

6. Übung „Nebenläufige und verteilte Programmierung“

Abgabe am 12. Dezember, Besprechung am 13. Dezember

Aufgabe 1

Eine Kommunikation bei RMI besteht immer aus einer Anfrage und einer Antwort. Auch wenn die Methode keinen Rückgabewert (void) liefert. Schreiben Sie ein Program mit dem Sie zeigen, dass der Client auch für Methoden ohne Rückgabewert auf die Antwort des Servers wartet. Durch die zusätzlichen Verwendung von Threads ist es auch möglich Nachrichten abzuschicken, ohne auf ein Ergebnis zu warten. Wie können Sie diese Kommunikationsvariante durch die Definition spezieller Klassen/Interfaces unterstützen? Überlegen Sie insbesondere auch, was Sie bei Kommunikationsfehlern (Remote Exceptions) machen können.

Aufgabe 2

Reimplementieren Sie den Chat unter Verwendung von RMI.

- a) Überlegen Sie genau, welche RMI-Server-Objekte registriert werden müssen und welche RMI-Objekt-Referenzen in Nachrichten weitergegeben werden können.
- b) Die programmierten Chats sind zwar robust gegen Ausfälle von Klienten. Sie sind aber nicht robust bezüglich des Ausfalls des Servers. Überlegen Sie sich eine neue Struktur, bei der kein ausgezeichneter Server mehr existiert, sondern alle Klienten gleichberechtigt miteinander kommunizieren. Der Chat soll solange existieren, bis sich kein Klient mehr im Chatraum befindet. Überlegen Sie sich auch einen Registrierungsmechanismus, der das Nadelöhr eines Servers verhindert, aber dennoch die Existenz mehrerer unabhängiger Chats erlaubt.
- c) An welchen Stellen in Ihren Programmen ist das Warten auf die Beendigung des Methodenaufrufs überflüssig? Verwenden Sie die in Aufgabe 1 definierten Klassen/Interfaces um die entsprechenden Aufrufe zu ersetzen.

Aufgabe 3

In dieser Aufgabe sollen Sie erste einfache Erlang Programme schreiben.

- a) Programmieren Sie `fibExp/1` und `finLin/1`, die die Fibonacci-Funktion mit exponentiellem bzw. linearem Aufwand berechnen.
- b) Programmieren Sie eine Funktion `qSort/1`, welche eine Liste von Zahlen mit Hilfe des Quicksort-Algorithmus sortiert.

- c) Programmieren Sie einen Prozesse, welcher `ping`-Nachrichten empfängt und mit `pong`-Nachrichten antwortet. Starten Sie diesen Prozess in einer Funktion `pingTest` und kommunizieren Sie mehrfach mit dem `ping`-Server.
- d) Schreiben Sie ein Testprogramm, welches die genaue Matching-Reihenfolge des `receive`-Ausdrucks verdeutlicht.