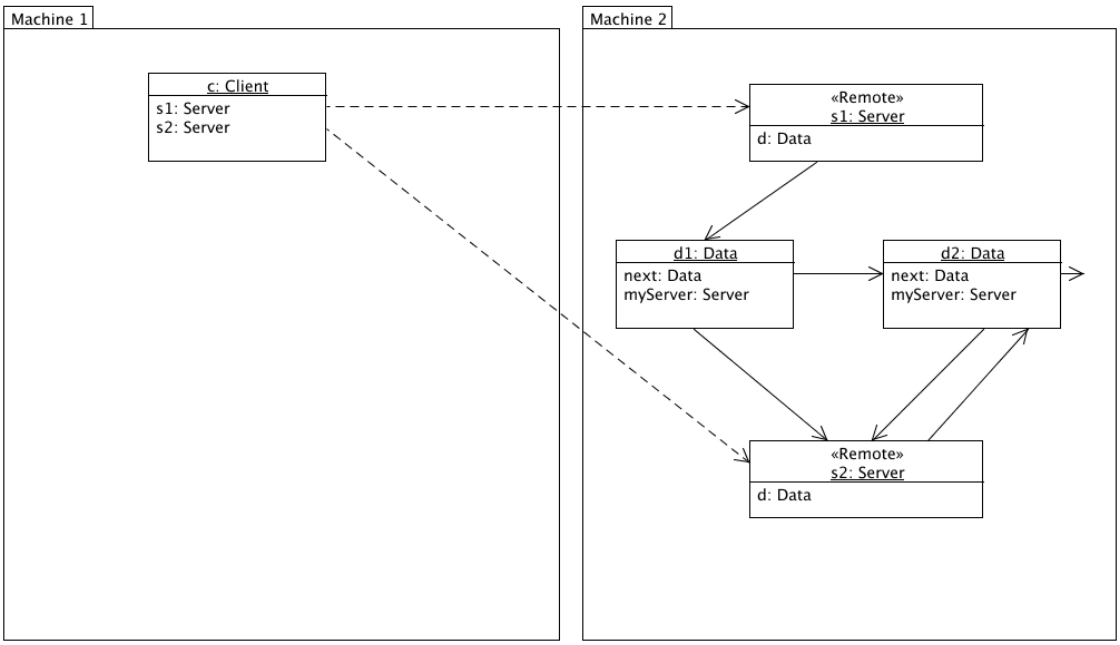
Übungsblatt 11

1. Erstelle eine verteilte „Chat“-Anwendung mit Java und **RMI**.
   Es gibt einen Server mit einer Methode zum Registrieren eines Chat-Partners und einer Methode, um eine Nachricht an alle registrierten Clients zu versenden.
   Die Clients registrieren sich beim Server, können Nachrichten versenden und werden vom Server mit neuen Nachrichten versorgt.
   Eine Nachricht soll aus einer Client-ID, der Uhrzeit und einer Textnachricht bestehen.

2. In Übung 4 haben wir die Verwendung des Observer-Patterns für aktive Werte betrachtet. Der Observer musste dabei ein bestimmtes Interface implementieren, damit er von Veränderungen benachrichtigt werden kann.
   Ändere die Implementierung so, dass Reflection verwendet wird um eine Methoden-Referenz zu registrieren. Die Observer sollen nicht mehr gezwungen sein ein bestimmtes Interface zu implementieren.

3. **Klausurbeispiel!**
   Verteilte Programmierung
   Gegeben ist die unten abgebildete Objektstruktur. Der Client-Prozess läuft auf Maschine 1, der Server-Prozess auf Maschine 2. Der Server-Prozess enthält zwei Remote-Objekte vom Typ Server sowie zwei Datenelemente vom Typ Data, die nicht remote sind. Im Client-Prozess befindet sich nur ein Objekt, welches Referenzen auf die beiden Server-Objekte des Server-Prozesses besitzt. Im Diagramm werden nur die Felder gezeigt; alle Klassen besitzen für ihre Felder entsprechende Zugri s-Methoden und eventuell weitere Methoden. Insbesondere enthält die Klasse Server die Methode Data getData() { return d; }.

   a) Wie funktioniert in Java RMI die Kommunikation zwischen zwei Prozessen?
   b) Welche Schritte erfolgen beim Kompilieren des Programms?
   c) Client c ruft auf dem Server-Object s1 die Methode getData auf. Zeichnen Sie in das obige Diagramm die dadurch auf Maschine 1 entstehende Objektstruktur.
   d) Client c ruft auf dem Server-Object s2 die Methode getData auf. Zeichnen Sie in das obige Diagramm die dadurch auf Maschine 1 entstehende Objektstruktur.
   e) Alle Argumente eines Methoden-Aufrufs werden in Java gemeinsam serialisiert. Schreiben Sie ein Programm-Fragment, an dem man den Unterschied zwischen gemeinsamer Serialisierung und getrennter Serialisierung sieht. Stellen Sie die dabei entstehenden Objektstrukturen graphisch dar. Die Klassen Client, Remote und Data können für dieses Beispiel verwendet werden.
   f) Kann man Java RMI als ortstransparent (location transparent) bezeichnen? Begründen Sie Ihre Antwort!


Machine 1
Machine 2
c: Client
s1: Server
s2: Server
«Remote»
s1: Server
d: Data
d1: Data
next: Data
myServer: Server
d2: Data
next: Data
myServer: Server
«Remote»
s2: Server
d: Data