— Übungsblatt 3 (T): Entfernte Aufrufe —

Aufgabe 1 (Nachrichtenaustausch) (T)

In der Vorlesung haben Sie verschiedene Methoden kennengelernt, wie Nachrichten in einem verteilten System ausgetauscht werden können. Ordnen Sie die untenstehenden Definitionen den jeweils richtigen Namen zu.

1. Streams und Sockets
2. Entfernte Prozeduraufrufe
3. Entfernte Methodenaufrufe
4. Webservices
5. Message Passing

Name: _____
Eine Anwendung, die mit einem Uniform Resource Identifier eindeutig identifizerbar ist und deren Schnittstelle als XML-Artefakt definiert, beschrieben und gefunden werden kann.

Name: _____
Alle an der Kommunikation beteiligten Systeme sind gleichberechtigt. Eine Unterscheidung zwischen Client und Server gibt es nicht, stattdessen werden die Systeme durch einen unabhängigen Vermittler entkoppelt.

Name: _____
Die Objekte der Anwendung befinden sich nicht mehr im Speicher nur eines Prozesses, sondern können sich im Speicher verschiedener Prozesse befinden, die auf unterschiedlichen Rechnern laufen können.

Name: _____
Client und Server unterscheiden sich nur durch die Rollenverteilung beim Verbindungsaufbau. Sobald die Verbindung hergestellt wurde, können beide Systeme jederzeit Daten senden, die dabei einzuhaltende Reihenfolge ergibt sich einzig aus dem Anwendungsprotokoll.

Name: _____
Eine Serveranwendung verwaltet eine Menge von Routinen, welche durch einen Anfrage/Antwort-Mechanismus über das Netzwerk aufgerufen werden können.

Aufgabe 2 (Lokale und verteilte Objektmodelle) (T)

Folgende Begriffe sind Ihnen aus der nicht verteilten Programmierung bereits bekannt. Übertragen Sie diese in das verteilte Objektmodell, indem Sie mit wenigen Sätzen die Unterschiede zwischen der herkömmlichen Definition und der Definition im Kontext von verteilten Systemen erläutern.

1. Objektsystem
2. Objektreferenz
3. Schnittstelle
4. Ereignisse und Aktionen
5. Ausnahmen (Exceptions)
6. Garbage Collection

Erläutern Sie zusätzlich, warum der Garbage Collector Probleme mit entfernten Referenzen verursacht. Skizzieren Sie einen Ablauf, wie die Middleware diese Probleme vermeiden kann.

Aufgabe 3 (Entfernte Aufrufe im Detail) (T)

1. Welche Aufgaben übernehmen Stub- und Skeletonobjekte und wie werden sie erzeugt?
2. Wie wird beim entfernten Aufruf das Proxymuster umgesetzt?
3. Fertigen Sie eine Skizze an und beschreiben Sie, wie ein entfernter Aufruf von der Middleware abgewickelt wird.
4. Grenzen Sie die beiden Übergabesemantiken `Call by Value` und `Call by Reference` voneinander ab. Wann kommt welche Methode zum Einsatz?
5. Welche Probleme können bei der Kopiersemantik entstehen, wenn ganze Objekte zwischen den Systemen ausgetauscht werden?
6. Was ist ein Namensdienst und wozu wird er beim entfernten Aufruf genutzt? Wie kann damit Orts- und Mobilitätstransparenz erreicht werden?

Aufgabe 4 (Wahr oder Falsch) (T)

Sind die folgenden Aussagen wahr oder falsch? Kreuzen Sie pro Zeile nur eine Antwort an. Für jedes korrekte Kreuz gibt es einen Punkt. Für falsch gesetzte Kreuze gibt es keinen Punktabzug. Ebenso werden Zeilen mit mehr oder weniger als einem Kreuz nicht bewertet. Sie können daher bis zu acht Punkte erhalten.

Aussage	wahr	falsch
Entfernte Objekte können nicht lokal aufgerufen werden.		
Entfernte Schnittstellen müssen in einer speziellen Beschreibungssprache, die nicht Teil der Sprache Java ist, beschrieben werden, um mit `Sun/Oracle RMI` verwendet werden zu können.		
Da Methodenaufrufe nicht wirklich den Speicher einer Java-Anwendung verlassen können, benutzt man ein Stellvertreterobjekt (Stub), auf dem die Methoden ganz normal aufgerufen werden. Dieses Objekt codiert dann den Aufruf in eine Byteform und verwendet direkte Socketprogrammierung, um die Daten an ein Skeletonobjekt einer anderen Anwendung zu schicken, das den Aufruf (ebenfalls lokal) vornimmt.		
Neben `Java RMI` gibt es noch eine Menge weitere Middleware-Technologien, welche die Durchführung entfernter Methodenaufrufe ermöglichen.		
Das Remote Interface wird, weil es sich ja um eine entfernte Schnittstelle handelt, nur auf Seiten des entfernten Objekts benötigt.		
`Java RMI` kennt zweierlei Arten entfernter Objekte: Solche die permanent im Speicher der Serveranwendung sind und solche, die von `RMI` bei Bedarf automatisch erzeugt werden. Für den Aufruf ergibt sich jedoch kein Unterschied.		
Die von `RMI` beim entfernten Aufruf vorgeschriebene `RemoteException` gehört zur Familie der sog. `Runtime Exceptions`, weshalb Sie nicht zwingend behandelt werden muss.		
Java kennt nur die Übergabesemantik `Call by Value`, weshalb automatisch ein Stub- und ein Skeletonobjekt entsteht, wenn Sie ein normales Objekt einem entfernten Objekt als Parameter übergeben.		

Aufgabe 5 (Webservices) (T)

1. Was ist ein Webservice und worin besteht der Unterschied zu anderen entfernten Aufrufen?
2. Zählen Sie Vor- und Nachteile von Webservices auf und geben Sie ein Beispiel, wo der Einsatz von Webservices sinnvoll sein kann.
3. Was machen der Anbieter, das Verzeichnis und der Konsument von Webservices?
4. Welche Informationen beinhaltet eine `WSDL`-Beschreibung?
5. Wie unterscheiden sich `RESTful` Webservices von `SOAP`-Webservices oder `XML-RPC`?

Aufgabe 6 (SOAP-Webservices mit JAX-WS) (P)

Vervollständigen Sie die nachfolgende Klasse so, dass ihre Methoden über einen `SOAP`-Webservice namens `Ticket2Rock` aufgerufen werden können. Sorgen Sie durch entsprechende Annotationen dafür, dass die Methodenparameter im Webservice genauso heißen, wie im Quellcode. Außerdem soll der Rückgabewert der Methoden `getAllArtists` und `searchArtist` in der Antwortnachricht `artist` heißen. Die Rückgabewerte der Methoden `getAllConcerts` und `searchConcerts` sollen entsprechend `concert` genannt werden.

```
import java.util.*;
import javax.jws.*;

public class ConcertServiceImpl {
    private List<Artist> artists = new ArrayList<Artist>();
    private List<Concert> concerts = new ArrayList<Concert>();

    public List<Artist> getAllArtists() {
        return this.artists;
    }

    public List<Concert> getAllConcerts() {
        return this.concerts;
    }

    public Artist searchArtist(             String query) {
        // Ganz komplizierter Quellcode hier
    }

    public List<Concert> searchConcerts(               String place,
                                                       String time,
                                                       String name) {
        // Noch mehr komplizierte Zeilen
    }
}
```