Übungen und Prüfungen

- Endnote ergibt sich zu 20% aus bewerteten Übungen und zu 80% aus schriftlicher Prüfung
- Übungen
  - 2 praktische Übungen (je 4.5%)
  - 2 theoretische Übungen (je 3%)
  - 1 Übung Alonso (5%)
- Schriftliche Prüfung in der Prüfungssession
  - Stoff der Vorlesung
  - Stoff der theoretischen Übungen
  - Konzeptionelles aus den praktischen Übungen

Voraussichtliche Termine

- 5.11. Besprechung praktische Übung 1
- 12.11. Besprechung theor. Übung 1
- 26.11. Abnahme praktische Übung 1
- 3.12. Besprechung praktische Übung 2
- 10.1. Abnahme praktische Übung 2
- 14.1. Nachbesprechung prakt. + theor. Übungen

Praktische Übungen

- Gruppen von 3-4 Studenten
- Rechnerräume IFW D 31 und IFW D 35
- Übungen können auch anderswo gelöst werden, die Lösungen müssen bei der Abnahme aber in obigen Räumen oder auf Laptop funktionieren
- Betreuung durch Tutoren einmal in der Woche
- Abnahme durch Tutoren in der Vorlesung am auf dem Übungsblatt angegebenen Termin, Anwesenheit *aller* Gruppenmitglieder erforderlich
- Besprechung der Lösungen in der Vorlesung

Praktische Übungen

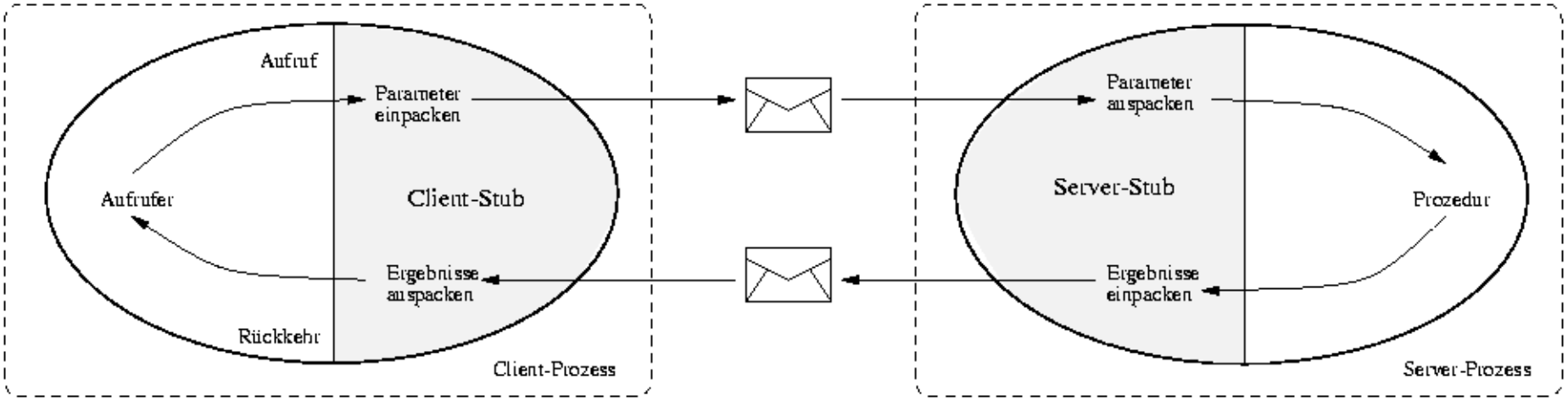
- Jeder sollte einen Account haben
- Webseite mit Aufgabenstellungen, Codegerüsten, Literaturhinweisen etc:

  `http://www.vs.inf.ethz.ch/edu/WS0405/VS/`

- Betreuer (Mattern)
  - Kay Römer: `roemer@inf.ethz.ch`
  - Matthias Ringwald: `mringwal@inf.ethz.ch`

Remote Procedure Call

- Aufruf von Methoden/Prozeduren über Rechnergrenzen hinweg



- Client-Stubs: Stellvertreter der Prozedur im Client
- Server-Stubs: Stellvertreter des Aufrufers im Server

Ablauf eines RPC

- Aufrufer ruft Client-Stub auf
- Client-Stub erzeugt Nachricht mit Namen und Parametern der aufgerufenen Prozedur
- Übertragung der Nachricht zum Server-Prozess
- Server-Stub dekodiert Prozedur-Namen und Parameter und ruft Prozedur auf
- Server-Stub verpackt Ergebnisse in Nachricht
- Übertragung der Nachricht zum Client-Prozess
- Client-Stub dekodiert Ergebnisse und übergibt sie dem Aufrufer

Lösungshinweise

- Codegerüste auf der Webseite
- Tips zur Implementierung:
  - Nachrichtenformat
  - (De-)Codieren der Nachrichten
  - Verwendung der Sockets in den Client-Stubs
  - Schnittstelle Dispatcher -> Server-Stubs
  - Schnittstelle Server-Stubs -> Objekte
  - Event-Loop des Dispatchers

Nachrichtenformat

- Was muss die Auftragsnachricht enthalten?
  - Identifikation des Zielobjektes
  - Name der aufgerufenen Methode
  - Parameter
- Was muss die Antwortnachricht enthalten?
  - Flag ob Exception aufgetreten ist
  - Ergebnisse bzw. Exception

(De-)Codieren der Nachrichten

- Java-Klassen `DataInputStream` und `DataOutputStream` sehr hilfreich!
- `DataOutputStream`-Methoden:
  - `writeUTF()`
  - `writeLong()`
  - `writeByte()`
  - `flush()`
- `DataInputStream`-Methoden:
  - `readUTF()`
  - `readLong()`
  - `readByte()`

Verwendung der Sockets

- Genau eine Socket-Verbindung zwischen Client und Server
- Wird gemeinsam von mehreren Stubs benutzt
  - Implementierung mittels `static` Klassenvariable
- Gemeinsame Basisklasse `ClientStubBase` für alle Client-Stubs

Schnittstelle Dispatcher/Server-Stubs

- Neu erzeugte Server-Stubs müssen beim Dispatcher registriert werden
- Trifft beim Dispatcher eine Nachricht ein, so bestimmt er den zugehörigen Server-Stub und übergibt ihm den Aufruf
- Server-Stubs bieten Methode `dispatch()` an
- Gemeinsame Basisklasse `ServerStubBase` für alle Server-Stubs

Schnittstelle Server-Stubs/Objekte

- Server-Stubs sind separate Objekte
- Beim Erzeugen eines Stubs wird im Konstruktor das zugehörige Objekt übergeben

Event-Loop des Dispatchers

- Nach der Initialisierung wird die Methode `run()` des Dispatchers aufgerufen
- `run()` wartet in einer Endlosschleife auf Verbindungswünsche von Clients
- Eintreffende Netzwerknachrichten werden an den zuständigen Server-Stub weitergeleitet