

Tagesprogramm

Ausgewählte Aspekte zu Programmierparadigmen

Basiskonzepte der objektorientierten Programmierung

Was Paradigmen ausmacht

Berechnungsmodell

Umgang mit **Querverbindungen** durch Seiteneffekte

- Seiteneffekte verbieten (funktional)
- Querverbindungen sichtbar machen (objektorientiert)

First-Class-Entities

- hoher Aufwand, lohnt sich nur für wichtigste Konzepte

Modularisierungseinheiten

Modul

Objekt

Klasse

Komponente

Namensraum

Parametrisierung

dynamisches Befüllen von Lücken (zur Laufzeit)

Konstruktor

Initialisierungsmethode

zentrale Ablage

statisches Befüllen von Lücken

Generizität

Annotationen

Aspekte

Ersetzbarkeit

A durch B ersetzbar wenn B überall verwendbar wo A erwartet

Schnittstellen von A und B spezifizierbar durch

- Signatur

- Abstraktion

- Zusicherungen

- Protokolle

Aufgabe: Ersetzbarkeit und Parametrisierung

Warum ist Ersetzbarkeit keine Form der Parametrisierung?

- A: Weil Ersetzbarkeit keine Wiederverwendung unterstützt.
- B: Weil Ersetzbarkeit keine Lücken füllt.
- C: Weil für Ersetzbarkeit alle Details von Anfang an definiert sein müssen.
- D: Weil Ersetzbarkeit nicht mit Generizität kompatibel ist.

Abstraktion und Typen

struktureller Typ → nur Signatur

nominaler Typ → Typname ermöglicht Abstraktion

abstrakter Datentyp (ADT)

= nominale Schnittstelle einer Modularisierungseinheit

Objekt aus realer Welt }
Softwareobjekt } gemeinsamer Name

Denken in Konzepten

Untertypen

Ein Typ U ist Untertyp eines Typs T wenn jede Instanz von U überall verwendbar ist wo eine Instanz von T erwartet wird.

für strukturelle Typen eindeutig

für ADT in der Verantwortung der Programmierer

Objekt

kapselt Variablen und Methoden zu Einheit

Eigenschaften:

Identität

Zustand

Verhalten

Softwareobjekt (ADT) simuliert und abstrahiert reales Objekt

Klasse

beschreibt Struktur der Objekte

Konstruktoren zur Initialisierung der Objekte

gleiche Klasse → gleiche Schnittstellen und gleiches Verhalten

jedes Objekt ist Instanz genau einer Klasse

Untertypen und Vererbung

Untertypen bedingen Ersetzbarkeit

- Variable hat **deklarierten** und **dynamischen Typ**
- dynamisches Binden

Vererbung = Übernehmen von Code aus Oberklasse

Praxis: Vererbung als Hilfsmittel für Untertypen

Erfolg in der OO-Programmierung

OOP = viele Möglichkeiten zur **Faktorisierung**

- erleichtert gezielten Einsatz von Erfahrung
- ermöglicht Wiederverwendung durch Ersetzbarkeit
- überfordert Anfänger

OOP gut für große, langlebige Programme

OOP schlecht für komplexe Algorithmen

Verantwortlichkeiten einer Klasse

definiert durch drei w-Ausdrücke, „ich“ ist Objekt der Klasse:

was ich weiß	(Zustand der Objekte)
was ich mache	(Verhalten der Objekte)
wen ich kenne	(sichtbare Objekte, Klassen)

wer Klasse entwickelt ist zuständig für Änderungen in den Verantwortlichkeiten

Klassen-Zusammenhalt

Klassen-Zusammenhalt (class cohesion)

= Grad der Beziehungen zwischen Verantwortlichkeiten der Klasse

hoch, wenn

Variablen und Methoden eng zusammenarbeiten
und durch Klassenname gut beschrieben

Klassen-Zusammenhalt soll hoch sein

- Hinweis auf gute Faktorisierung
- verringert Wahrscheinlichkeit für nötige Änderungen

Objekt-Kopplung

Objekt-Kopplung (object coupling)

= Abhängigkeit der Objekte voneinander

stark, wenn

 viele sichtbare Methoden und Variablen

 viele Nachrichten im laufenden System

 viele Parameter in Methoden

Objekt-Kopplung soll schwach sein

→ Hinweis auf gute Kapselung

→ weniger unnötige Beeinflussung bei Änderungen