# Testate und Prüfungen

- Testate bei erfolgreicher Bearbeitung von 70% der Aufgaben
  - ca. 3-4 praktische Aufgaben
  - ca. 2-3 theoretische Aufgaben
- Schriftliche Prüfung in der Prüfungssession:
  - Stoff der Vorlesung
  - Konzeptionelles aus den 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 funktionieren
- Betreuung durch Tutoren einmal in der Woche in den Rechnerräumen
- Abnahme durch Tutoren in der Vorlesung am auf dem Übungsblatt angegebenen Termin, Anwesenheit *aller* Gruppenmitglieder erforderlich
- Besprechung der Lösungen in der Vorlesung

# Praktische Übungen 2

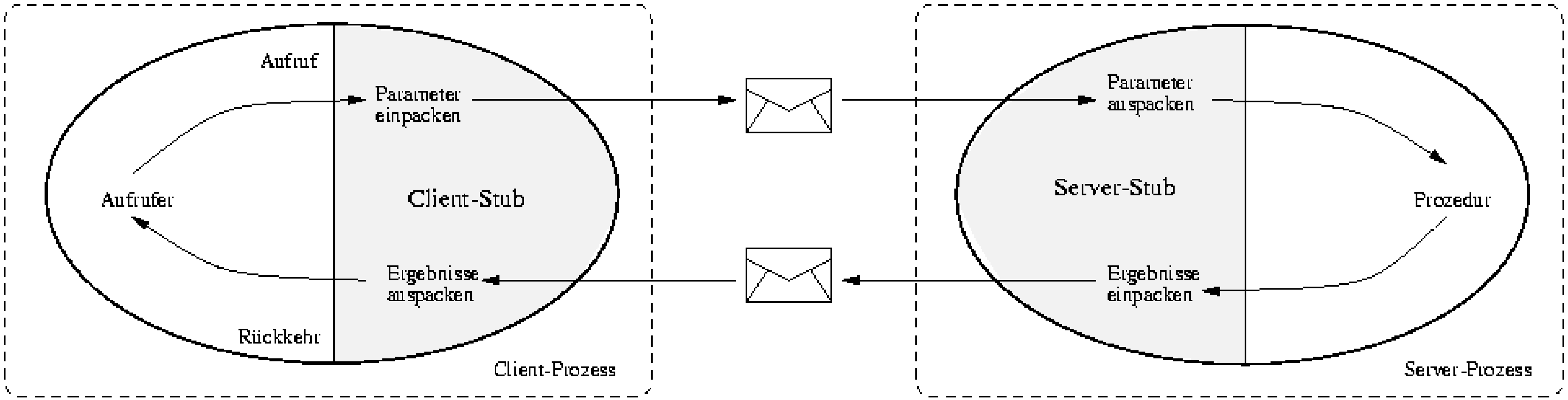
- Nicht-Informatiker bitte nach der Vorlesung wegen Account bei mir melden
- Webseite mit Aufgabenstellungen, Codegerüsten, Literaturhinweisen etc:

  `http://www.inf.ethz.ch/vs/education/WS0203/VS/`

- Betreuer
  - 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