Technische Hochschule Ingolstadt
Fakultät EI
Prof. Dr. Christian Facchi

Übung Verteilte Systeme
WS 2018/19
Übungsblatt 6

1) Aufgabe (RMI - Echo Server):

Es soll nun eine einfacher Echo-Dienst erzeugt werden, der eine als Argument gegebene Zeichenreihe wiederholt und das Ergebnis zurückgibt. Der Client und der Server sind auf verschiedenen Maschinen lokalisiert (beim ersten Versuch auf der gleichen Maschine *localhost*). Der Server-Prozess erzeugt zu einer als Argument gegebener Zeichenreihe als Ausgabezeichenreihe die verdoppelte Eingabezeichenreihe („Hallo″ → „HalloHallo″) Hierzu soll RMI verwendet werden.

Client:

a) Kopieren Sie sich den Wraper des Client-Codes *EchoClient.java*. Sehen Sie sich den Code an. Hinweis: Der Zugriff auf ein entferntes Objekt (der Klasse *Echo*) wird mit einem Namensdienst (*Name.binding*) realisiert. Nach dieser Aktion kann auf das entfernte Objekt wie auf ein lokales Objekt zugegriffen werden.

b) Nun definieren Sie eine Klasse *Echo* als Interface von der Klasse *Remote* abgeleitet. Importieren Sie *java.rmi.**. Definieren Sie die Schnittstelle einer Methode *getEcho*, die sowohl als Eingabe, als auch als Ausgabe eine Zeichenreihe erfordert. Reagieren Sie auf *RemoteException* in einer Throws-Klausel.

Sever:

c) Kopieren Sie sich den Wraper des Server-Codes *EchoServer.java*. Sehen Sie sich den Code an. Hinweis: Mit *remote* und *rebind* binden Sie da Objekt *EchoImpl* an den dementsprechenden Port. Der Prozess terminiert nicht wegen des RMI-Systems.

d) Erstellen Sie jetzt eine Klasse *EchoImpl*. Diese Klasse ist von *UnicastRemoteObject* abgeleitet und somit wird das entfernte Objekt exportiert. Der Echo-Dienst ist eine Implementierung des Remote-Interface zur Klasse *Echo*, die als lokale Klasse ansprechen können. Reagieren Sie auf *RemoteException* in einer Throws-Klausel. Importieren Sie *java.rmi.**.

Komplettsystem:

e) Kopieren Sie die Sicherheitsregeln (Datei policy.txt) in das Projekt-Verzeichnis.

f) Starten Sie die Registrierung des Dienstes mit
start /Dbuild rmiregistry
oder falls Sie die Kommunikation protokollieren wollen:
start /Dbuild rmiregistry -J-Djava.rmi.server.logCalls=true

Es kann erforderlich sein, dass Sie die Path-Variable auf das Java jdk setzen müssen. Bitte passen Sie auf, dass der Dienst im Verzeichnis bin, in dem die class-Files enthalten sind.

Unter LINUX rufen Sie *rmiregstry* direkt auf.

g) Starten Sie den *EchoServer*
Hinweis: Es kann nötig sein, dass sie den Classpath für zu implementierende Klasse durch eine VM-Konfiguration –Djava.rmi.server.codebase=″file:<pfad zu class files>“ angeben müssen.

h) Starten Sie den *EchoClient* mit den Argumenten *localhost Hallo*

i) Lassen Sie den Client und den Server auf verschieden Rechnern laufen.

2) Aufgabe (RMI – Mobile Agenten):

Nun soll ein Programm implementiert werden, das einen zur Laufzeit bestimmten Programmcode vom Client zum Server überträgt und dann am Server zum Ablauf bringt. Das Ergebnis wird dem Client zurückgegeben. Das Herunterladen des Byte-Codes geschieht mit einem http-Server.
Hinweis: Richten Sie für die jeweiligen Anwendungen geeignete Verzeichnisse (*server*, *client*, *http*) ein.

Server:

a) Kopieren Sie sich die Dateien *Agent.java* und *ServerAgent.java*. Hinweis: Das Interface *Agent* und *ServerAgent* legen fest, dass die entfernte Methode *execute* einen Transport des Agenten initiiert.

b) Nun definieren Sie eine Klasse *ServerAgentImpl* als Implementierung des Interfaces *ServerAgent* und von der Klasse *UnicastRemoteObject* abgeleitet. Importieren Sie *java.rmi.** und *java.rmi.server.**. Definieren Sie den Konstruktor, sowie die Methode *execute*, die als Eingabeparmeter ein Objekt der Klasse *Agent* erfordert. Bei dem Eingabeobjekt wird die Methode execute ausgefuehrt und dann dieses Objekt als Rückgabewert dem Aufrufer übergibt. Reagieren Sie auf *RemoteException* in einer Throws-Klausel bei beiden Methoden.

c) Implementieren Sie nun in der Klasse *Server* das Server-Hauptprogram. Hinweis: Äquivalent zu Aufgabe 1 c).

d) Kopieren Sie die Sicherheitsregeln (Datei policy.txt) in das server-Verzeichnis.

Client:

e) Kopieren Sie sich die Datei *Agent.java, ServerAgent.java* und *DemoAgent.java*. Hinweis: Das Interface *Agent* und *ServerAgent* legen fest, dass die entfernte Methode *execute* einen Transport des Agenten initiiert. Die Klasse *DemoAgent* implementiert das Interface *Agent* und die Methoden *execute*, bei der einfach eine Summe gebildet wird (das könnte man mit Gauss einfacher haben ☺). Das Ergebnis wird über die Methode *getResult* zurückgeliefert.

f) Erstellen Sie jetzt eine Klasse *Client*. Hierbei lesen Sie sich den Hostnamen als erstes Kommandozeilenargument ein. Binden Sie ein Objekt der Klasse *ServerAgent* an den entfernten Dienst mit der Methode lookup unter den Namen *//<host>/agent*. (In Analogie zur Aufgabe 1: *EchoClient)*. Erzeugen Sie ein entferntes Objekt der Klasse *Agent* mit *new DemoAgent(100).* Dann starten Sie dieses Objekt mit der Methode *execute* und geben das Ergebnis aus. Importieren Sie *java.rmi.**.

http-Server:

g) Kopieren Sie den Code für einen einfachen http-Server *HTTPserver.java* in das entsprechende Verzeichnis und übersetzen Sie dies.

Komplettsystem

h) Starten Sie die Registrierung des Dienstes mit
start /Duild rmiregistry
oder falls Sie die Kommunikation protokollieren wollen:
start /Dbuild rmiregistry -J-Djava.rmi.server.logCalls=true

i) Starten Sie den *Server* mit den Java-Runtime-Parametern
-Djava.security.manager -Djava.security.policy=policy.txt -cp build

j) Starten Sie den *HTTPServer* mit den Argumenten *8080 ../client/build.* Hierbei beschreibt *8080* den betreffenden Port und *../client/build* den Ort unter den der Bytecode des Agenten zu finden ist.

Hinweise: Bei einem anderen Verzeichnis für den Java-Bayte-Code als build, verwenden Sie bitte diese Bezeichnung (z.B. bin)

k) Jetzt starten Sie den Client. Mit dem Parameter *-Djava.rmi.server.codebase=http://localhost:8080/*für die Java-Runtime-Umgebung geben Sie die Quelle für den Agentencode an (der Http-Server), an dem der Byte-code vom Java-Laufzeitsystem gefunden werden kann. Mit dem Kommandozeilenparameter *localhost* beschreiben Sie den Ort, an dem der Client ausgeführt werden soll.

l) Lassen Sie den Client, http-Server und den Server auf verschieden Rechnern laufen.

Hinweis: Falls Sie Server und Client auf verschiedenen Rechnern installieren wollen, müssen Sie bei der Virtualbox die Option Netzwerkbrücke (networkbridge) aktivieren. Somit erhalten Sie eine von außen zugängliche Netzwerkadresse.