

Java RMI

**Systemtechnik Labor**

**4CHIT 2015/16, Gruppe C**

**Martin Weber**

|  |  |
| --- | --- |
|  | **Version 1.0** |
| **Note:** | **Begonnen am 1. April 2016** |
|  | **Beendet am x. April 2016** |

**Betreuer: Borko**

Inhaltsverzeichnis

[1. Einführung 3](#_Toc433312024)

[1.1. Ziele 3](#_Toc433312025)

[1.2. Voraussetzungen 3](#_Toc433312026)

[1.3. Aufgabenstellung 3](#_Toc433312027)

[2. Ergebnisse 4](#_Toc433312028)

# Einführung

Verteilte Objekte haben bestimmte Grunderfordernisse, die mittels implementierten Middlewares leicht verwendet werden können. Das Verständnis hinter diesen Mechanismen ist aber notwendig, um funktionale Anforderungen entsprechend sicher und stabil implementieren zu können.

## Ziele

Diese Übung gibt eine einfache Einführung in die Verwendung von verteilten Objekten mittels Java RMI. Es wird speziell Augenmerk auf die Referenzverwaltung sowie Serialisierung von Objekten gelegt. Es soll dabei eine einfache verteilte Applikation in Java implementiert werden.

## Vorraussetzungen

* Grundlagen Java und Software-Tests
* Grundlagen zu verteilten Systemen und Netzwerkverbindungen
* Grundlegendes Verständnis von nebenläufigen Prozessen

## Aufgabenstellung

Folgen Sie dem offiziellen Java-RMI Tutorial, um eine einfache Implementierung des PI-Calculators zu realisieren. Beachten Sie dabei die notwendigen Schritte der Sicherheitseinstellungen (SecurityManager) sowie die Verwendung des RemoteInterfaces und der RemoteException.

Implementieren Sie ein Command-Pattern [2] mittels RMI und übertragen Sie die Aufgaben/Berechnungen an den Server. Sie können am Client entscheiden, welche Aufgaben der Server übernehmen soll. Die Erweiterung dieser Aufgabe wäre ein Callback-Interface auf der Client-Seite, die nach Beendigung der Aufgabe eine entsprechende Rückmeldung an den Client zurück senden soll. Somit hat der Client auch ein RemoteObject, welches aber nicht in der Registry eingetragen wird sondern beim Aufruf mittels Referenz an den Server übergeben wird.

## Quellen

[1] "The Java Tutorials - Trail RMI"; online: http://docs.oracle.com/javase/tutorial/rmi/  
[2] "Command Pattern"; Vince Huston; online: http://vincehuston.org/dp/command.html  
[3] "Beispiel Konstrukt für Command Pattern mit Java RMI"; Michael Borko; online: https://github.com/mborko/code-examples/tree/master/java/rmiCommandPattern

Bewertung: 16 Punkte  
- Java RMI-Tutorial lauffähig (5 Punkte)  
- Implementierung des Command-Patterns mittels RMI (6 Punkte)  
- Implementierung des Client-Callbacks (2 Punkte)  
- Protokoll entsprechend der Richtlinien mit entsprechendem theoretischen Background (3 Punkte)

# Ergebnisse

## Java RMI Tutorial

### Theorie

### Praxis

Der Code wurde uns schon in den richtigen Files zur Verfügung gestellt. Wir mussten noch die Argumente die beim Starten übergeben anpassen, sowie Änderungen in der Java Policy vornehmen.

# Zeitaufzeichnung

Schätzung:

|  |  |
| --- | --- |
| Aufgabenstellung | Dauer |
| Java RMI | 3h |
| Command Pattern | 3h |
| Client-Callback | 2h |
| Protokoll | 2,5h |
| Gesamt | 10,5h |

Wirklich benötigt:

|  |  |
| --- | --- |
| Aufgabenstellung | Dauer |
| Java RMI | 3h |
| Command Pattern | 1,5h |
| Client-Callback | 1h |
| Protokoll | - |
| Gesamt | 5,5h |

# GitHub Link

https://github.com/mweber-tgm/JavaRMI