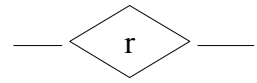
Kap.7.5 UML-Modellierung

Kap.7.5.1 Differenzierung von Beziehungen:

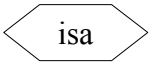
bisher



1	1
1	n
n	m

in UML zusätzlich:

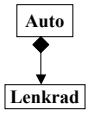
Generalisierung:



Komposition:

Komposition: Eine Instanz von B kann Komponente von höchstens einer Instanz von A sein,

z.B.



1:1 immer implizit

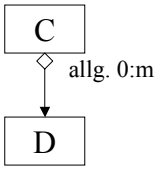
1

Auto

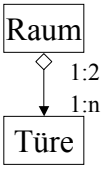
Rad

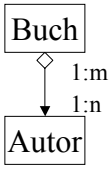
Übung: Auto mit 1 Motor, 1 Karosse, 5 Rädern, Sonderausstattung

Aggregation:



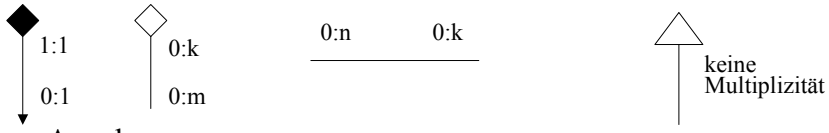
Eine Instanz d von D kann zu mehreren Instanzen c von C gehören,
z.B.





2

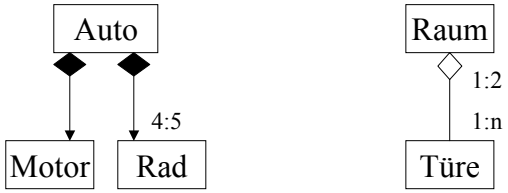
Multiplizitäten allgemein



ohne Angabe
per default, d.h. implizit

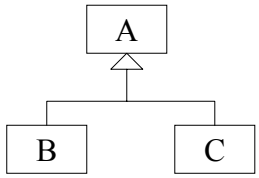
sonstige Multiplizitäten sind explizit
anzugeben, z.B. 4:4 ≡ 4 bei Autorädern
oder 4:5 mit Ersatzrad
oder 4:8 mit Winterbereifung
oder 4:9 mit Ersatz- und Winterbereifung

Beispiel:



3

Rel. Schema für Isa:



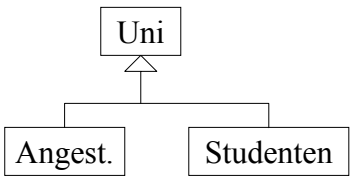
B, C erben alle Attribute von A
3

⇒ gemeinsame Attribute
in Relation für A
C

Isa-Schema 1:
key **ak**

A (**ak,** a1, ..., ai)
B (**ak,** b1, ..., bj)
C (**ak,** c1, ..., ce)

⇒ keine Redundanz oder Integritätsprobleme, dafür Joins zur Konstruktion von B-Instanzen und C-Instanzen



es gibt aber auch Uni-Mitglieder z.B. EHL oder Ehrensenatoren, die weder Angestellte noch Studenten sind.

4

Isa-Schema 2: mit redundanter Speicherung: d.h. alle B-Instanzen und C-Instanzen sind in A repliziert.

A (**ak,** a1, ..., ai)
B (**ak,** a1, ..., aj, b1, ... bj) mit geerbten Attributen
C (**ak,** a1, ..., ai, c1, ..., ce) mit geerbten Attributen

Integritätsbedingungen:

$A.ak = B.ak \Rightarrow A(ak, a1, ..., ai) = B(ak, a1, ...ai)$
$A.ak = C.ak \Rightarrow A(ak, a1, ..., ai) = C(ak, a1, ...ai)$
$B.ak = C.ak \Rightarrow B(ak, a1, ..., ai) = C(ak, a1, ...ai)$

d.h. ak ist systemweite OID

5

Isa-Schema 3: Ohne redundante Speicherung,
Vermeidung von Joins, sonst wie Isa-Schema 2

$A \cap B = A \cap C = B \cap C = \varnothing \wedge$

$\forall a \in A \; \forall b \in B \; \forall c \in C : a.ak \neq b.ak \wedge a.ak \neq c.ak \wedge b.ak \neq c.ak$

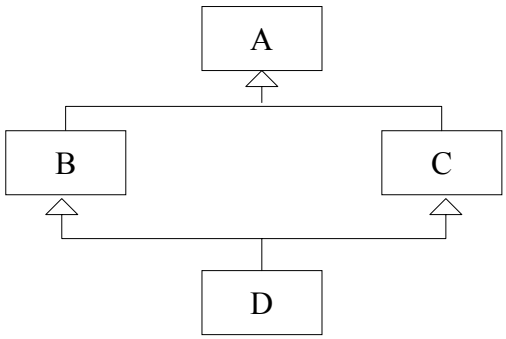
d.h. jede Instanz wird nur in speziellster Klasse gespeichert,
vollständige Instanzenmenge durch π und $\cup$ über Subhierarchie

Isa-Schema 4: *ohne Rel. für A*
define view A **as**
select ak, a1 ..., ai **from** B
union
select ak, a1, ... ai **from** C

macht nur Sinn für abstrakte Klasse A, die selbst keine Instanzen hat

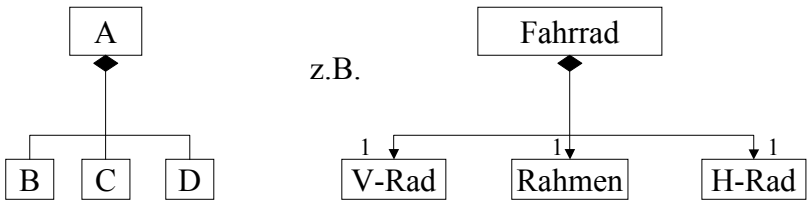
6

Übung: Entsprechende Schemata für multiple Vererbung, i.e.



7

Rel. Schema für Komposition:



Komp.-Schema 1: Komposition als schwache Entität: Fahrrad wie **schwache Entität** behandeln mit 4 Relationen, wobei Fahrrad aus Fremdschlüsseln und Zusatzattributen besteht,

z.B. Fahrrad 1= (**Vid**..., **Rid**..., **Hid**..., Preis)
zwei Löschoperationen:
schwaches delete = flat delete = disassemble
starkes delete = deep delete = mit Teilen zerstören

8

Komp.-Schema 2:
Fahrrad bekommt eigenen Indentifikator Fid, der bei den Komponenten gespeichert wird: ⇒ Komposition als starke Entität

Fahrrad2 = (**Fid**, Preis)
V-Rad2 = (**Vid**, Fid, ...)
Rahmen2 = (**Rid**, Fid, ...)
H-Rad2 = (**Hid**, Fid, ...)

Hinweis: $\overset{\blacklozenge}{\uparrow}_1$ erlaubt Speicherung der Schlüssel auf einer der beiden Seiten. $\overset{\blacklozenge}{\uparrow}_*$ erfordert Liste auf *-Seite, deshalb (ListID, El#).

```
define view  Fahrrad1 as
        select  v.Vid,   v. ... ,
                r.Rid,   r. ... ,
                h.Hid,   h. ... ,
                f.Preis
        from    V-Rad2   v,
                Rahmen2  r,
                H-Rad2   h,
                Fahrrad2 f
        where   f.Fid = v.Vid   and
                f.Fid = r.Rid   and
                f.Fid = h.Hid
```

9

Komp.-Schema 3:
Komp. als starke Identität und zusätzlich mit Fremdschlüssel

Fahrrad3 = (**Fid**, Vid, Rid, Hid, Preis)
V-Rad3 = (**Vid**, ...) ohne Fid
Rahmen3 = (**Rid**, ...) ohne Fid
H-Rad3 = (**Hid**, ...) ohne Fid

```
define view  Fahrrad1 as
        select  v.Vid,   v.   ... ,
                r.Rid,   r.   ... ,
                h.Hid,   h.   ... ,
                f.Preis
        from    V-Rad3   v,
                Rahmen3  r,
                H-Rad3   h,
                Fahrrad3 f
        where   f.Vid = v.Vid  and
                f.Rid = r.Rid  and
                f.Hid = h.Hid
```

10

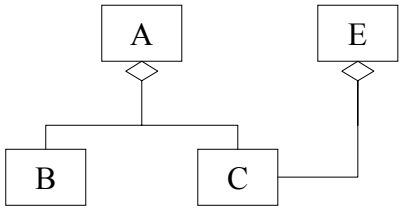
Komp.-Schema 4:
Kombination aus 2. und 3. mit Redundanz,
d.h.

Fahrrad = (**Fid**, Vid, Rid, Hid, Preis)
V-Rad = (**Vid**, Fid, ...)

define view Fahrrad **as** ... ?

11

Rel. Schema für Aggregation:



Aggr. Schema 1: A als schwache Entität:
völlige Analogie zu Komp.-Schema 1

Aggr.-Schema 2:
Analogie zu Komp.-Schema 2 geht nicht, weil B, C in mehreren Instanzen von A vorkommen dürfen.

Aggr.-Schema 3:
genaue Analogie zu Komp.Schema 3, A als starke Identität

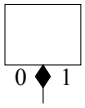
Aggr.-Schema 4:
geht nicht, Begründung wie in Aggr.-Schema 2

12

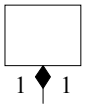
Modellvarianten und zugehörige Rel.Schemata

eigenständiger Key

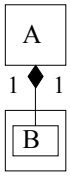
key = composition von foreign keys



flat delete, d.h. Komponente bleibt bei delete erhalten



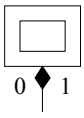
deep delete, d.h. eigentlich



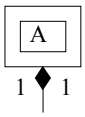
das nur möglich,

B kann dann OID von A bekommen

13



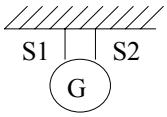
flat delete



very deep delete, weil delete einer Komponente von A auch das delete der anderen Komponenten erzwingt,

(or transitive delete, auch cascading oder recursive delete genannt)

z.B. elektromagnetisches Feld, Flügel eines Flugzeugs im Flug



Aufhängseile S1, S2 mit Tragkraft

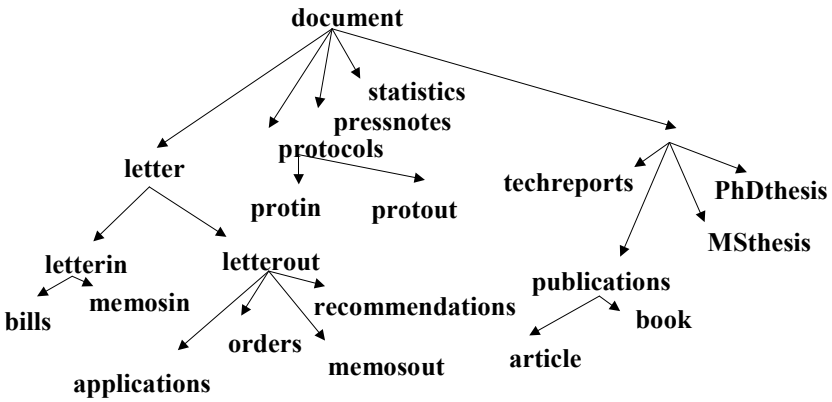
½ G < S1 < G

½ G < S2 < G

bzw. Stützpfeiler & World-Trade-Centers Einsturz

14

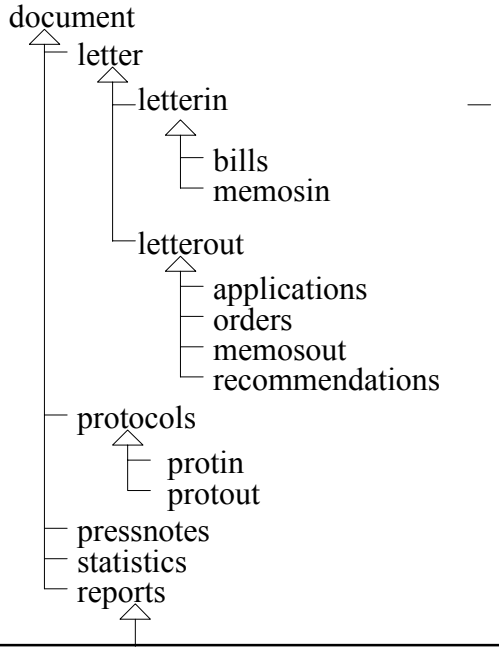
Beispiel für reine ISA-Hierarchie von Dokumenten



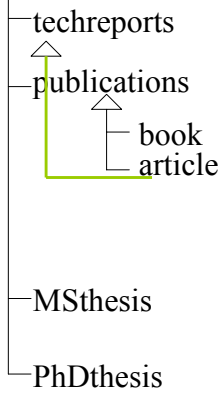
Vererbung von
- Struktur
- Werten
- Integritätsbedingungen
- Invarianten

15

Darstellung für Isa-Hierarchien



16

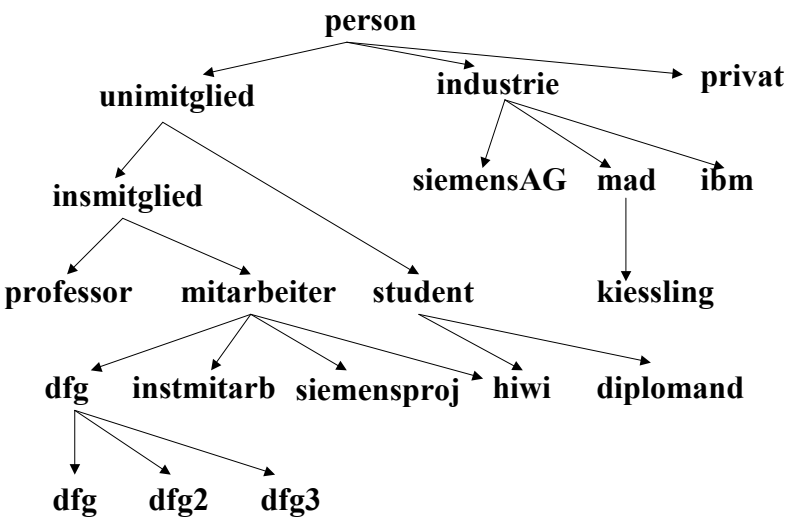


Beispiel für mehrfache Vererbung

d.h. Baum in Vorordnung (Vaterknoten kommt vor Kindern) dargestellt.

17

Personenhierarchie und Strukturvererbung



18

Strukturvererbung

person	**diplomand**	**mitarbeiter**	**industrie**
name	thema	vertrag	abteilung
vorname	anmeldung	seit	chef
adresse	betreuer	bis	email
telefon	abgabe	stelle	
	danote		

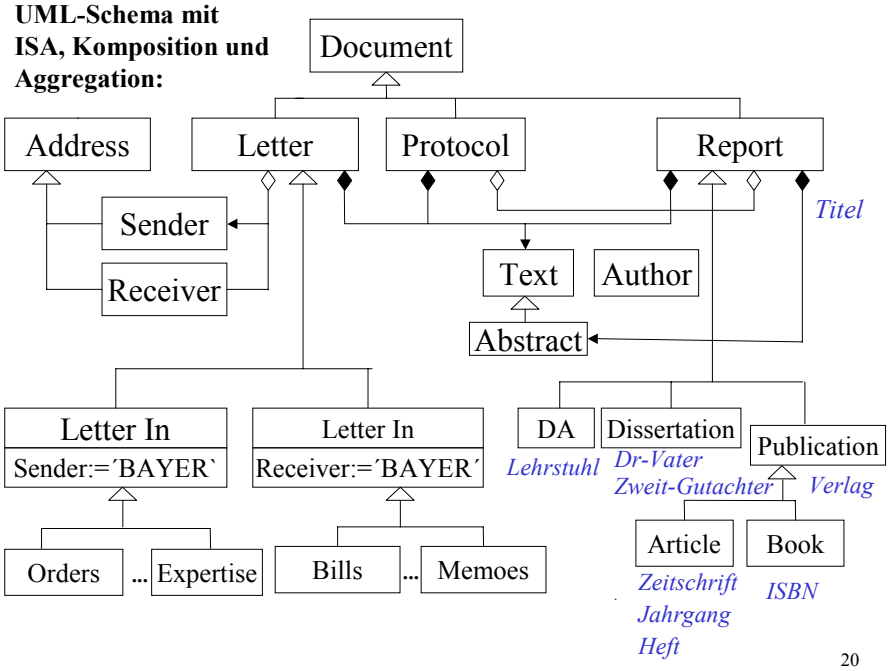
Attribute

Vererbung von Attributen

instmitglied	**student**	**hiwi**
tel#	matrikel#	stunden
zi#		

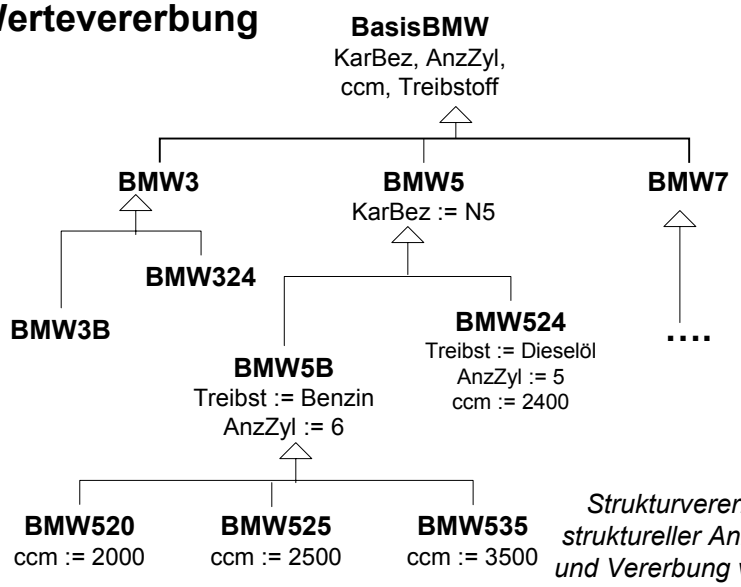
19

UML-Schema mit ISA, Komposition und Aggregation:



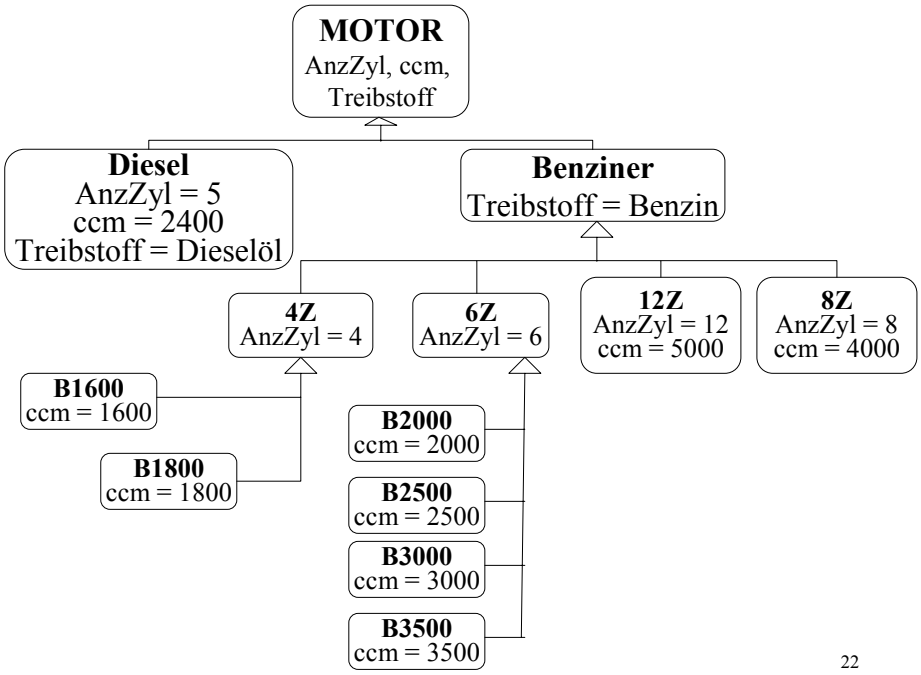
20

Wertevererbung



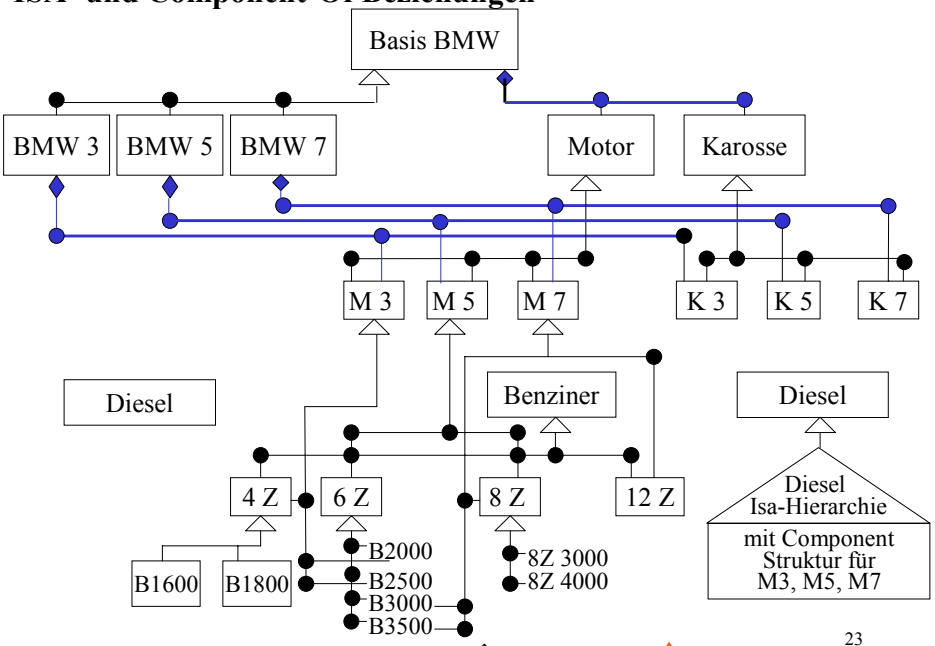
Strukturvererbung mit struktureller Anreicherung und Vererbung von Werten

21



22

ISA- und Component-Of Beziehungen



23

Systematik: schwarz für △, blau für ◆, orange für ◊

Relationale Darstellung als Kantenmenge

ISA	
Kind	**Vater**
BMW 3	Basis BMW
BMW 5	„
BMW 7	„
M3	Motor
M5	„
M7	„
K3	Karosse
K5	„
K7	„
4Z	M3
B 2000	„
B 2500	„
B 1600	4Z

ISA	
Kind	**Vater**
B 1800	4Z
B 2000	6Z
B 2500	„
B 3000	„
B 3500	„
8Z 3000	8Z
8Z 4000	„
B 3000	M7
B 3500	„
8Z	„
12Z	„
6Z	M5
8Z	„

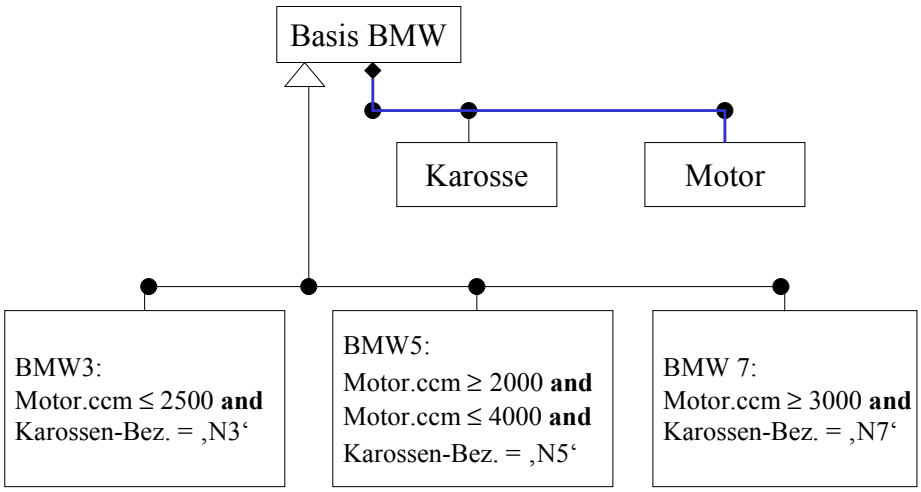
Component

Comp	of
Motor	Basis BMW
Karosse	„
M3	BMW3
K3	„
M5	BMW5
K5	„
M7	BMW7
K7	„

Hinweis:
Transitive Hülle zur Berechnung aller Motoren, die z.B. für M3 in Frage kommen

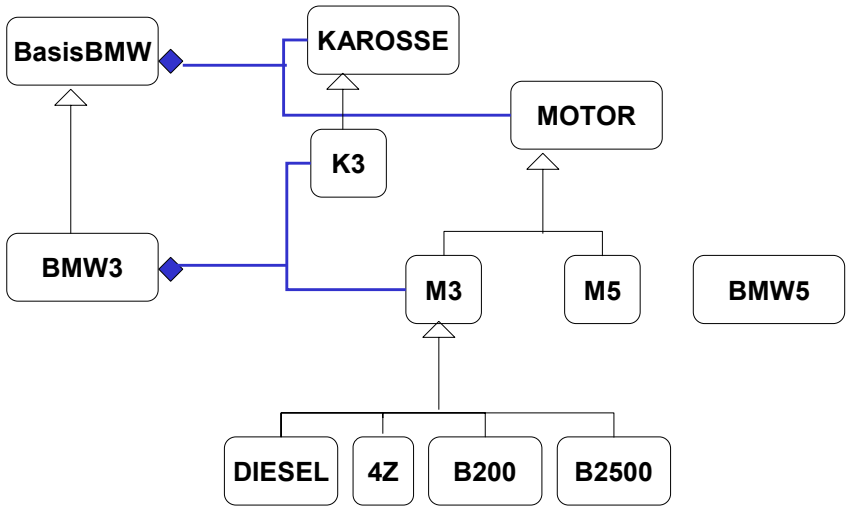
24

Verbindung Graphik und Prädikat:



25

Part-of-Beziehungen
mit Vererbung

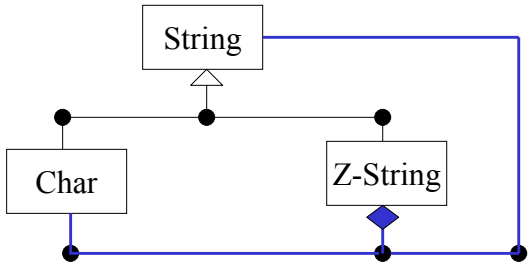


Ausschnitt aus der Motorhierarchie

26

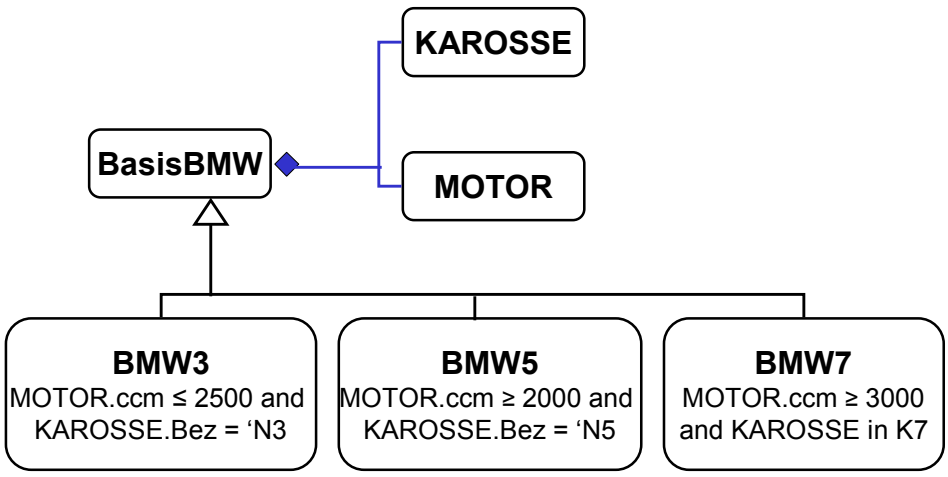
UML und Rekursive Strukturen

String ::= Char | Char String



String ::= Char | Char Z-String
Z-String ::= Char String

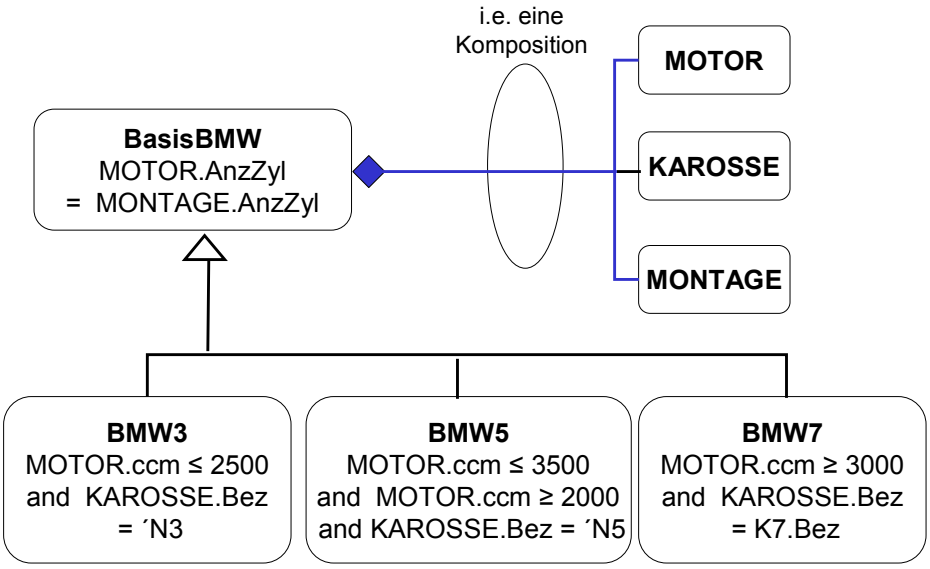
27



Invarianten

28

Vererbung von Invarianten



29