Methoden I

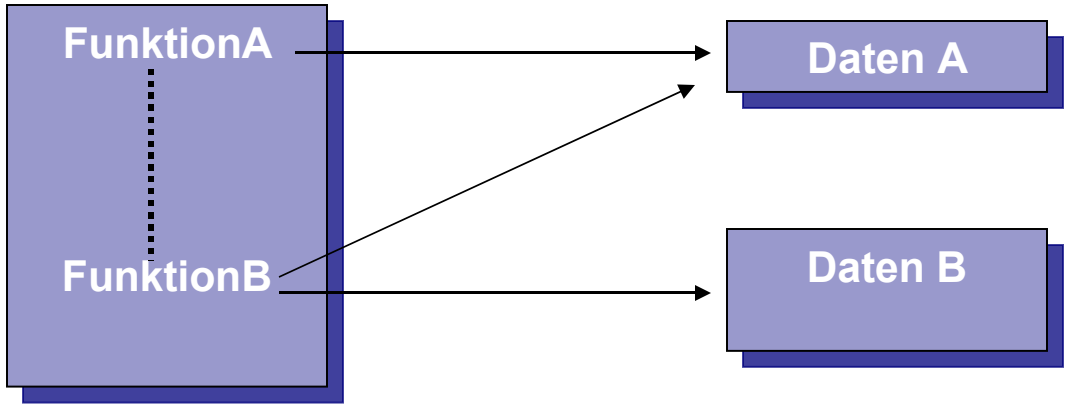
- Der Mensch begegnet der Komplexität von Problemstellungen, in dem er versucht methodisch vorzugehen

Methoden II

- Methoden...
  - ...sind Anleitungen zur Durchführung von verschiedenen voneinander abhängigen Aktivitäten in bestimmter Reihenfolge
  - ...haben Regeln zur Vorgehensweise (Syntax)
  - ...geben u.U. auch Hinweise auf inhaltliche Fragen (Semantik)
  - ...benötigen textliche und/oder graphisch Notation zur Dokumentation der Ergebnisse

Strukturierte Methoden I

- Grundideen
  - Trennung von Funktionalität und Daten
  - Klar strukturierte Vorgehensweise während den Projektphasen Analyse, Design und Implementation

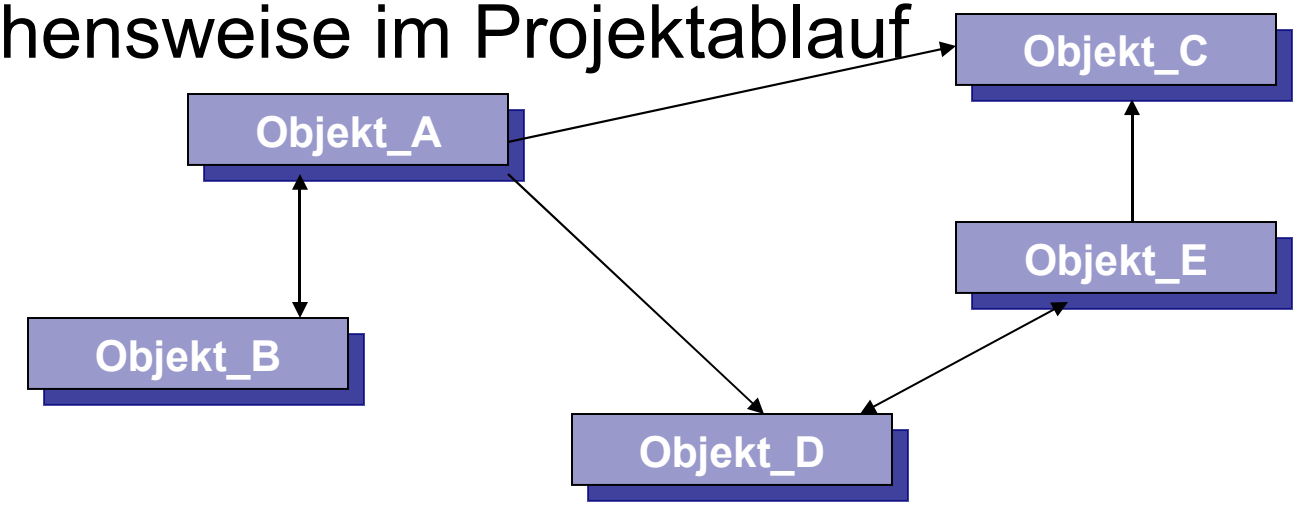


Strukturierte Methoden II

- Strukturierte Methoden verwenden verschiedene Modelle, um verschiedene Sichten der Problemstellung zu betrachten
  - Datenflussdiagramm — Funktion
  - Zustandsdiagramm — Logik
  - Entity Relationship Diagramm — Daten

Objektorientierte Methoden I

- Grundideen
  - Bildung von Organisationen und Einheiten mit Daten- und Funktionseigenschaften (Objekte)
  - Den realen Gegebenheiten angepasste Vorgehensweise im Projektablauf



Objektorientierte Methoden II

- Bauen auf bewährten Konzepten der strukturierten Methoden auf
- Erweitern diese mit einer durchgehenden Notation (UML) für die Phasen
  - Analyse
  - Design
  - Implementierung

Objektorientierte Methoden III

- **Vorteile**
  - Einfachere Kommunikation zwischen Entwickler und Benutzer
  - Einfache Abbildung der OO-Sichtweise auf die reale Welt
  - Bessere Handhabung grosser Systemkomplexität
  - Bessere Qualität und Konsistenz der Software durch durchgängige Modellierung
  - Objektorientierte Modelle sind stabiler, Änderungen meist lokal begrenzt
    - vereinfacht Wartung und Erweiterung
  - Komponentenbasierte Entwicklung erhöht die Software-Wiederverwendbarkeit

Objektorientierte Methoden IV

- Nachteile
  - Objektorientierte Modellierung kann zu unterschiedlichen Ergebnissen führen
  - Objekte sind nicht eindeutig bestimmt
    - Siehe Vexierbild
  - u.U. schlechtere Performance

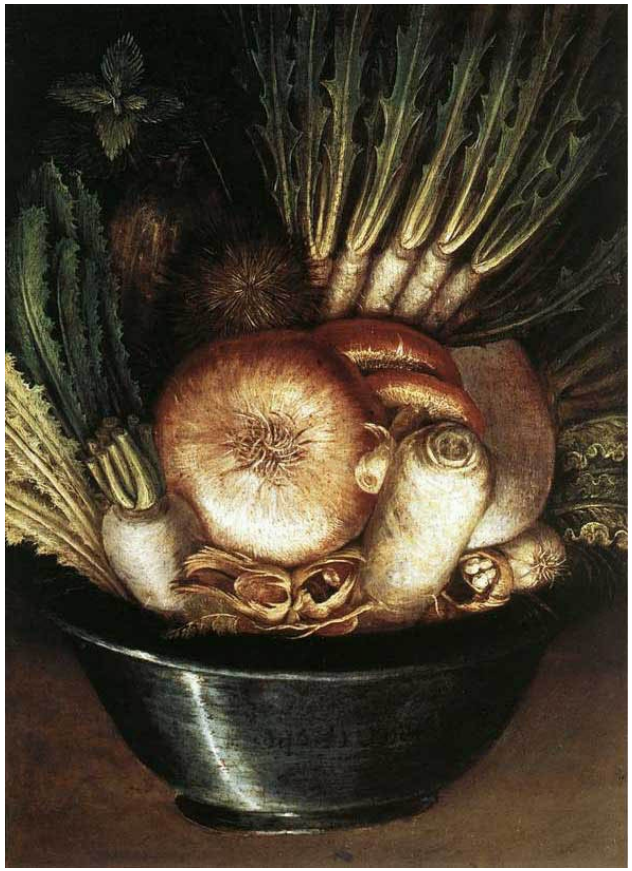
Prinzipien der objektorientierten Denkweise

- OO SW-Entwicklung ist keine Zauberei!
- Ist „nur“ eine etwas andere Denkweise
- Was wir lernen müssen:
  - Anwendung von neuen Denkmustern um Modelle entwickeln zu können
  - Eine OO Programmiersprache um diese umzusetzen (z.B. Java).

Objektorientierten Denkweise I

- Objektorientierung stammt aus der Psychologie nicht aus der Informatik
  - Menschliches Erkennen funktioniert genau so!
- Somit ist Objektorientierung nichts, was wir im eigentlichen Sinne lernen müssen
- Es beschreibt lediglich, wie wir sowieso schon alltäglich erkennen und denken

Objektorientierten Denkweise I

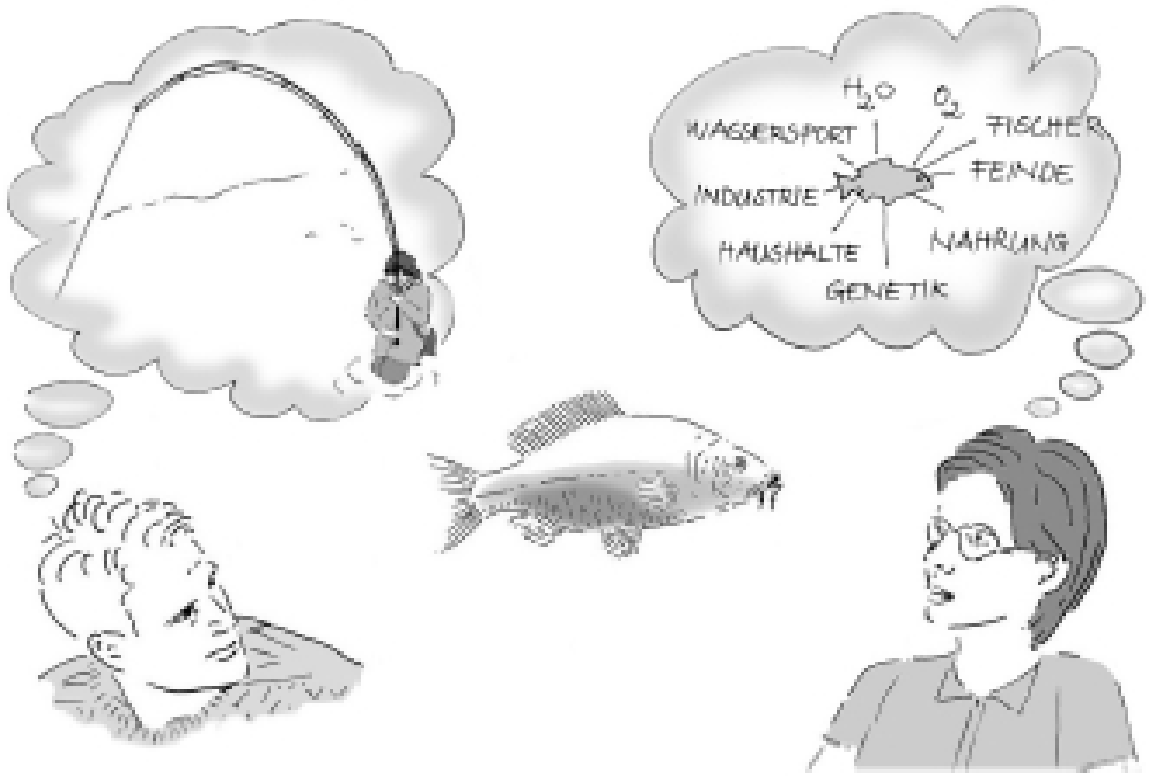


- Vexierbild von Guiseppe Arcimboldo (1527-1593)
  - The Vegetable Gardener Vexierbild

Objektorientierte Konzepte

- Abstraktion
- Kapselung
- Hierarchie
- Polymorphismus

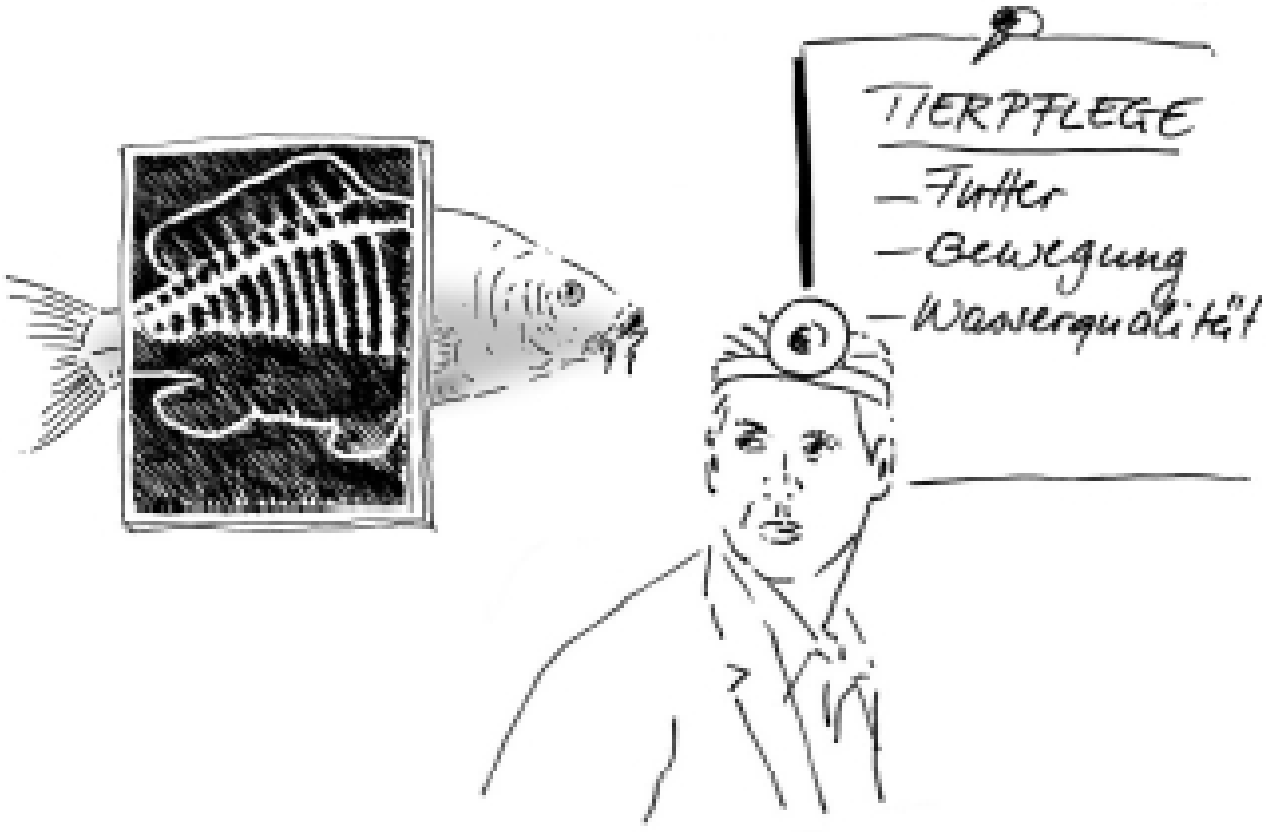
Abstraktion I



Abstraktion II

- Definition
  - Vereinfachte Beschreibung oder Spezifikation eines Systems, die sich auf einige Details oder Eigenschaften des Systems konzentriert und andere dafür vernachlässigt.
- Nutzen
  - Verständlichkeit
- Merke
  - Konzentration auf das Wesentliche

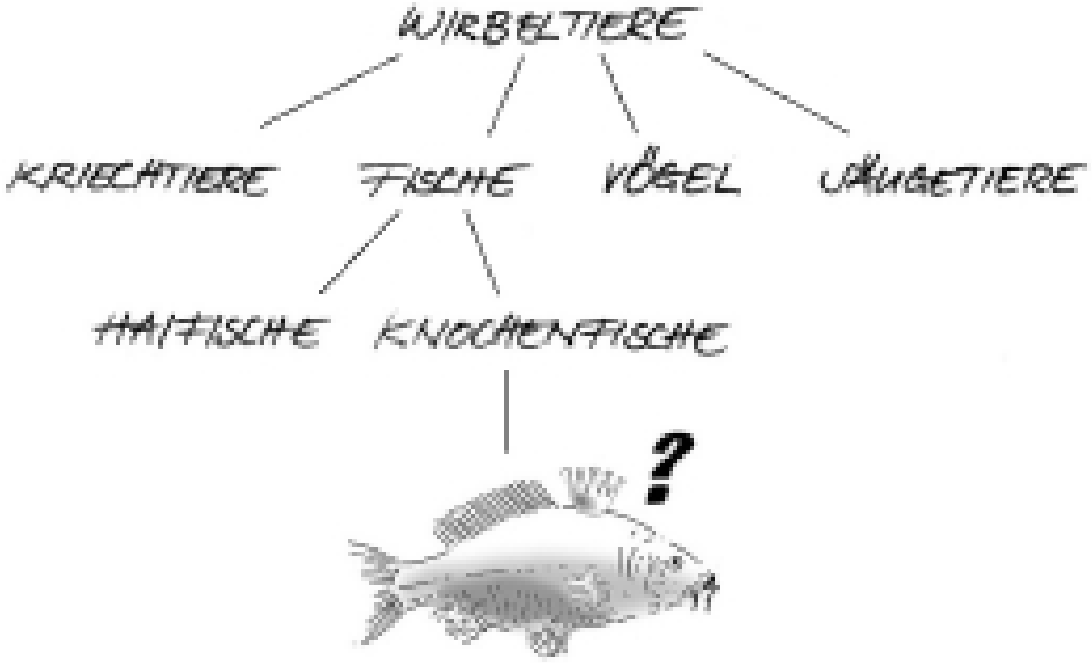
Kapselung I



Kapselung II

- Definition
  - Verbergen von Informationen über ein System, die nicht zu den wesentlichen Charakteristika gehören. Das Prinzip der Abstraktion wird verwendet, um die Schnittstelle einer Abstraktion von ihrer Implementierung zu trennen
- Nutzen
  - Änderbarkeit
- Merke
  - Denken in Black-Boxen

Hierarchie I



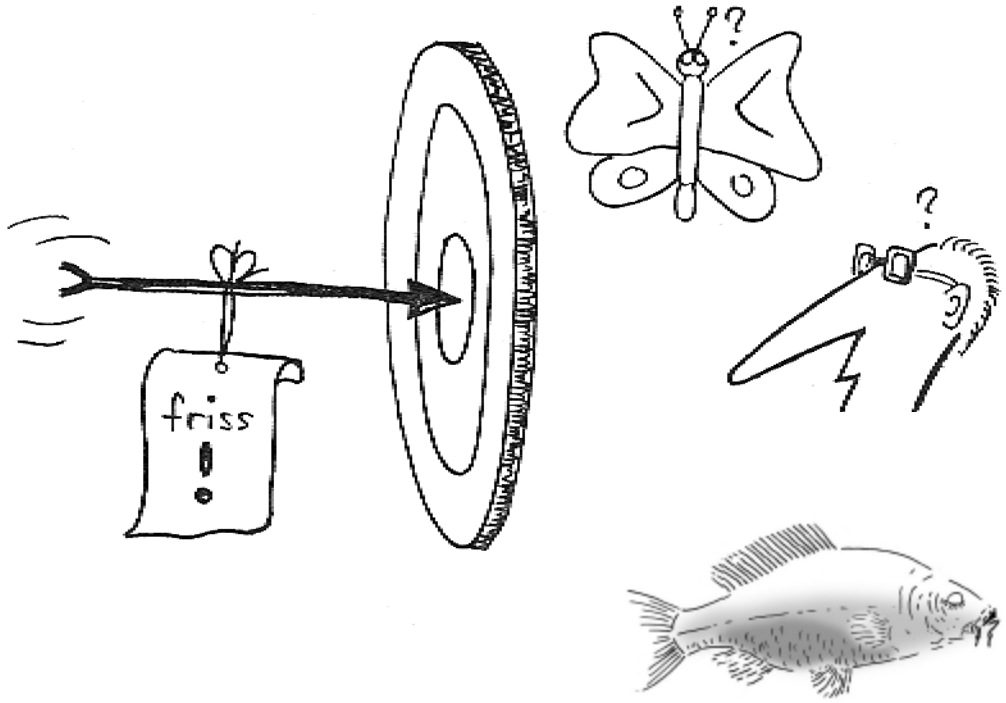
Hierarchie II

- Definition
  - Hierarchie ist eine Rangfolge oder Anordnung von Abstraktionen
- Generalisierung
  - Zusammenfassen von gemeinsamen Eigenschaften oder Verhaltensweisen
- Spezialisierung
  - Erweiterung durch zusätzliche oder abgeänderte Eigenschaften oder Verhaltensweisen

Hierarchie III

- Nutzen
  - Erweiterbarkeit
- Merke
  - Wir suchen Gemeinsamkeiten

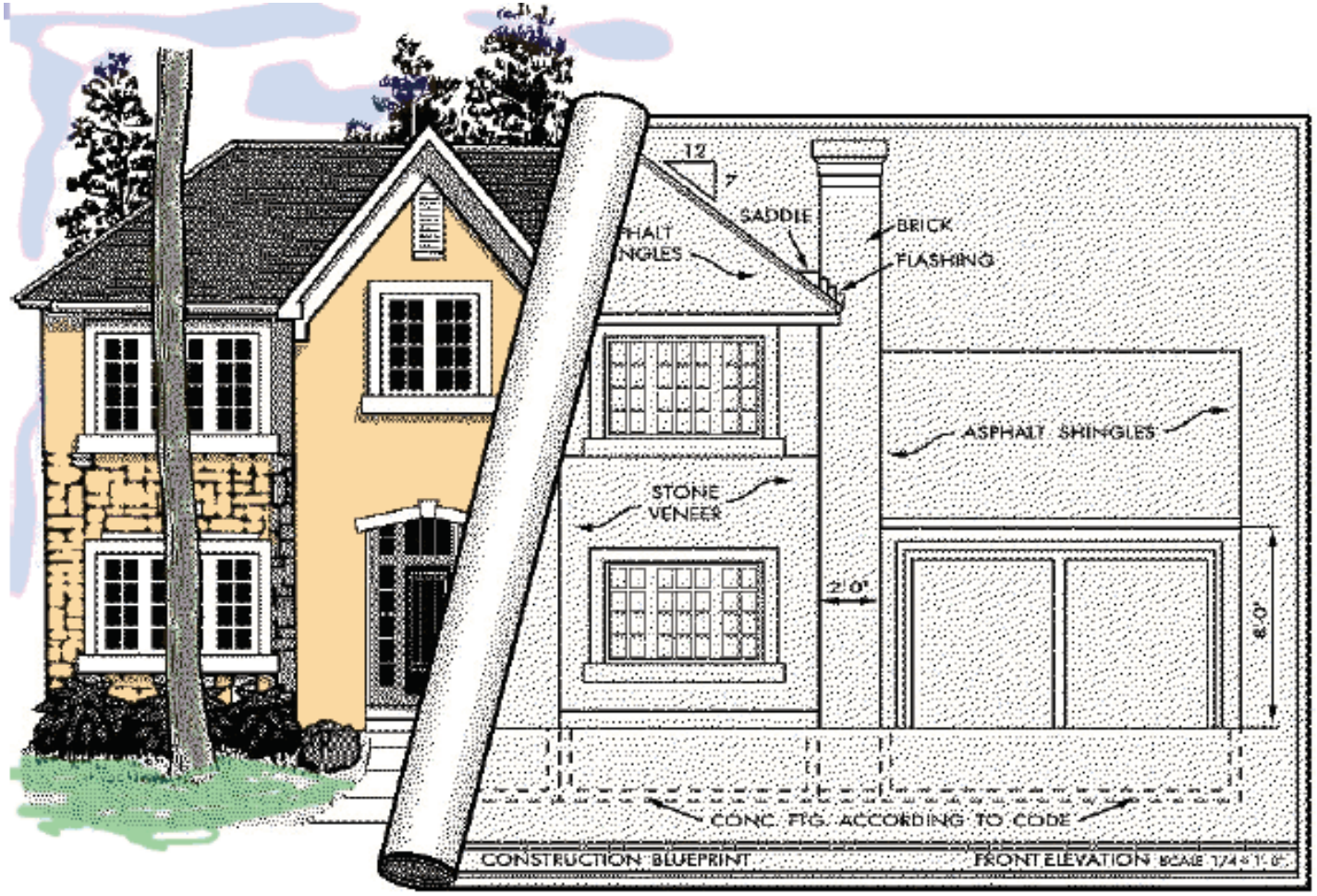
Polymorphismus I



Polymorphismus II

- Definition
  - Hinter einem einzigen Interface können sich mehrere verschiedene Implementierungen verstecken
- Nutzen
  - Vereinfachung
- Merke
  - Wir senden Nachrichten, kümmern uns aber nicht darum, wie die betroffenen Nachrichten individuell bearbeitet werden

Objektorientiertes Modellieren



Unified Modeling Language

- Entstehung als Gemeinschaftsarbeit von Gardy Booch, Jim Rumbaugh und Ivar Jacobson (Drei Amigos).
- Zweck
  - Spezifikation
  - Visualisierung
  - Dokumentation
- Merke
  - UML ist die Beschreibung einer Notation und **keine** Entwicklungsmethode

UML Überblick

- Graphische Modellierungssprache für OO-Modellierung zur Beschreibung von strukturellen, verhaltens- und zustands-orientierten Aspekten
- Die drei Beschreibungsebenen
  - Anforderungsmodell
    - Gewünschtes Zusammenwirken
  - Statisches Modell
    - Anwendung von Vererbung, Assoziation, Abstraktion
  - Dynamisches Modell
    - Verhalten, Austausch von Nachrichten, Aufruf von Methoden

Nutzen der UML

- Definiert nahtlosen Übergang zwischen Analyse, Design und Implementierung
- Definiert aussagekräftige und konsistente Notation
  - Erleichterte Kommunikation
  - Hilfsmittel um Lücken und Inkonsistenz aufzuzeigen
  - Unterstützung von Grob- und Fein-Analyse und auch Grob-und Fein Design

OO Elemente

- Grundlegende Elemente
  - Objekte
  - Klassen
  - Attribute
  - Methoden

OO Elemente

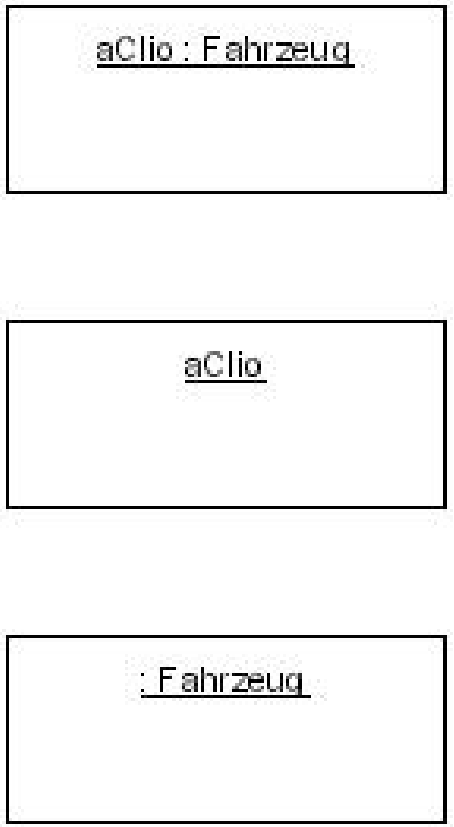
- Beziehungen
  - Abhängigkeit (uses)
  - Assoziation
  - Aggregation
  - Komposition
  - Generalisierung
  - Realisierung

Objekte

- Definition
  - Ein Objekt ist eine Abstraktion eines realen oder gedachten „Gegenstandes“.
  - Jedes Objekt wird durch Eigenschaften beschrieben
  - Mit Objekteigenschaften meint man:
    - seine Identität
    - seine Daten
    - und sein Verhalten

Objekte

- UML Notation
  - Rechteck mit Namen
  - Name fett unterstrichen
  - Bei Bedarf Angabe der Klasse, getrennt mit Doppelpunkt
  - Bei Bedarf anonym mit Angabe der Klasse



Objekte

- Java Notation

```
Fahrzeug aClio = new Fahrzeug("Clio");
```

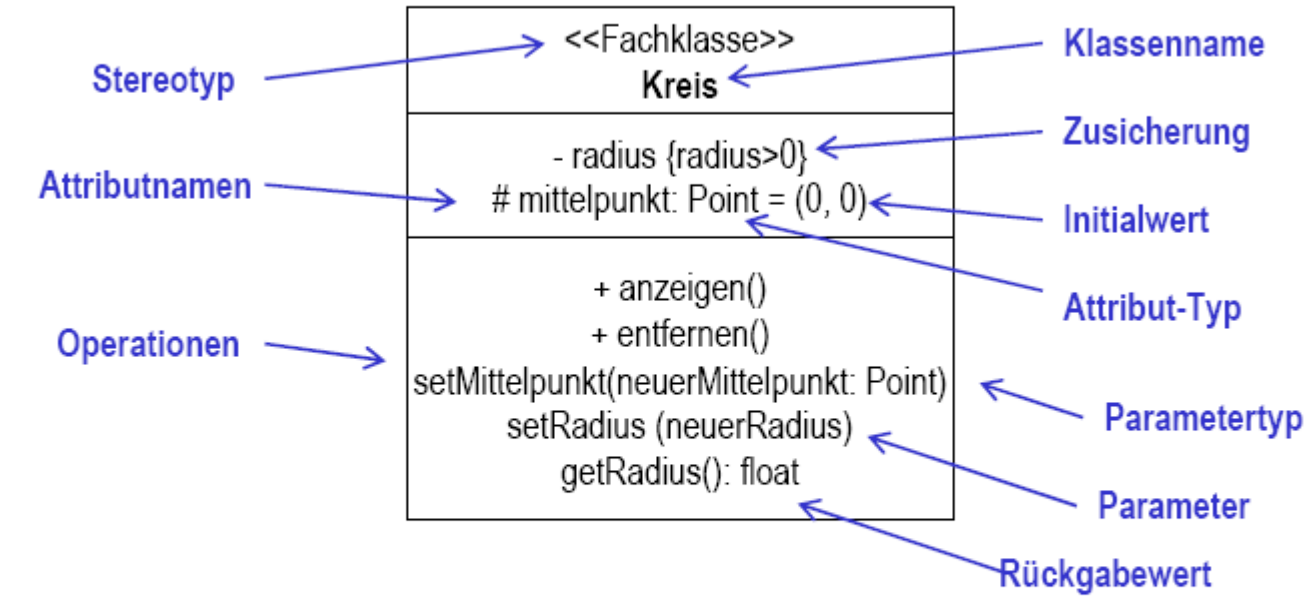
Klassen

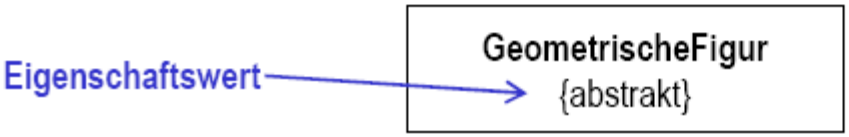
- Definition
  - Eine Klasse beschreibt eine Menge von gleichartigen Objekten (Bauplan)
  - Dazu gehören:
    - Eigenschaftsstrukturen
    - Verhalten
  - Dazu gehört nicht:
    - Eigenschaftswerte
  - Eine Klasse wird auch als abstrakter Datentyp bezeichnet

Klassen

- UML Notation
  - Rechteck mit Namen der Klasse
  - Name fett

Klassen


<<Fachklasse>>
Kreis
- radius {radius>0}
mittelpunkt: Point = (0, 0)
+ anzeigen()
+ entfernen()
setMittelpunkt(neuerMittelpunkt: Point)
setRadius (neuerRadius)
getRadius(): float
Stereotyp
Klassenname
Attributnamen
Zusicherung
Initialwert
Attribut-Typ
Operationen
Parametertyp
Parameter
Rückgabewert


GeometrischeFigur
{abstrakt}
Eigenschaftswert

Klassen

- Java Notation

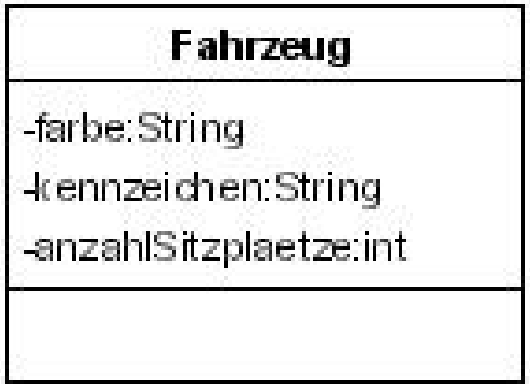
```
class Fahrzeug {

}
```

Attribute

- Definition
  - Ein Attribut ist ein Datenelement, das in jedem Objekt einer Klasse gleichermassen vorhanden ist, und in jedem Objekt mit einem individuellen Wert repräsentiert wird
  - Andere Bezeichnungen: Instanzvariable, Datenelemente oder „member variable“

Attribute

- UML Notation
  - Unterhalb dem Klassennamen, getrennt durch einen Strich



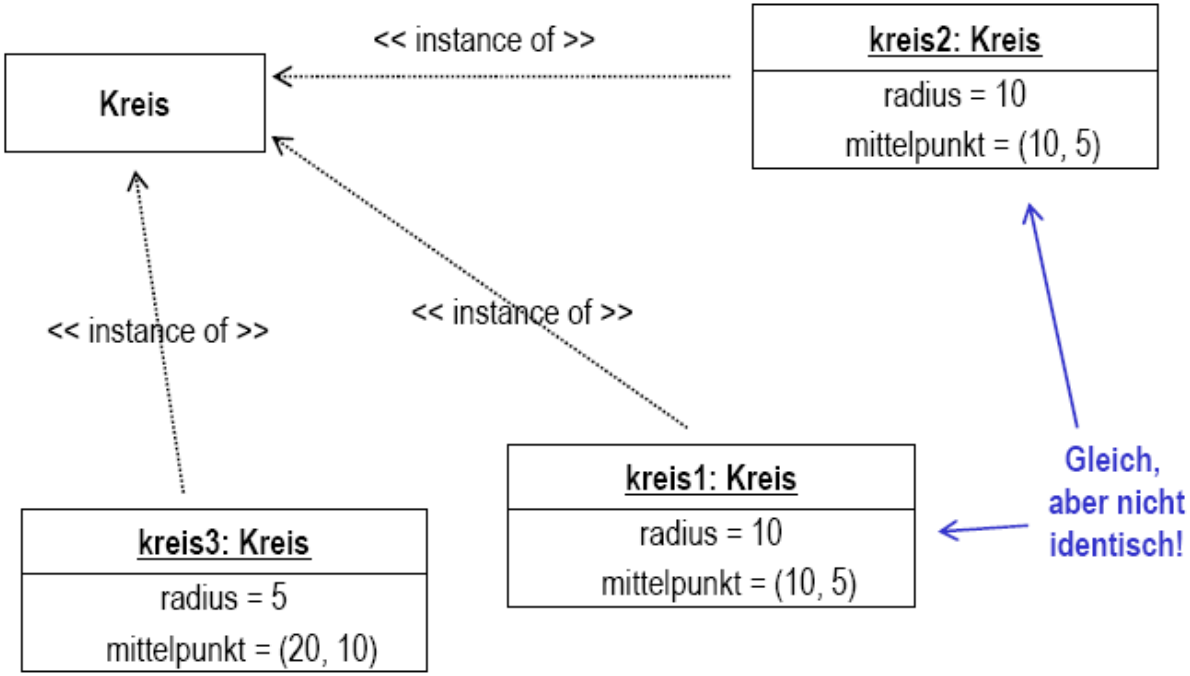
Attribute

- Java Notation

```
class Fahrzeug {

        String farbe;
        String kennzeichen;
        int anzahlSitzplätze;

}
```

Gleichheit versus Identität

- Gleich: Gleiche Werte in den Attributen
- Identisch: dasselbe Objekt, gleiche Speicheradresse

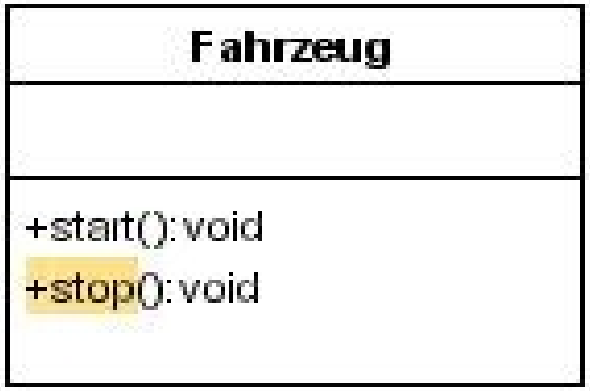


Methoden

- Definition
  - Eine Methode implementiert eine Operation d.h. eine Sequenz von Anweisungen
  - Andere Bezeichnungen: Elementfunktion, Operation

Methoden

- UML Notation
  - Unterhalb dem Klassennamen und der Attribute, getrennt durch Strich



Methoden

- Java Notation

```
class Fahrzeug {

      void start() {
      }

      void stop() {
      }
}
```

Beziehungen

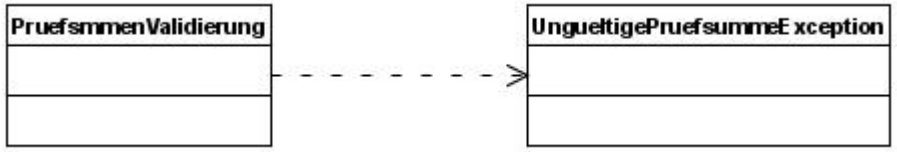
- Einführung
  - Genauso wie reale oder begriffliche Gegenstände des Alltags miteinander in Verbindung stehen, so pflegen auch Objekte bzw. Klassen Beziehungen untereinander

Abhängigkeit (uses)

- Definition
  - Die Abhängigkeit zeigt eine lose Kopplung zwischen zwei Klassen

Abhängigkeit (uses)

- UML Notation
  - Gestrichelter Pfeil von der Klasse die braucht (client) zu derjenigen, die verwendet wird (supplier)



Abhängigkeit (uses)

- Java Notation

```
class PruefsummenValidierung {

  void validate() throws UngueltigePruefsummenException  {

       try {

       }
       catch (Exception ex) {
         throws new UngueltigePruefsummenException();
       }
  }

}
```

Assoziation

- Definition
  - Als Assoziation bezeichnet man eine gleichrangige Beziehung zweier Klassen

Assoziation

- UML Notation
  - Verbindungslinie zwischen den beteiligten Klassen