

IPD
Universität Karlsruhe
AVG, 2. OG
Adenauerring 20a
76131 Karlsruhe

Prof. Dr. Gerhard Goos Zi. 201 Tel. 0721/608-4760
ggoos@ipd.info.uni-karlsruhe.de

Dr. Dirk Heuzeroth Zi. 232 Tel. 0721/608-4759
heuzer@ipd.info.uni-karlsruhe.de

Übungsblatt: 2

Ausgabe: 5.5. 2004

Besprechung: 12.5. 2004

Aufgabe: 2.1 (CVS)

1. Was ist der Unterschied zwischen *Version* und *Revision*?
2. Legen Sie eine zentrale Ablage an (z.B. in $\${HOME}/cvs$).
3. Konfigurieren Sie CVS so, daß es die soeben erstellte Ablage standardmäßig verwendet.
4. Erstellen Sie ein Verzeichnis `myproj` und legen Sie in diesem Verzeichnis zwei Dateien (`f1` und `f2`) mit verschiedenem Inhalt an. Importieren Sie dieses Projekt in die zuvor erstellte zentrale Ablage.
5. Beschaffen Sie sich eine Arbeitskopie aus der zentralen Ablage.
6. Modifizieren Sie die Dateien `f1` und `f2` nach Belieben. Übertragen Sie die Dateien dabei zwischendurch gelegentlich in die zentrale Ablage, also z.B.: Modifizieren von `f1`, übertragen von `f1`; modifizieren von `f1`, übertragen von `f1`; modifizieren von `f2`, übertragen von `f2`.
7. Lassen Sie sich den Status der Dateien ihres Projektes sowie deren Veränderungen zwischen den verschiedenen Revisionen anzeigen.
8. Fügen Sie ihrem Projekt eine neue Datei `f3` hinzu.

Aufgabe: 2.2 (Klammern und Records)

Ein Speichermedium s erlaubt nur blockweisen Zugriff mit konstanter Blockgröße. Es enthält Daten, die durch geschachtelte Klammerpaare strukturiert sind. Klammerpaare können sich über mehrere Records erstrecken.

- Bestimmen Sie den Anfang des n -ten Klammerpaars auf oberster Ebene mit Speicherbedarf $O(1)$.
- Setzen sie Ihre Lösung in Java um.

Aufgabe: 2.3 (Pipes)

UNIX Pipes koppeln zwei Programme über einen Bytestrom. Das Pipe-Modell kann den Kontrollfluß in vielen Fällen vereinfachen.

- Lösen Sie Aufgabe 2 mit UNIX Pipes in Java. (Hinweis: Ein Programm liest Records aus einer Datei und gibt einen Bytestrom auf die Standardausgabe. Ein zweites Programm liest den Bytestrom von der Standardeingabe und bearbeitet die Klammerpaare.)

Aufgabe: 2.4 (Koroutinen)

Koroutinen sind eine Form kooperativen Multitaskings. Jede Koroutine ist ein Ausführungsfaden mit eigenem Keller. Koroutinen werden einfädig ausgeführt. D.h., ein Koroutinenaufruf suspendiert die Ausführung der aktuellen Koroutine und setzt die aufgerufene Koroutine fort.

Programmversion nach Jackson überführt Koroutinenaufrufe nach folgendem Schema in Methodenaufrufe:

1. Zeichne eine Koroutine als Aktivitätsträger aus.
2. Zerteile jede Koroutine an den Aufrufstellen in Methoden.

3. Verschiebe den Kellerzustand der nicht-aktiven Routinen in Klassenvariable.
4. Ersetze Koroutinenaufrufe durch Methodenaufrufe.
(Aufrufe an den Aktivitätsträger werden zu einer Methodenrückkehr)

Oft ist die Umsetzung einer gegebenen Aufgabe mit Koroutinen einfacher als eine sequentielle Lösung. Letztere kann mechanisch durch Programminversion erreicht werden.

- Lösen Sie Aufgabe 2 mit Koroutinen. Nehmen Sie dabei Koroutinen-Semantik für Methoden in Java an.
- Transformieren Sie Ihre Lösung mit Programminversion in ein sequentielles Programm.

Aufgabe: 2.5 (*Initialisierung*)

Geben Sie ein einfaches Erzeuger-Verbraucher-System in Java an, bei dem die Erzeugerklass eine Methode **produce** definiert, die direkt die Methode **consume** der Verbraucherklasse aufruft und dabei ein Datum als Argument mitgibt. Woher kennt die Erzeugerklass die Verbraucherklasse, genauer woher kennen Erzeugerobjekte ihre zugehörigen Verbraucherobjekte?