

Objektorientierte Programmierung

(Programmieren 2)
Teil OOA / OOP

Einführung in die objektorientierte Software-Entwicklung

Prof. Dr. Bernd Ruhland

Objektorientierte Software-Entwicklung

Die *funktions-* oder *datenorientierte* Software-Entwicklung basieren jeweils nur auf einen einseitigen und somit unvollständigen Blick auf das System.

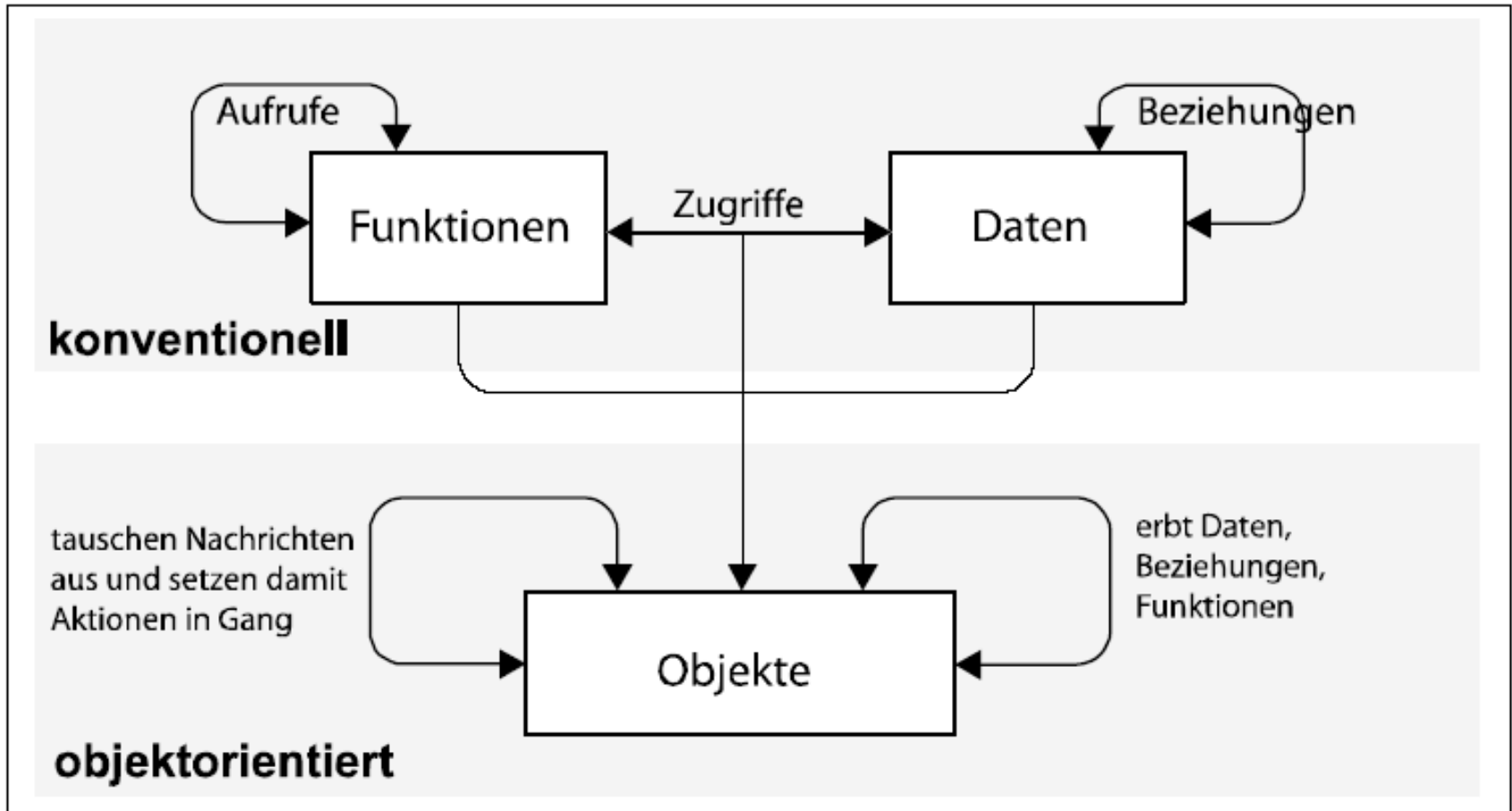
Bei der *funktionsorientierten* Vorgehensweise werden die für ein zu entwickelndes System relevanten Funktionen oder Tätigkeiten festgelegt. Davon ausgehend werden die erforderlichen Datenbestände bestimmt. Die Struktur der Datenbestände orientiert sich an den speziellen Funktionen.

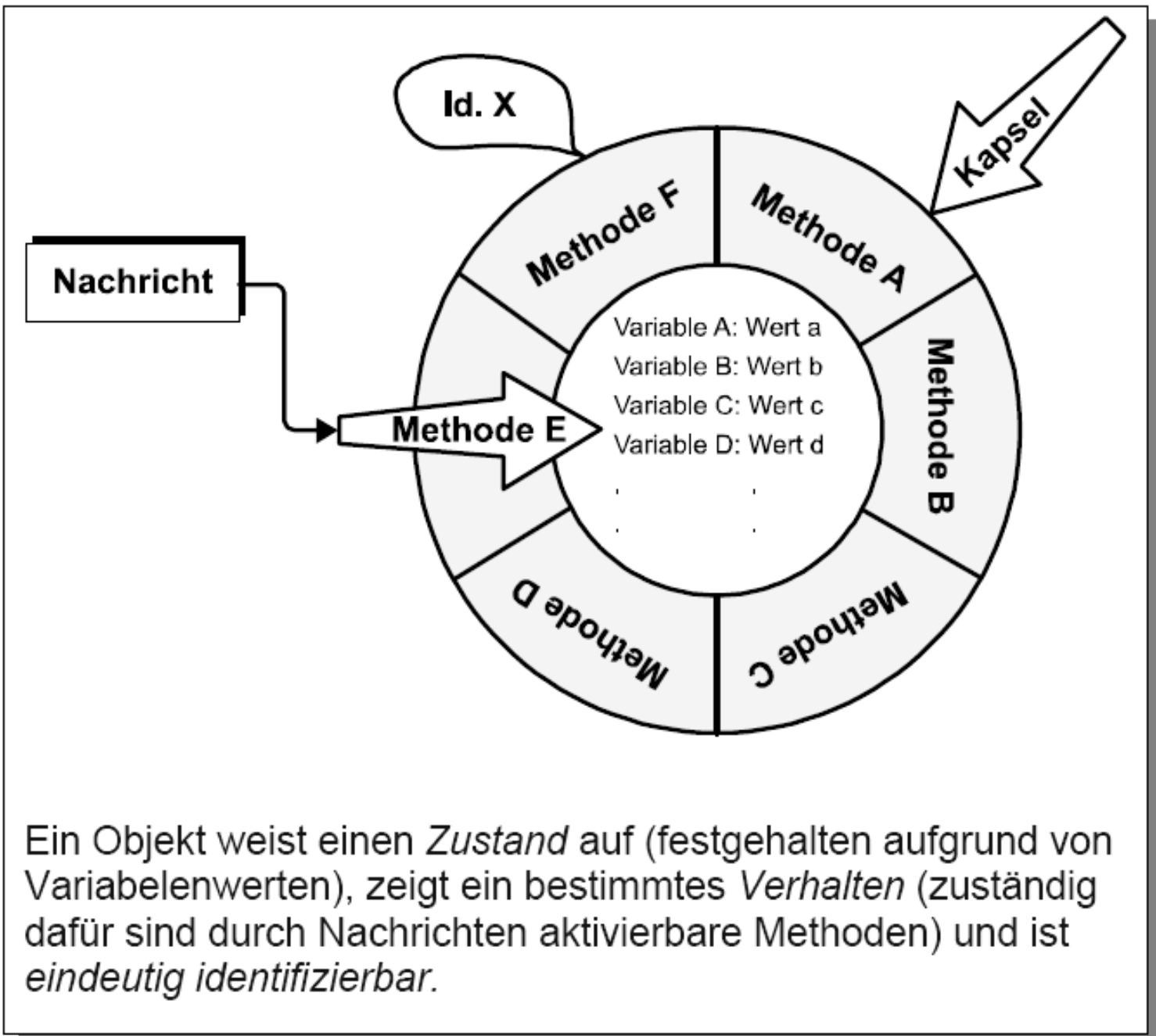
Informationstechnologisches Objekt

Ein informationstechnologisches Objekt (im folgenden nur noch als Objekt bezeichnet) enthält abgeschirmte Daten sowie Methoden zu deren Verarbeitung. Die Daten bestimmen den *Status* (Zustand) eines Objekts, die Methoden das *Verhalten*, also die Reaktion auf einen äußeren Einfluss.

Ein Objekt verkapselt sowohl seine Daten (Variablen, Attribute) wie auch seine Methoden und verkehrt mit der Außenwelt über eine wohldefinierte Schnittstelle

Unterschied zwischen prozeduraler und objektorientierter Programmierung:





Ein Objekt weist einen *Zustand* auf (festgehalten aufgrund von Variablenwerten), zeigt ein bestimmtes *Verhalten* (zuständig dafür sind durch Nachrichten aktivierbare Methoden) und ist *eindeutig identifizierbar*.

Fragen zum Stoff

1. Was ist ein informationstechnisches Objekt?
2. Womit ist der Zustand eines Objekts festzuhalten ?
3. Was bestimmt das Verhalten eines Objekts ?
4. Erklären Sie den Begriff „Datenkapsel“
5. Was ist das Geheimnisprinzip (information hiding) ?
6. Was bewirkt das Geheimnisprinzip ?
7. Wie erfolgt die Kommunikation zwischen Objekten ?

Statische Objektmodellierung

Objekte repräsentieren individuelle und identifizierbare *Exemplare* von Dingen, Personen oder Begriffen der realen oder der Vorstellungswelt.

Bei der Realitätsmodellierung *abstrahiert* man und arbeitet mit *Konstrukten*, die stellvertretend für viele Einzelfälle sind.

Diese Konstrukte werden auch als *Klassen* bezeichnet.

Bei der *Entitäten-Beziehungs-Modellierung* (ER-Modell) verwendet man bei der Realitätsmodellierung Vererbungs-, Beziehungs- und Aggregationsstrukturen.

Klassen

Eine Klasse enthält *Objekte des gleichen Typs*.

Es gibt zwei Typen von Klassen:

- *konkrete* (d. h. instanzierbare) Klassen enthalten im Normalfall Objekte.
- *abstrakte* (d. h. nicht instanzierbare) Klassen enthalten niemals Objekte. Der Zweck einer abstrakten Klasse liegt darin, Variable (Attribute) und Methoden an konkrete Klassen zu vererben.

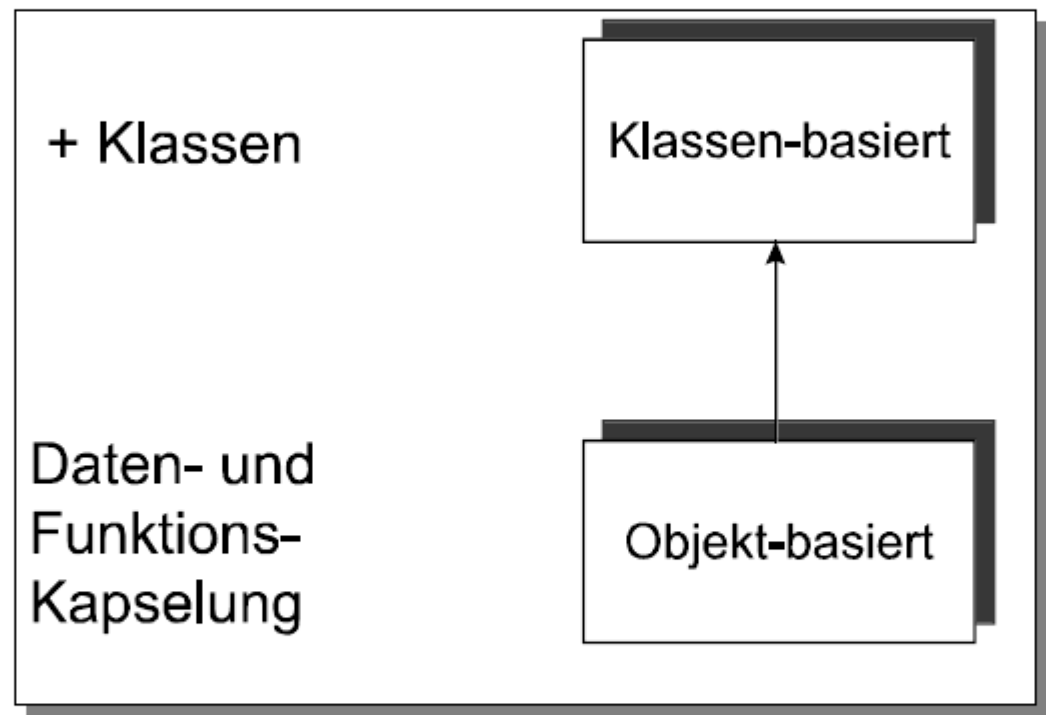
Jedem Objekt wird bei der Generierung automatisch ein eindeutiges Identifikationsmerkmal zugeordnet.

Objektorientierter Entwicklungszyklus

- Analyse:
 - Erfassung der Objekte der realen Welt
 - Mit den Begriffen der Fachwelt (Domäne)
 - Modellierung der Zusammenhänge
 - ➔ „Analysemodell“ oder „Begriffsmodell“ oder „Domänenmodell“
- Design:
 - Umsetzung Analysemodell in das „Designmodell“ aus Klassenkonzept („Klassenmodell“) und System-Komponenten
- Implementierung:
 - Programmierung (Attribute und Methoden)
- Test:
 - Klassentest (Unit Test)
 - Integrationstest

Ein methodisches Vorgehen bezeichnet man gemäß „Wegner“[★] als klassenbasiertes Vorgehen falls es

- die Daten- und Funktionskapselung nutzt,
- mit Klassen arbeitet u. damit die Möglichkeit zur Abstraktion verwendet.



★ Peter Wegner 1978

Ordnungsstrukturen

Die Komplexität eines Problems kann mittels Ordnungsstrukturen reduziert werden.

Klassifikation: Objekte können nach einem bestimmten Kriterium geordnet werden

Aggregation: Objekte können Bestandteile anderer Objekte sein; Objekte setzen sich aus anderen Objekten zusammen

Generalisierung

Spezialisierung

Säugetier

Löwe

Pferd

Büffel

Wal

Delphin

Wassertier

Fisch

Forelle

Aal

Seeteufel

Vogel

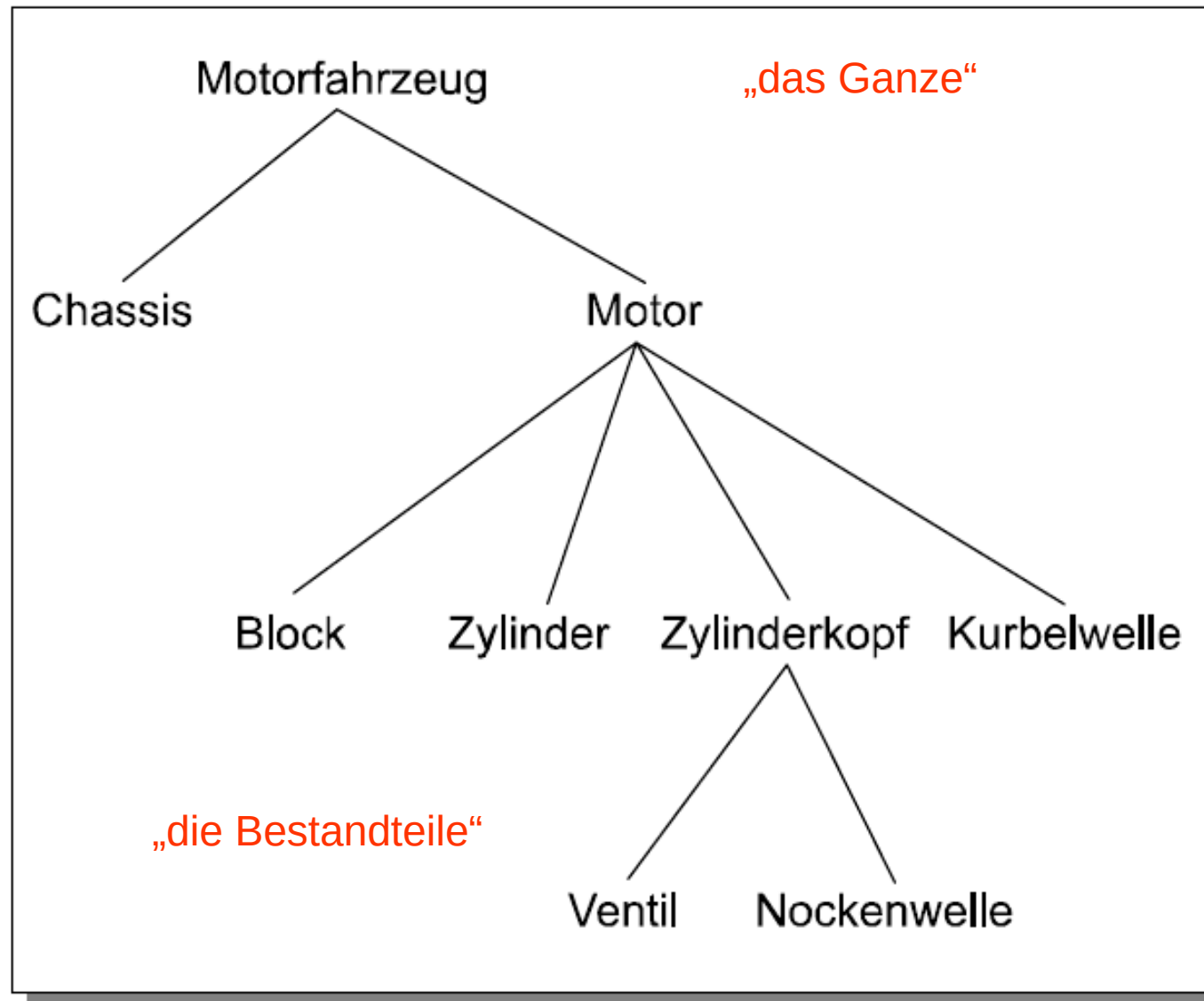
fliegend

Specht

Adler

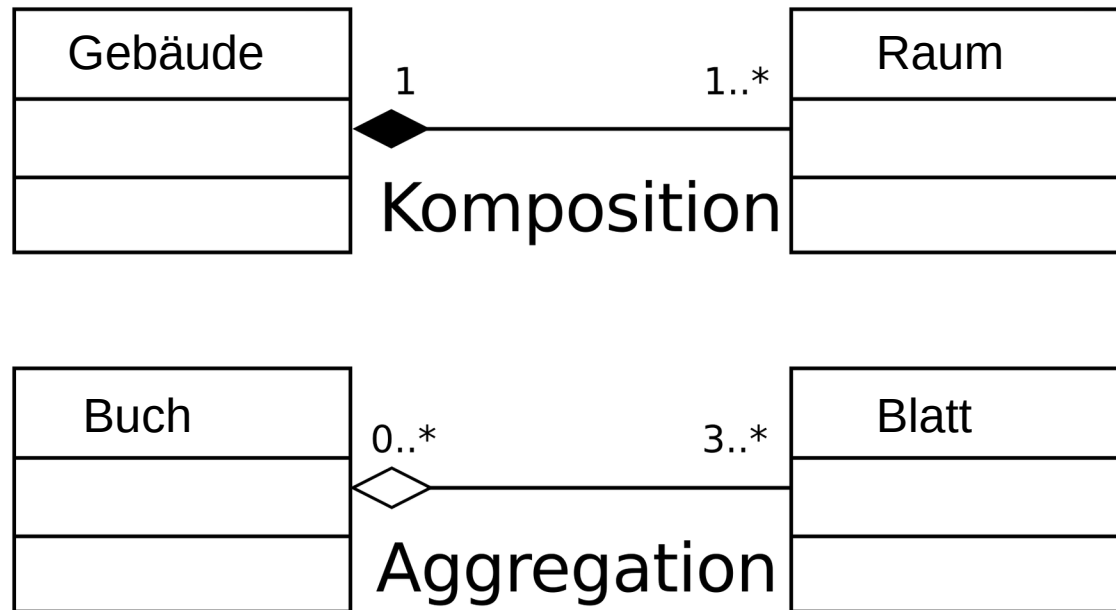
Pinguin

willkürliche
Klassifikation
als Beispiel



Aggregationsstrukturen

Mit einer Aggregationsstruktur wird die Zusammensetzung eines Ganzen aus mehreren Teilen aufgezeigt.



wikipedia

Die Komposition ist die Spezialform der Aggregation, bei der die Komponente genau einem Ganzen zugeordnet ist (und mit Zerstörung des Ganzen mit zerstört wird).

Fragen zum Stoff

Anstelle von Objekten spricht man auch von einer Klasse:

- A) Kapseln
- B) Abstraktionen
- C) Instanzen
- D) Methoden

Fragen zum Stoff

Wodurch sind Objekte gleichen Typs charakterisiert ?

- A) Sie sind Instanzen derselben Klasse
- B) Sie haben dieselben Objektnamen
- C) Sie bilden eine Aggregation untereinander

Fragen zum Stoff

Man unterscheidet zwischen und Klassen:

- A) starren und flexiblen
- B) abstrakten und konkreten
- C) krassen und blassen
- D) Wegner 1 und Wegner 2

Fragen zum Stoff

Was bedeutet Klassifikation?:

- A) Klassenbildung über unterschiedliche Objekttypen
- B) Einordnung von Klassen anhand von Generalisierung und Spezialisierung
- C) Zusammensetzung von Objekten aus Bestandteilen, die Objekte anderer Klassen sind
- D) Generalisierung klärt die Klassifikation, Spezialisierung löst sie auf

Fragen zum Stoff

Was bedeutet Aggregation?:

- A) Klassenbildung über unterschiedliche Objekttypen
- B) Einordnung von Klassen anhand von Generalisierung und Spezialisierung
- C) Zusammensetzung von Objekten aus Bestandteilen, die Objekte anderer Klassen sind
- D) Wechsel von Aggregatzuständen von Klassen anhand der Temperatur

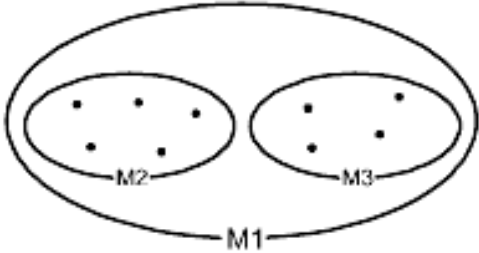
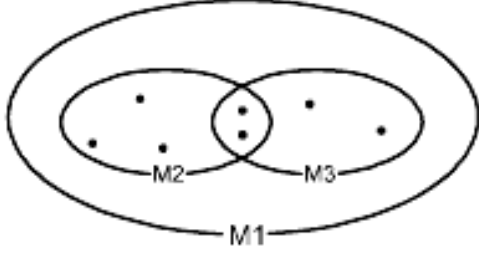
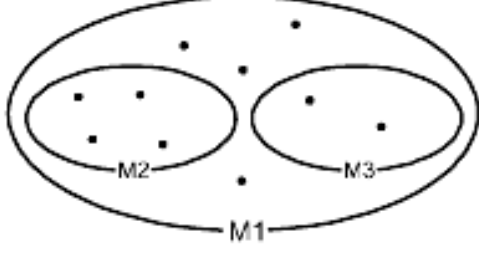
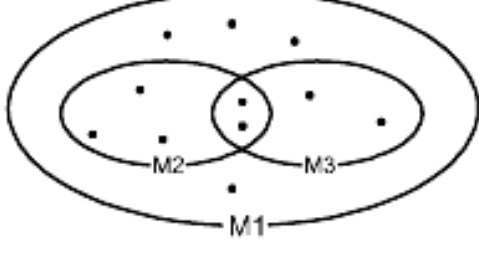
Erste Übung

Klassifikation und Aggregation

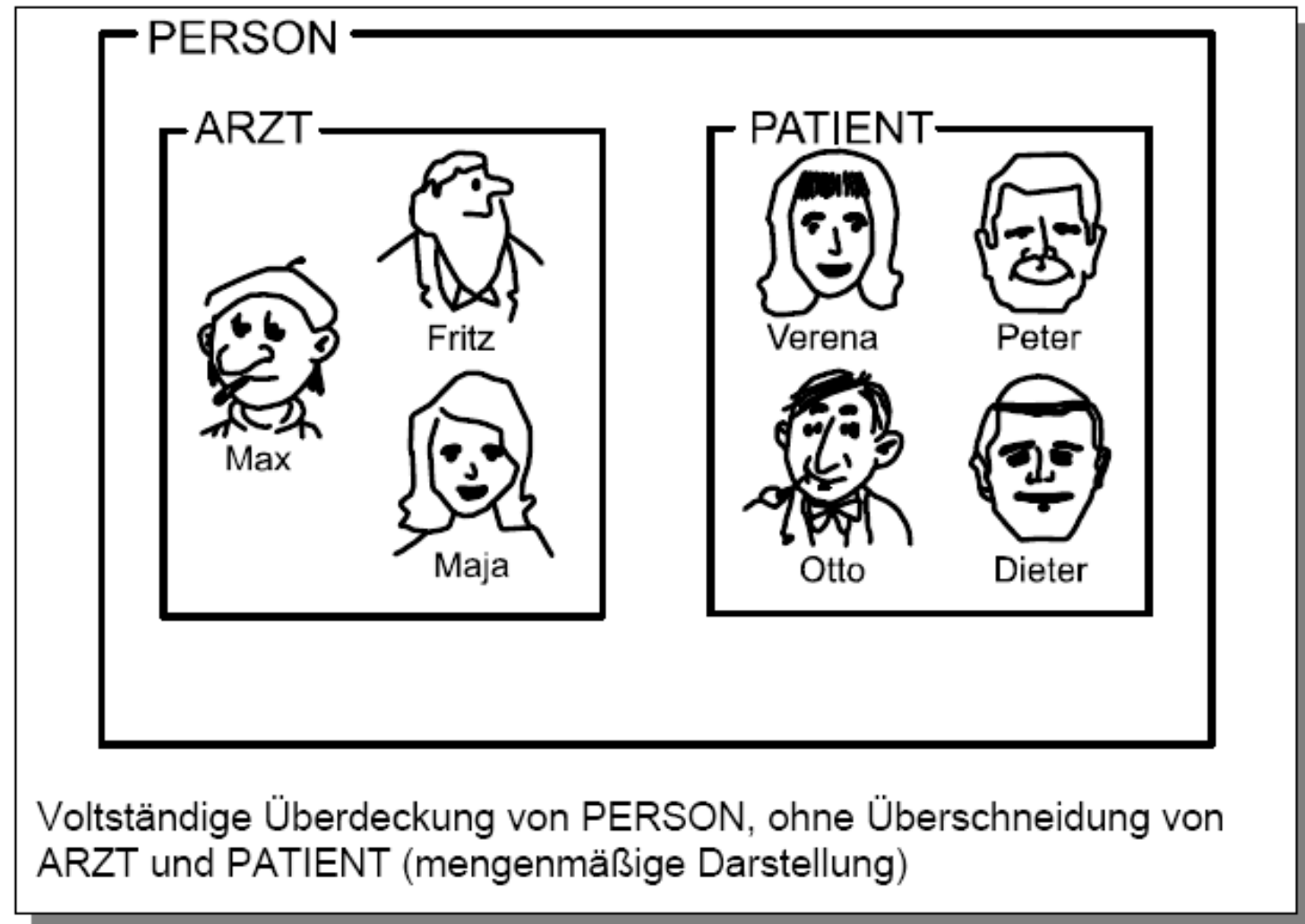
Bei der Spezialisierung bzw. Generalisierung wird zwischen den folgenden Fällen unterschieden:

Legende:

- Ovale sind Klassen
- Punkte sind Objekte

1.		vollständige Überdeckung von M1, ohne Überschneidung von M2 und M3
2.		vollständige Überdeckung von M1, mit Überschneidung von M2 und M3
3.		teilweise Überdeckung von M1, ohne Überschneidung von M2 und M3
4.		teilweise Überdeckung von M1, mit Überschneidung von M2 und M3

Darstellung der theoretischen Ausführungen am praktischen Beispiel:
Beispiel:



Vererbung

- Als *Vererbung* bezeichnet man den Umstand, dass Variablen (Attribute) und Funktionen (Methoden) der Basisklasse an abgeleitete Klassen weitergegeben werden.
- Falls eine vererbte Variable oder Methode in einer Spezialisierung nicht passt, so kann sie überschrieben werden.
- *Einfache Vererbung* liegt vor, wenn jede Spezialisierung nur eine Generalisierung aufweist.
- *Mehrfache Vererbung* liegt vor, wenn eine Spezialisierung mehrere Generalisierungen aufweist.

Fragen zum Stoff

Was versteht man unter Vererbung ?

A) Dass eine vollständige Überdeckung einer Basisklasse durch ihre Ableitungen erfolgt.

B) Dass die Attribute und Methoden der Basisklasse auch in der abgeleiteten Klasse enthalten sind.

Wie entstehen Vererbungsstrukturen ?

A) Durch Ableitungen, es ist die Umsetzung der Klassifikation.

B) Durch die Abstimmung zwischen Generalisierungen, welche Spezialisierungen etwas erben sollen.

Fragen zum Stoff

Was ist zu tun, wenn eine geerbte Variable oder Methode in einer Spezialisierung nicht passt ?

- A) überladen
- B) überschreiben
- C) das Klassenmitglied löschen
- D) je nach Instanz einzeln entscheiden

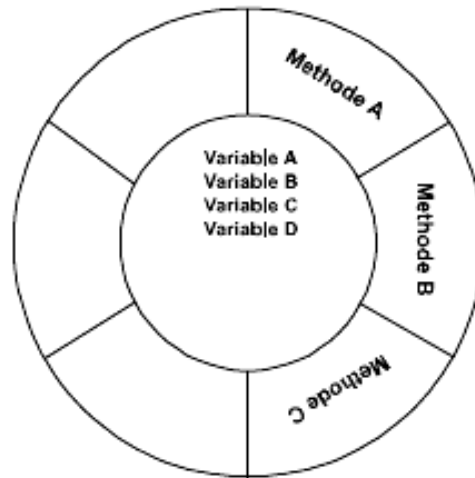
Mehrfachvererbung

Wenn eine Spezialisierung mehr als eine Generalisierung aufweist, so bezeichnet man das als mehrfache Vererbung.

Mehrfachvererbungen können unter Umständen zu Problemen führen, wenn z. B. eine Subklasse ein und dieselbe Variable oder Methode von mehreren Superklassen erbt

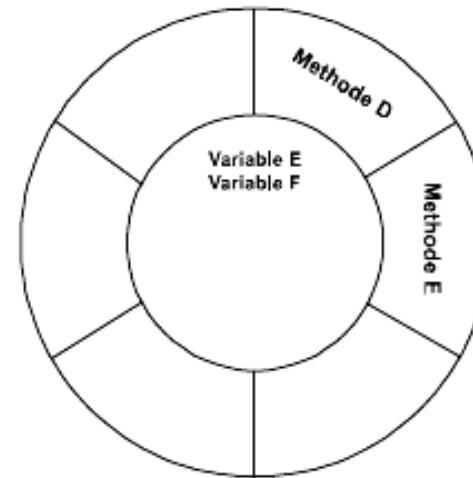
Klasse K1

(Basisklasse)



Klasse K2

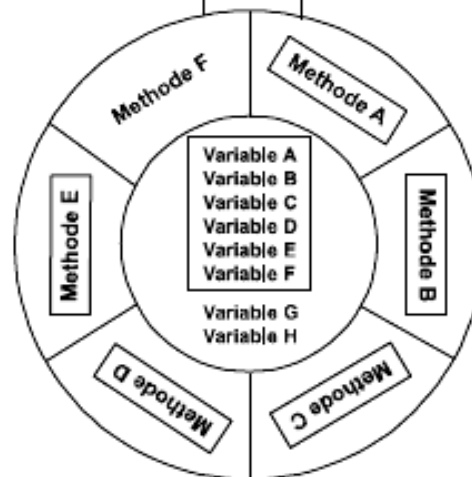
(Basisklasse)



Mehrfachvererbung

Klasse K3

(abgeleitete Klasse)

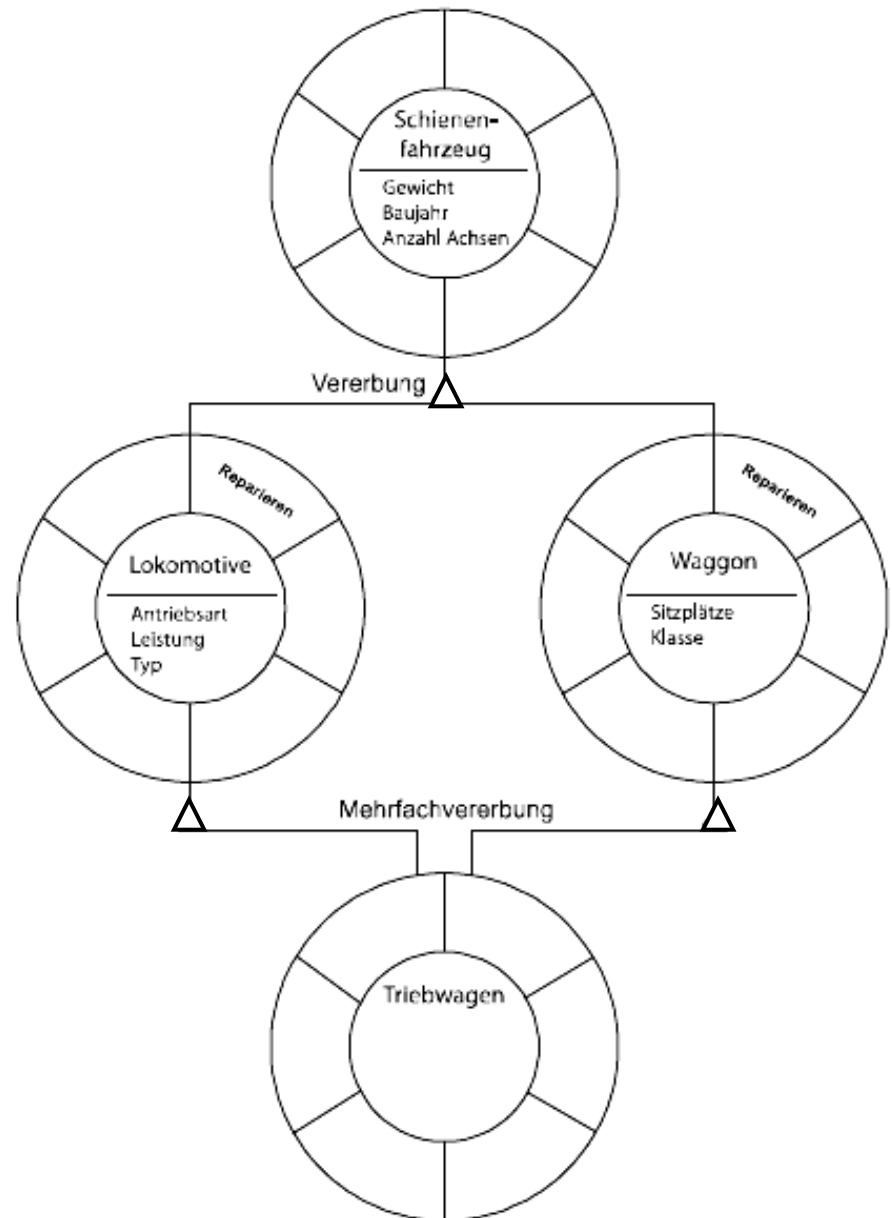


Vererbte
Methoden &
Variablen

Problematische Mehrfachvererbung:

Es gibt zwei
Möglichkeiten, diese
Probleme zu lösen:

- a) Fälle der hier
aufgezeigten Art
vermeiden
- b) Mehrfach geerbte
Variablen und Methoden
in den Subklassen
überschreiben

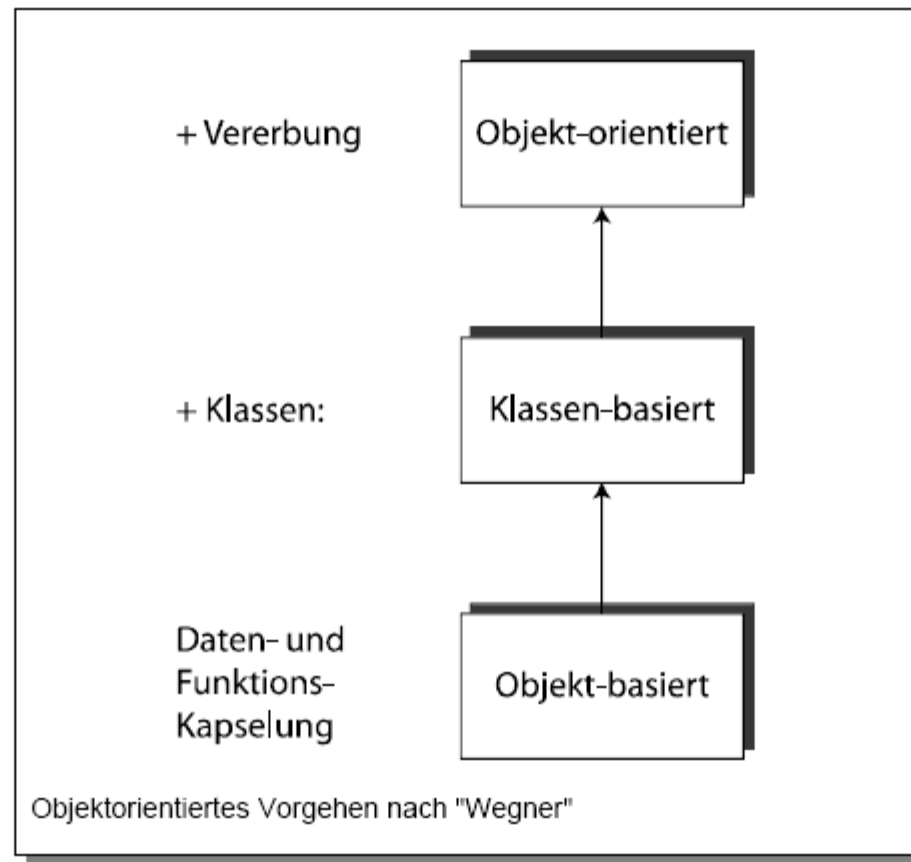


Fragen zum Stoff

Wann führt eine Mehrfachvererbung zu Problemen ?

- A) Es gibt keine Probleme bei der Mehrfachvererbung.
- B) Wenn die gleichen Methoden von mehreren Basisklassen Konflikte oder Uneindeutigkeiten entstehen lassen.
- C) Wenn mehrmals vererbt wird in einer Klassenhierarchie und die Ableitung der Ableitung die ursprüngliche Basisklasse nicht mehr kennt.

Ein *objektorientiertes Vorgehen* nutzt das Prinzip der *Daten- und Funktionskapselung* sowie jenes von *Klassen*. Zudem arbeitet es mit *Vererbungsstrukturen* und gewährleistet damit einen hohen Grad an *Wiederverwendbarkeit*.

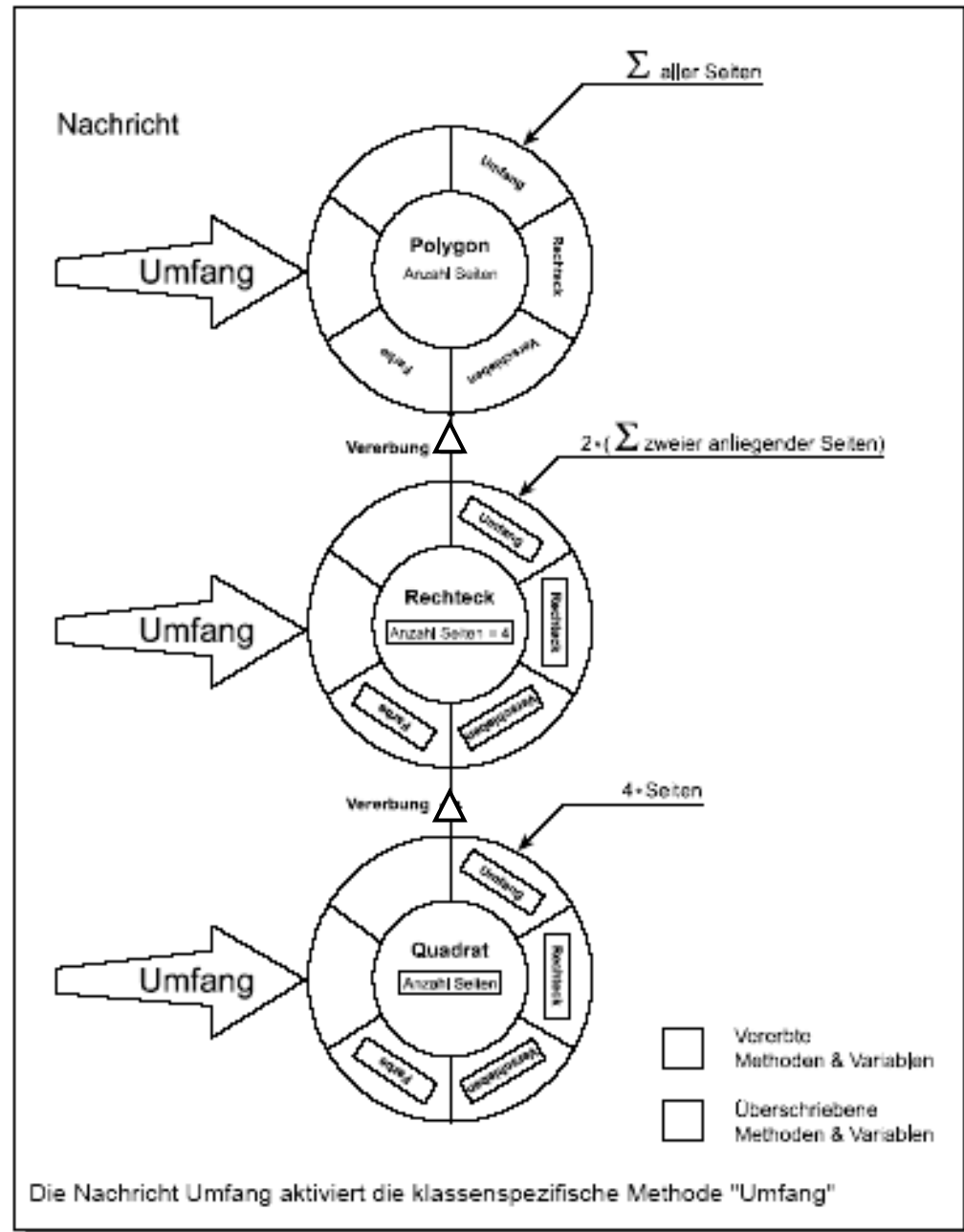


Polymorphismus

„Vielgestaltigkeit“:

Instanzen von unterschiedlichen Klassen können aufgrund der selben Nachricht verschieden reagieren.

(In einer Klassifikation also in einer Klassenhierarchie oder Ableitungshierarchie)

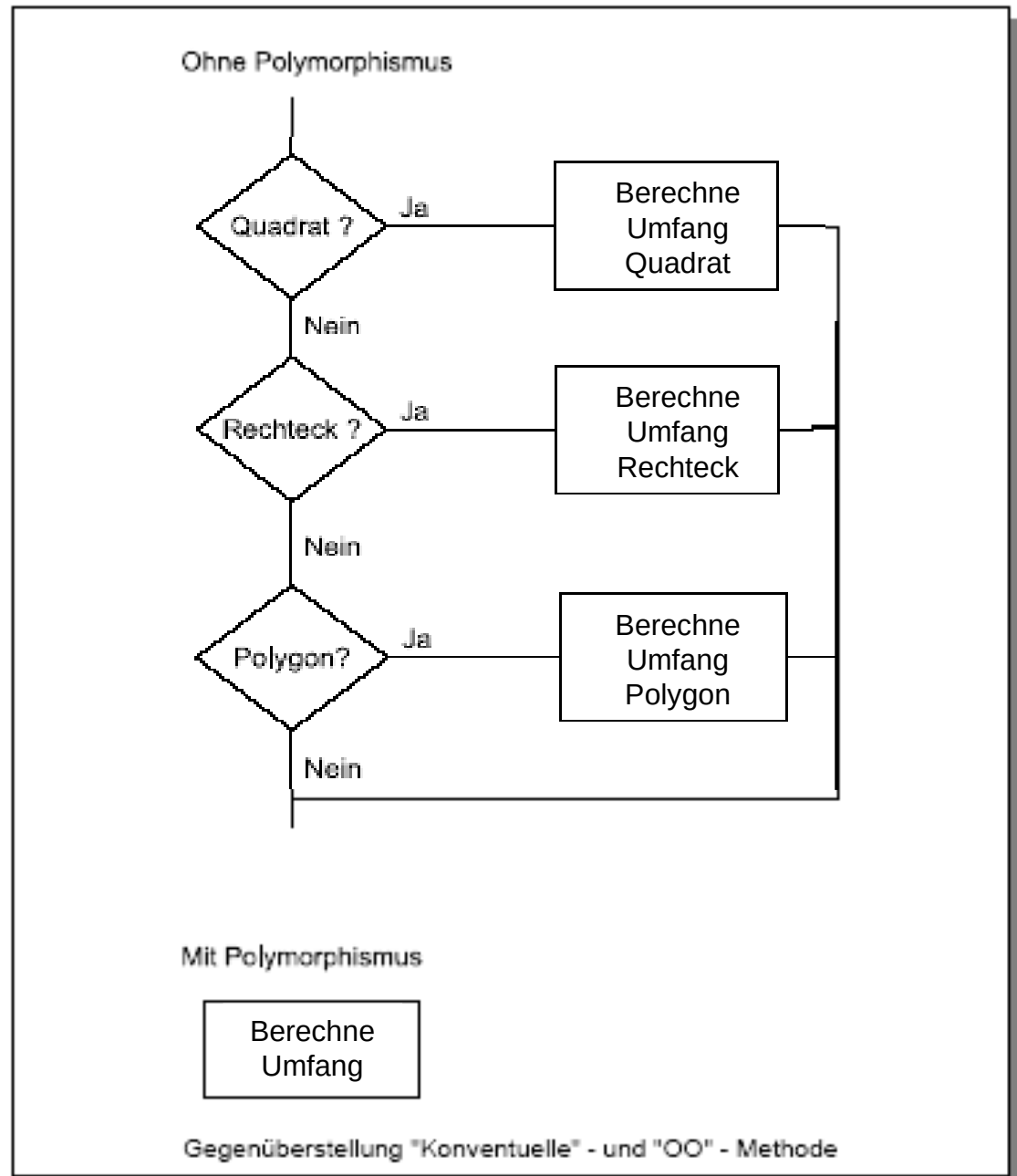


Mit Polymorphismus ist eine hohe Flexibilität, eine signifikante Vereinfachung der Logik von Programmen und eine Reduktion des Wartungsaufwandes zu erzielen.

Ein Algorithmus muss nur einmalig allgemein umgesetzt werden.

Jedes zu einer Aktion aufgeforderte Objekt kennt die spezifische auszuführende Methode.

Berechne



Fragen zum Stoff

Was ist Polymorphismus ?

- A) Dass überschriebene Methoden optional verwendet werden können.
- B) Dass bei einer Nachricht an ein Objekt einer Klasse in einer Vererbungshierarchie automatisch die zur Spezialisierung passende Überschreibung einer Methode aufgerufen wird.
- C) Dass Klassen in einer Vererbungshierarchie je nach Häufigkeit Ihrer Methodenüberschreibungen höher oder niedriger eingestuft werden können.
- D) Dass die erstmals übersetzte konkrete Version einer Methode an eine abstrakte Instanz gebunden wird, die ihre Gestalt ändern kann.

Fragen zum Stoff

Welche Vorteile sind mit Polymorphismus zu erzielen ?

A) Man kann auf private Attribute von Objekten zugreifen.

B) Man kann auf Ableitungen verzichten und die unterschiedlichen Spezialisierungen innerhalb einer einzigen Klasse implementieren.

C) Man kann Algorithmen implementieren, ohne auf den genauen Typ eines Objekts achten zu müssen.

Fragen zum Stoff

Charakterisieren Sie (welche Aussagen sind richtig?):

Objektbasiertes Vorgehen:

- A) Kapsel und Geheimnisprinzip
- B) globale Variablen als Verbindung von Methoden
- C) Klassenbildung durch Abstraktion bei gleichartigen Objekten

Klassenbasiertes Vorgehen

- A) Klassifikation durch Ableitungshierarchien mittels Vererbung
- B) Klassenbildung durch Abstraktion bei gleichartigen Objekten
- C) Jede Klasse ist ihre eigene Basis

Objektorientiertes Vorgehen

- A) Kapsel und Geheimnisprinzip
- B) Klassenbildung durch Abstraktion bei gleichartigen Objekten
- C) Klassifikation durch Ableitungshierarchien mittels Vererbung

Techniken zur Ermittlung von Klassen:

Das Kernproblem objektorientierter Anwendungsentwicklung liegt in der Ermittlung der richtigen Klassen.

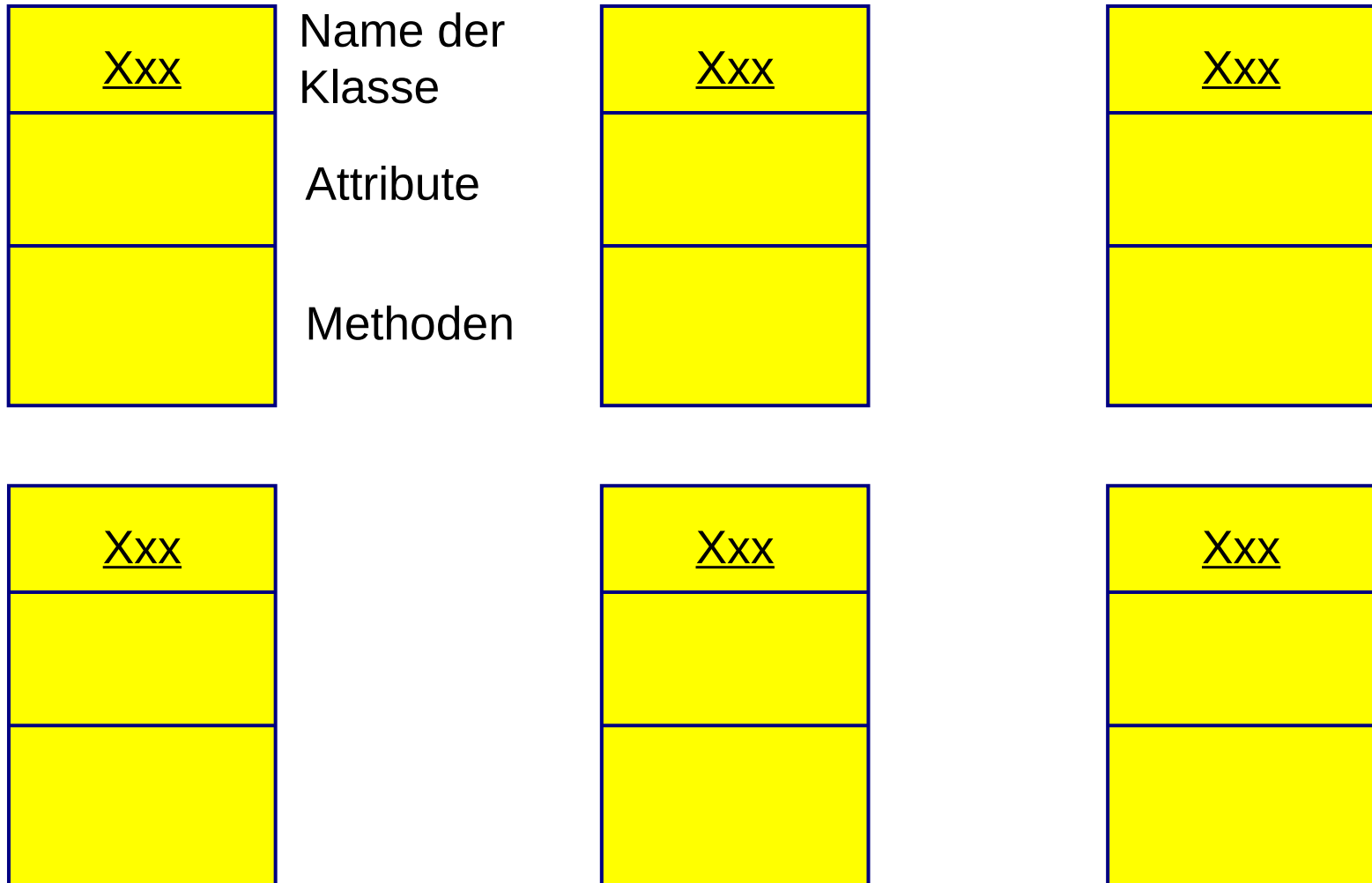
Textanalyse (Fachmodell erstellen):

Liegt für das Problem eine (natürlichsprachliche) textuelle Beschreibung vor, so wird diese systematisch nach *Substantiven* und somit nach Kandidaten für Klassen durchsucht.

Attribute werden anhand von *Adjektiven* bestimmt.

Methoden (auch als Geschäftsvorfälle bezeichnet) werden anhand von *Verben* bestimmt.

Einfache Klassen-Notation:



Textbeispiel:

Ein grünes Auto fährt eine 20 km lange Straße entlang um ein 20 kg schweres Paket auszuliefern. Das Auto beschleunigt nach einer engen Kurve und verzögert wegen eines Wildschweins, das über die Straße läuft, weil der Fahrer aufmerksam ist, reagiert und bremst.

Klassenkandidaten

Auto
Straße
Paket
Kurve
Wildschwein
Fahrer

Kandidaten für Attribute

grün
20 km Länge
20 kg Gewicht
eng
aufmerksam

Kandidaten für Methoden

fahren
ausliefern
beschleunigen
verzögern
laufen
aufmerksam sein (?)
reagieren
bremsen

Einfaches Klassenmodell:

<u>Auto</u>
Farbe
fahren() ausliefern() beschleunigen() verzögern()

<u>Straße</u>
Länge

<u>Paket</u>
Gewicht

<u>Kurve</u>
Radius

<u>Wildschwein</u>
laufen() straßeBetreten()

<u>Fahrer</u>
Aufmerk- samkeit
reagieren() bremsen()

Verhaltensanalyse:

In der Verhaltensanalyse kommen folgende Aspekte zum Tragen:

- *Initiator*: das einen Vorgang auslösende Objekt
- *Action*: die auszuführende(n) Aktion(en)
- *Participant*: zur Durchführung des Vorganges beitragende(s) Objekt(e)

Beispiel:

<i>Initiator</i>	<i>Action</i>	<i>Participant</i>
Kunde	Wählt Einzahlung auf Konto	Kassierer
Kassierer	Addiert Einzahlungsbetrag	Konto
Kassierer	Erstellt Einzahlungsbeleg	Kunde

Ermittlung von Methoden:

1. Objektstatusdiagramm erstellen.
2. Aus Objektstatusdiagramm erforderliche Methoden ableiten.
3. Erforderliche Message-Connections identifizieren.
4. Methoden spezifizieren.

Im Leben eines Objekts sind in der Regel verschiedene, unter Umständen sich wiederholende Phasen zu unterscheiden.

Beschrieben wird ein derartiger Lebenszyklus mittels eines sogenannten *Objektstatusdiagramm*.

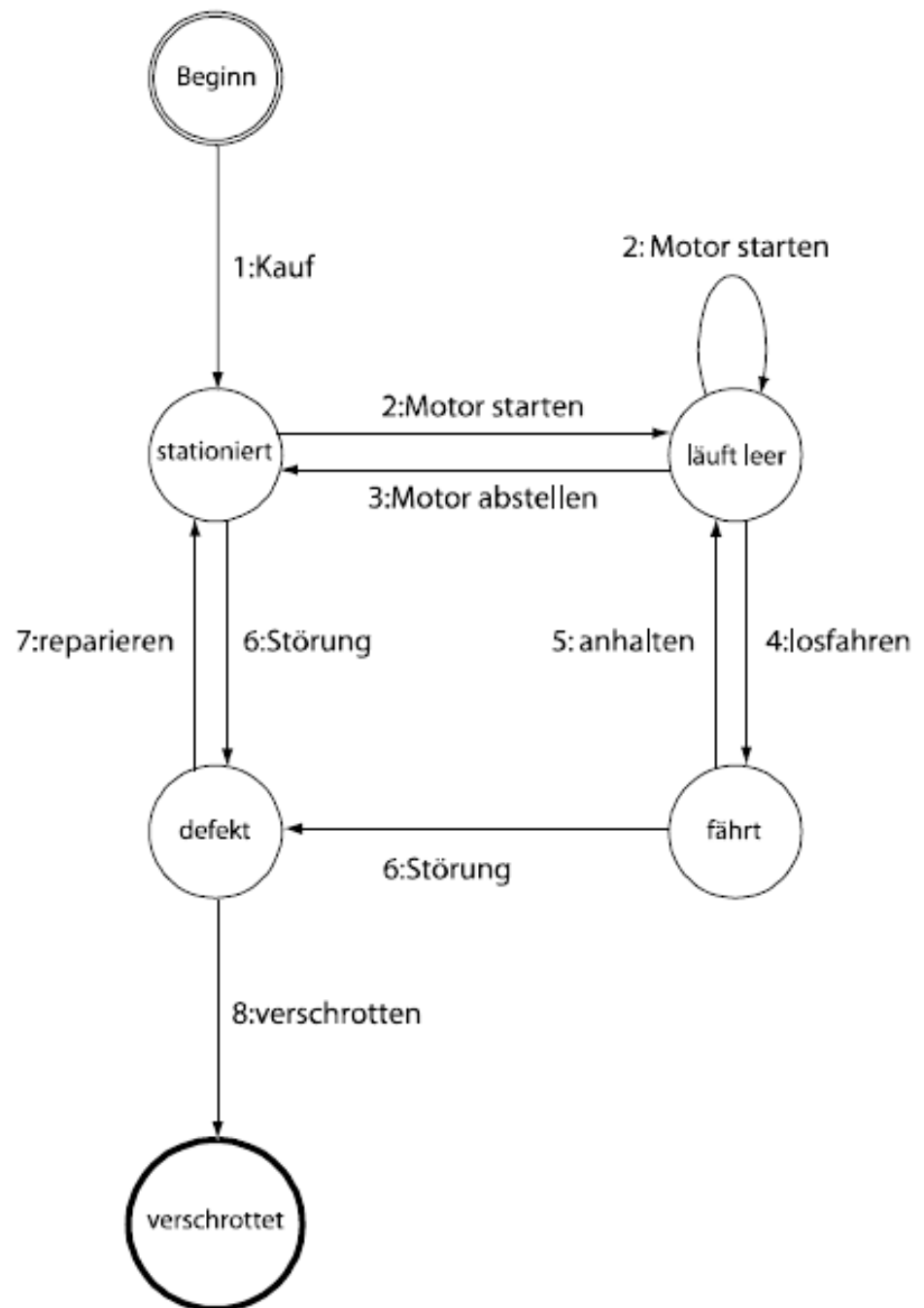
Aus Objektstatusdiagrammen / Statusübergangstabellen sind Methoden abzuleiten, da es zu jedem Statusübergang mindestens eine den Übergang veranlassende Methode geben muss.



Klasse AUTO mit erforderlichen Variablen und Methoden

Beispiel:

Objektstatusdiagramm
für ein Objekt der
Klasse Auto



Die Erstellung von einer *Statusübergangstabelle* ist dann erforderlich.

Ereignis Status	1 Kauf	2 Motor starten	3 Motor abstellen	4 losfahren	5 anhalten	6 Störung	7 reparieren	8 verschrotten
Beginn	stationiert	-	-	-	-	-	-	-
stationiert	-	läuft leer	-	-	-	defekt	-	-
läuft leer	-	ignoriert	stationiert	fährt	-	-	-	-
fährt	-	-	-	-	läuft leer	defekt	-	-
defekt	-	-	-	-	-	-	stationiert	verschrottet
verschrottet	-	-	-	-	-	-	-	-

Statusübergangstabelle für ein Objekt der Klasse AUTO

Fragen zum Stoff

Wie sind Methoden zu ermitteln ?

A) Durch erraten

B) Durch eine Textanalyse, Verben sind Kandidaten für Methoden

C) Durch die Öffnung der Kapsel für Nachrichten

Fragen zum Stoff

Was ist in einem Objektstatusdiagramm festzuhalten ?

- A) Der Aufbau der Klasse aus Attributen und Methoden
- B) Anfang und Ende der Nutzung eines Objektes
- C) Mögliche Zustände von Objekten und deren Übergänge durch die Anwendung ihrer Methoden
- D) Wie es dem Objekt im Moment gerade geht

Fragen zum Stoff

Was ist mit einer Statusübergangstabelle festzuhalten ?

- A) Das was im Objektstatusdiagramm dargestellt wird in Form einer Matrix aus Methoden und Statuswerten
- B) Schwellwerte für neuronale Netze
- C) Die Vererbungsstruktur ergänzt um die möglichen Wertebereiche der Attribute
- D) Der Aufrufzusammenhang von Methoden zwischen beteiligten Objekten

Dokumentation von Methoden

Für jede Methode ist festzulegen, welche Dienstleistungen nach außen angeboten werden und was mit einer Methode im einzelnen zu geschehen hat.

Zu diesem Zwecke sind sowohl *Interface* (d. h. eine mittels *Input*- und *Output*-Parametern festzulegende Schnittstelle) wie auch *Aktionen* zu bestimmen.

Fragen zum Stoff

Aus welchen Teilen besteht eine Methodenspezifikation ?

Mit *Beziehungsstrukturen* sind die Objekte einer Klasse oder verschiedener Klassen miteinander in Beziehung zu setzen. Je nach Anzahl der Objekte, mit denen ein bestimmtes Objekt in Beziehung (Instance Connections [Coad/Yourdon], Associations [Booch]) stehen kann, unterscheidet man:

- Einfache Beziehungen
- Konditionelle Beziehungen
- Komplexe Beziehungen
- Komplex-konditionelle Beziehungen

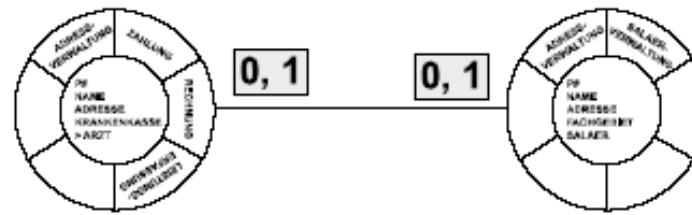
Die Anzahl der an der Beziehung beteiligten Objekte wird mit Zuordnungskardinalitäten angegeben.

Einfache Beziehung:

Wenn jedes Objekt einer Klasse A jederzeit mit einem Objekt einer Klasse B in Beziehung steht, so liegt von A nach B eine einfache Beziehung vor (im Spezialfall steht jedes Objekt der Klasse A mit einem Objekt der gleichen Klasse in Beziehung).

Konditionelle Beziehung:

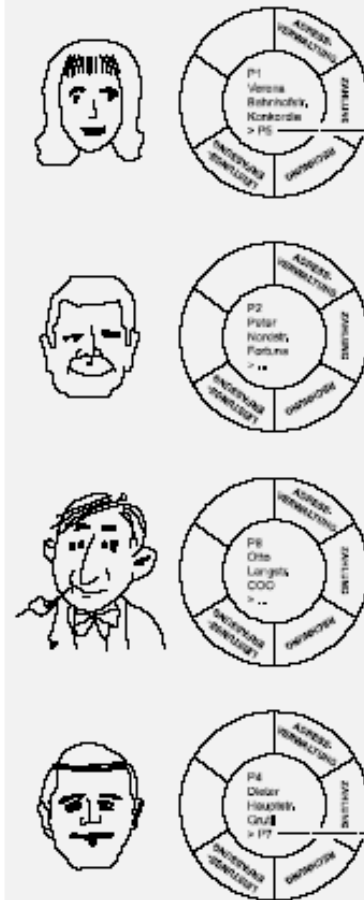
Wenn jedes Objekt einer Klasse A jederzeit mit einem oder keinem Objekt einer Klasse B in Beziehung steht, so liegt von A nach B eine konditionelle Beziehung vor (im Spezialfall steht jedes Objekt der Klasse A mit einem oder keinem Objekt der gleichen Klasse in Beziehung).



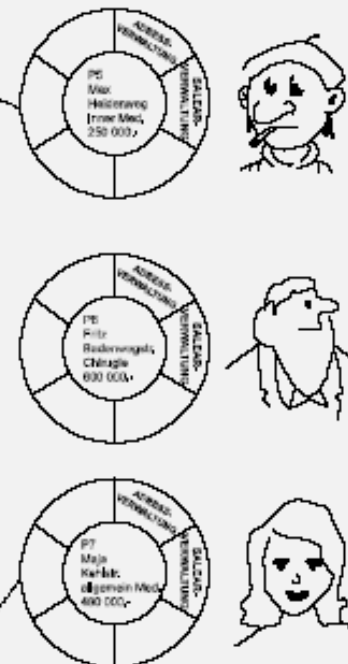
Klasse PATIENT

Klasse ARZT

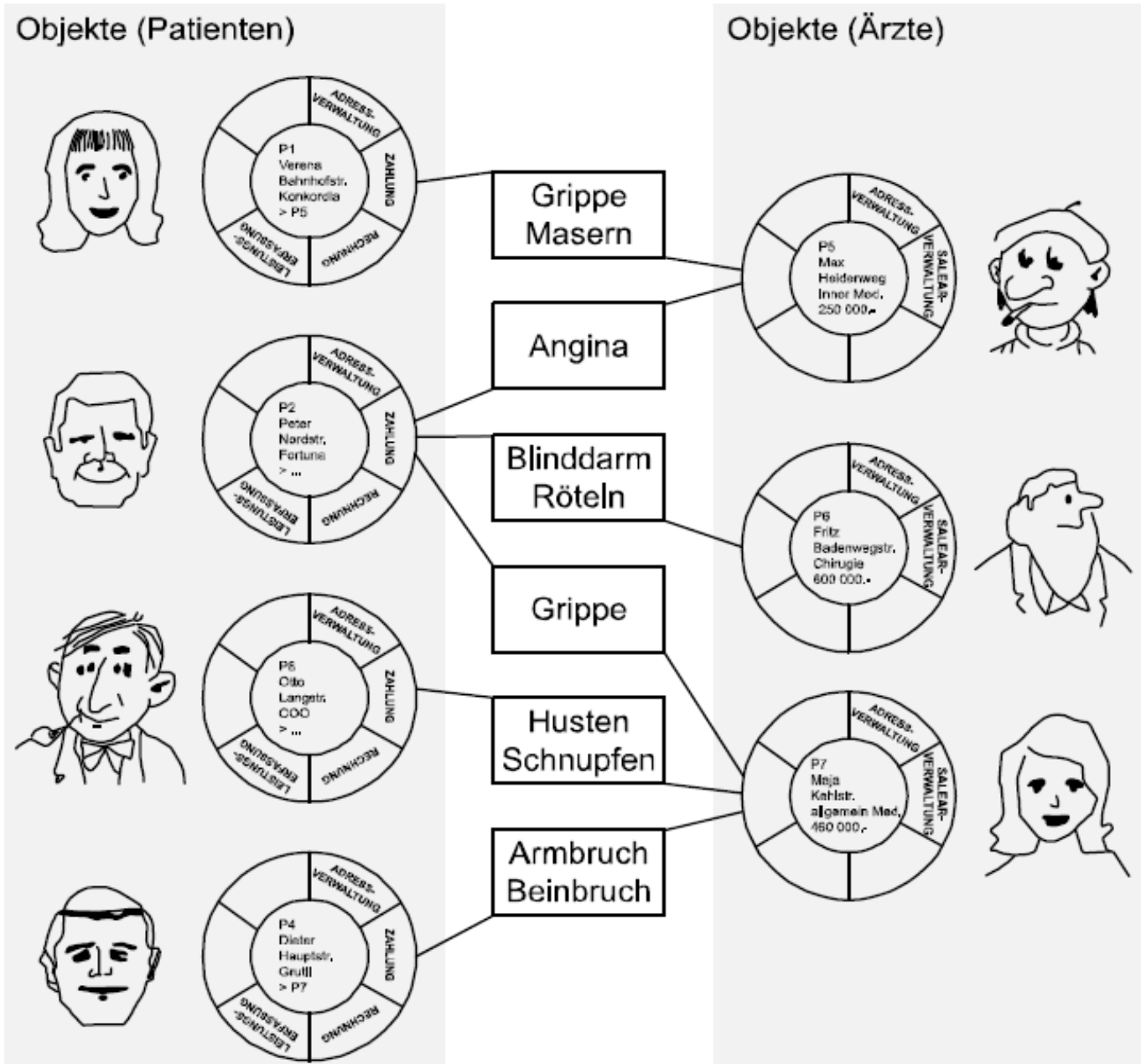
Objekte (Patienten)



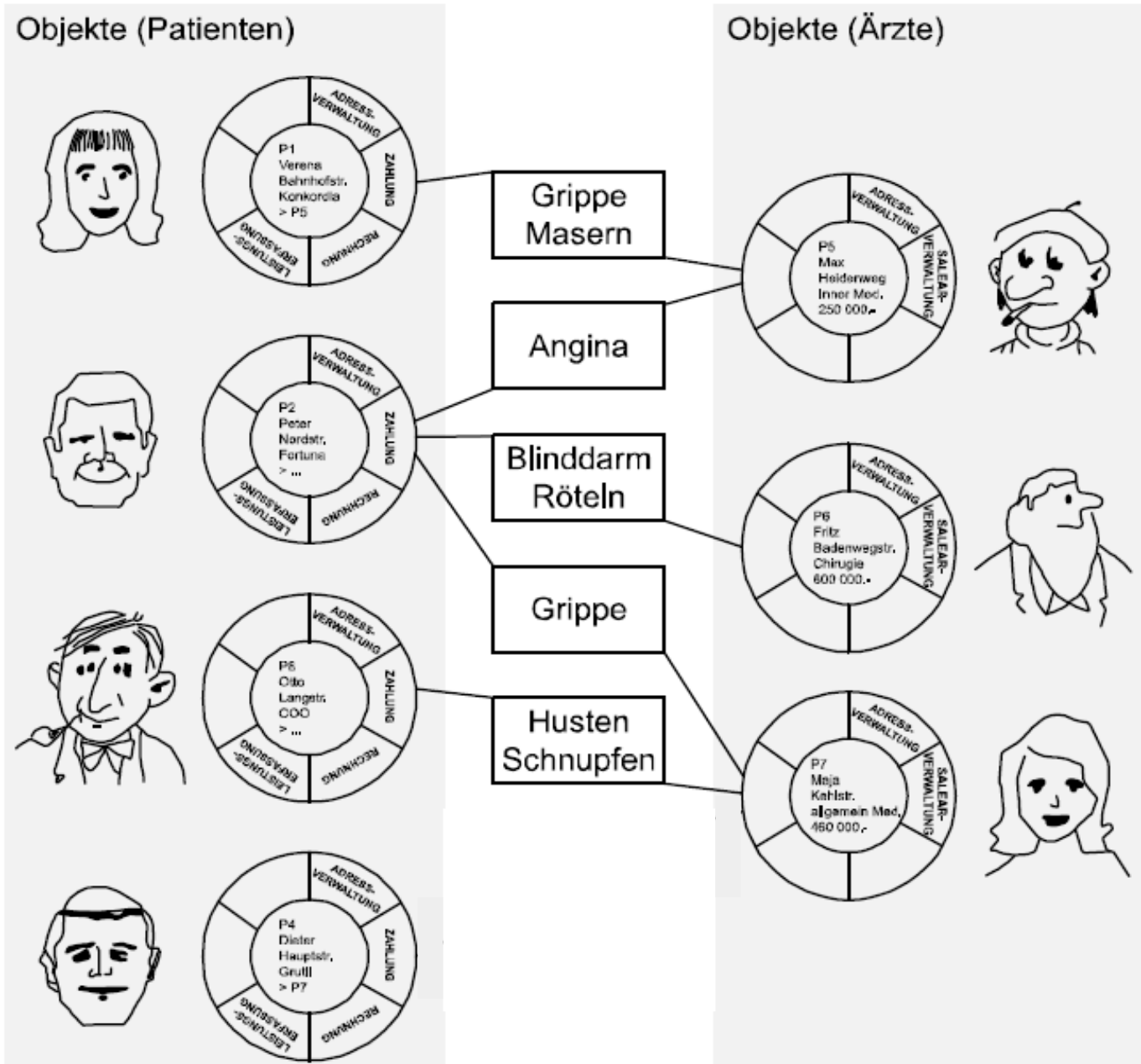
Objekte (Ärzte)



Komplexe Beziehung



Komplex-konditionelle Beziehung

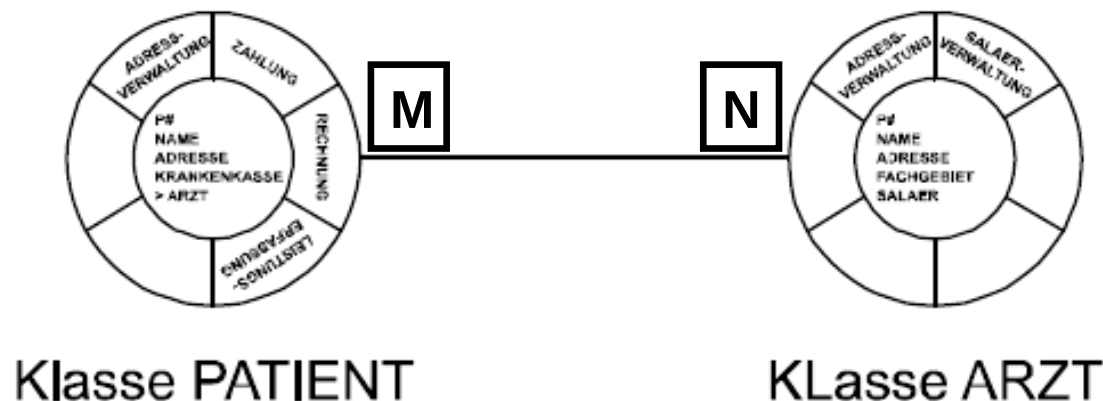


Dietmar hat nichts

Ereignisklassen

Ähnlich wie Objekte durch Variable (Attribute) zu charakterisieren sind, sind auch Beziehungen durch Variable (Attribute) zu umschreiben.

Im objektorientierten Umfeld sind Ereignisklassen zu bilden (s. dazu Ereignisanalyse). Mit Ereignisklassen wird das Geschehen auf die Zeitachse modelliert.



Die Objekte der Ereignisklasse BEHANDLUNG bringen zum Ausdruck, welche Patienten von welchen Ärzten im Verlaufe der Zeit behandelt werden.



Ereignisklasse zur Modellierung einer Beziehungsvariablen

Fragen zum Stoff

Wozu dienen Beziehungsstrukturen ?

A) Um modellieren zu können, welche Objekte mit welchen Objekten Nachrichten tauschen können

B) Zur Darstellung der Vererbungshierarchie

C) Um die Modellierung zu vereinfachen, indem unnötige Attribute wegfallen

Fragen zum Stoff

Worin unterscheiden sich Beziehungsstrukturen von Vererbungsstrukturen ?

A) Nichts, beide beschreiben dasselbe in unterschiedlicher Darstellung (Notation)

B) Vererbungsstrukturen zeigen die statische Herkunft von Objekten, Beziehungsstrukturen zeigen die dynamische Wirkung von Objekten

C) Beziehungsstrukturen können konditionell zu Vererbungsstrukturen führen

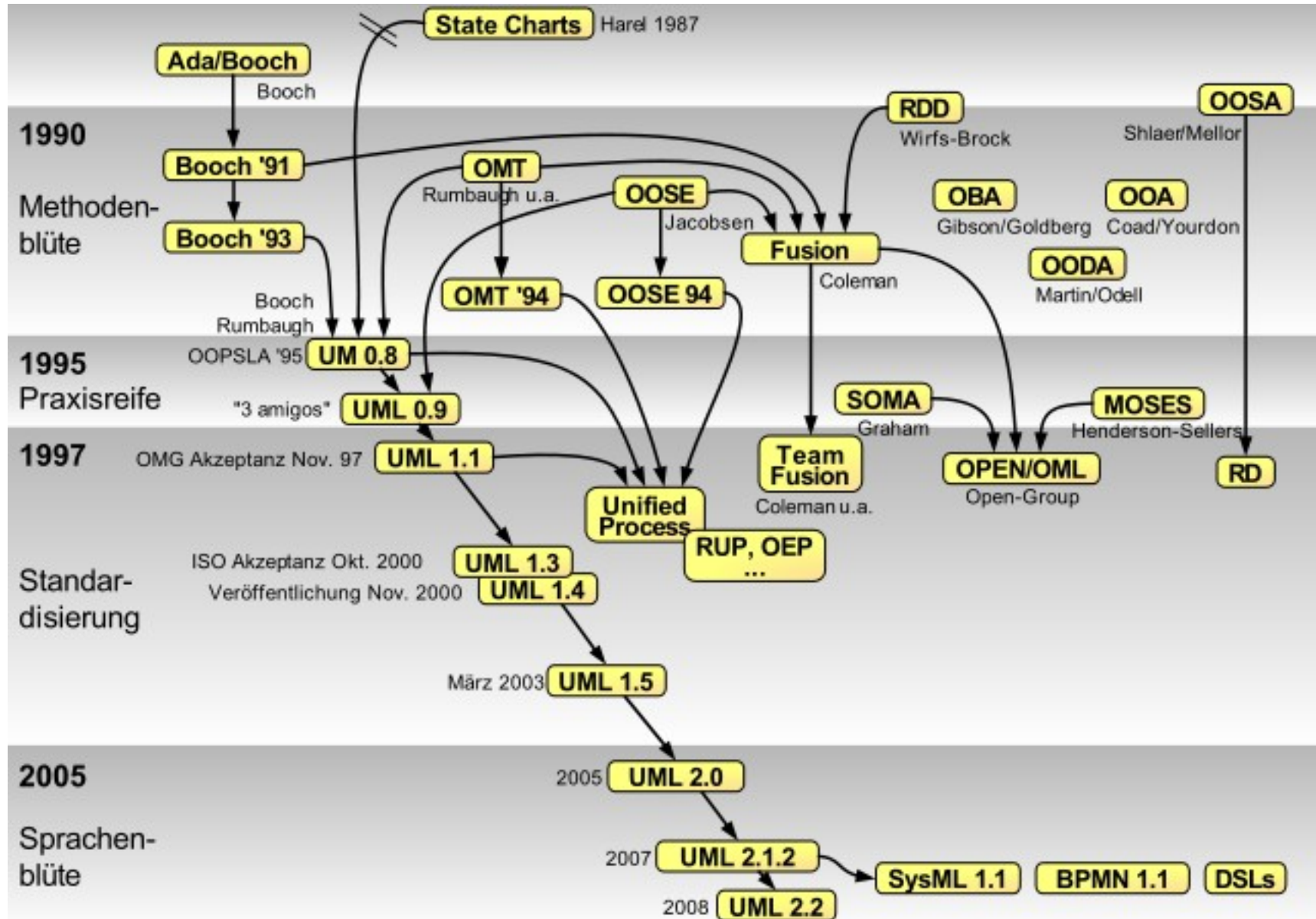
Fragen zum Stoff

Welche Beziehungsarten (Zuordnungskardinalitäten) unterscheidet man ?

Eine Ereignisklasse entspricht in der klassischen Datenmodellierung einer

==> Kreuztabelle (Ausblick auf die Datenbank-Vorlesung und die Persistierung von Objekten)

OOA/OOD-"Stammbaum"



Quelle: wikipedia / creative commons / Innovative Informatik GmbH

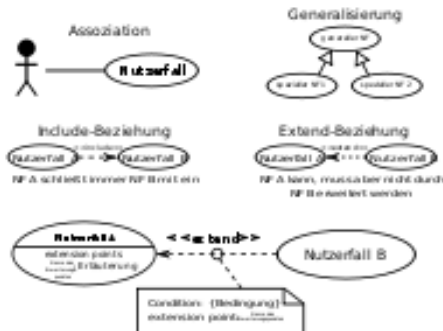
Kleiner Exkurs: SOLID

- von C. Martin 1994 geprägt:
 - **Single Responsibility Principle (SRP)**: fachliche Zuständigkeit einer Klasse
 - **Open Closed Principle (OCP)**: offen für Erweiterung in der Ableitung (Spezialisierung), geschlossen für Manipulationen in der Basisklasse
 - **Liskov Substitution Principle (LSP)**: ein Objekt der Ableitung muss sich so verhalten, wie es auch ein Objekt der Basisklasse tun würde
 - **Interface Segregation Principle (ISP)**: Interfaces dürfen keinen Ballast haben, sonst ziehen sie ggf. unpassende Abhängigkeiten nach (nicht rel. in C++)
 - **Dependency Inversion Principle (DIP)**: Methoden der Basis dürfen keine Methoden der Ableitung aufrufen (Aufrufe gerichtet, frei von Zyklen)
- die 5 Prinzipien haben jeweils eigene Autorinnen und Autoren, zeitlich in der „Methodenblüte“ bis zur „Praxisreife“
- die Ideen flossen bei B. Stroustrup in den Entwurf von C++ teilweise mit ein, teilweise sind sie dafür nicht relevant
- externer [Link zu t3n](#), dort schön zusammengefasst

=> geht über die Ziele der Prog2-Lehrveranstaltung hinaus ;-)

UML-“Kondensat“

Nutzerfalldiagramm



Zustandsdiagramm

Verhaltenszustandsmaschine (BSM)

Transition: $\text{in} [\text{Ausdruck}] [\text{Bedingung}] [\text{Zustand}]$
 Ausdruck: $\text{in Ereignis} [\text{Zustand}]$
 Ereignis: $\text{in Ereignis} [\text{Ereignis}]$
 Zustand: $\text{in Zustand} [\text{Zustand}]$
 Zustand: $\text{in Zustand} [\text{Zustand}]$

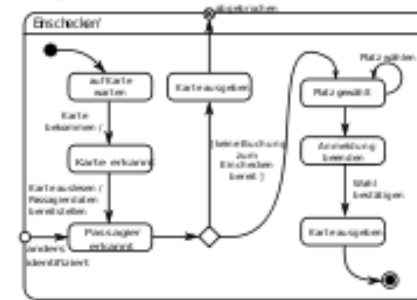
Protokollzustandsmaschine (PSM)

Transition: $\text{in} [\text{Veränderung}] [\text{Methodeaufruf}] [\text{Nachbedingung}]$

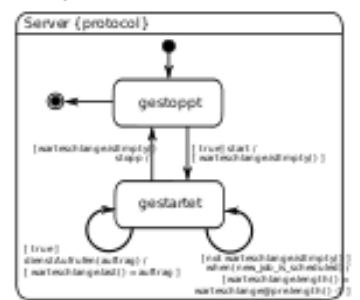
Notationen:



Beispiel: Verhaltenszustandsmaschine



Beispiel: Protokollzustandsmaschine

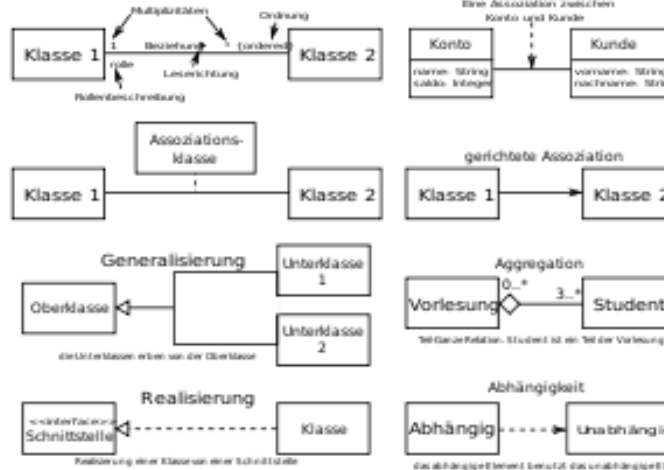


Klassendiagramm

Analysemodell (fachliche Sicht)

- Klassen sind Fachbegriffe
- Attribute
 - i. Allg. ohne Datentypen
 - ggfs. mit Multiplizitäten
- Methoden ohne Parameter und Rückgabewert
- Bidirektionale Assoziationen/Aggregationen
- Beschreibung von Assoziationen (Multiplizitäten, Rollennamen, Assoziationsnamen (mit Leserichtung))
 - ggfs. mit Qualifier
- Assoziationsklassen und n-äre Beziehungen
- Generalisierung / Spezialisierung (Vererbung)
- Abgeleitete Attribute und Methoden
- Aufzählungen (Enumerationen)

Eine Assoziation beschreibt eine Beziehung zwischen zwei oder mehr Klassen. An den Enden von Assoziationen sind häufig Multiplizitäten vermerkt. Diese drücken aus, wie viele dieser Objekte in Relation zu den anderen Objekten dieser Assoziation stehen.



Entwurfsmodell (fachliche+tech. Sicht)

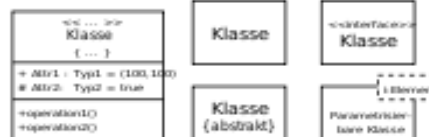
- Klassen -> abstrakt, Interface, Stereotyp, ggf. Klassen streichen, hinzufügen, umbenennen
- Attribute -> Sichtbarkeiten, Ableitung, Klassenattribute, Initialisierung, weitere spezielle Eigenschaften
- Operationen -> Parameter, Sichtbarkeiten, Rückgabewert, Klassenoperation
- Assoziationen -> gerichtet, geordnet / sortiert
- Auflösen von Assoziationsklassen / n-äre Beziehungen
- Abhängigkeiten
- Packages
- Hilfsmethoden (Konstruktoren, getter/setter, toString() u.a.)

Syntax für Attribute

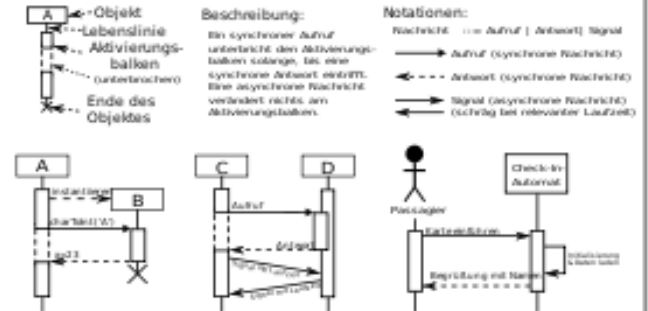
Sichtbarkeit Attributname : Paket : Typ [Multiplizität Ordnung] = Initialwert (Eigenschaftswerte)
 Eigenschaftswerte: {readOnly}, {ordered}, {composite}

Syntax für Operationen

Sichtbarkeit Operationsname (Parameterliste): Rückgabewert
 Sichtbarkeit: + public element, # protected element, - private element, ~ package element
 Parameterliste: Richtung Name : Typ = Standardwert
 Richtung: in, out, inout



Sequenzdiagramm



Aktivitätsdiagramm

Notationen:

- Startpunkt
- Endpunkt
- Kreuzung oder Entscheidung
- Vereinigung oder Gabelung

Beispiel: Aktivitätsdiagramm



Zweite Übung

Modellierung einer Hafeneinfahrt

Fin!

Bernd Ruhland

email: ruhland@hs-worms.de