

IPD
Universität Karlsruhe
AVG, 2. OG
Adenauerring 20a
76131 Karlsruhe

Prof. Dr. Gerhard Goos Zi. 201 Tel. 0721/608-4760
ggoos@ipd.info.uni-karlsruhe.de

Mamdouh Abu-Sakran Zi. 232 Tel. 0721/608-8350
msakran@ipd.info.uni-karlsruhe.de

Übungsblatt: 3

Ausgabe: 26.5. 2004

Besprechung: 1.6. 2004

Aufgabe: 3.1 (Entfernter Methodenaufruf)

Verwenden Sie entfernten Methodenaufruf (engl. remote method invocation, rmi) in Java, um aus einem Applet heraus die Methode `String sayHello()` auf dem Dienstgeberrechner aufzurufen, von dem Sie das Applet geladen haben.

Gegeben sei dazu der folgende HTML-Code, in den das Applet eingebettet ist:

```
<HTML>
<title>Hello World</title>
<center> <h1>Hello World</h1> </center>
```

The message from the HelloServer is:

```
<p>
<applet codebase="myclasses/"
        code="examples.hello.HelloApplet"
        width=500 height=120>
</applet>
</HTML>
```

Das Gerüst für die Dienstnehmerklasse ist gegeben durch:

```
package examples.hello;

import java.applet.Applet;
import java.awt.Graphics;
import java.rmi.Naming;
import java.rmi.RemoteException;

public class HelloApplet extends Applet {

    String message = "blank";

    // "obj" is the identifier that we'll use to refer
    // to the remote object that implements the "Hello"
    // interface
    Hello obj = null;

    public void init() {
        try {
            obj = ;
            message = obj.sayHello();
        } catch (Exception e) {
            System.out.println("HelloApplet exception: " +
                               e.getMessage());
            e.printStackTrace();
        }
    }
}
```

```

    }
}

public void paint(Graphics g) {
    g.drawString(message, 25, 50);
}
}

```

Stellvertreterklassen zur entfernten Methodeninteraktion müssen Sie mit dem folgenden Aufruf des rmi-Übersetzers generieren:

```
rmic -d $HOME/public_html/myclasses examples.hello.HelloImpl
```

Die `.class`-Dateien der von Ihnen übersetzten Java-Dateien ihres Programmes müssen dazu im Verzeichnis `$HOME/public_html/myclasses` liegen. Falls Ihre Infrastruktur anders eingerichtet ist, ändern Sie die Verzeichnisangabe bitte entsprechend.

Den Dienstgeber starten Sie mit:

```
java -Djava.rmi.server.codebase=http://meinrechner.meinedomäne.de/~ich/myclasses/
-Djava.security.policy=$HOME/policy examples.hello.HelloImp
```

Die Datei `policy` hat beispielsweise folgenden Inhalt:

```
grant {
    // Allow everything for now
    permission java.security.AllPermission;
};
```

Auf dem Dienstnehmerrechner starten Sie das Applet, indem Sie die oben angegebene HTML-Seite in ein entsprechendes Internet-Programm laden, beispielsweise:

```
appletviewer http://meinrechner.meinedomäne.de/~ich/helloRMI.html &
```

Hinweis: Denken Sie daran die RMI-Ablage und ihren WWW-Dienstgeberprozeß zu starten.

Aufgabe: 3.2 (Sprachbindung)

Gegeben sei eine IDL `MINIDL`, die folgende primitive Wertetypen und Konstruktoren für Wertetypen definiert:

Primitiver Typ	Definition
<code>bool</code>	Boolescher Wahrheitswert: <code>true</code> oder <code>false</code>
<code>int</code>	Vorzeichenbehaftete 64-bit Ganzzahl
<code>string</code>	UTF-8 Zeichenkette

Typkonstruktor	Definition
<code>array (base , int length)</code>	Reihung gegebener Länge des Basistyps (nullbasiert)
<code>record(t1 n1, t2 n2, ..., tn nn)</code>	Record. Enthält alle gegebenen, namentlich adressierbaren Felder.
<code>union (t1 n1, t2 n2, ..., tn nn)</code>	Typsichere Variante (genau eine der Alternativen)

- Definieren Sie eine Sprachbindung $b_j : \text{MINIDL} \rightarrow \text{JAVA}$ mit struktureller Induktion. Geben Sie dazu zunächst Abbildungen der primitiven Typen an, dann Abbildungen der Konstruktoren. (Hinweis: Verwenden sie Klassen für `record` und `union`. Vererbung ist der Schlüssel zur Umsetzung von `union`.)
- Definieren Sie eine Abbildung u_j in umgekehrter Richtung. Welche Probleme treten auf?
- Ist $u_j \circ b_j = id_{\text{MINIDL}}$? Ist $b_j \circ u_j = id_{\text{JAVA}}$? Wenn nicht, wie erreicht man das?
- Können Sie eine Sprachbindung $b_c : \text{MINIDL} \rightarrow \text{C}$ angeben? Welche Probleme treten auf?

Aufgabe: 3.3 (Serialisierung)

Gegeben sei folgendes Serialisierungsformat für Wertetypen in MINIDL:

Primitiver Typ	Serialisierungsformat
bool	Ein Byte: $\neq 0 \sim \text{true}$, $0 \sim \text{false}$
int	Acht Bytes in <i>little-endian byte order</i>
string	Länge der Zeichenkette als int, entsprechend viele Bytes UTF-8

Typkonstruktor	Serialisierungsformat
array (base , int length)	Länge des Arrays als int, entsprechend viele Serialisierungen des Basistyps
record(t1 n1, t2 n2, ..., tj nj)	Serialisierungen der Feldtypen in Reihenfolge der Definition
union (t1 n1, t2 n2, ..., tj nj)	Index der Variante (n1 ~ 0) als int, Serialisierung der entsprechenden Variante

- Schreiben Sie Serialisierer s_j für die primitiven Typen in JAVA (Hinweis: Ziehen sie für Fließkommazahlen `java.lang.Float` und `java.lang.Double` zu Hilfe). Vorgabe für alle primitiven Typen X in JAVA :


```
final class Serializer {
    public static void serializeX(ByteOutputStream os, X value) { }
}
```
- Schreiben Sie einen Serialisierer für Objekte (Java-Arrays sind Objekte). Verwenden Sie `java.lang.reflect`, um die Felder eines Objekts zu ermitteln und zu lesen. Setzen Sie für objektwertige Felder Rekursion ein, ansonsten greifen Sie auf primitive Serialisierer zurück. Verwenden Sie die obige Vorgabe mit `X = Object`.
- Wie vollständig ist Ihr Serialisierer? Wo liegen Probleme?
- Ist es möglich, einen allgemeinen Deserialisierer anzugeben? Geben Sie eine Lösung oder ein Gegenbeispiel und einen alternativen Lösungsansatz an.

Aufgabe: 3.4 (stubs und skeletons)

Klassische Middleware-Architekturen wie CORBA erzeugen mit einem IDL-Übersetzer spezifische *stubs* und *skeletons*. Moderne Enterprise JavaBeans-Container wie JBOSS verwenden stattdessen Reflexion und dynamische Aufrufe, um allgemeine *stubs* und *skeletons* zu realisieren.

- Implementieren sie einen universellen Stub mit `java.lang.reflect.Proxy` und ein passendes universelles Skeleton. Verwenden Sie dabei Java-Serialisierung und Deserialisierung.
- Diskutieren Sie Vor- und Nachteile der dynamischen und statischen Ansätze.