Seminar Objektorientiertes Programmieren
Theorie, Methodik und Anwendung

N. Schmeißer*
Forschungszentrum Rossendorf e.V.,
Abt. Kommunikation und Datenverarbeitung

5. Dezember 1997

Inhaltsverzeichnis

*Telefon: (+49 351) 260/2207, E-Mail: nils@schmeisser.com,
URL: http://www.fz-rossendorf.de/FVTK/MITARB/schmei

1 Motivation

- natürliche Artikulation
- "sauberes Programmieren"
- Vermeidung von Fehlern
- Wiederverwendbarkeit
- imperativer Programmierstil
- strukturiertes Programmieren (funktionale PS)
- modulares Programmieren
- OO Programmieren

2 Begriffe

Objekt : eigenständiges Individuum mit charakteristischen Eigenschaften, inneren Zuständen und Methoden

Attribut : Variable, die einem Objekt zugeordnet ist und dessen inneren Zustand beschreibt

Methode : semantische Einheit, die auf ein Objekt einwirkt

Klasse : Zusammenfassung aller Objekte mit gleichen Eigenschaften, besitzt eine Methode zur Erzeugung von Instanzen

Instanz : Realisierung eines Objektes, d.h. Variable vom Objekt-Typ mit zugeordnetem Speicherplatz

Nachricht : Datum, das an eine Instanz gesendet, von einer Methode verarbeitet wird; Instanzen interagieren durch Nachrichtenaustausch

Vererbung : Weitergabe von Eigenschaften an eine neue Klasse

Polymorhpie : ein Konzept (Methode, Funktion, Operator) kann viele verschiedene Formen annehmen

Abstrakter Datentyp : Hilfsmittel zur Beschreibung von Objekten und ihren Klassen

3 Abstrakter Datentyp

Typ : (algebraische) Struktur mit Operationen

Sorte : Kombination von Datentypen
Notation: $\omega = i_1 : \tau_1 + ... + i_n : \tau_n$

abstrakter Datentyp : Struktur aus einer Menge von Sorten, (Ω), Operatoren über Sorten ($\delta \in \Delta$), deren Spezifikation ($\Gamma_{\delta_n} : \Omega^n \rightarrow \Omega$) und einer Semantik ($B : \Omega \rightarrow S$); ein ADT ist eine mehrsortige $\Omega\Delta\Gamma$-Algebra

universell polymorphe Abbildung : Operatoren δ und δ' in zwei ADT's $\mathcal{A}$ und $\mathcal{A}'$ mit $\Omega \subseteq \Omega'$ und einer Sortenkonvertierung $\kappa : \Omega' \rightarrow \Omega$, für die
$\kappa \circ \delta'|_{\Omega'|kern(\kappa)} \equiv \delta$ gilt

abgeleiteter ADT : $\mathcal{A} \leq \mathcal{A}'$; für jeden Operator aus Δ gibt es einen polymorphen Operator in Δ'

implizite Typkonversion : es existiert eine Konversionsfunktion zwischen zwei Typen resp. Sorten (ad-hoc Polymorphismus)

Klasse : ADT zusammen mit Methoden zur Erzeugung und Zerstörung von Instanzen, Ein- und Ausgabemenge

Referenzregel : Referenz $\&\mathcal{I}$ = Verweis auf eine Instanz; $\mathcal{I}.x = (\&\mathcal{I}) \rightarrow x$

4 Ableitungsbegriff

Ableitung : Klasse B, für die $ADT(A) \leq ADT(B)$ und $A.M \subseteq B.M$ gilt; $B :: A$

einfache : $B :: A$

mehrfache : $B :: A_1$ und $B :: A_2$

zyklische : $B :: A$ und $A :: B$

direkt abgeleitete Klasse :
$B ::_d A$ gdw. $\nexists C : B :: C \wedge C :: A$

Basisklasse : Klasse B, für die es keine Klasse A mit $B :: A$ gibt

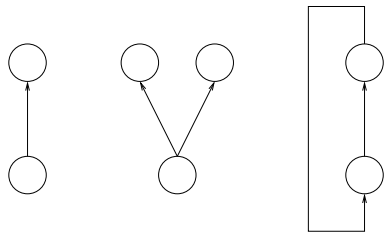
abstrakte : es gibt keine Methode zur Erzeugung von Instanzen

virtuelle : bei Mehrfachvererbung werden die Eigenschaften nur höchstens einmal weitergegeben

Ableitungskette : Tupel aus Klassen $C_0 :: ... :: C_n$

:: ist eine transitive Relation.

Ableitungsgraph : Graph mit Knoten aus Klassen und Kanten zwischen allen Knoten, für die $B ::_d A$ gilt



5 Polymorphie und späte Bindung

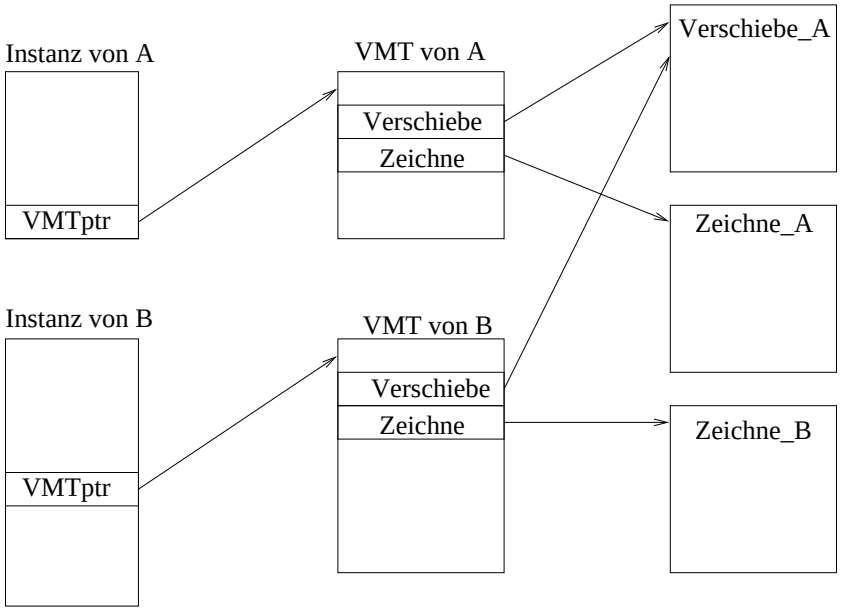
erweiterte Referenzregel :

sei $B :: A$, dann ist $\mathcal{R}_A = \&_A\mathcal{I}_B$ zulässig;
$\mathcal{R}_A{\rightarrow}$ liefert die Einschränkung auf A

virtuelle Methode :

polymorphe Methode m in $B :: A$ für die
$(\&_A\mathcal{I}_A){\rightarrow}m = A.m \wedge (\&_A\mathcal{I}_B){\rightarrow}m = B.m$ gilt

Funktionsweise der späten Bindung (VMT)



6 Instanzierung

1. Speicherplatz für die Attribute bereitstellen
2. Konstruktion der Objekte der Elternklassen
3. Konstruktion des Objektes der Klasse
4. Konstruktor der Klasse rufen

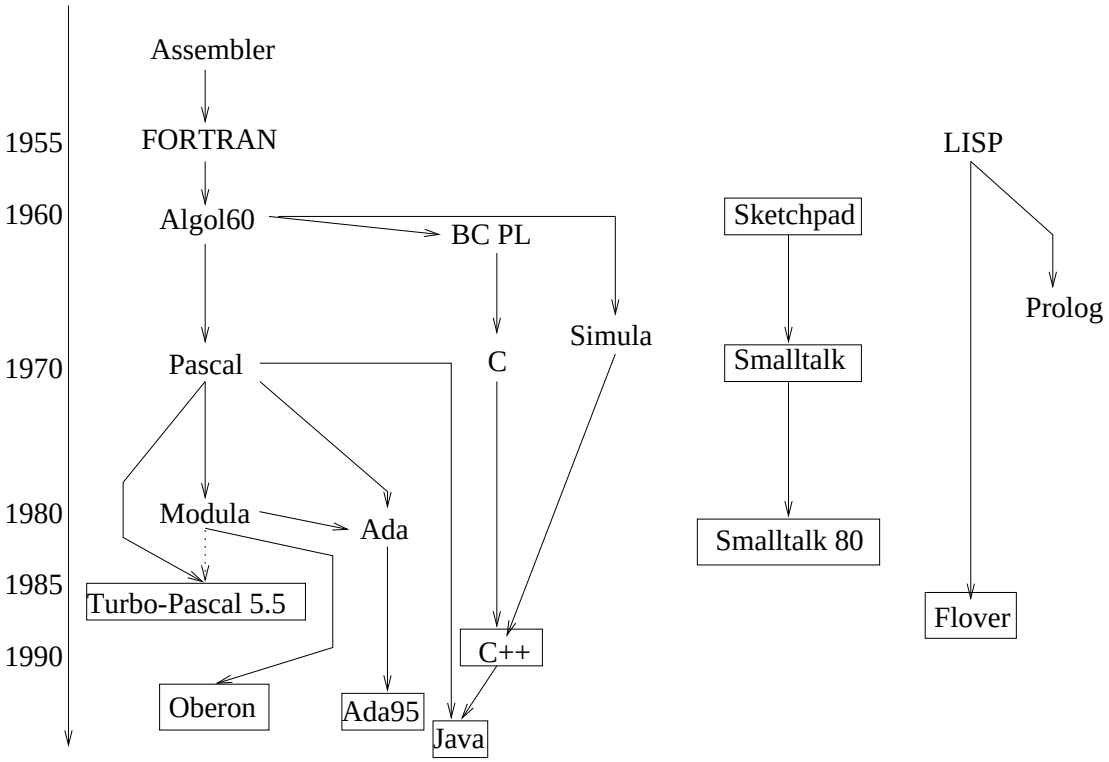
statische Instanz : Instanz wird bei Betreten des Gültigkeitsbereiches erzeugt und bei Verlassen zerstört

dynamische Instanz : Instanz wird durch expliziten Konstruktorruf erzeugt bzw. Destruktorruf zerstört

garbage-collection : nicht mehr referenzierte Instanzen werden automatisch zerstört

7 Geschichte der Programmiersprachen

- imperative Programmiersprachen (BASIC, C, FORTRAN)
- funktionale/applikative PS (LISP)
- prädikative PS (PROLOG)
- objektorientierte PS (Smalltalk, Ada, C++)



8 OO-Sprachen

1. Turbo-Pascal, Delphi
2. Oberon
3. Smalltalk-80
4. C++
5. Java
6. Ada 95
7. Eiffel

8. CLOS (Common Lisp)
9. Modula-3
10. Objective-C
11. Python
12. Sather
13. Simula
14. Tcl/Tk
15. VBScript
16. Visual Basic

9 Datenbanken

RDBM : relationales Datenbankmodell, dem RDBM liegt ein Relationsschema für die Beziehungen zwischen den Datensätzen zugrunde
(SQL-Datenbanken (DB2, SQL/DS), Oracle, INGRES), DBase IV, Open Access)

SDM : semantisches Datenbankmodell, diesem liegen Modelle wie Entity-Relationship, prädikatenlogische Modelle, mengentheoretische Modell, statische Modelle mit semantischer Hierarchie etc. zugrunde
(IFO, Iris, TAXIS)

OODM : objektorientiertes Datenbankmodell, Verwendung des OO- Paradigmas zur Realisierung von RDBM, SDM (GemStone, ONTOS/VBASE, O_2, POET, POSTGRES)

SQL : Abfragesprache und gleichzeitig Sprache zur Definition von Relationsschemata

ODBC : Open Database Connectivity Technology; Datenbank-Programmierschnittstelle

DBMS : Datenbase-Managementsystem

10 GUI's und Betriebssysteme

1. MS-Windows 3.1, MS-Windows NT
2. Oberon
3. OS/2 Presentation Manager
4. Mac OS

IDL : Interface Description Language

OLE : Object Linking and Embedding

- OLE-1: Client/Server Architektur
- OLE-2: Objekt/Instanz ist direkt mit Server verknüpft (In-PlaceActivation)
- drag-and-drop

OMG : Object Management Group

11 Verteiltes OOP

CORBA : Common Object Request Broker Architecture; Client-Server Architektur, bei der die Implementation von Methoden auf einem Server erfolgt, währen eine Klient (Instanz) über eine genormte Schnittstelle (in IDL formuliert) zugreift

DCOM : Kürzel von Microsoft für Distributed Component Architecture (das ist verteiltes OLE2), das schließt OLE und Microsoft Spezifika wie ActiveX ein, entsprechend verwendet MS eine MIDL - Microsoft Interface Description Language

ActiveX/OLE : Zitat MS: "ActiveX is a set of technologies from Microsoft that enables interactive content for the World Wide Web. With ActiveX, Web sites come alive ... ActiveX provides the glue that ties together a wide assortment of technology building blocks to enable these "active" Web sites."

Java Beans/RMI : Zitat Sun: "Java Beans is an architecture and platform neutral API for creating and using dynamic Java components."
RMI - Remote Method Invocation - Schema zum Aufruf von Serverfunktionalität

12 Aufgaben

12.1 Axiomsystem der euklidischen Geometrie

Formulieren Sie Elemente des Axiomsystems der euklidischen Geometrie in der OO Terminologie.
Ziel: Verständnis der Begriffe des OOP

12.2 Algebraische Strukturen

Formulieren Sie ein System von Klassen mit denen Menge und algebraische Strukturen wie Gruppe, Ringe und Körper erfaßt werden können.
Ziel: Vererbung, Instanzierung, Wahl einer geeigneten OO Sprache

12.3 Vektoren und Matrizen

Erweitern Sie das obige System um die Strukturen Matrix und Vektorraum.
Ziel: Templates, Vererbung, Wiederverwendung von Code

12.4 Taylorarithmetik

Generieren Sie eine Klasse, mit der Sie Taylorarithmetik ausführen können.
Ziel: Operatorüberladung, Funktionsüberladung

12.5 Formelmanipulation

Entwerfen Sie ein System von Klassen zur Formelmanipulation.
Ziel: abstrakte Basisklasse, virtuelle Methoden, Bäume, Traversierung

12.6 attributierte Graphen

Ein Bild als Wort eine Sprache enthält eine gewisse Information. Mit Hilfe attributierter Graphen kann nach einer syntaktischen Analyse die Semantik des Bildes relativ einfach bestimmt werden und daraus kann dann eine Aktion generiert werden z.B. ein Satz, der den Inhalt beschreibt. Entwerfen Sie eine Klassenhierarchie zum Umgang mit attributierten Graphen.
Ziel: virtuelle Methode, Verknüpfung verschiedener Objekte

12.7 grafisches Basissystem

Entwerfen Sie eine kleine objektorientierte grafische Nutzerschnittstelle.
Ziel: abstrakte Basisklassen, virtuelle Methoden, dynamische Instanzierung, Nachrichtenparadigma

12.8 minimales Datenbanksystem

Schreiben Sie ein kleines objektorientiertes Datenbanksystem.
Ziel: effiziente Verwaltung einer großen Anzahl Instanzen, Umgang mit Graphen, Kombination OOP mit herkömmlicher Programmierung