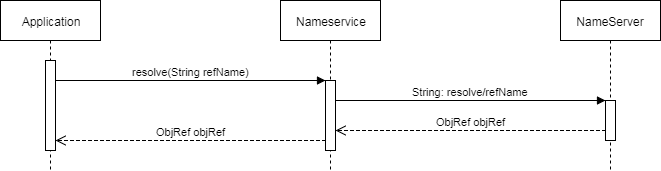


Abbildung 3 Resolve

1. Der Client fragt per resolve mit dem entsprechenden refName nach dem refObj.
2. Der NameService baut eine Anfrage bestehend aus resolve/refName zusammen und schickt diese ab.
3. Der NameServer erhält die Anfrage und schickt das zum refName passende, zuvor gespeicherte refObj zurück.
4. Der NameService erhält das refObj und gibt dieses zurück.

## Method call

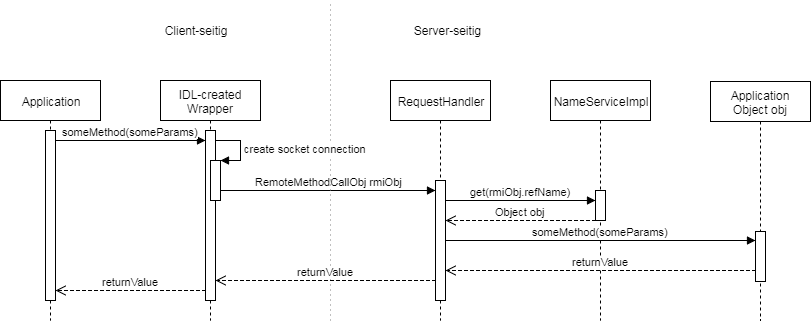


Abbildung 1 Method call

1. Die Client Applikation ruft die Methode eines vom Wrapper erstellten Objektes auf.
2. Das Wrapperobjekt stellt eine Verbindung her und sendet den Aufruf mit den entsprechenden Parametern weiter.
3. Der RequestHandler fängt diese ab und fragt beim NameService nach dem zugehörigen Objekt.
4. Per Java Reflection wird die Methode anhand der vom rmiObj gegebenen Daten (Methodenname, Parameter, Parametertypen) ausgeführt und das Ergebnis wird zurückgeschickt.
5. Der Wrapper erhält das Ergebnis und prüft, ob es vom Typ „Exception“ ist. Falls ja, so wird diese geworfen, wodurch der Client die Serverseitige Exception bekommt. Anderenfalls wird es zum entsprechenden Rückgabetyp der aufgerufenen Methode gecastet und zurückgegeben.