

**Team:** 08, Dieter Pisarewski, Maxim Rjabenko

### Aufgabenaufteilung:

Maxim Rjabenko und Dieter Pisarewski:  
Gemeinsame Entwurf und Implementierung

### Quellenangaben:

Base64 Coder: <http://stackoverflow.com/questions/917604/decode-base64-string-java-5/917692#917692>

### Bearbeitungszeitraum:

28.11.2013 – 14.12.2013

### Aktueller Stand:

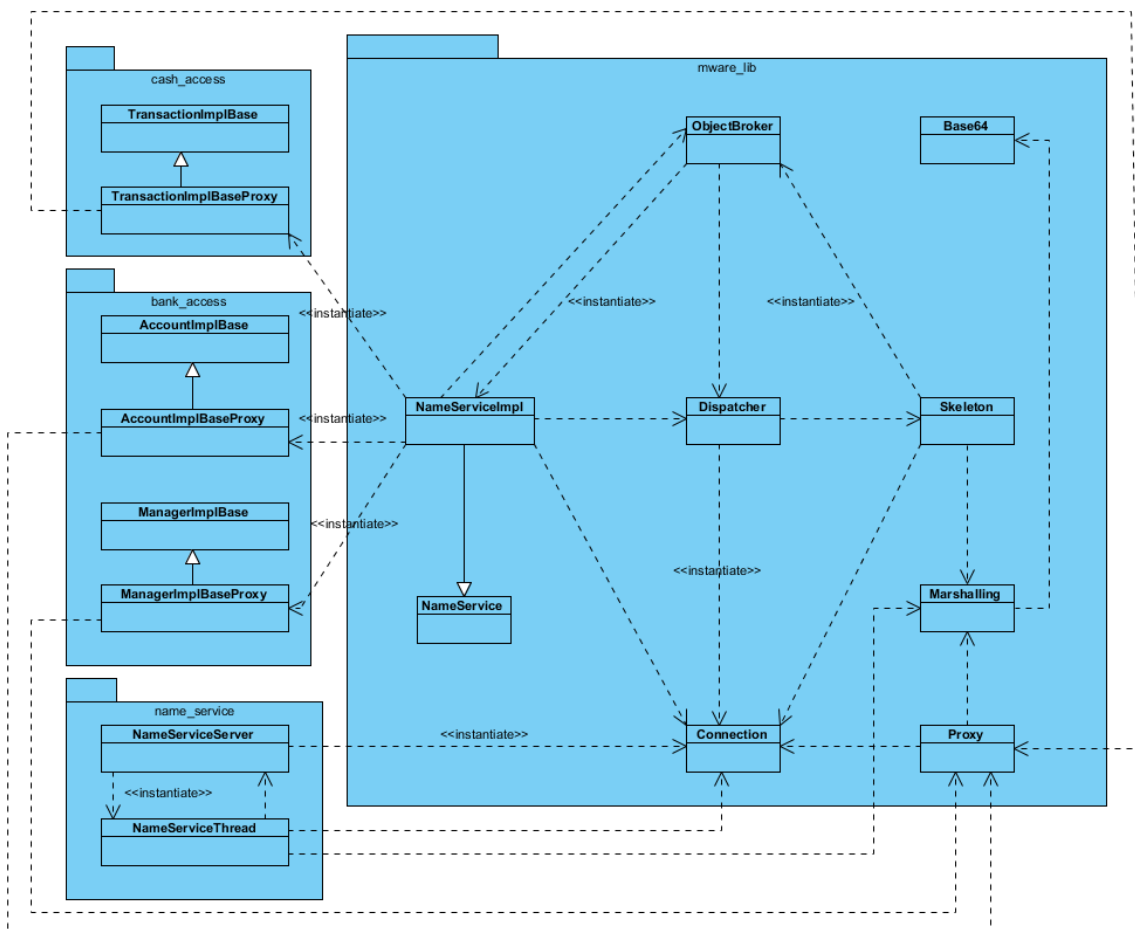
Algorithmus ist implementiert und auf einem Rechner getestet.

### Änderungen im Entwurf:

Alle Diagramme geändert

### Architektur:

### Classendiagramm



Wir haben folgende Klassen implementiert:

- Proxys:
  - AccountImplBaseProxy
  - ManagerImplBaseProxy
  - TransactionImplBaseProxy
- Vordefinierte Classen:
  - AccountImplBase
  - ManagerImplBase
  - TransactionImplBase
  - ObjectBroker
  - NameService
- Eigene Classen:
  - NameServiceServer
  - NameServiceImpl
  - Dispatcher
  - Skeleton
  - Connection
  - Marshalling
  - Proxy

**NameServiceServer** startet einen Server-Thread, der Kommandos fürs Binden bzw. Auflösen von entfernten Objekten verarbeitet. Dabei registriert er die Objekte bei sich in einem HashMap.

**NameServiceImpl** sendet Anfragen zum Binden bzw. Auflösen eines entfernten Objekts an den NameServiceServer.

Beim Binden überträgt er an den NameServiceServer einen Objektnamen, den Klassennamen des Objekts, eigenen Hostnamen und Port des lokalen Dispatchers.

Beim Auflösen erhält er die oben genannten Daten vom NameServiceServer und instatiert ein Proxyobjekt für entferntes Objekt.

**Proxy** dient als Helper-Klasse für entfernte Methodenaufrufe.

Sie implementiert Methode invoke, die eine Verbindung zum Host eines entfernten Objekts herstellt und Kommando „INVOKE“ mit dazugehörigen Namen des Objekts, Methodennamen und Methodenparametern sendet.

Danach liest Sie die Antwort auf diesen Aufruf, deserialisiert das Ergebnis und liefert es zurück.

**AccountImplBaseProxy** erbt von AccountImplBase, delegiert Methodenaufrufe an die Proxy Klasse.

**ManagerImplBaseProxy** erbt von ManagerImplBase, delegiert Methodenaufrufe an die Proxy Klasse.

**TransactionImplBaseProxy** erbt von TransactionImplBase, delegiert Methodenaufrufe an die Proxy Klasse.

**Dispatcher** starten einen Server-Thread, der entfernte Methodenaufrufe annimmt und einen Skeleton dafür startet.

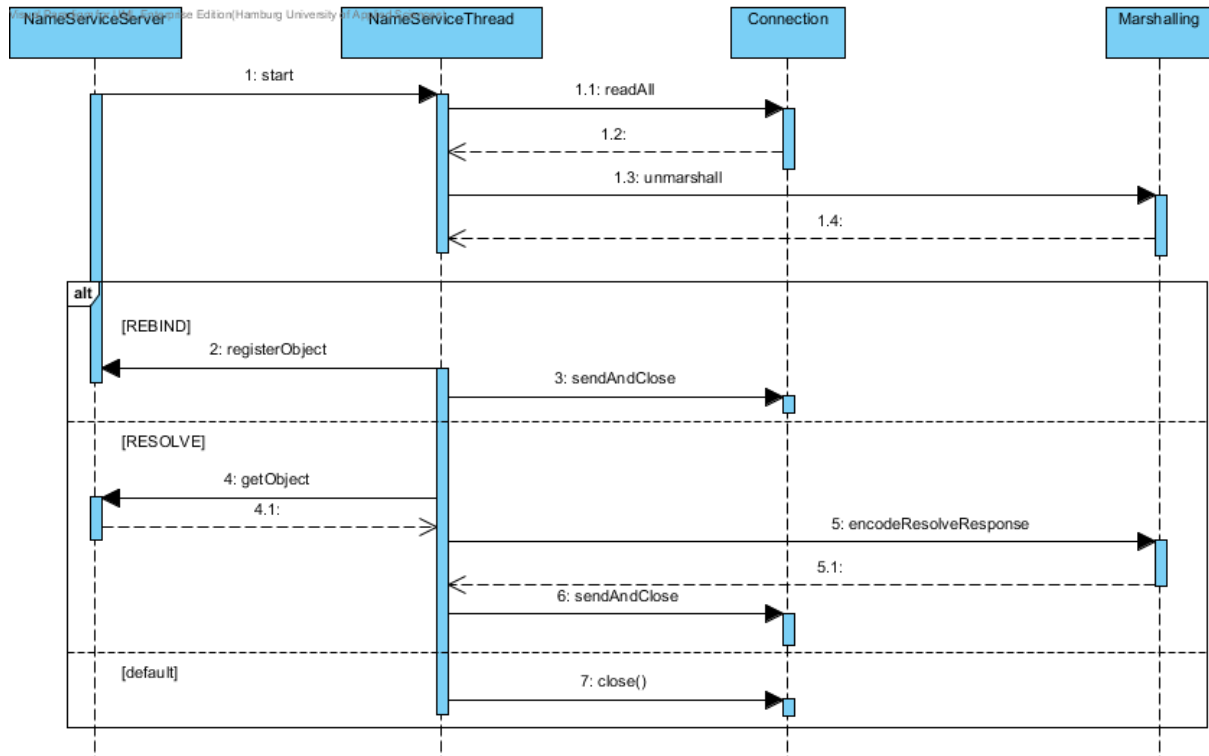
**Skeleton** verarbeitet einen entfernten Aufruf an einem lokalen Objekt.

**Connection** implementiert Methoden für Netzwerkkommunikation.

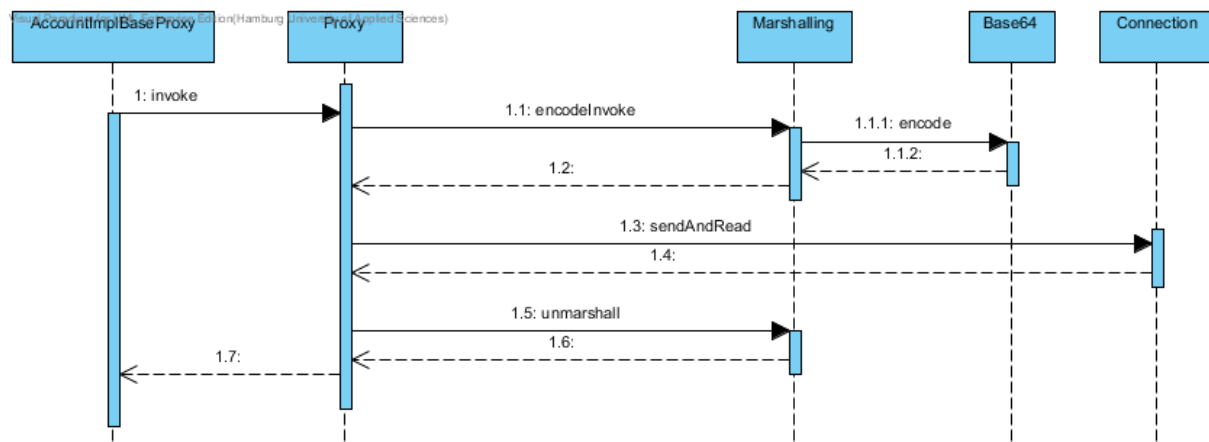
**Marshalling** dient als Helper-Klasse für Serialisierung. Sie benutzt Objektserialisierung von Java.

## Kommunikationsdiagramme

### Ablauf beim Binden/Auflösen eines Objekts



### Ablauf von einem Aufruf einer Methode an einem entfernten Objekt



## Ablauf beim Ausführen eines entfernten Methodenaufrufs an einem lokalen Objekt

