

Was ist Objekt-Orientierung?

ACM / IEEE CS Curriculum (v2008)

- Topics
 - Object-Oriented Design
 - Encapsulation and Information Hiding
 - PF: Separation of behavior and implementation
 - Classes and subclasses
 - Inheritance (overriding, dynamic dispatch)
 - Polymorphism (subtype polymorphism vs inheritance)
 - PL: class hierarchies
 - PL: collection classes and iteration protocols
 - PL: internal representation of objects and method tables
-
-

Was ist Objekt-orientierung?

Kritik am Curriculum

- Punkte auf unterschiedlichem Abstraktionsniveau
 - “object-oriented design” vs “method tables”
 - Object-oriented design
 - Nicht weiter spezifiziert, wo ist Analyse?
 - Datenmodellierung?
 - Responsibilities? RPD, CRC
 - Rekursion nicht erwähnt – unwichtig?
 - Keine Fixierung: by concept vs by example?
 - Typ wichtiger als Klasse
 - Typ = Methoden, die das Objekt versteht
 - OO ungleich Java (“overloading vs overriding”)
 - High-level Konzept / low-level Erklärung
-
-

OO Konzepte

- Abstrakter Datentyp
 - Verkapselung von Zustand und Verhalten
 - generell unerlässliches Konzept
 - Erklärung nicht an OO gebunden
 - Subtyppolymorphie (Interface)
 - Abstraktion über Verhalten
 - Kollaborationsgedanke (von Objekten)
 - Protokoll statt Algorithmus
 - Vererbung (Klassenhierarchie)
 - Wichtig zu verstehen
 - Wichtig zu wissen, wann nicht angebracht (oft)
-
-

Probleme mit Objekt-Orientierung

- Problem: Gute Beispiele für OO?
 - Viele Lehrbuchbsp sind “geerbt” von imperativen Vorgängern
 - Konzeptvermittlung durch Bsp oder durch Prozess
- OO muss nicht an einer PS festgemacht werden
- Natürliche Modellierung durch Objekte?
 - aber nicht nach der Natur
 - Künstliche virtuelle Entitäten

Probleme II

- Die korrekte Anwendung der OO-Konzepte muss erlernt werden, falsche Anwendung führt zu Problemen bei Wiederverwendung, Wartung
- Vorgehensweise: top-down vs bottom-up?
- Kochrezepte für Probleminstanzen
 - Unterhalb von Designpatterns
 - Herangehensweise für Daten- /Problemanalyse

Zielsetzungen

- Klientel ist wichtig
 - Hauptfach vs Nebenfach
 - Ingenieure vs Kerninformatiker
 - Brauchen Ing nur Programmieren-light?
 - Weniger Modellierung, mehr GUI-Programmierung
 - Bsp:
 - Sprachen mit/ohne Klassen
 - Designmuster / -rezepte?
 - Beherrschung der Konzepte
-
-

Diskussionspunkte

- Computational model
 - High-level / low-level
 - Protokolle vs Algorithmen
 - Vorteil OO vs ADT bei Anfängern
 - CSCC: “oo is difficult”
 - Früh und lange unterrichten vs erst Grundlagen
 - Ausbeutung der OO / funktional Dualität
 - Typische Schwierigkeiten (s.o.)
 - Endrekursion fundamental?
 - Stellenwert der Vererbung
 - Welche OO Element müssen die Studenten beherrschen?
-
-

Bemerkungen

- Rekursive Datenstrukturen nicht Core-Topic
 - [core] recursion für divide-and-conquer
 - [elective] recursion on lists
 - [core] Iteration über Collections