

## Aufgabenblatt 3

### Musterlösung

## Vorlesung Verteilte Systeme

### Wintersemester 2025/26

#### Aufgabe 1: RPC und Unions

Falls das Laufzeitsystem nicht erkennt, welcher Wertetyp sich in dem Feld befindet, kann es ihn nicht korrekt verpacken. Damit können Unions in einem RPC-System nicht toleriert werden, es sei denn, es gibt ein kennzeichnendes Feld, das eindeutig beschreibt, was in dem varianten Feld enthalten ist. Das kennzeichnende Feld darf nicht der Kontrolle des Benutzers unterliegen.

#### Aufgabe 2: Entfernte Objektreferenzen

Unter Verwendung von Java können wir eine solche Implementierung z.B. durch die folgende Klasse ausdrücken:

```
public class Object_reference {
    InetAddress server_address; // Netzwerkadresse des Objekt-Servers
    int server_port;           // Port, der Server idnetifiziert
    int object_id;             // Identifier für dieses Objekt
    URL client_code;           // (entfernte) Datei, die den Code des
                                // Client-Stubs enthält
    byte[] init_data;         // Möglicherweise zusätzliche
                                // Initialisierungsdaten für den Stub
}
```

Die Objektreferenz sollte mindestens die Transportschichtadresse des Servers enthalten, auf dem sich das Objekt befindet, hier ebenso auch den Port. Außerdem brauchen wir eine Objekt-ID, weil der Server mehrere Objekte enthalten kann. In unserer Implementierung verwenden wir eine URL, um auf eine (entfernte) Datei zu verweisen, die den gesamten erforderlichen Client-seitigen Code enthält. Eine alternative Implementierung wäre es, direkt einen serialisierten Client-Stub als Objektreferenz zu verwenden. Dieser Ansatz wird beispielsweise in Java RMI befolgt, wo Proxies als Referenz übergeben werden.

#### Aufgabe 3: Exceptions

Weil die Exceptions ursprünglich auf der Server-Seite geworfen werden, kann der Server-Stub die Ausnahme nur abfangen, als spezielle Fehlerantwort verpacken und an den Client zurückgeben. Der Client-Stub dagegen muss die Nachricht entpacken und die selbe Exception werfen, um den Zugriff auf den Server transparent zu halten. Demzufolge müssen auch die Ausnahmen in einer Schnittstellendefinitionssprache beschrieben werden.

#### **Aufgabe 4: Threads**

**Pflichtaufgabe, Abgabe über moodle!**

#### **Aufgabe 5: RMI- Grundlagen und Architektur**

**Pflichtaufgabe, Abgabe über moodle!**

Siehe Kap. 2 und 3 der Vorlesung.

#### **Aufgabe 6: Programmierung: Java-RMI - Hello World!**

Die Lösung dieser Aufgabe finden Sie im Archiv [103Files.zip](#)<sup>1</sup> auf der Vorlesungswebseite.

---

<sup>1</sup><http://www.bs.informatik.uni-siegen.de/web/wismueller/v1/vs/103Files.zip>