

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: 1

Ausgabe: 21.4. 2004

Besprechung: 28.4. 2004

Aufgabe: 1.1 (*Schnittstellen und Wiederverwendung*)

Warum reicht es für Wiederverwendung noch nicht aus, wenn eine Komponente eine saubere Schnittstelle besitzt?

Aufgabe: 1.2 (*Bibliotheken als Komponenten*)

Verschaffen Sie sich eine klassifizierende Übersicht über

- das Java Development Kit (JDK): <http://java.sun.com>
- die Standard Template Library (STL): <http://www.sgi.com/tech/stl/>
- das .NET Rahmensystem: http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpref/html/cpref_start.asp

Aufgabe: 1.3 (*Begriffe, Bezüge und Abgrenzungen*)

Definieren Sie folgende Begriffe:

- Objekt
- Klasse (generische K., abstrakte K.)
- Modul
- Typ

In welchem Bezug stehen sie zu:

- Exemplar einer Komponente
- Bilden von Komponenten
- Ausfalten von (generischen) Komponenten

Diskutieren Sie:

- Ist eine generische Klasse eine Klasse?
- Welche Eigenschaften objektorientierter Systeme sind für Komponentensysteme wünschenswert?

Aufgabe: 1.4 (*Generizität*)

Gegeben sei ein Graph mit Adjazenzmatrix $A = (a_{ij})$ mit folgenden Problemen für $i \neq j$:

- Existiert ein Pfad von i nach j ?
- Länge des kürzesten Pfades von i nach j ?
- Maximaler Fluß von i nach j ?
(die Kapazitäten sind durch a_{ij} gegeben)

Der generische FLOYD-WARSHALL-Algorithmus löst alle drei Probleme:

$$\begin{aligned} \forall k \in [1, n] : a_{kk} &:= c \\ \forall k \in [1, n] : \\ \quad \forall i \in [1, n] : \\ \quad \quad \forall j \in [1, n] : \\ \quad \quad \quad a_{ij} &:= a_{ij} \sigma(a_{ik} \tau a_{kj}) \end{aligned}$$

- Bestimmen Sie die Konstante c und die Operationen σ, τ für alle Probleme.
- Welche algebraischen Eigenschaften müssen σ, τ aufweisen?
- Setzen Sie Ihre Lösungen als Schablonen (*templates*) in C++ um.
- Können Sie das Beispiel auch in Pascal oder C lösen?

Aufgabe: 1.5 (Mißbrauch von Generizität)

C++ erlaubt das Überladen von Schablonen (*template specialization, full or partial*). Analog zum Überladen von Methoden wählt dieser Mechanismus anhand der zur Übersetzungszeit bekannten Parametertypen eine von mehreren Varianten aus. Schreibweise:

```
// Template
template<T1 t1, T2 t2, ...> class U {
    ...
};

// Partial template specialization for t1=v1
template<T2 t2, ...> class U<v1, t2, ...> {
    ...
};
```

- Setzen sie Fallunterscheidung als Schablone in C++ um. Hinweis: Atomare Datentypen und ihre Werte sind als Parameter und Spezialisierungen in Schablonen zulässig.

Eine while-mächtige Sprache ist Turing-mächtig. Eine Sprache mit Fallunterscheidung und Rekursion ist while-mächtig.

- Wie kann man Rekursion mit Schablonen in C++ umsetzen?
- Erstellen Sie eine Schablone, die Fakultäten berechnet ($f(0) = 1, f(n) = n * f(n - 1)$).

Turing-mächtig ist nicht gleichbedeutend mit leicht verwendbar. Berichten Sie über Probleme bei der Umsetzung von

- Komplexen Datenstrukturen (Listen, Bäume)
- Seiteneffekten

Aufgabe: 1.6 (Merkmalsmodellierung)

Erstellen Sie ein Merkmalsdiagramm für eine Automobil-Produktlinie. Beschränken Sie sich dabei auf die wesentlichen Merkmale.

Aufgabe: 1.7 (Abgrenzung Produktlinien und -familien)

Erklären Sie die Begriffe Programmfamilie und Produktlinie. Worin unterscheiden sich diese?