Beantwortung der Fragen zu RMI

1. RMI (Remote Method Invocation) erlaubt den Aufruf von Methoden von Objekten, die sich auf einem anderen – einem entfernten – Rechnersystem befinden.

   • Man unterscheidet zwischen entfernten und lokalen Methoden bzw. entfernten und lokalen Objekten.

   • Der Aufruf einer entfernten Methode kann nicht vom Aufruf einer lokalen Methode unterschieden werden. Folglich realisiert RMI Ortstransparenz.

   Vorteile:

   – hohes Maß an Ortstransparenz

   – Einsatz eines Namensdienstes (RMIRegistry)

   – dynamisches Laden von Code möglich

   – Quasi-Standard

   Nachteile:

   – Integration mit anderen Verteilungstechniken schwierig

   – Synchronisation der entfernten Aufrufe durch Programmierer notwendig

   – Aufruf entfernter Methoden kann fehlschlagen

   • RMI-Client

   • Stub (-Objekte)

   • Skeleton (-Objekte)

   • RMI-Registry

   • RMI-Server und entfernte Objekte

   • Stub

   – Ein Stub-Objekt ist ein (client-seitiger) Stellvertreter für das entfernte Objekt beim RMI-Server.

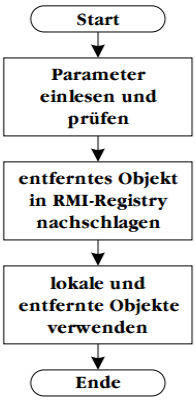
   • Skeleton

   – Ein Skeleton-Objekt ist ein (server-seitiger) Stellvertreter für das aufrufende Objekt.

   • RMI-Registry

   – Die RMI-Registry implementiert einen Namensdienst, der eine Abbildung von Dienstnamen auf entfernte Objekte realisiert.

Arbeitsweise eines RMI-Clients



Entwicklung einer RMI-Anwendung

Definition der Schnittstelle entfernter Objekte

⇓

Implementierung der entfernten Schnittstelle

⇓

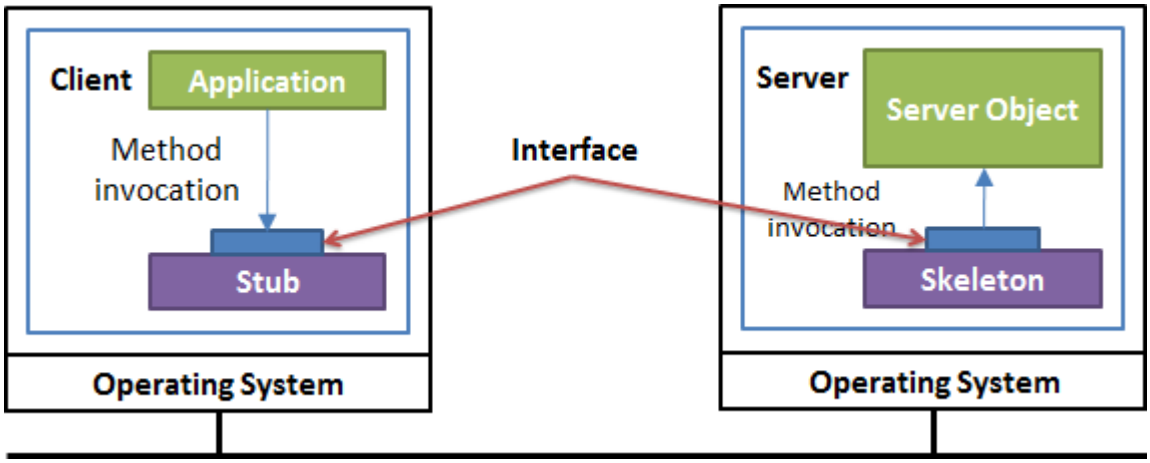
Generierung der Stubs/Skeletons

⇓

Starten des RMI-Servers

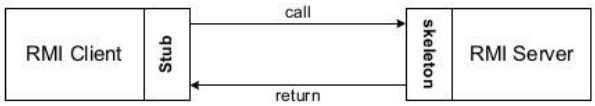
⇓

Starten des RMI-Clients



The Stub and Skeleton

SunilOS



- A client invokes a remote method, the call is first forwarded to stub.
- The stub is responsible for sending the remote call over to the server-side skeleton.
- The stub opens a socket to the remote server, marshals the object parameters and forwards the data stream to the skeleton.
- A skeleton contains a method that receives the remote calls, unmarshals the parameters, and invokes the actual remote object implementation.

Definition der Schnittstelle entfernter Objekte

Alle entfernt aufrufbaren Methoden müssen zunächst in einem Interface definiert werden, das von dem Interface Remote erbt.

• Das Interface Remote selbst ist leer und dient nur zur Markierung von so genannten RemoteInterfaces (Marker-Interface).

• Beispiele:

– Remote-Interface für DAYTIME

– Remote-Interface für TIME

– Remote-Interface für ECHO

CODEBASE

The client, which calls the Remote object can load the Method definitions from the codebase.

RMI SECURITY MANAGER

Obige Beispiele sind zwar lauffähig, aber mit Vorsicht zu genießen, da keinerlei Sicherheitsvorkehrungen getroffen wurden. Server und Client sind JavaApplikation mit allen Rechten. Irgendein Client auf irgendeinem Rechner kann zu unserem Server Verbindung aufnehmen und alle Methoden des Interfaces rufen. Solange man bloß ein Datum geliefert bekommt, ist das nicht problematisch, wenn aber andere Daten geliefert werden, so kann das schon anders aussehen. Um Mißbrauch einzudämmen gibt es SecurityManager. Wird ein SecurityManager gesetzt, so überprüft er, ob für einen bestimmten Vorgang die Genehmigung erteilt worden ist. Ohne Genehmigung endet der Versuch den Vorgang auszuführen mit einer **java.security.AccessControlException**. Genehmigungen werden erteilt durch einen entsprechenden Eintrag in die Datei **java.policy**, die sich Im Verzeichnis **$JAVA_HOME/jre/lib/security** befindet.

java.rmi.server.hostname

The value of this property represents the host name string that should be associated with remote stubs for locally created remote objects, in order to allow clients to invoke methods on the remote object. The default value of this property is the IP address of the local host, in "dotted-quad" format.

java.rmi.server.useCodebaseOnly

If this value is true, automatic loading of classes is prohibited *except* from the local CLASSPATH and from the java.rmi.server.codebase property set on this VM. Use of this property prevents client VMs from dynamically downloading bytecodes from other codebases. This property is ignored in the implementations of 1.2 and 1.2.1 because of a bug.

java.rmi.server.useLocalHostname

Java RMI now uses an IP address to identify the local host when the java.rmi.server.hostname property is not specified and a fully qualified domain name for the localhost cannot be obtained. In order to force Java RMI to use the fully qualified domain name by default, you need to set the this property to true.

2. Antworten auf die Fragen aus Arbeitsauftrag 2:
    a. Das Interface Java.rmi.Remote?
    → Die Klasse enthätl folgendes:

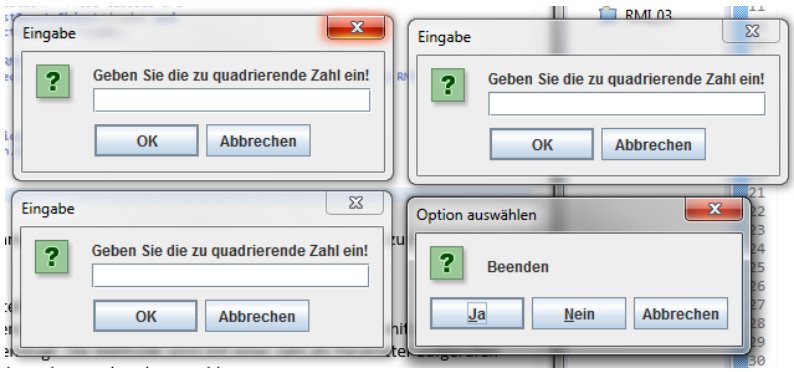
```
package java.rmi;

/**
 * The <code>Remote</code> interface serves to identify interfaces whose
 * methods may be invoked from a non-local virtual machine.  Any object that
 * is a remote object must directly or indirectly implement this interface.
 * Only those methods specified in a "remote interface", an interface that
 * extends <code>java.rmi.Remote</code> are available remotely.
 *
 * <p>Implementation classes can implement any number of remote interfaces and
 * can extend other remote implementation classes.  RMI provides some
 * convenience classes that remote object implementations can extend which
 * facilitate remote object creation.  These classes are
 * <code>java.rmi.server.UnicastRemoteObject</code> and
 * <code>java.rmi.activation.Activatable</code>.
 *
 * <p>For complete details on RMI, see the <a
href=../../../platform/rmi/spec/rmiTOC.html>RMI Specification</a> which describes the RMI API and system.
 *
 * @since   JDK1.1
 * @author  Ann Wollrath
 * @see     java.rmi.server.UnicastRemoteObject
 * @see     java.rmi.activation.Activatable
 */
public interface Remote {}
```

Wie man erkennen kann, enthält sie nichts. Das Interface wird nur zum Markieren verwendet.

b. Die Aufgabenstellungen lauten wie folgt:
→ Programmiere eine RMI Anwendung, die ein entferntes Objekt mit einer Methode quadrat(long) erzeugt. Die Methode wird mit einer Zahl als Parameter aufgerufen wird und berechnet das Quadrat dieser Zahl.
Vorteil: Leichter in der Implementierung
Nachteil: Bietet weniger Möglichkeiten (umständlich erweiterbar)
→ Der Server soll eine weitere Methode enthalten, die generische Tasks lösen kann. Es soll ein Interface Task geben, das als Argument dem Server übergeben wird.
Vorteil: Bietet mehrere Möglichkeiten (leichter zu erweitern)
Nachteil: Aufwändiger und komplexer in der Implementierung

c. Ja es funktioniert, es ist möglich mehrere Clients können gleichzeitig abgearbeitet werden und ein Client kann mehrere Aufrufe simultan durchführen.



Anmerkung: Bei der Ausführung der Programme, muss unter Run-configurations „localhost“ als Argument übergeben werden.