FHD

Fachhochschule Darmstadt
Fachbereich Informatik

Softwaretechnik I

Kapitel 3 ff

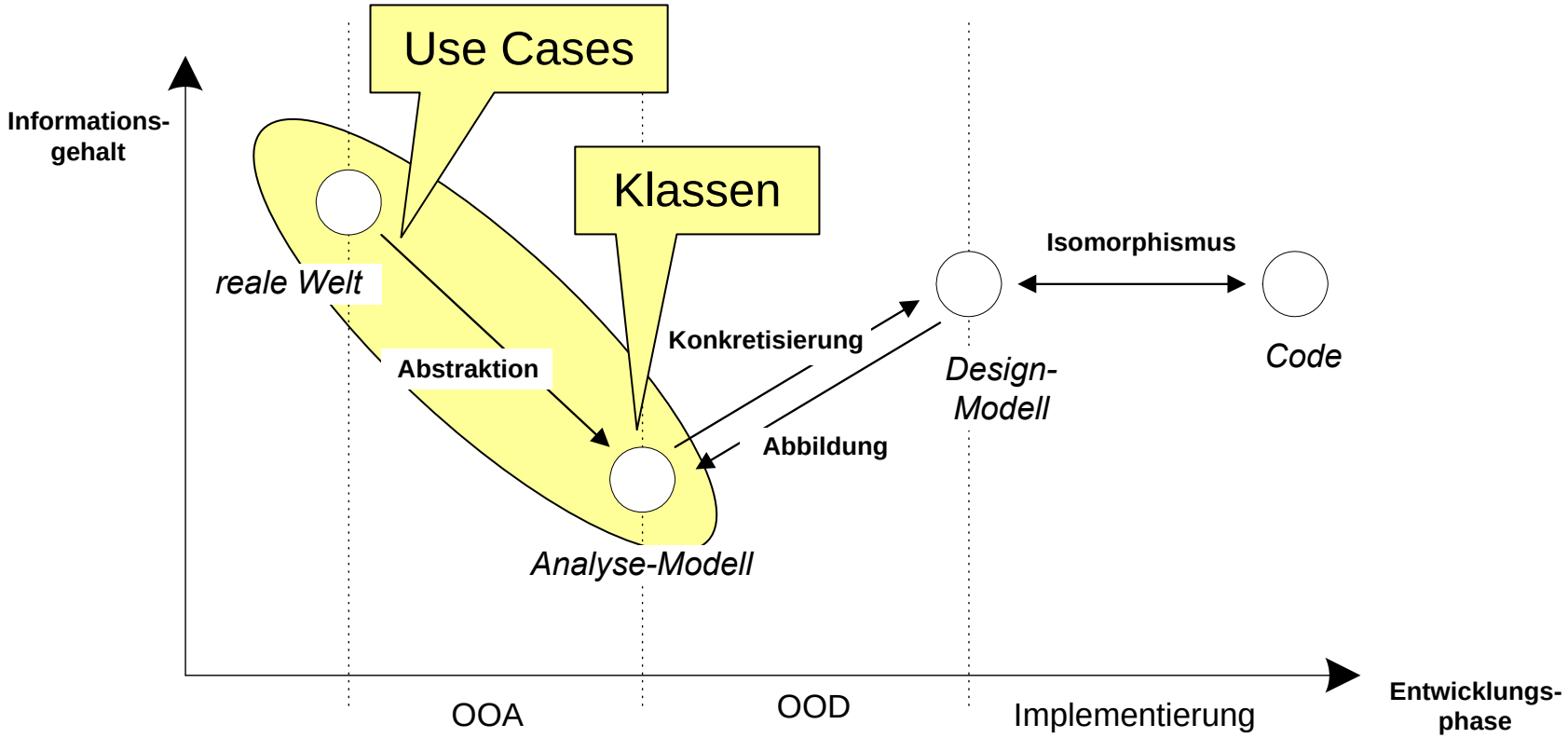
- Klassen & UML

Quellenhinweis:
Einige Folien zu dieser Vorlesung entstammen Präsentationen von Prof. G. Raffius und Prof. W. Weber

Stand im Phasenmodell

Objektorientierte Analyse

- ⇨ **Klassen**



Klassen und Beziehungen zwischen Klassen

Eigenschaften ↘ *Verhalten* ↘

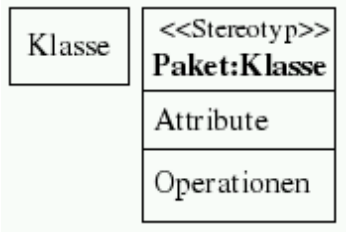
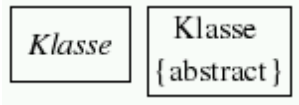
Klasse: Attribute + Operationen

Beziehungen zwischen Klassen:

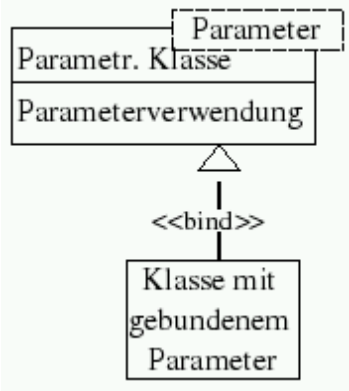
- Generalisierung / Spezialisierung (Vererbung)
- Assoziation
- Aggregation
- Komposition

⇨ **Beschreibung der <u>Statik</u> des Systems**

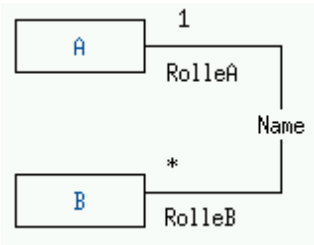
Notation in UML 2.0 (I)

	**Klasse** Ein Klasse kann als Rechteck dargestellt werden, das den Klassennamen enthält. Üblicherweise bestehen Klassen aus drei Bereichen; der obere Bereich enthält den Stereotyp, das Paket zu dem die Klasse gehört und den Namen. Im mittleren Bereich werden die Attribute angegeben und im unteren Bereich stehen die Operationen der Klasse. Laut UML-Spezifikation kann die Darstellung einer Klasse zusätzliche Bereiche enthalten.
	**Abstrakte Klasse** Der Name einer abstrakten Klasse wird kursiv geschrieben. Alternativ kann die Eigenschaft {abstract} angegeben werden.

Notation in UML 2.0 (II)

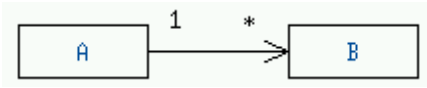
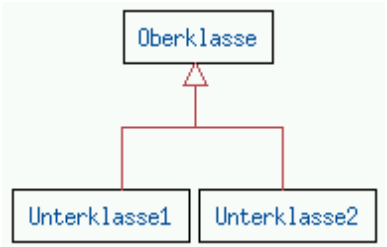


Parametrisierte Klasse
auch Template oder Schablone genannt. Die parametrisierte Klasse hat in der rechten, oberen Ecke ein das Klassensymbol überlappendes Rechteck, das die Schablonen-Parameter der Klasse enthält. Die Funktion, die den Parameter verwendet wird angegeben. Die Klasse, die den Parameter bindet, wird über eine gestrichelte Linie mit Pfeil an dem Template verbunden. Diese trägt die Bezeichnung <<bind>>.

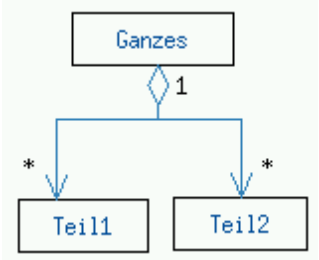


Assoziation
Eine Linie zwischen den Klassen stellt eine Assoziation dar. Eine Assoziation ist eine Beziehung zwischen Klassen. Die Objekte der Klassen kommunizieren über die Assoziationen miteinander. Die Assoziation kann einen Namen haben. Ein Pfeil an dem Assoziationsnamen gibt die Leserichtung des Namens an. An den Assoziationsenden können die Rollen der beteiligten Klassen und die Multiplizität angegeben werden. Die zweigliedrige Assoziation kann, wie die mehrgliedrige Assoziation, durch eine Raute markiert werden.

Notation in UML 2.0 (III)

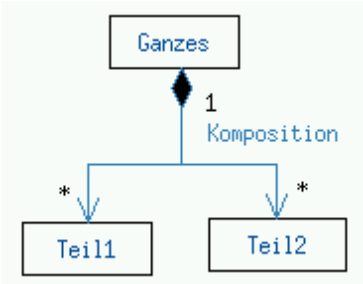
	**Gerichtete Assoziation** Mit einem Pfeil an der Assoziation kann die Navigationsrichtung angegeben werden. Der Pfeil drückt die Zugriffsrichtung der Objekte aus. Objekt A greift auf B zu, B greift nie auf A zu.
	**Vererbung** auch Generalisierung/Spezialisierung genannt. Vererbungsbeziehungen werden mit einem Pfeil dargestellt. Die Pfeilspitze zeigt auf die Oberklasse. Die Oberklasse vererbt ihre Eigenschaften an die Unterklassen.

Notation in UML 2.0 (IV)



Aggregation

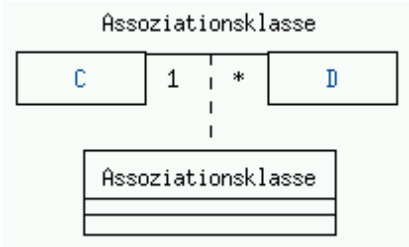
Eine Aggregation drückt eine Teile-Ganzes-Beziehung aus. Das Ganze-Objekt besteht aus Teil-Objekten. Die Raute befindet sich an dem Ende des Ganzen. Die Aggregation ist eine spezielle Art der Assoziation. Da das Ganze die Teile enthält, sollten am Assoziationsende der Teile ein Navigationspfeil stehen.



Komposition

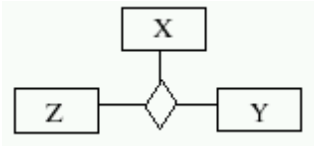
Die Komposition ist auch eine Beziehung, die Teile zu einem Ganzen in Beziehung setzt. Die Teile und das Ganze sind bei dieser Beziehung existenzabhängig; die Teile können nicht ohne das Ganze existieren. Wird das Ganze gelöscht, so beenden auch die Teile ihre Existenz.

Notation in UML 2.0 (V)



Assoziationsklasse

Ist eine Klasse von dem Vorhandensein einer Assoziation zwischen zwei Klassen abhängig, so kann dies durch eine Assoziationsklasse ausgedrückt werden. Die Assoziationsklasse beschreibt Eigenschaften, die keiner der an der Assoziation beteiligten Klassen sinnvoll zuordenbar sind. Die Assoziationsklasse wird über eine gestrichelte Linie mit der Assoziation, von der sie abhängt, verbunden. Hat die Assoziation einen Namen, dann muss die Assoziationsklasse den selben Namen erhalten. Die Assoziationsklasse ist ein Analysekonzept.



Mehrgliedrige Assoziation

Eine mehrgliedrige Assoziation drückt eine Beziehung zwischen mehr als 2 gleichwertigen Klassen aus. Die Beziehung wird mit einer Raute markiert.

Wie findet man Klassen ?

- Aus den Beschreibungen der einzelnen Use Cases:
  Suche nach Substantiven, z.B.
  - Personen, Orte, konkrete Dinge
    (Artikel, Rechnungen, Abteilung, Meßwerte etc.)
  - Schnittstellen eines Systems
    (Masken, Anbindungen an Datenbanken etc.)
  - Abstrakte Dinge (ein Algorithmus, eine Information etc.)
- Identifiziert zuerst Objekte
- Aussondern überflüssiger Klassen. (Ist Person eigenes Objekt oder nur Rolle? (z.B. Klasse Person. Rollen: Kunde, Vorgesetzter etc.)
- Spätere Ergänzungen aus Sequenzdiagrammen, Zustandsdiagrammen.
- Weitere Hinweise für Klassen: Anwendungen von Entwurfsmustern (Design).

Wie findet man Operationen und Attribute ?

Attribute ergeben sich aus

- ⇨ Beobachtungen des Anwendungsgebiets (Domäne) *(= Eigenschaften der Objekte)*
- ⇨ oder aus Vorgängersystemen. *(Formulare, Bildschirmmasken, Datenmodell etc.)*
- ⇨ oder aus den Use Cases
- ⇨ und auch bei dynamischer Modellierung *(Szenarien, Zustandsdiagramme)*

Operationen (Methoden)

- ⇨ Aus dynamischer Modellierung des Systems. *(Hier wird beschrieben, wie sich die Objekte des Systems verhalten und interagieren.)*
- ⇨ Ableitung aus Sequenzdiagrammen und Zustandsdiagrammen.

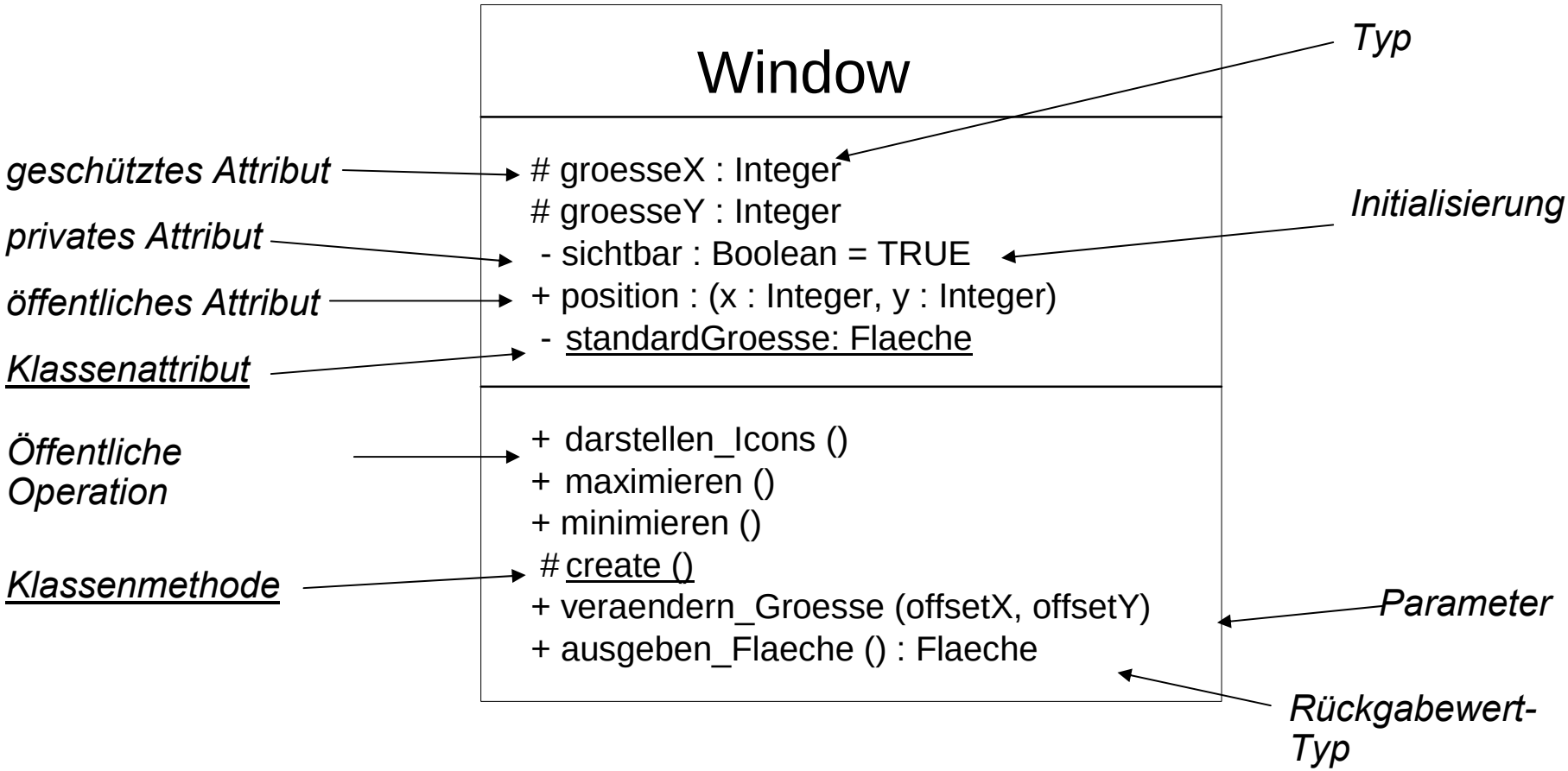
Wie findet man Klassen, Operationen und Attribute ? Beispiel (Diskussion)

Aus einer Use Case Beschreibung:

Ein Dauerauftrag überweist einen bestimmten Betrag in regelmäßigen Zeitabständen von einem Girokonto auf ein Anderes.
Der Dauerauftrag gehört immer zu einem bestimmten Girokonto.
Daueraufträge können angelegt und wieder gelöscht werden.

⇨ Dauerauftrag	Klasse
⇨ Betrag	Attribut für Klasse Dauerauftrag
⇨ Zeitabstand	Attribut für Klasse Dauerauftrag
⇨ Girokonto	Assoziation von Dauerauftrag? Oder Attribut?
⇨ Anderes (Girokonto)	Attribut für Klasse Dauerauftrag
⇨ Anlegen	Operation für Klasse Girokonto? Oder Konstruktor?
⇨ Löschen	Operation für Klasse Girokonto? Oder Destruktor?

Darstellung von Klassen, Attributen und Methoden



Anzeigemodus kann im CASE-Tool eingestellt werden!

Darstellung von Klassen, Attributen und Methoden (II)

Klassenmethode:

- Bezieht sich nicht auf Instanz (Instanzmethode) sondern auf die Klasse als die Menge aller ihrer Objekte.
  z.B.: Konstruktor, Berechnung Durchschnittsgehalt etc.

Klassenattribute:

- Attributwert ist nicht einer Instanz (Instanzattribut) sondern der Klasse zugeordnet und existiert nur 1x für die Klasse.
  z.B.: maxGeschwindigkeit eines Fahrzeugtyps

Methoden für das Schreiben und Lesen von Attributen:

- Sind trivial $\Rightarrow$ werden nicht in die Klassendef. auf Analyse-Ebene aufgenommen.

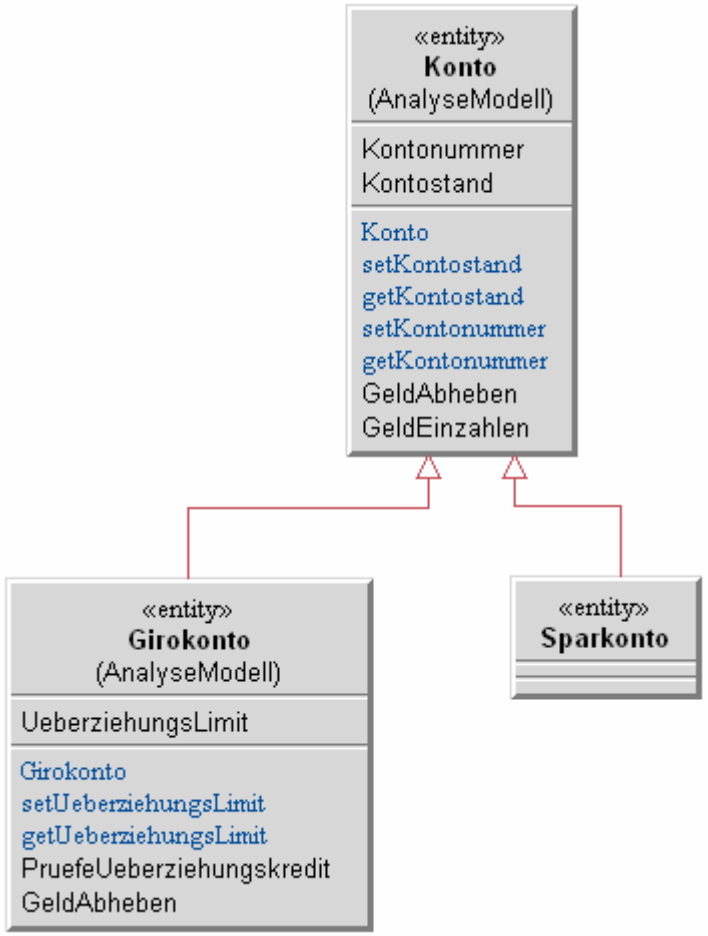
Bei Verwendung eines CASE-Tools:

- Einzelangaben können, um die Übersicht des Gesamtdiagramms zu erhöhen, ausgeblendet werden.

Vererbung / Generalisierung / Spezialisierung

Aus der Menge der Klassen sucht man nach Klassenpaaren wobei:

- ⇨ Die eine Klasse (Superklasse) ist ein allgemeinerer Typ und die andere Klasse (Subklasse) ist ein speziellerer Typ, den die Superklasse erweitert.
  - ⇨ Es besteht eine „Ist-Ein“ – Beziehung !
- ⇨ Ein Objekt der Superklasse ist durch ein Objekt der Subklasse ersetzbar!
- ⇨ Unter der Subklasse ist eine Teilmenge der Menge der Instanzen der Superklasse angesiedelt.



Vererbung / Generalisierung / Spezialisierung

Die Subklasse besitzt alle Merkmale (Attribute, Operationen, Assoziationen, Aggregationen) der Superklasse und noch zusätzliche Merkmale.

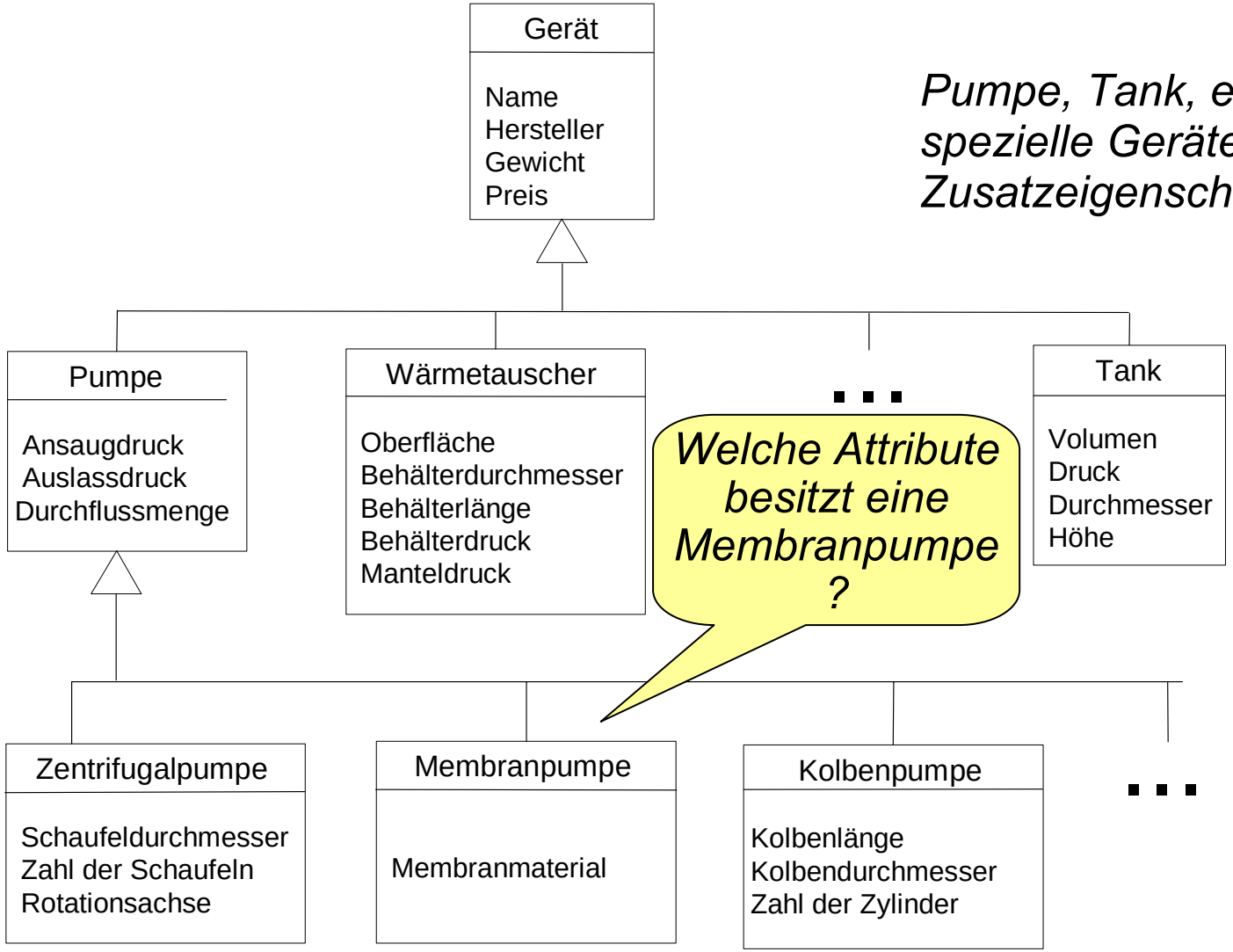
⇨ *Die Subklasse erbt die Merkmale der Superklasse.*

Vererbung: Alle Merkmale werden von der Superklasse auf die Subklasse vererbt.

Generalisierung: "Ist-ein-Beziehung" von der Subklasse aus gesehen.
Ein Konto ist ein (verallgemeinertes) Girokonto

Spezialisierung: "Ist-ein-Beziehung" von der Superklasse aus gesehen.
Ein Girokonto ist ein (spezialisiertes) Konto

Vererbung / Generalisierung / Spezialisierung: Beispiel



Pumpe, Tank, etc. als spezielle Geräte mit Zusatzeigenschaften.

Frage: Welche Attribute besitzt das Exemplar Membranpumpe ?

Instanzen:

(Membranpumpe)

Name = P101
Hersteller = Simplex
Gewicht = 100 kg
Preis = 8.000 €
Ansaugdruck = 1,1 atm
Auslassdruck = 3,3 atm
Durchflussrate = 300 l/h
Membranmat. = Teflon

(Wärmetauscher)

Name = E302
Hersteller = Braun
Gewicht = 5000 kg
Preis = 30.000 €
Oberfläche = 300 m
Behälterdurchm. = 2 cm
Behälterlänge = 6 m
Behälterdruck = 15 atm
Manteldruck = 1,7 atm

(Tank)

Name = T111
Hersteller = Simplex
Gewicht = 10.000 kg
Preis = 80.000 €
Volumen = 400.000 l
Druck = 1,1 atm
Durchmesser = 8 m
Höhe = 9 m

Vererbung / Generalisierung / Spezialisierung: Sichtbarkeit

- Man unterscheidet (allgemein) 4 verschiedene Sichtbarkeitsstufen:
  - Private: außerhalb der Klasse nicht sichtbar
    (in der UML: -), auch nicht für Unterklassen
  - Protected: für alle Unterklassen sichtbar
    (in der UML: #)
  - Public: für alle anderen Klassen sichtbar, d.h. öffentlich
    (in der UML: +)
  - Package: für Klassen im gleichen Paket sichtbar
    (in der UML: ~)
- Verschiedene Stufen der Sichtbarkeit innerhalb einer Vererbungshierarchie
- Jede Klasse „kennt“ nur ihre eigenen Attribute, Operationen und Assoziationen und die ihrer Oberklassen, sofern diese für sie sichtbar sind

Vererbung / Generalisierung / Spezialisierung: Regeln

- In Subklassen dürfen nicht nur zusätzliche Merkmale definiert werden, sondern auch geerbte Merkmale überschrieben werden:

  - Operationen (Reimplementierung / Polymorphie)
  - Attribut-Typen
  - Argumente und Rückgabewerte von Operationen

Aber:

Durch ein Überschreiben darf

nie die Semantik des Attributs bzw. der Operation geändert werden!

(Die Ist-ein-Beziehung darf nie verletzt werden !)

Einschränkungen bezüglich des Überschreibens der Merkmale (I)

- **Der Wertebereich von Attributen darf eingeschränkt, jedoch nicht erweitert werden.**

  z.B.: Superklasse Mitarbeiter: Gehalt 1... 150.000,- €

  Subklasse Arbeiter: Gehalt 1... 80.000,- €

- **Der Typ eines Attributs der Subklasse muss vom Typ der Superklasse isomorph umschlossen sein.**

  z.B.: Superklasse Attribut von Typ REAL

  in Subklasse Attribut von Typ INTEGER

  z.B.: Superklasse Attribut von Typ der Klasse KONTO

  in Subklasse Attribut von Typ einer Subklasse von KONTO (z.B.: GIROKONTO)

Einschränkungen bezüglich des Überschreibens der Merkmale (II)

- **Integritätsbedingungen einer Klasse dürfen nicht verletzt werden:**

  z.B.: Ellipse-Kreis:

  Eine Ellipse ist keine Spezialisierung eines Kreises, da die Kreiseigenschaft zweier gleichlanger Achsen durch die Ellipse verletzt würde. Ein Kreis kann jedoch eine spezielle Ellipse sein, bei der die Achsen gleich lang sind.

- **Überschreiben von Methoden:**

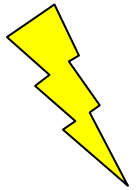
  Andere Implementierungen (z.B. Performance-Verbesserung) mit gleicher Funktionalität sind erlaubt. Die Schnittstelle (Name, Anzahl von Argumenten) der Operation der Superklasse muss jedoch eingehalten werden. Typen von Argumenten und Rückgabewerten müssen den o.g. Bedingungen für das Überschreiben von Attributen genügen.

Hinweise zum Finden von Generalisierungs- & Spezialisierungs-Beziehungen

Gleiche Attribute, Operationen, Assoziationen, Aggregationen in verschiedenen Klassen

⇒ gemeinsame Merkmale in der Superklasse.

Aber: Es muss eine „ist-ein-Beziehung“ bestehen und darf nicht nur alleine der Erleichterung der Implementierung dienen.

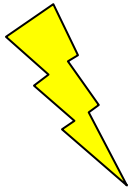


***Bsp**.: Dieselben Attribute, aber keine „ist-ein-Beziehung“.*

1. Klasse Fenster und Klasse Rechteck:

Attribute: Fläche und Größe

Aber haben aus semantischer Sicht keine gemeinsame Superklasse.



2. Klasse Rollstuhl und Klasse Kraftfahrzeug:

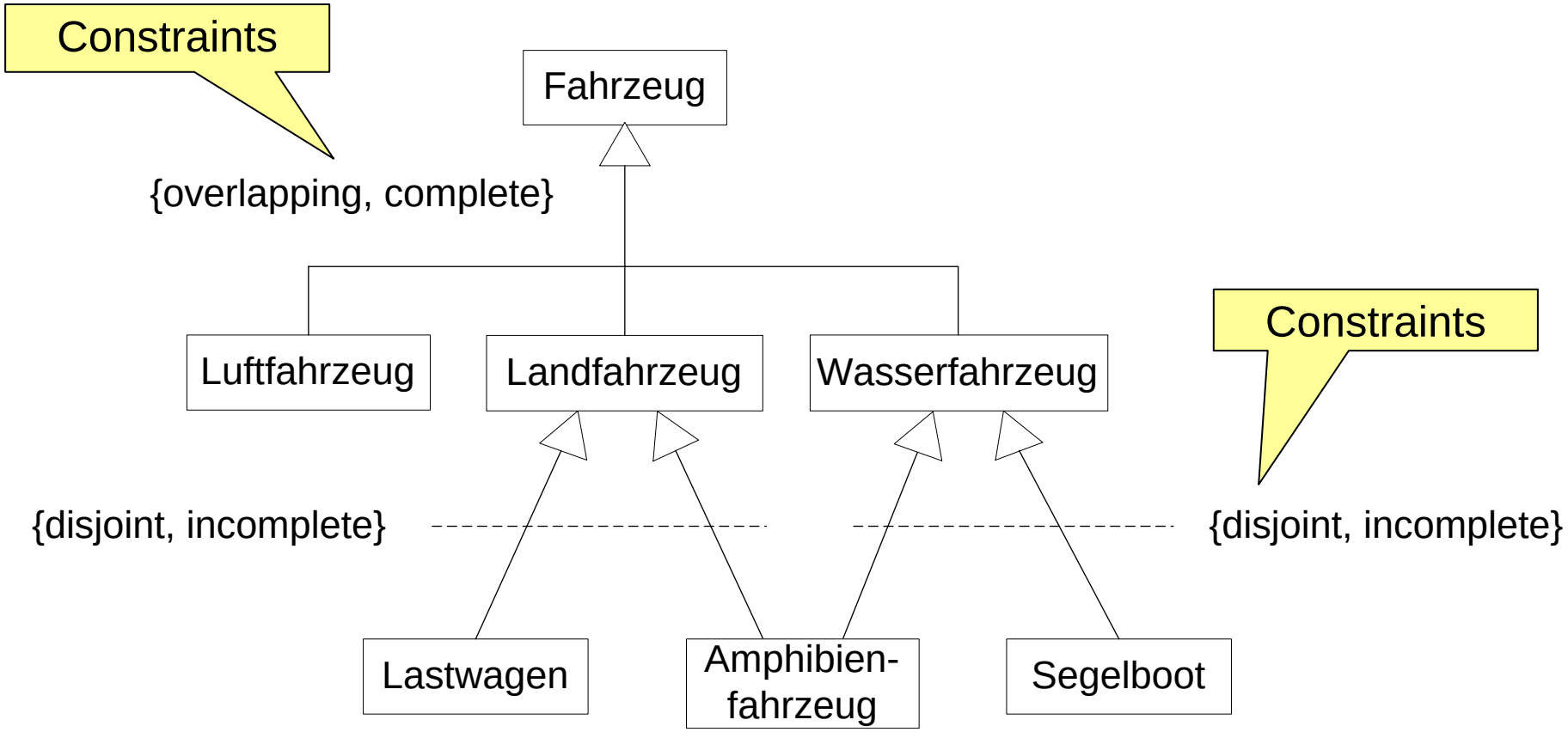
Attribute: Gewicht, Anzahl Räder, Farbe

Operation: fortbewegen

Kraftfahrzeug besitzt zusätzliches Attribut kW

Trotzdem ist Kraftfahrzeug keine Spezialisierung eines Rollstuhls.

Randbedingungen für Vererbung / Mehrfachvererbung



Default: {overlapping, incomplete} – wird weggelassen

Unterschiede zwischen Methoden und Operationen

Operation: Ein best. Verhalten, das eine Klasse für ihre Objekte definiert. Dazu gehören auch alle geerbten Operationen. Eine Operation definiert Schnittstelle sowie Vor- und Nachbedingungen eines bestimmten Verhaltens.

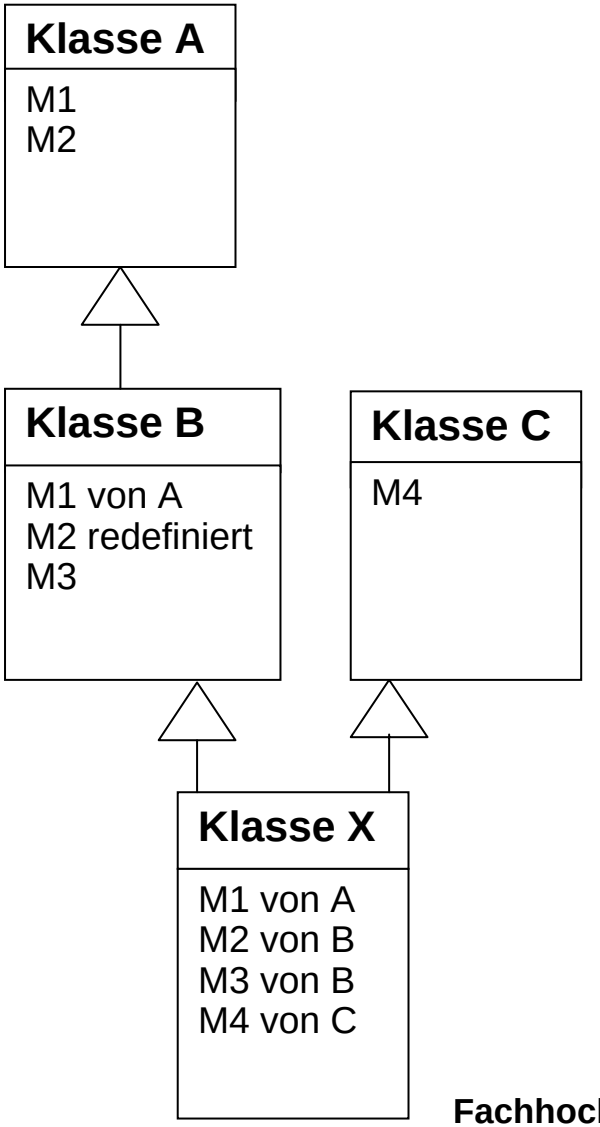
Methode: Die Implementierung/Ausprägung einer Operation. Die Implementierung muss die Vorgaben der Operationsdefinition einhalten.

Bsp.: graph. Objekt: *Operationen:* *berechneFlaeche(), drehe()*
Methoden: *g*h (Rechteck), π r² (Kreis), entspr. Operation zum Neuzeichnen*

Unterschiede zwischen Methoden und Operationen (Aufgabe)

Frage:

In welchen Klassen sind M1 und M2 Methoden?
In welchen Operationen?



Die Unterscheidung wird häufig nicht gemacht!

Die Begriffe werden synonym verwendet!

Abstrakte Klassen (I)

Eine abstrakte Klasse ist eine Klasse, in der mindestens eine abstrakte Operation vorkommt.

abstrakte Operation:

- besitzt keine Implementierung (keine Methode)
- definiert nur Schnittstelle, Vor- und Nachbedingungen

⇒ **abstrakte Klassen können keine Instanzen bilden.**

Wozu ?

Klassenhierarchien sind „Begriffshierarchien“.

Abstrakte Klassen dienen dem Finden von Oberbegriffen für Klassen, um allgemeingültige Klassenhierarchien und intuitives Verständnis zu erreichen.

⇒ **Abstrakte Klassen definieren Schnittstellen für Verhalten !**

Abstrakte Klassen (II)

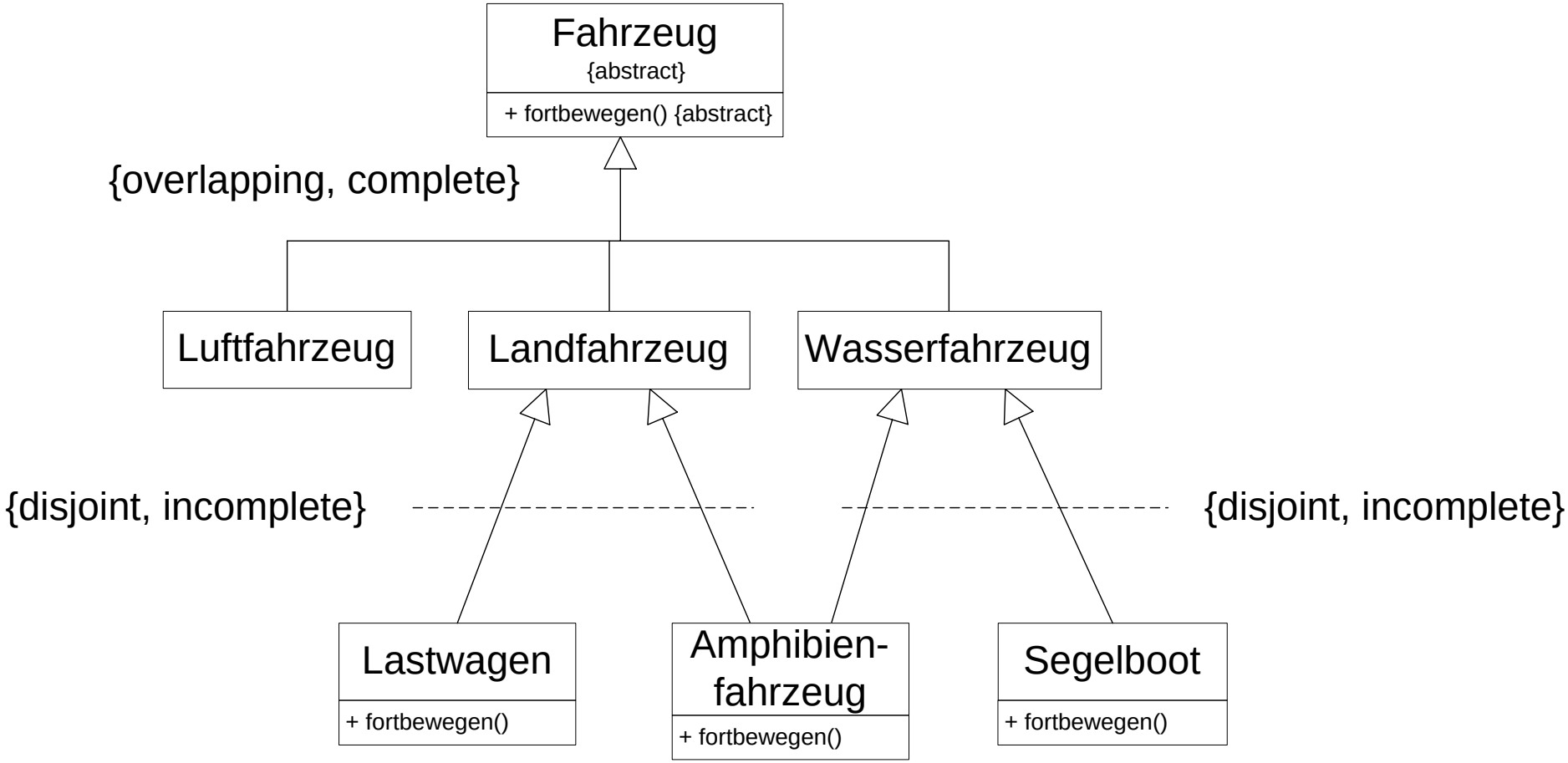
Klassen, die nicht abstrakt sind und Instanzen bilden können, werden *konkrete* Klassen genannt.

Konkrete Klassen, die von abstrakten Klassen erben, definieren die abstrakten Operationen ihrer Superklassen, d.h. sie implementieren Methoden dafür.

⇨ Polymorphie
gleiche Methoden anwendbar auf unterschiedliche Objekte (innerhalb einer Vererbungshierarchie)

⇨ dynamisches / spätes Binden
Die Verknüpfung zwischen Methodenaufruf und Methodenimplementierung findet erst zur Laufzeit statt (Wird nicht von allen Sprachen unterstützt!)

Abstrakte Klassen (Beispiel)



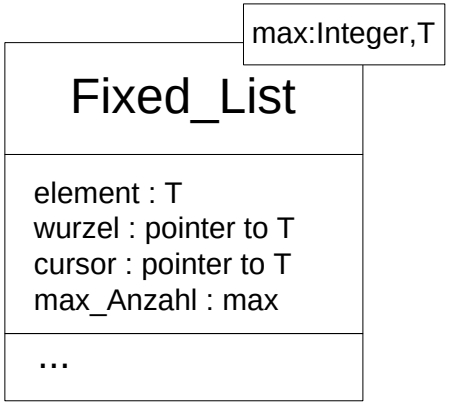
Was ist das ?

Bei der Definition einer Instanz einer parametrisierten Klasse:

⇨ Angabe konkreter Parameter ⇒ Ausprägung zur Kompilierzeit.
Dadurch ist es möglich typenunabhängige Algorithmen zu definieren und zur Kompilierzeit zu konkretisieren.

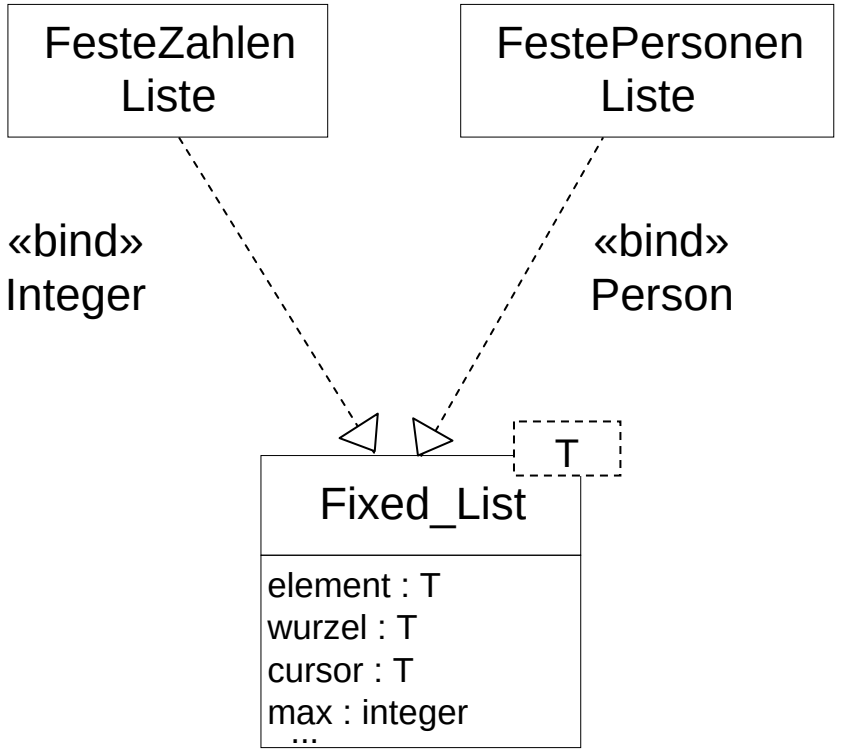
z.B.: Fixed_List:
*Instanz kann die unter **max** angegebene Anzahl von Elementen von dem angegebenen Typ (Integer, String, Kunde,...) enthalten.*

Anwendung:
Standardalgorithmen und Datenstrukturen
z.B. Graphen, Bäume, Listen, Sortierungen, Suche



Parametrisierte Klassen (Templates) (II)

Eine parametrisierte Klasse kann nicht direkt von anderen Klassen benutzt werden. Ihre generischen Parameter müssen erst an einen konkreten Typ „gebunden“ werden. Diese gebundenen Klassen können dann von anderen Klassen assoziiert werden.



Diskriminator (Klassifizierungsmerkmal)

- Unterscheidungsmerkmal für die Strukturierung der Modellsemantik in Generalisierungs- bzw. Spezialisierungsbeziehungen

- Es kann zu einer Klasse eine oder auch mehrere Spezialisierungen (Klassifizierungen) geben.

***Bsp.**: Fahrzeug (s.o.):*
Klassifizierung nach Umgebung, in der sich das Fahrzeug bewegt (eine Klassifizierung).

oder:
zusätzliche Klassifizierung nach Antriebsart (zweite Klassifizierung)

Diskriminator (Klassifizierungsmerkmal) (Beispiel)