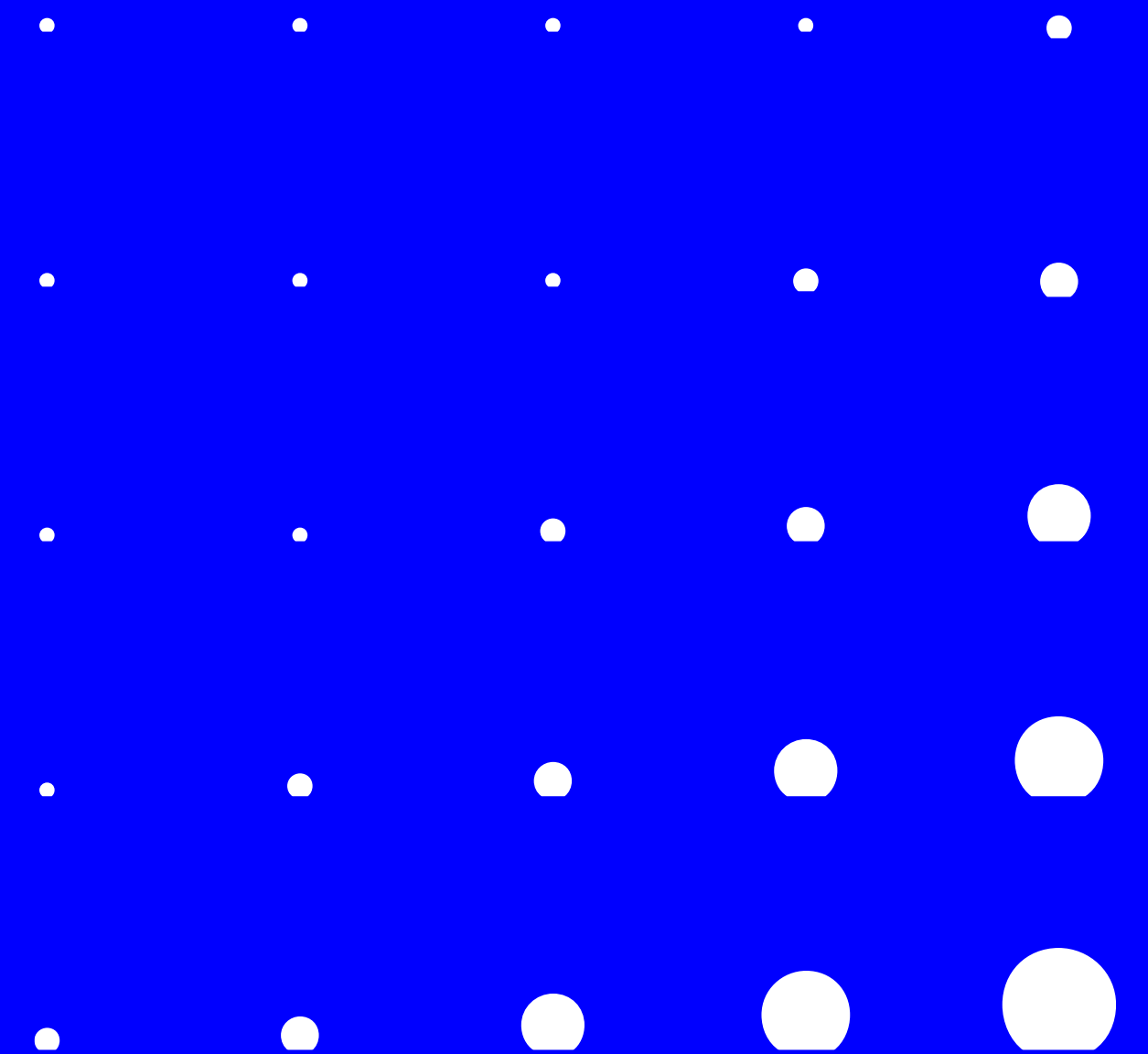
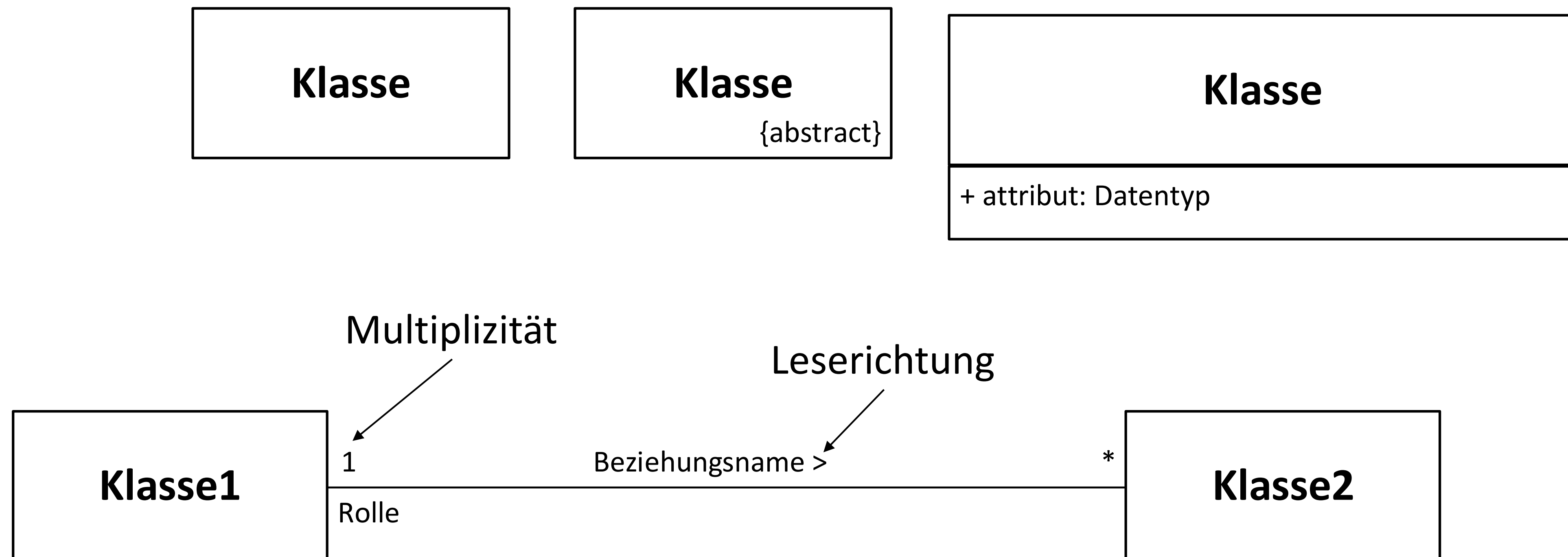


## Datenmodellierung mit UML



# Verwendung der UML 2.5 Notationsübersicht...

- Datenmodellierung über das Klassendiagramm, Operationen/Methoden sind dabei unbedeutend...



# UML-Klassendiagramm mit Java-Datentypen

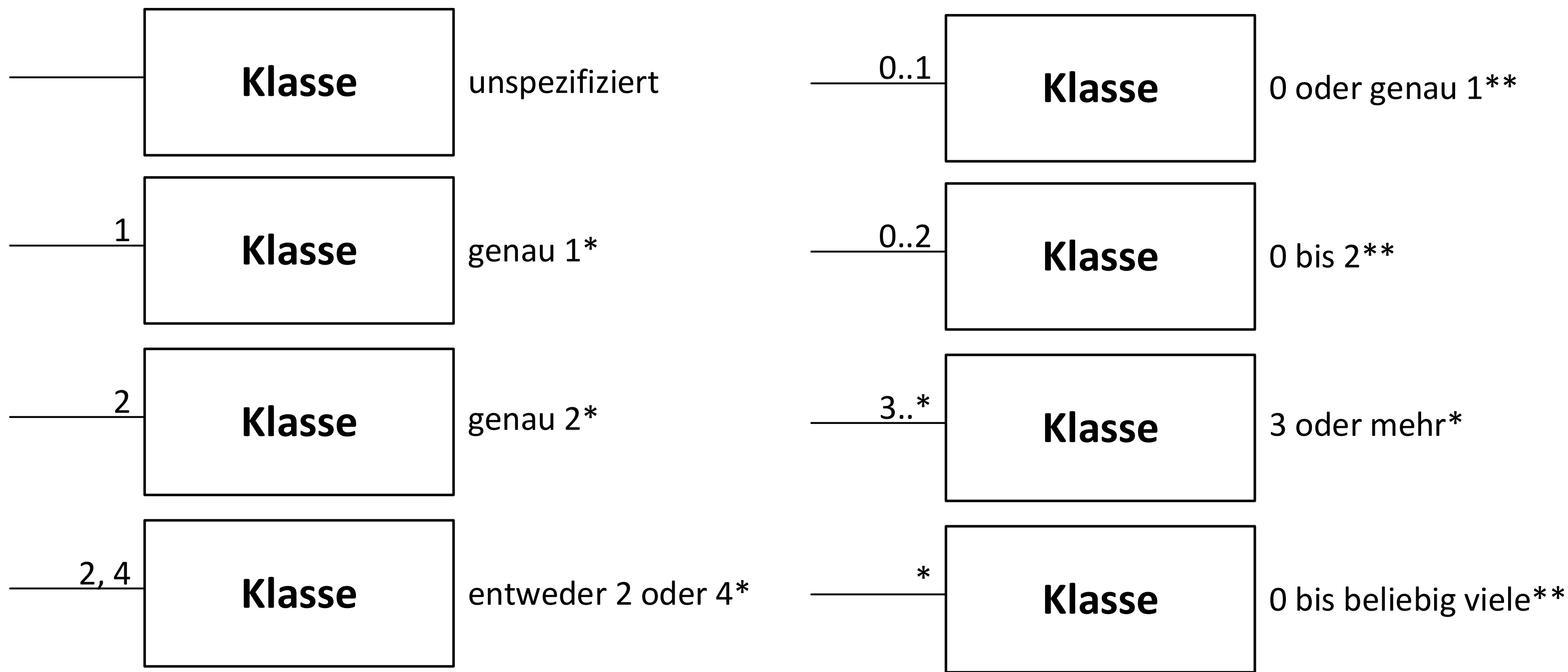
- Neben den primitiven Datentypen...

Datentyp	Größe	Wrapper-Klasse	Wertebereich	Beschreibung
byte	8 Bit	java.lang.Byte	-128 ... +127	Zweierkomplement-Wert
short	16 Bit	java.lang.Short	-32.768 ... +32.767	Zweierkomplement-Wert
int	32 Bit	java.lang.Integer	-2.147.483.648 ... +2.147.483.647	Zweierkomplement-Wert
long	64 Bit	java.lang.Long	-9.223.372.036.854.775.808 ... +9.223.372.036.854.775.807	Zweierkomplement-Wert
float	32 Bit	java.lang.Float	$\pm 1,4E-45$ ... $\pm 3,4E+38$	Gleitkommazahl )
double	64 Bit	java.lang.Double	$\pm 4,9E-324$ ... $\pm 1,7E+308$	Gleitkommazahl doppelter Genauigkeit (IEEE 754)

...verwenden wir noch Date, String und BigDecimal.

# UML-Klassendiagramm mit möglichen Multiplizitäten

- Eine Assoziation sagt zunächst nur, daß ein Objekt andere Objekte kennen kann; es ist also eine „kennt“-Beziehung.
- Die Angabe einer Multiplizität legt fest, wieviele Objekte ein Objekt kennen kann oder muss.
  - Sind es mehr als 1: Speicherung in einem Array oder einer Collection
  - Die Multiplizität wird in der UML am Ende der Assoziations-Linie notiert:

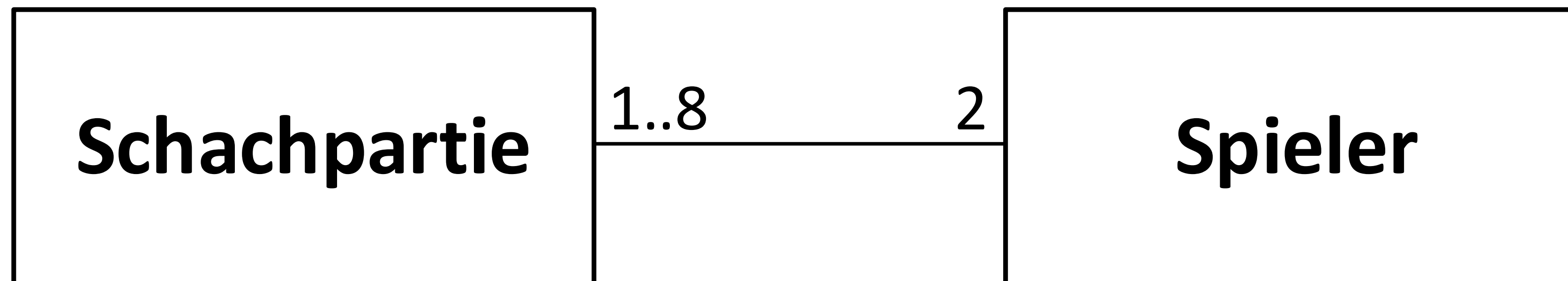


\* Muss-Assoziation:  
Das Objekt muss in  
Beziehung zu anderen  
stehen.

\*\* Kann-Assoziation:  
Das Objekt kann, muss  
aber nicht zwingend in  
Beziehung stehen

# Beispiel zur Multiplizität: Simultan-Schach

- Jede Schachpartie wird von zwei Spielern gespielt.
- Ein Spieler spielt 1 bis 8 Partien gleichzeitig.



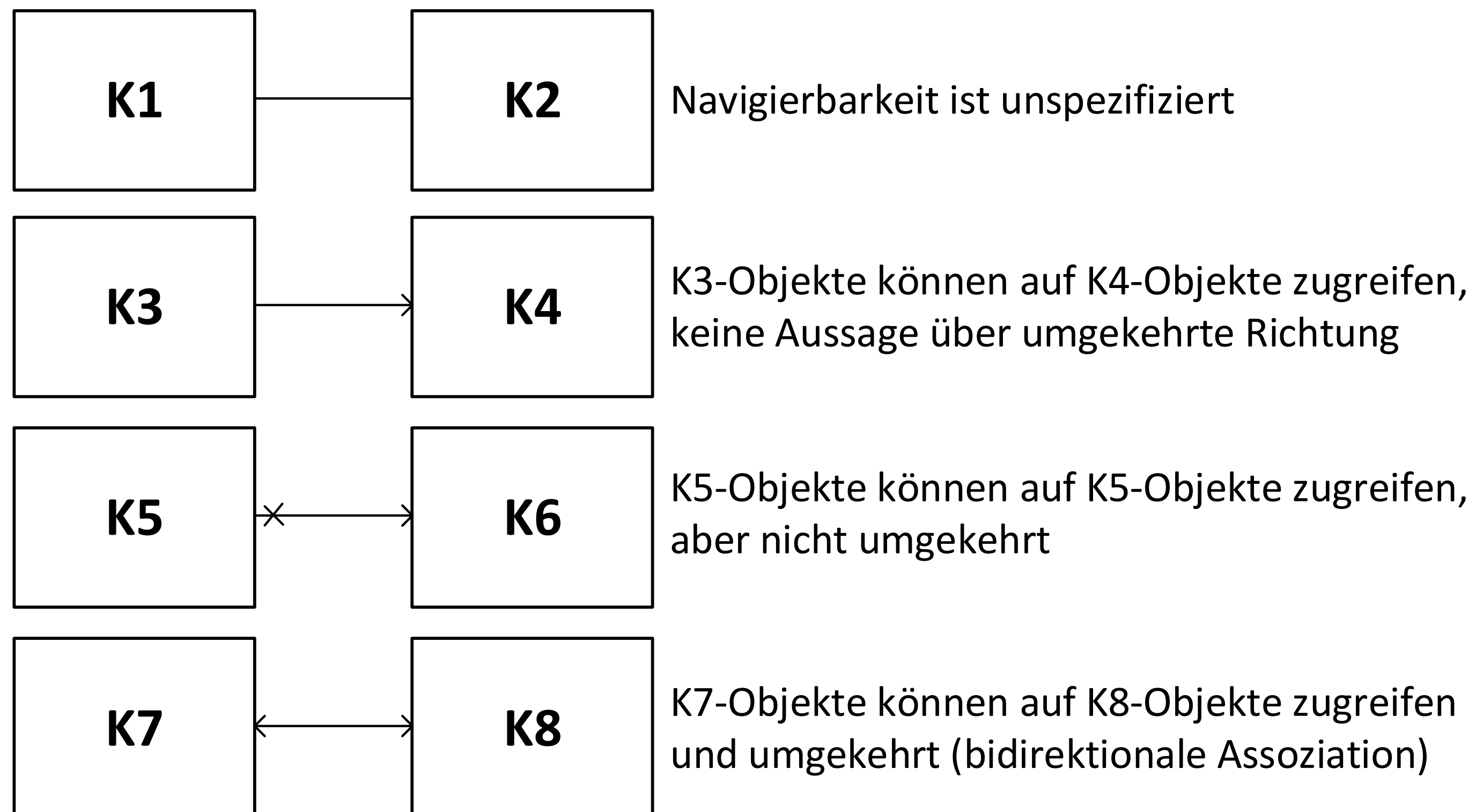
# Beispiel zur Multiplizität: Eine Bank wie die ING

- Ein Kunde muß mindestens ein Konto besitzen.
- Ein Konto gehört zu genau einem Kunden.
  - Wenn der Kunde gelöscht wird, so muss auch das Konto gelöscht werden!
- Ein Kunde kann kein oder beliebig viele Depots besitzen.
- Ein Depot gehört zu genau einem Kunden.
  - Wenn der Kunde gelöscht wird, dann müssen auch alle Depots gelöscht werden!



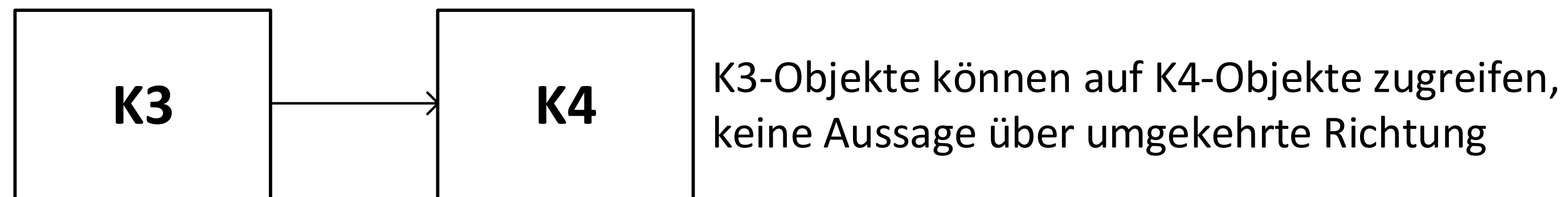
# Assoziation & Navigierbarkeit

- Die Assoziation ist von A nach B navigierbar.
  - Objekte von A können auf Objekte von B zugreifen, aber nicht notwendigerweise umgekehrt.
  - Objekte von A können auf Objekte von B kennen.
  - Jeder Zugriff erfolgt über einen Methodenaufruf; also muss ein A-Objekt eine Referenz auf ein B-Objekt haben.



# Assoziation & Navigierbarkeit

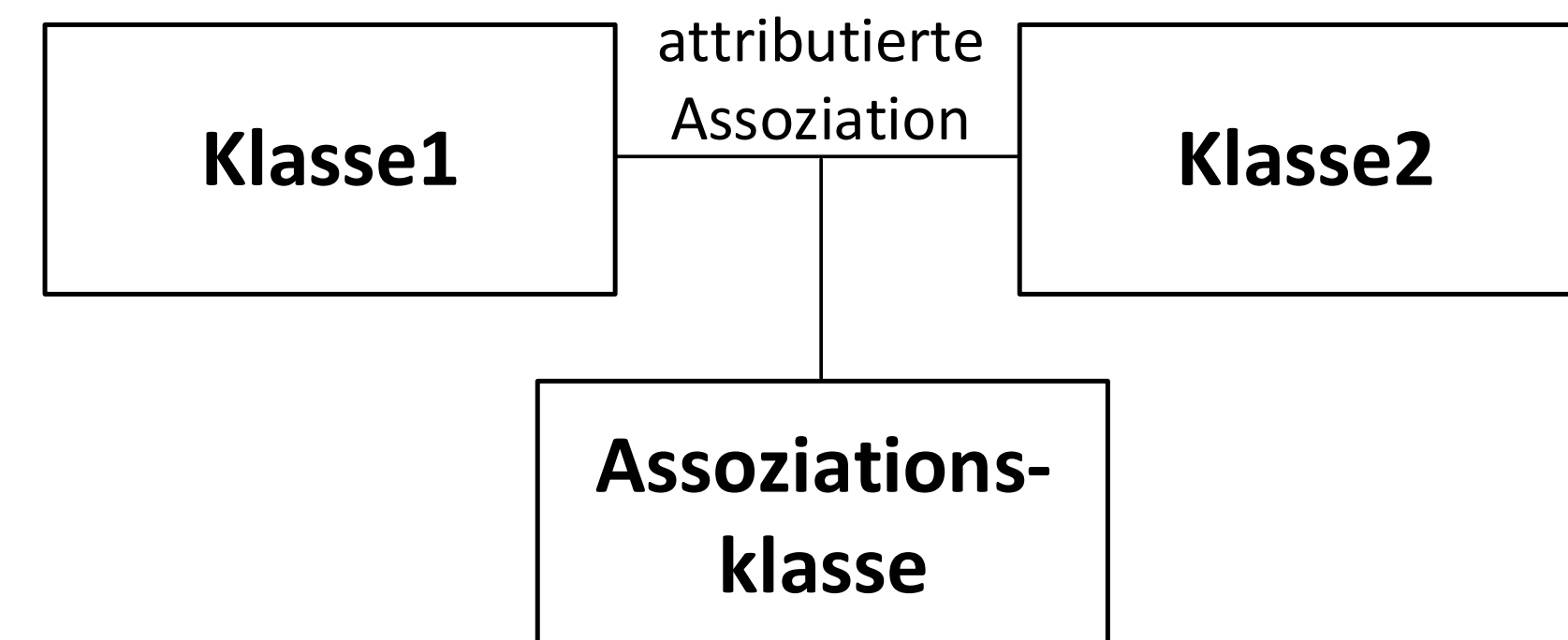
- Wenn ein K3-Objekt ein K4-Objekt kennen können muss, so muss das K3-Objekt eine Referenz auf ein K4-Objekt haben!
  - Die Referenz muss nicht stets ausgeprägt sein, sie kann auch null sein.
  - Die Kenntnis eines anderen Objektes ist damit bereits realisiert:
  - „Jedes Tier hat einen Namen“ bedeutet, dass jedes Tier-Objekt bereits bei seiner Erstellung eine Referenz „name“ auf ein Objekt der Klasse String besitzt.





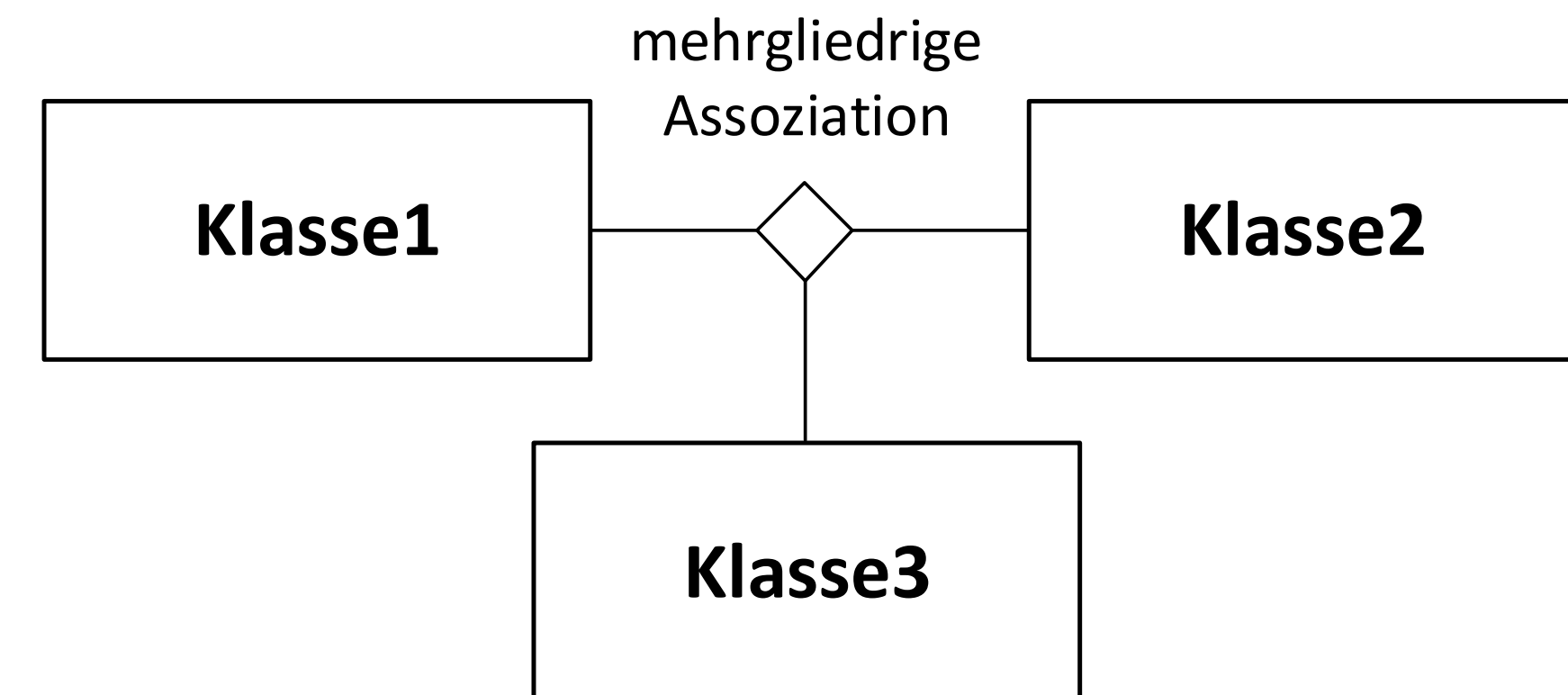
# Komplexere Assoziationen

- Eine Assoziation zwischen einem Objekt der Klasse1 und der Klasse2 besteht aus mehr als nur Referenzen...
  - Die Assoziation hat eigene Eigenschaften, eigene Attribute.
  - Diese werden dann in einer separaten Assoziationsklasse hinterlegt, die das Objekt von Klasse1 mit dem Objekt der Klasse2 verbindet.
  - Beispiel: Lieferant, Produkt und individuelle Rabatte von einigen Lieferanten für bestimmte Produkte ab einer bestimmten Menge.



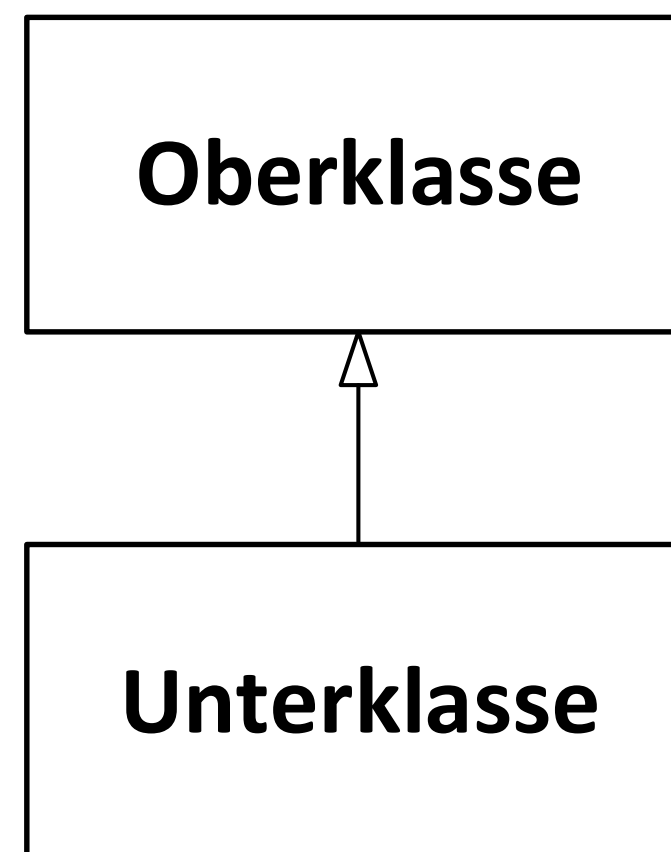
# Komplexere Assoziationen

- Manchmal sind die Verbindungen so komplex, dass eigene Klassen über eine bestimmte komplexe Klasse miteinander verbunden sind, die dann als Raute dargestellt wird.
  - Diese Klasse ist dann oft kein physisches Objekt, sondern etwas Virtuelles, Organisatorisches.
  - Beispiel: Interessent, Verkäufer, Produkt und als verbindende Klasse das Verkaufsgespräch.



# Die Vererbung

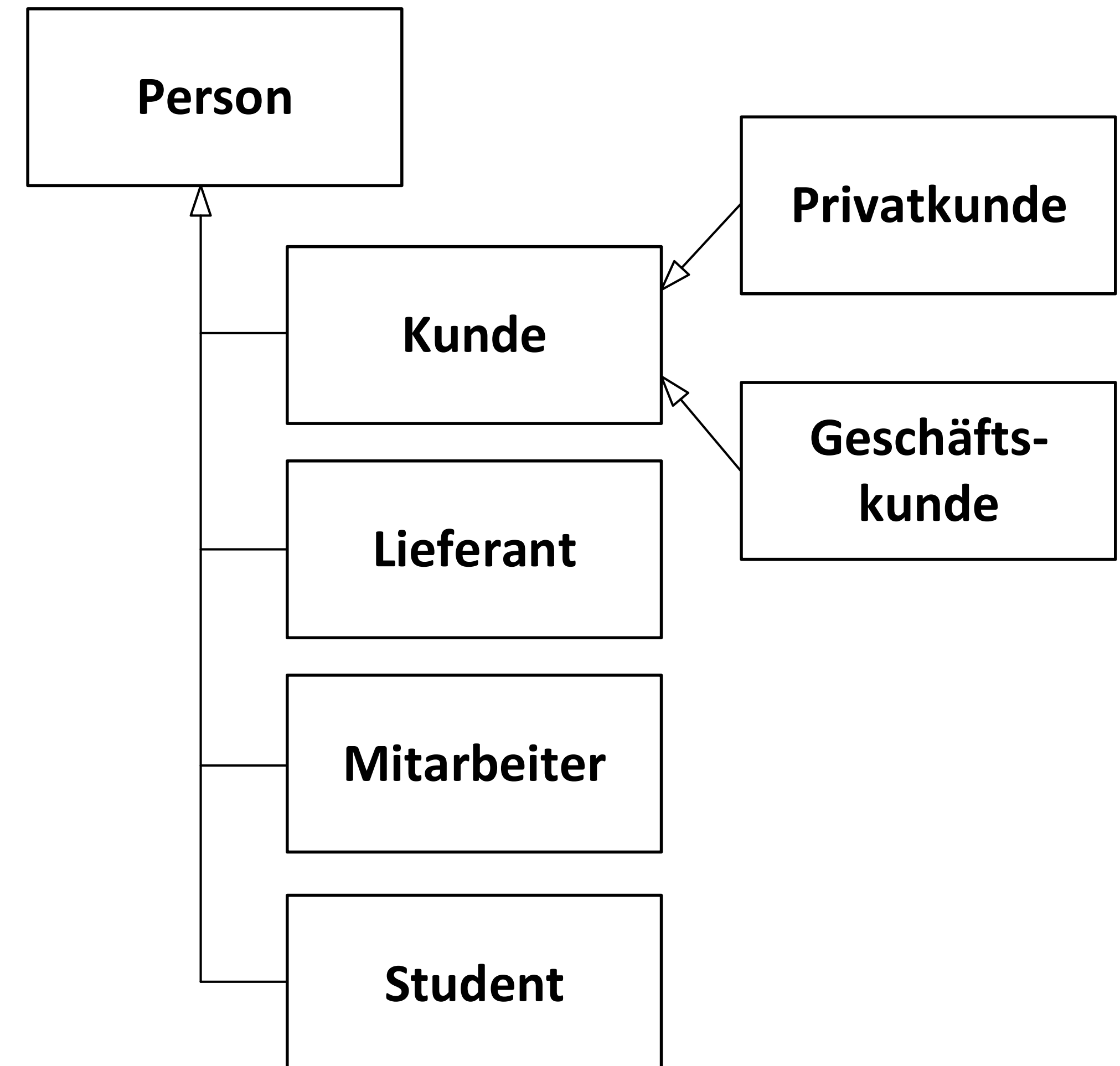
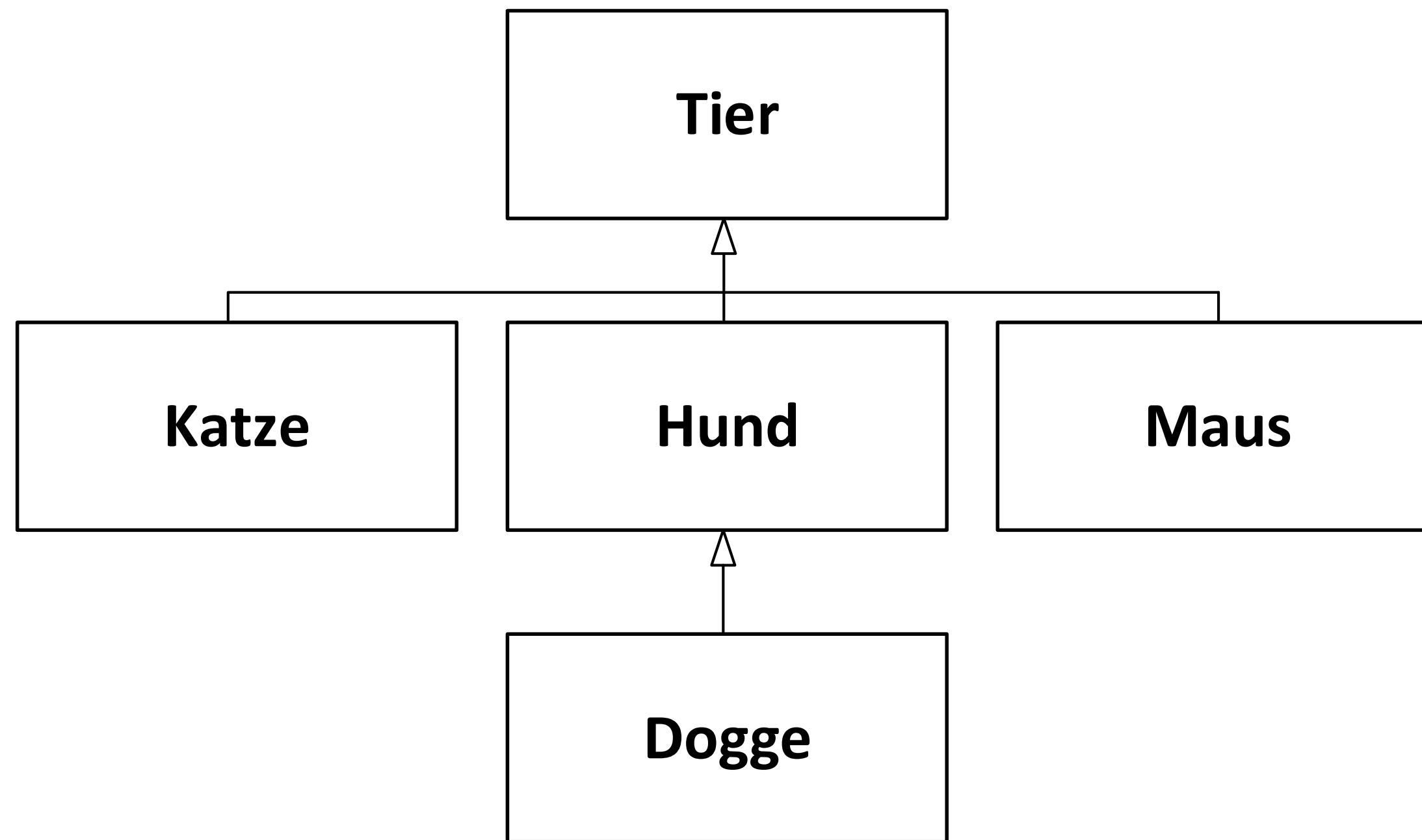
- Die Oberklasse generalisiert von den Unterklassen.
- Die Unterklasse ist spezieller als die Oberklasse.
- Ein Objekt der Unterklasse „ist ein“ Objekt der Oberklasse.



```
class Oberklasse {  
    ...  
}
```

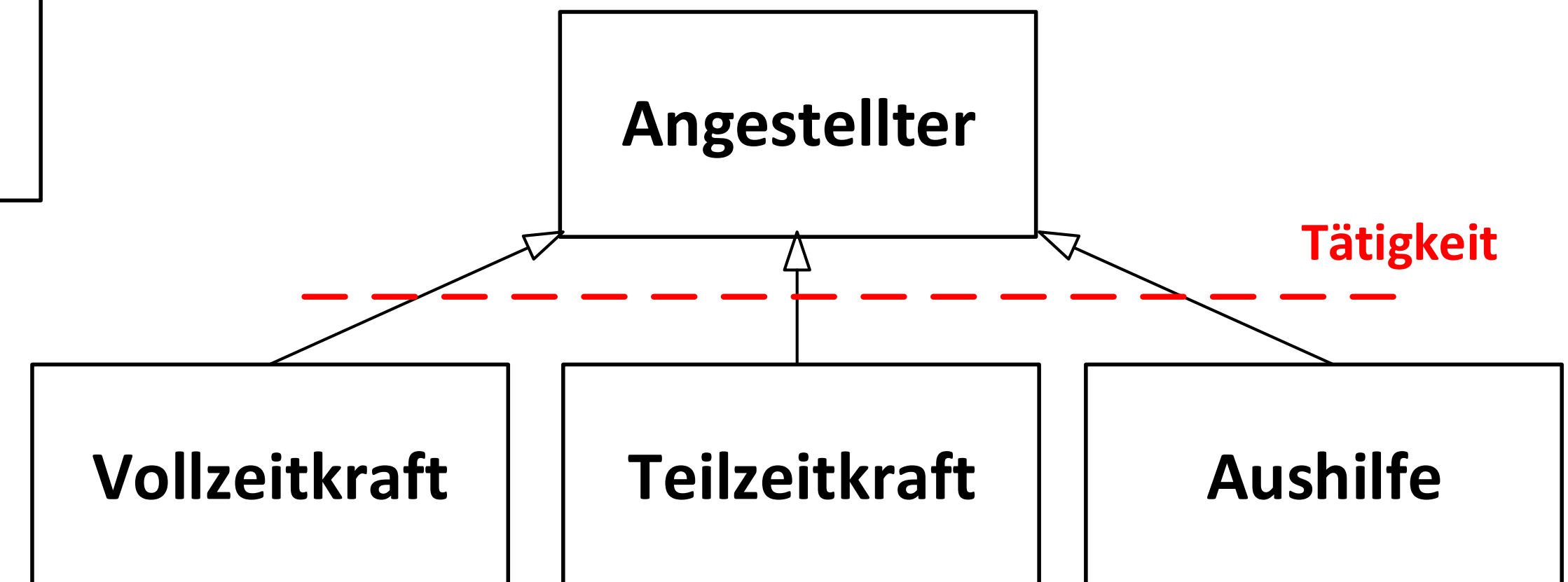
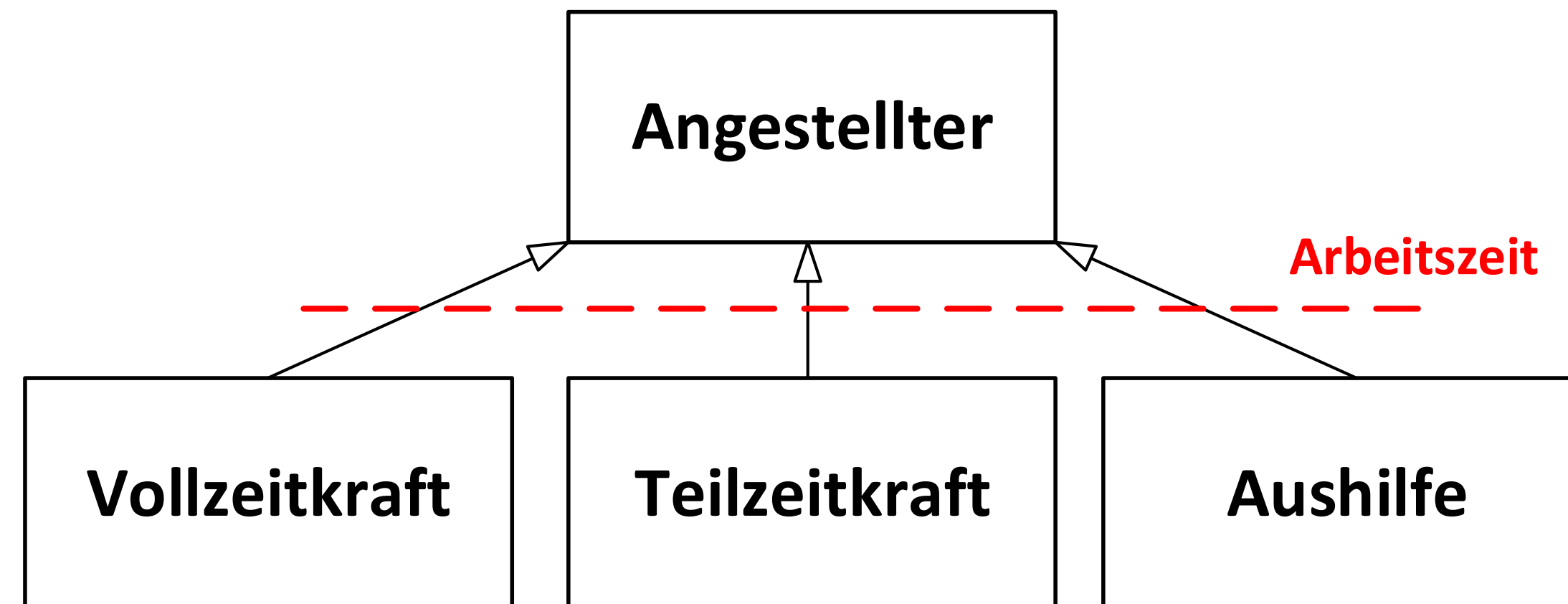
```
class Unterklasse extends Oberklasse {  
    ...  
}
```

# Die Vererbung: Gültige Beispiele



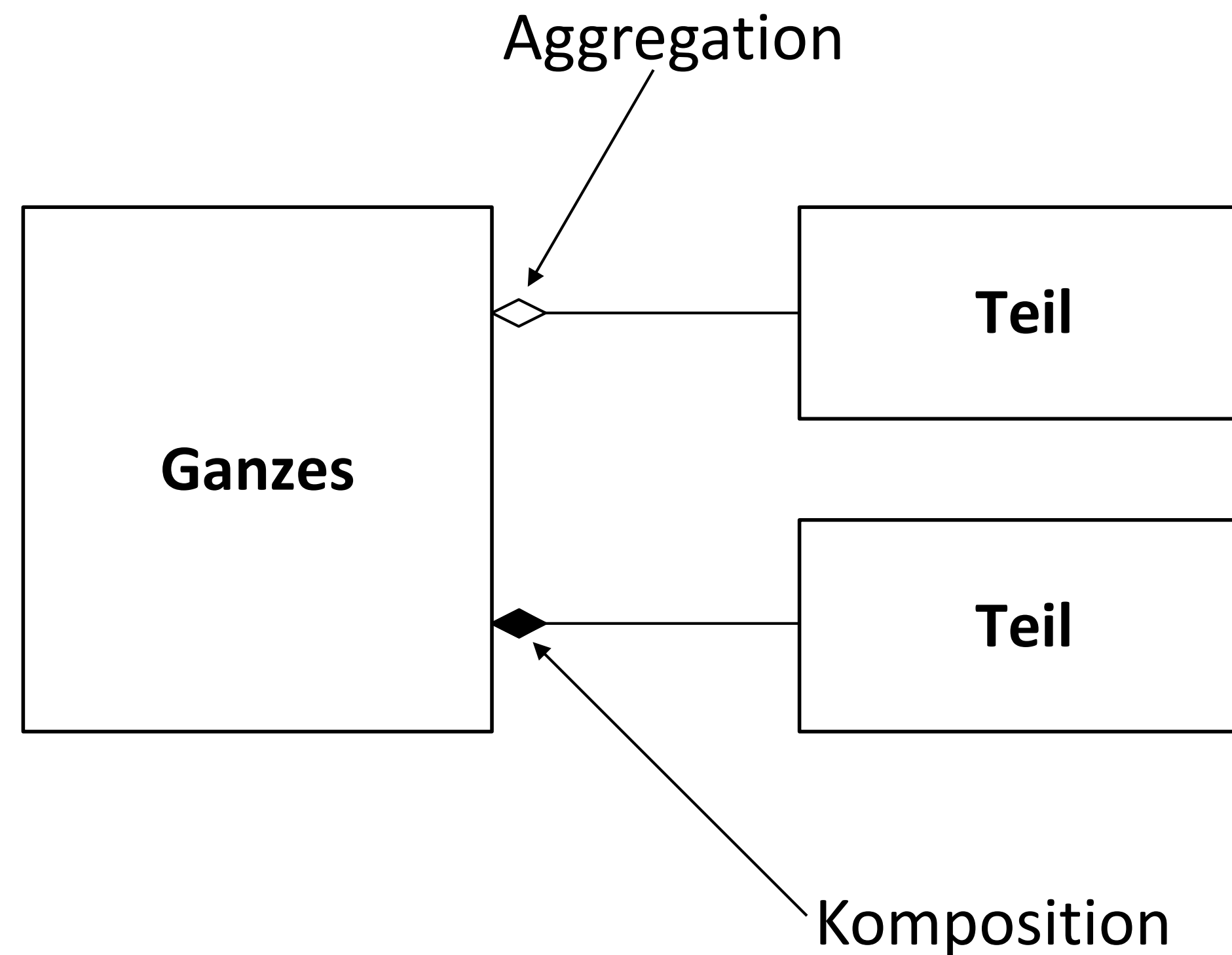
# Vererbung & Diskriminator

- Ein zusätzlicher Diskriminator als Unterscheidungsmerkmal kann das Kriterium angeben, nach dem klassifiziert/unterschieden wird.
- Dies hilft, die Vermischung von mehreren Kriterien zu vermeiden, was zu Mehrfachvererbung führen würde.



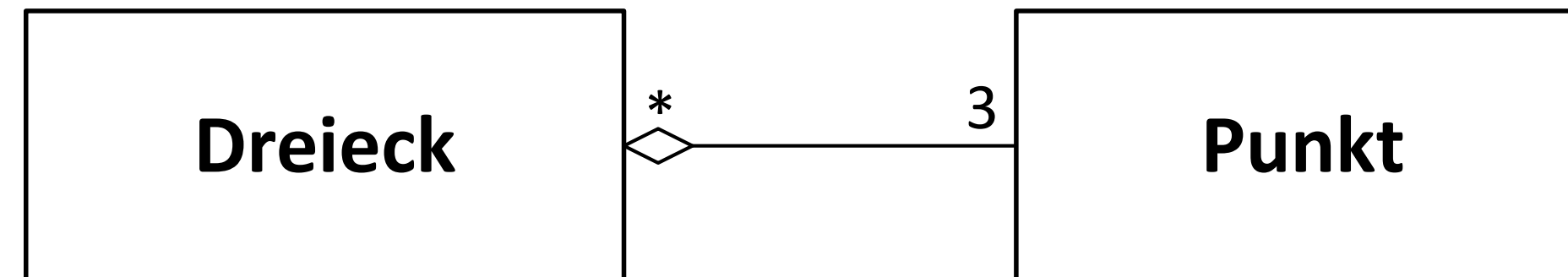
# Aggregation & Komposition

- Ein Ganzes besteht aus seinen Teilen.
- Bei der Komposition kann das Teil nicht ohne sein Ganzes existieren, Beispiel: Rechnung und Rechnungsposition

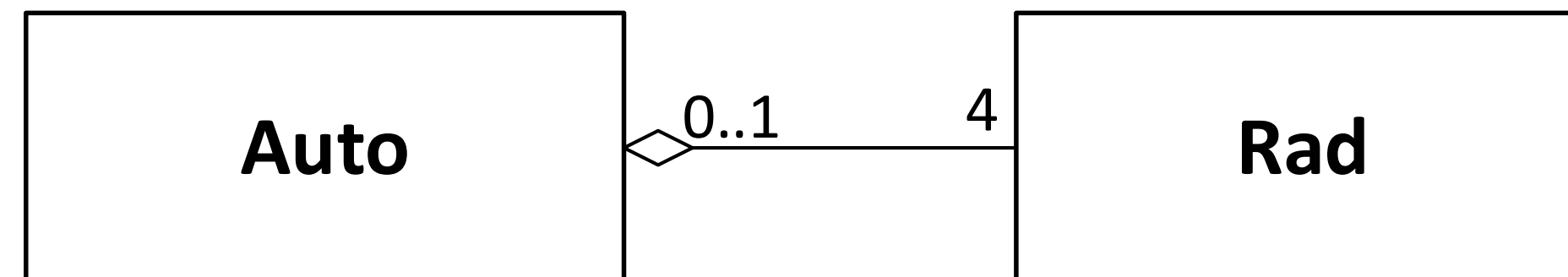


# Aggregation: Beispiele

- Ein Dreieck besteht immer aus genau 3 Punkten. Ein Punkt ist Teil von 0 bis beliebig vielen Dreiecken. Ein Punkt kann auch allein existieren ohne ein Dreieck:



- Ein Auto besteht unter anderem aus 4 Rädern. Ein Rad ist Teil von höchstens einem Auto. Es gibt auch Räder, die rumliegen und sich an keinem Auto befinden:



# Komposition: Beispiel

- Ein Verzeichnis besteht aus beliebig vielen Dateien.
- Dateien stehen immer in einem Verzeichnis, sonst existieren sie nicht.
- Dateien können als Datenstrom in einem Netzwerk übertragen werden, während dessen sind es aber keine Dateien mehr.
- Wird das Verzeichnis gelöscht, so werden auch all seine Dateien darin gelöscht.
- Die 1 an der ausgefüllten Raute kann man auch weglassen, denn die 1 gilt immer.

