# LV 4342
# Skriptsprachen-Praktikum
# Übung 10

## Distributed Ruby und Marshaling

# Organisatorisches

- Arbeitsverzeichnis:
  `~/lv/skriptspr/10/`

- Dateinamen:

```
10-lottoclient.rb      # Entwickeln und abgeben
10-teilnehmerdaten.rb  # Vorlage erweitern, abgeben
10-sportclient.rb      # Entwickeln und abgeben
10-sportsrv.rb         # Entwickeln und abgeben
10-ffm.csv             # Eingabedaten für den Server
```

- Werkzeuge:

```
ruby              # Der Interpreter
emacs             # mit Ruby-Mode. Auch X-Emacs ok
scite, irb, ri    # Optionale Tools, wie üblich
```

- Vorlagen:
  `10-teilnehmerdaten.rb`

# Überblick zur Aufgabe

- Die Aufgabe: **Einfache Client/Server-Anwendung**

- Die Aufgabe besteht aus zwei Teilen:

  **A: Vorübung: Variante zur Lottozahlen-Ziehung**
  Entwicklung eines einfachen DRb-Clients, der Dienste eines zentralen Serverprozesses verwendet.

  **B: Ein Auskunfts-Server für Sportergebnisse, und ein Client-Programm zur Abfrage**
  Grundlage der C/S-Anwendung ist wieder DRb

# Die Aufgabe

## Teil A, Vorgaben:

- Auf dem Rechner "edi01" läuft ein DRb-Server auf Port 14328. Er stellt die folgenden Methoden bereit:

```
init( key )          --> nil
draw_all( key )      --> anArray (Inhalt: Fixnum-Obj.)
draw_one( key )      --> aFixnum
```

- Schreiben Sie einen DRb-Client namens "10-lottoclient.rb", der Verbindung mit dem DRb-Server aufnimmt.
- Rufen Sie zuerst die Methode init() auf, verwenden Sie Ihren Benutzernamen als Wert von "key". Beispiel:

```
key = "bsimp001"; my_drb_obj.init( key )
```

- "init" füllt die Lotto-Trommel für Ihre Ziehung. Die Anzahl Kugeln ist i.a. nicht 49, sondern hängt ab vom Wert von "key"!
- Ermitteln Sie nun mit den o.g. Server-Methoden, wie viele Kugeln der Server für Ihren Accountnamen in die Trommel füllt.
- **Tragen Sie das Ergebnis als Kommentar in Ihre Datei ein!**

# Die Aufgabe

## Teil A, Hinweise:

- Die Aufgabe versteht sich als **einfache Vorübung**.
- Sie lässt sich mit ca. 10 Quellcode-Zeilen lösen.
- Tests erfordern i.d.R., dass Ihr Client-Programm auf einem der Cluster-Rechner des Fachbereichs läuft.

- Testmöglichkeiten "*remote*" über das Internet können nicht garantiert werden. Beachten Sie hierzu die mündlichen Ergänzungen.
- *Remote login* auf "login1", dann **ssh** auf einen der Clusterrechner sollte stets möglich sein.
- Sollten Sie Probleme mit dem Server bemerken: Bitte *E-Mail* an den Dozenten.

# Teil B, Vorgaben

- Vervollständigen Sie "10-teilnehmerdaten.rb":
  - **Erarbeiten & verstehen Sie zunächst den Programmcode**
  - Die Methode "matches?" existiert in der Vorlage nur als Rumpf, der stets "true" ergibt. Implementieren Sie die Methode richtig!

    ```
    matches?( aHash )    --> true or false
    ```

    (Die Musterlösung benötigt 4 bzw. 9 Zeilen ohne bzw. mit "rescue")
  - Der an "matches?" übergebene Hash enthalte nur "key"-Werte, die den Namen der "getter" der Klasse "Teilnehmerdaten" entsprechen. Die Werte des Hashes seien reguläre Ausdrücke!
  - Beispiel:

```
# Suchkriterien definieren. Beispiel, statisch erzeugt:
# Alle aus Altersklasse MHK, deren Name mit "A" anfängt
bedingungen = { "name" => /^A/, "ak" => /MHK/ }

# Benutzung von "matches?" dann etwa so:
einTeilnehmerdatensatz.matches? bedingungen
```

# Teil B, Vorgaben

- Entwickeln Sie "10-sportsrv.rb":
  - Verwenden Sie das DRb-Beispiel aus der Vorlesung!
  - Arbeiten Sie nur mit "localhost" und Port 1432**9**
  - Binden Sie Klasse "Teilnehmerdaten" ein mittels:

```
load "10-teilnehmerdaten.rb"
```

  - Implementieren Sie die Klasse "SportServer" mit den Methoden

```
initialize()  # CSV-Zeilen per "gets" einlesen, in Array
              # aus "Teilnehmerdaten"-Objekten wandeln
query( cond ) # Soll dem Client ein Array aus Treffern
              # für die Suchbedingung in "cond" liefern.
```

  - Erzeugen Sie ein Objekt der Klasse "SportServer" und starten Sie den DRb-Server mit diesem Objekt als Grundlage, analog zum Vorlesungsbeispiel.
  - Zur Orientierung: Die Musterlösung enthält

```
4 Quellcode-Zeilen in init(),
5 Quellcode-Zeilen in query()  bei 29 Zeilen insgesamt.
```

# Teil B, Vorgaben

- Start des Servers:

```
$ 10-sportsrv.rb 10-ffm.csv & # Hintergrundprozess!
Angelegt: 7091 Teilnehmerdaten
```

- Entwickeln Sie "10-sportclient.rb"
  - Der Client besteht aus einer einfachen Benutzerschnittstelle, mit der ein Anwender ein Suchkriterium eingibt (in Form regulärer Ausdrücke!), der Anfrage beim Server, und der Ausgabe der gefundenen Treffer.
  - Tipp: Fragen Sie die regulären Ausdrücke als Strings ab (ohne die "/") und wandeln Sie die Strings dann in reguläre Ausdrücke um.
  - Auf Sortierung der Ausgabe verzichten wir hier.

- Zur Orientierung:
  - Die Musterlösung enthält gerade mal 19 Quellcode-Zeilen.
  - Zur Interaktion mit dem Server passen Sie das Vorlesungsbeispiel zum DRb-Client geeignet an.

# Teil B, Vorgaben

- Beispiel:

```
$ 10-sportclient.rb
name: ^Ar
ak: M20
zeit_netto: (Ctrl-D)
 9464 Arnold, Tobias                 M20 3:32:35  1766 ..
10466 Arrosson, Sven SWE             M20 3:43:23  2471 ..
10045 Armatys, Krysztof              M20 4:30:47  5745 ..
```

Alle Teilnehmer der Altersklasse „**M20**“, deren Namen mit „**Ar**“ beginnen

- Testaufgaben:
  - Suchen Sie die Teilnehmer aus Ihrem Heimatort bzw. Ort Ihrer Wahl
  - Wie viele Teilnehmer aus Schweden finden Sie?
  - Wie viele Teilnehmer der Altersklassen M30 und M35 benötigten zwischen 2:54:00 und 2:54:59 Stunden?
  - Wie viele Frauen blieben unter 3 Stunden?
- Anregung: Ein GUI für den Client ☺

# Teil B, Hinweise

- Anmerkungen
  - Die Vorgaben sind bewusst recht allgemein gehalten, insbesondere für den Client.
  - Die praktische Problemlösung (Suchanfrage bei einem Server) steht dabei im Vordergrund, nicht die Details des Lösungswegs.
  - Beachten Sie die Demo bei der Vorbesprechung.
  - Suchen Sie selbst nach einem geeigneten Weg im Rahmen der gestellten Randbedingungen!

  - Das hier geforderte Maß an Selbständigkeit entspricht dem Fortschritt im Laufe dieses Kurses und ist daher bei dieser vorletzten Praktikumsaufgabe besonders hoch.
  - Die Aufgabe bietet Gelegenheit zur Wiederholung vieler Vorlesungsteile!
  - Wegen des Schwierigkeitsgrades – nicht des Umfangs! - von Teil (B) ist dieser erst bis in 14 Tagen abzugeben; Teil (A) bitte normal, also nächste Woche abgeben!