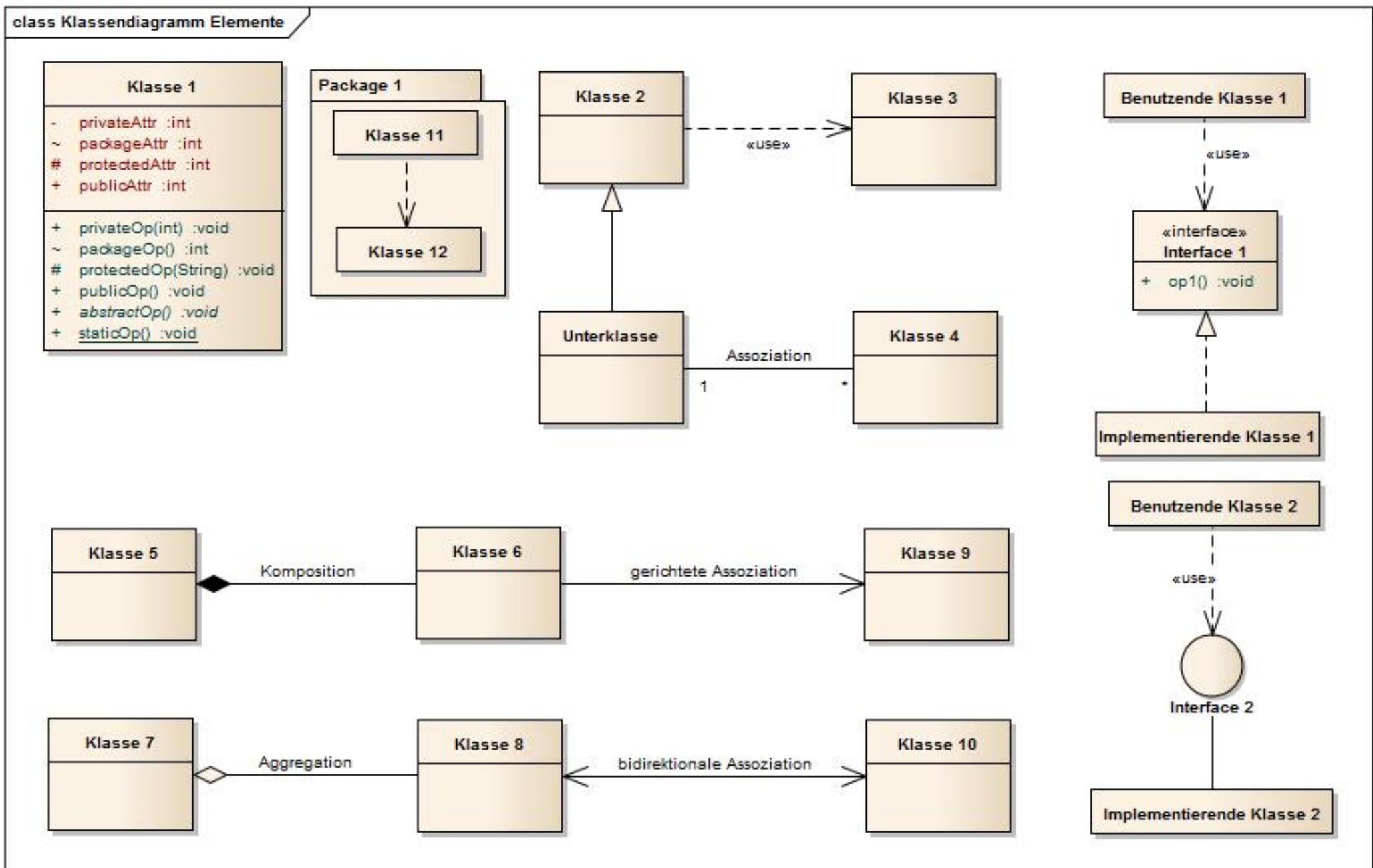


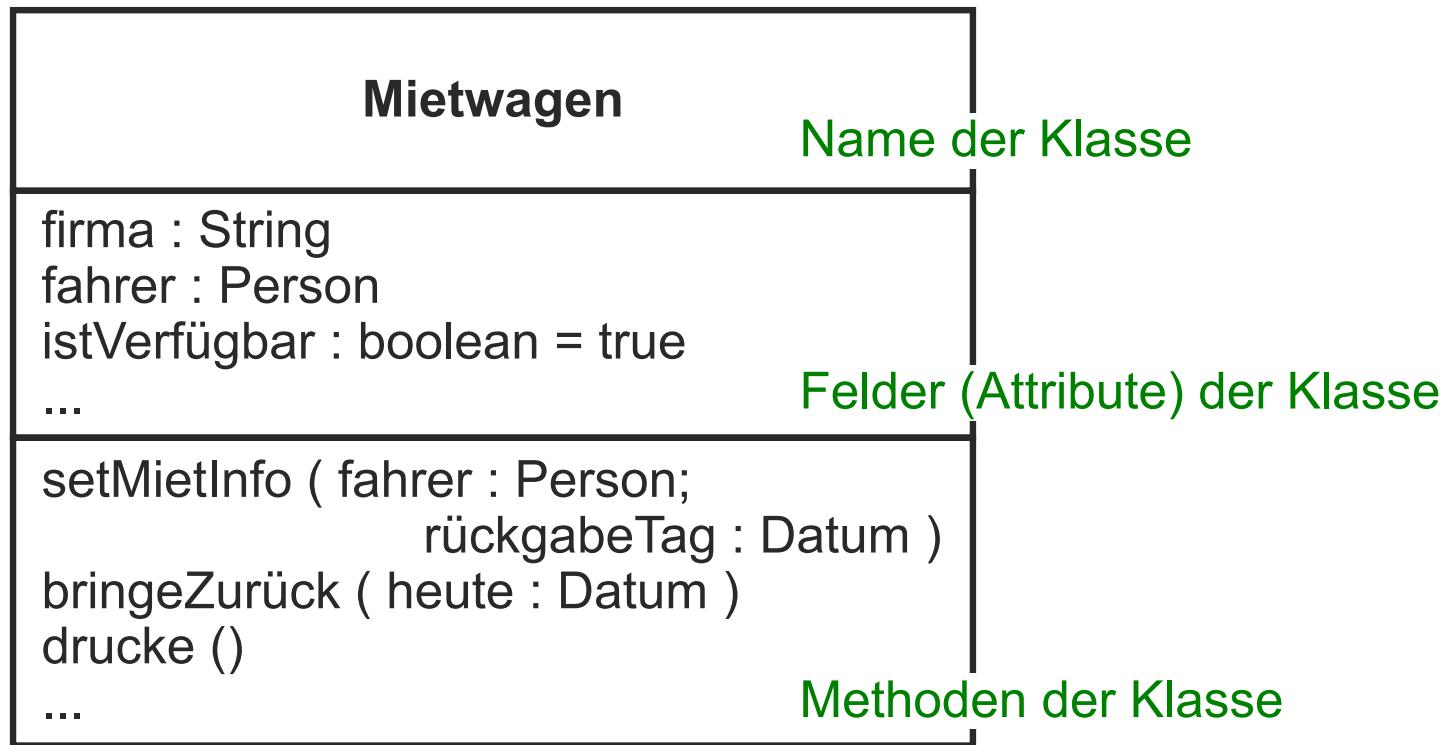
Klassendiagramme

- Übersicht
- Klassen
- Beziehungen zwischen Klassen
 - Assoziationen
 - Aggregation vs. Komposition
 - Vererbung
- Interfaces
- Aufgabe

Elemente von Klassendiagrammen

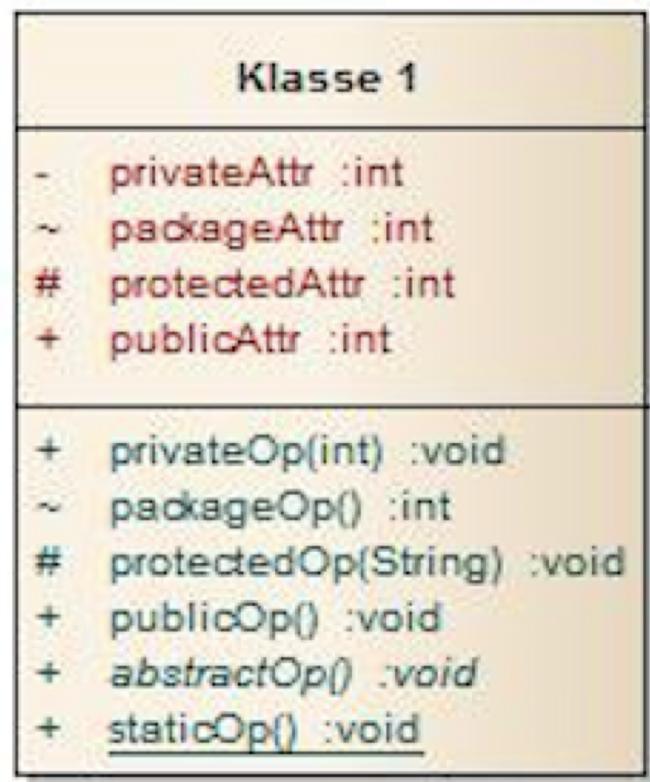


Klassen in der UML: Beispiel



Klassennamen beginnen mit einem **Großbuchstaben**

Klassen



Klassen haben...

- Attribute (Felder) mit...

- Sichtbarkeit
- Namen
- Typ
- optional: Initialwert

Attribute können statisch sein

„protected“ ist anders als in Java definiert!

- Operationen (Methoden) mit...

- Sichtbarkeit
- Namen
- Formaler Parameterliste
- Ergebnistyp

Operationen können abstrakt sein

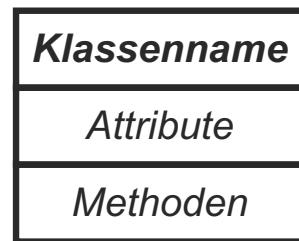
- Die zugehörige Klasse kann nicht instanziiert werden!

Operationen können statisch sein

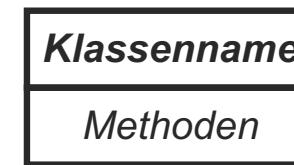
Klassen in der UML

Klassendiagramm

vollständig:



Kurzformen



Attribut: *attributname ":" Attributtyp*
attributname ":" Attributtyp = Attributwert

Methode: *methodename "(" ... ")" : Ergebnistyp*

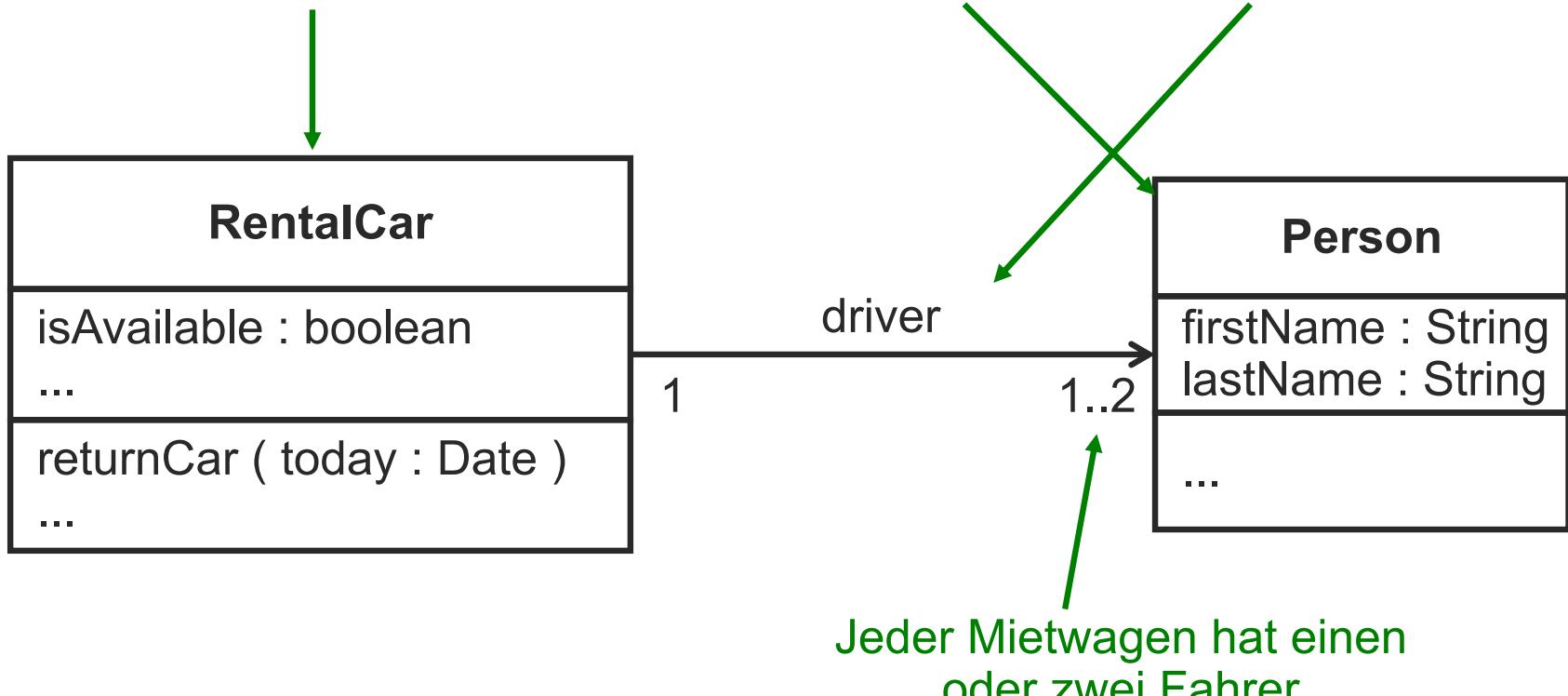
Konvention:

- Klassennamen beginnen mit Großbuchstaben
- Attributnamen und Methodennamen beginnen mit Kleinbuchstaben
- Es werden keine Trenn- oder Unterstriche verwendet

Java hat diese Konventionen von der UML übernommen

Beziehungen zwischen Klassen in der UML: Beispiel

Aus Sicht dieser Klasse handelt es sich hierbei um den Fahrer

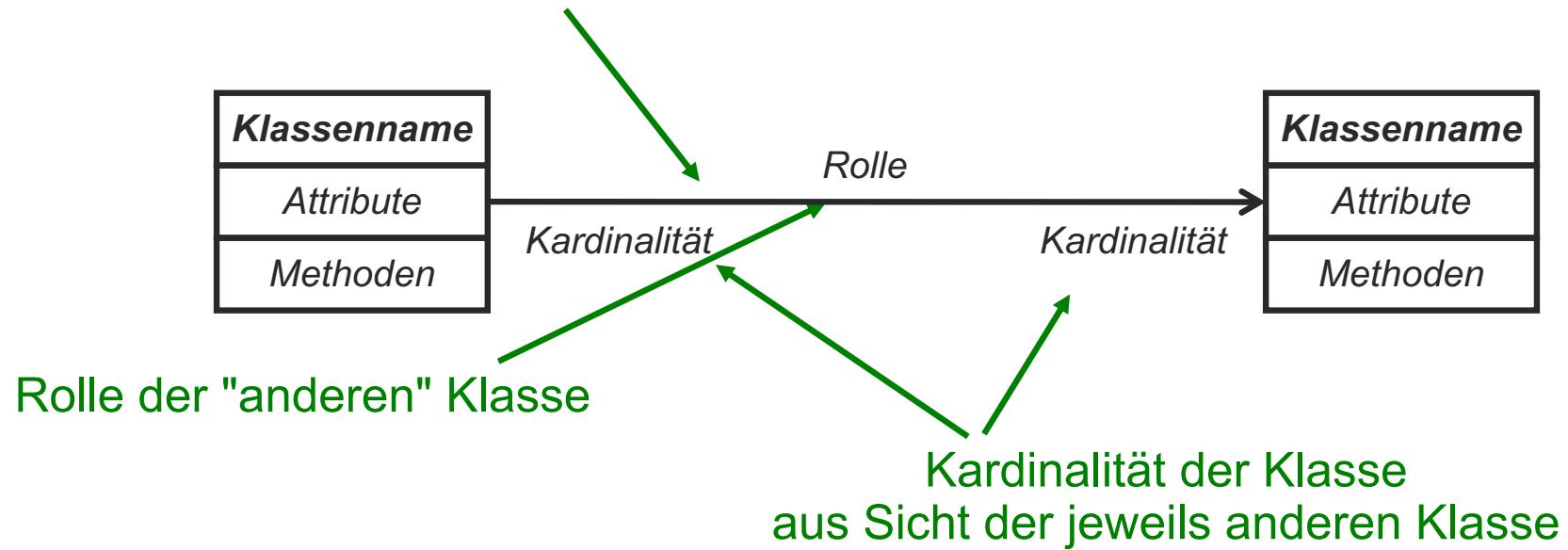


Jeder Fahrer gehört zu genau einem Mietwagen

Können Sie sich den Java-Code dazu vorstellen?

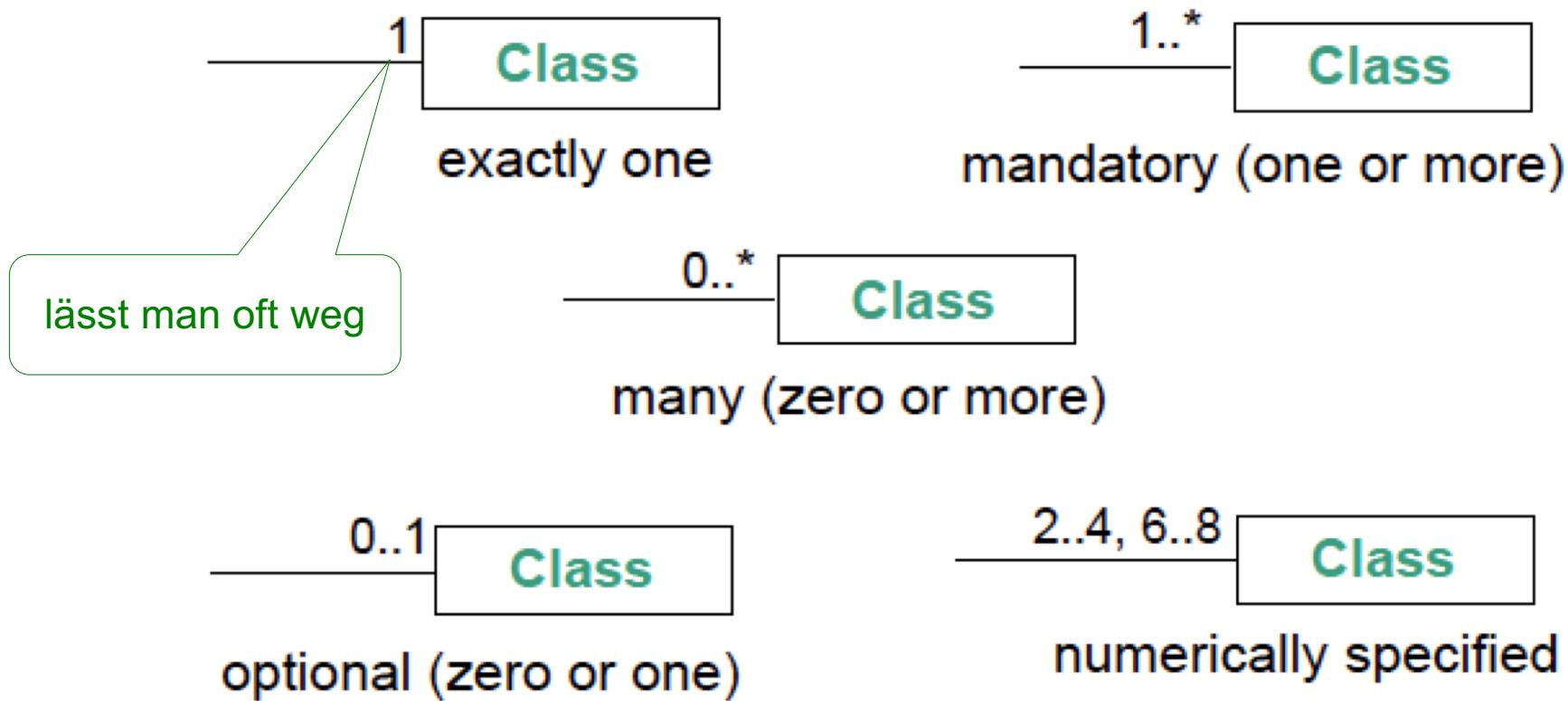
Beziehungen zwischen Klassen in der UML

Gerichtete Assoziationsbeziehung zwischen zwei Klassen

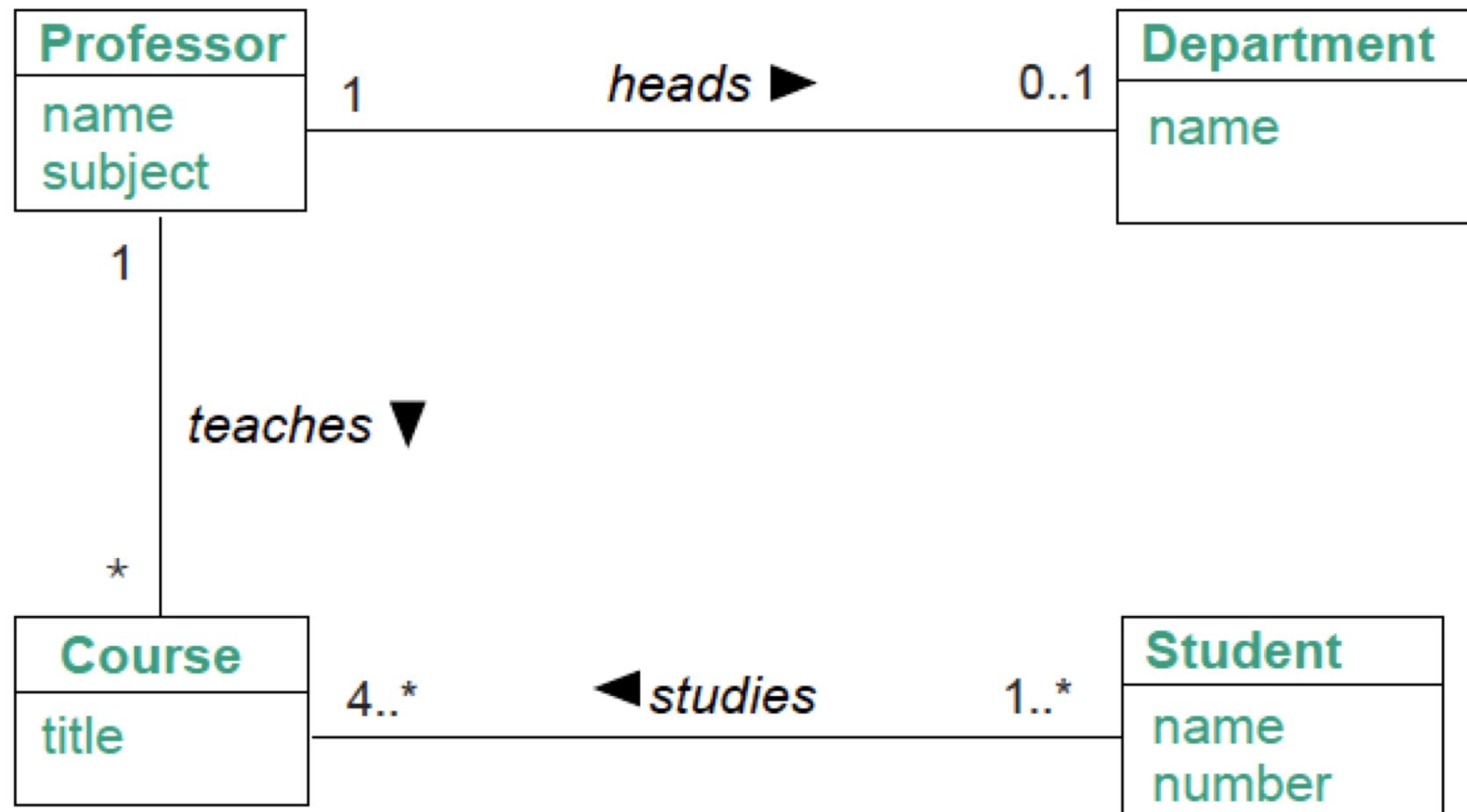


Kardinalität, (Häufigkeit, multiplicity)

Die Kardinalität beschränkt die Anzahl der Instanzen einer Klasse, die zu einer Instanz einer anderen Klasse in Bezug stehen

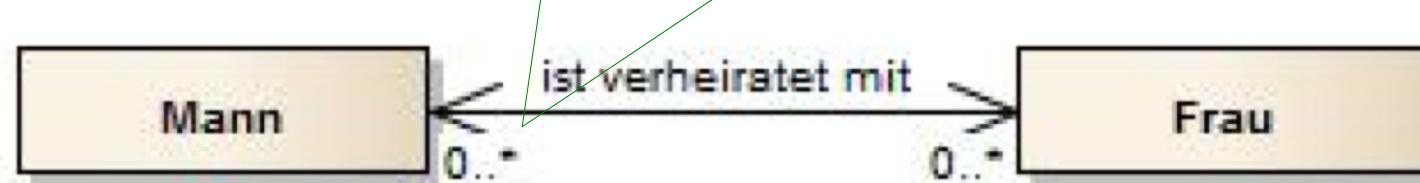


Beispiel mit Kardinalitäten



Kardinalität: Schnappschuss vs. Geschichte

wenn z. B. für die Polizei auch vergangene Beziehungen relevant sind



Mann

Frau

ist verheiratet mit

0..*

0..*

Mann

Frau

ist verheiratet mit

0..1

0..1

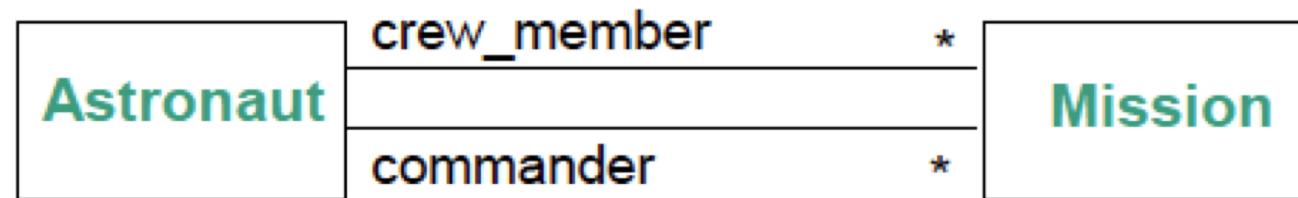
wenn z. B. für die Steuer nur der aktuelle Status relevant ist

Zwei gleichbedeutende Diagramme

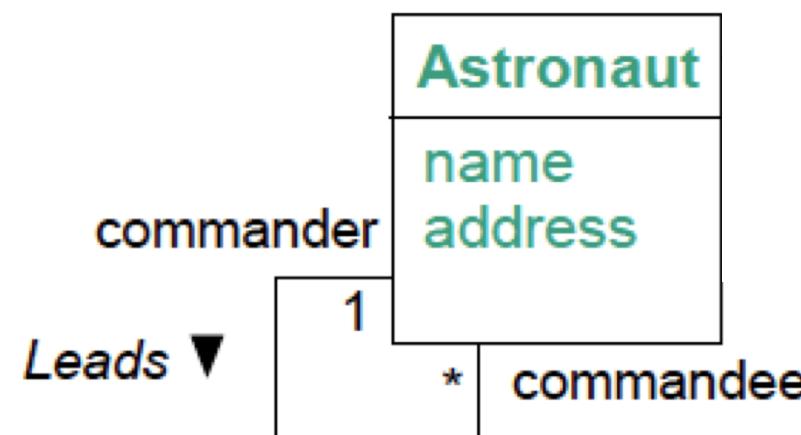


Sonderfälle

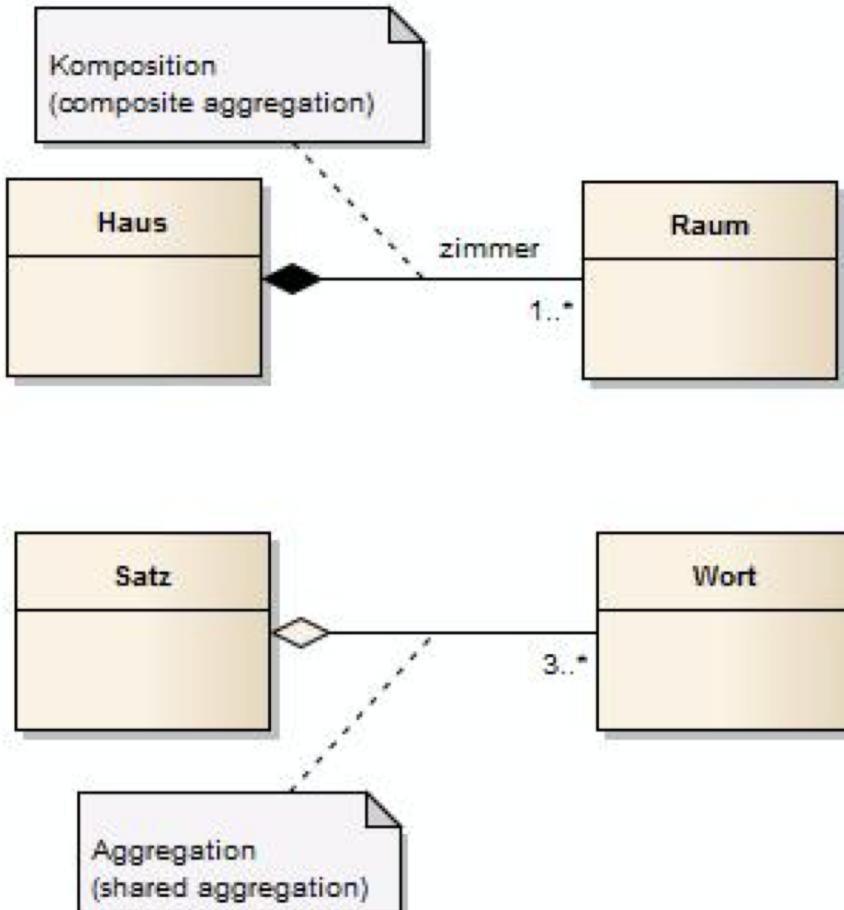
- Es kann mehr als eine Beziehung zwischen zwei Klassen existieren



- Eine Klasse kann mit sich selbst assoziiert sein

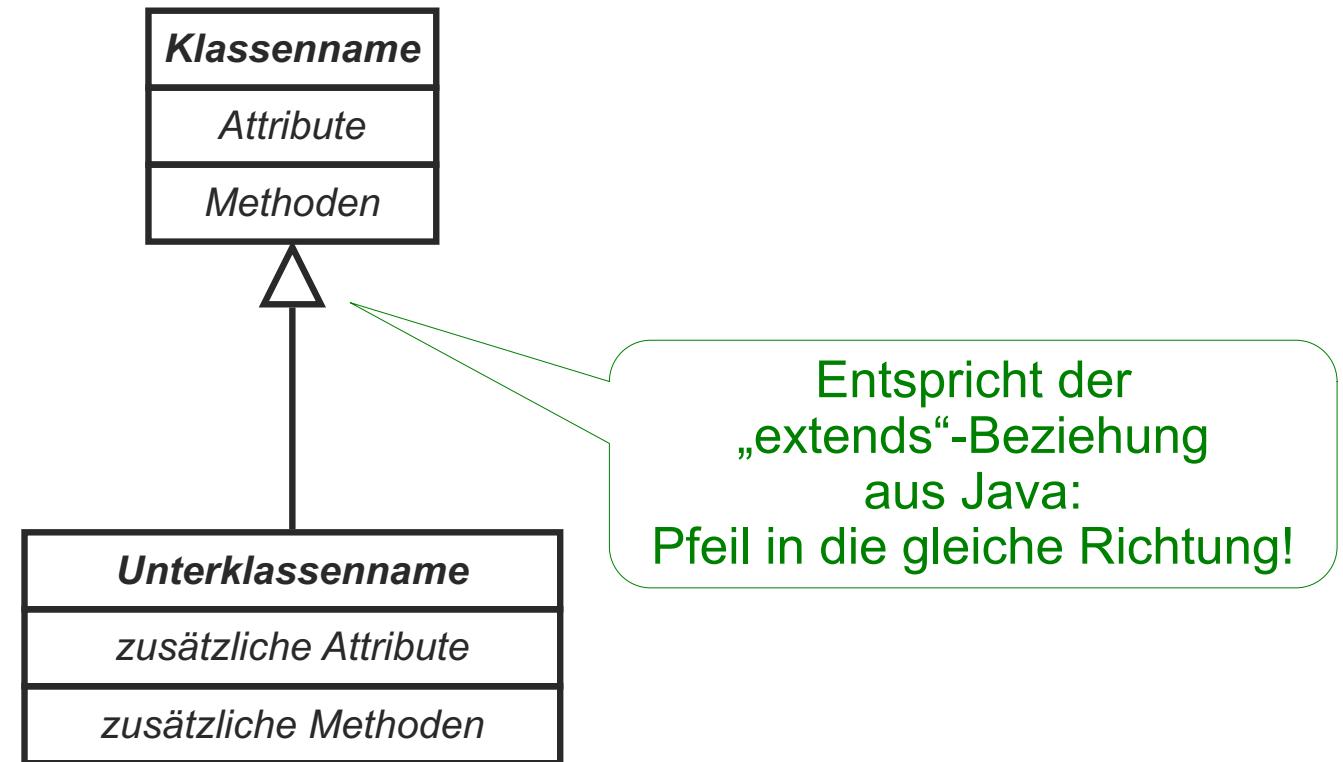


Aggregation vs. Komposition



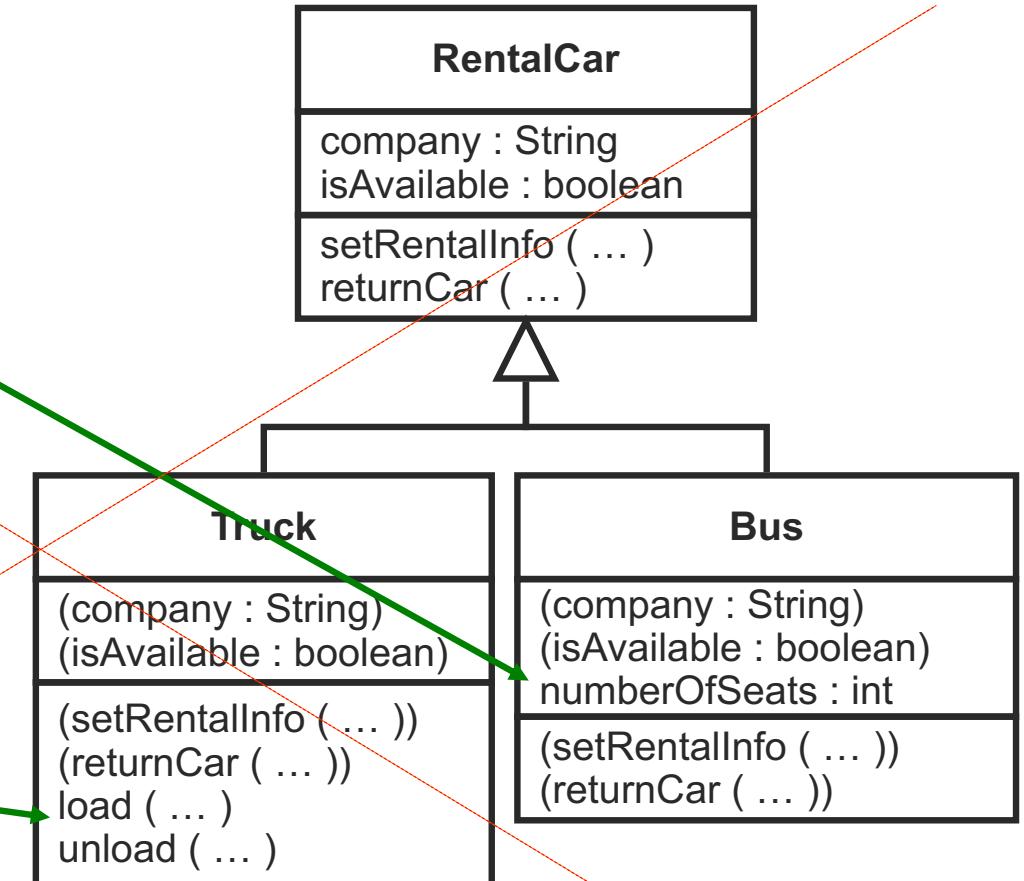
- Besondere Art der Assoziation: ein „Ganzes“ und seine „Teile“
- Auf der Seite des „Ganzen“ steht eine Raute (diamond)
- Ausgefüllte Raute
 - Die „Teile“ können *nicht* zu mehr als einem „Ganzen“ gehören
 - *Meistens* ist es nicht sinnvoll, dass „Teile“ alleine stehen
- Leere Raute
 - Die „Teile“ *können* zu mehr als einem „Ganzen“ gehören
 - Die „Teile“ können alleine stehen

Vererbung in der UML



Beispiel für Vererbung in der UML

- Ein Bus
 - ist ein (spezieller) Mietwagen,
 - der eine (große) Anzahl von Sitzplätzen bietet
- Ein Lkw
 - ist ein (spezieller) Mietwagen,
 - der beladen und entladen werden kann

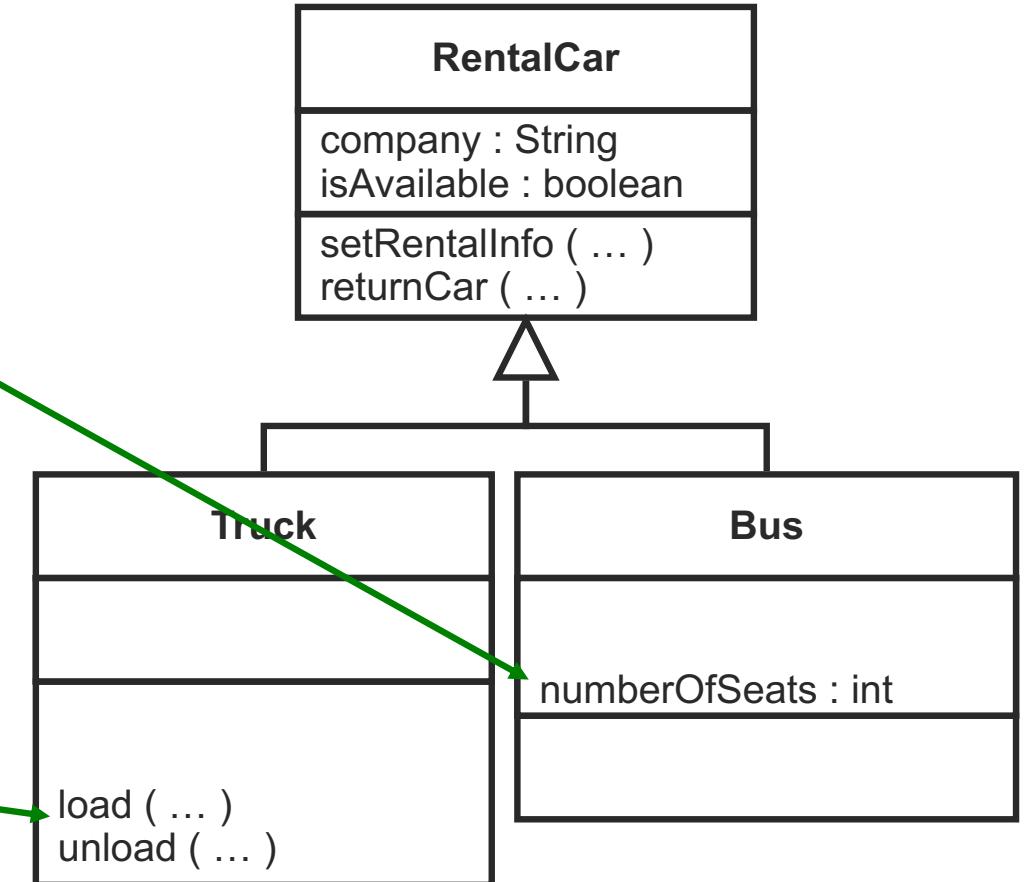


Erbte Attribute und Methoden
sind hier in Klammern gesetzt

Erbte Attribute und Methoden
werden in der UML
nicht aufgeführt

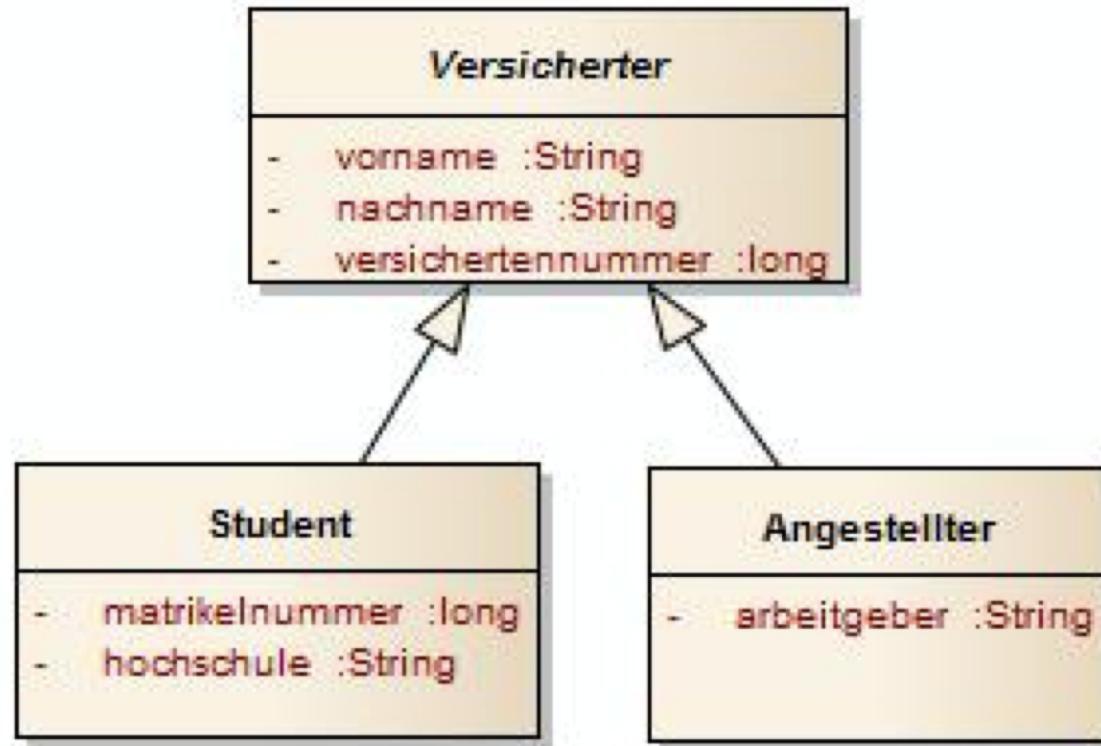
Beispiel für Vererbung in der UML

- Ein Bus
 - ist ein (spezieller) Mietwagen,
 - der eine (große) Anzahl von Sitzplätzen bietet
- Ein Lkw
 - ist ein (spezieller) Mietwagen,
 - der beladen und entladen werden kann

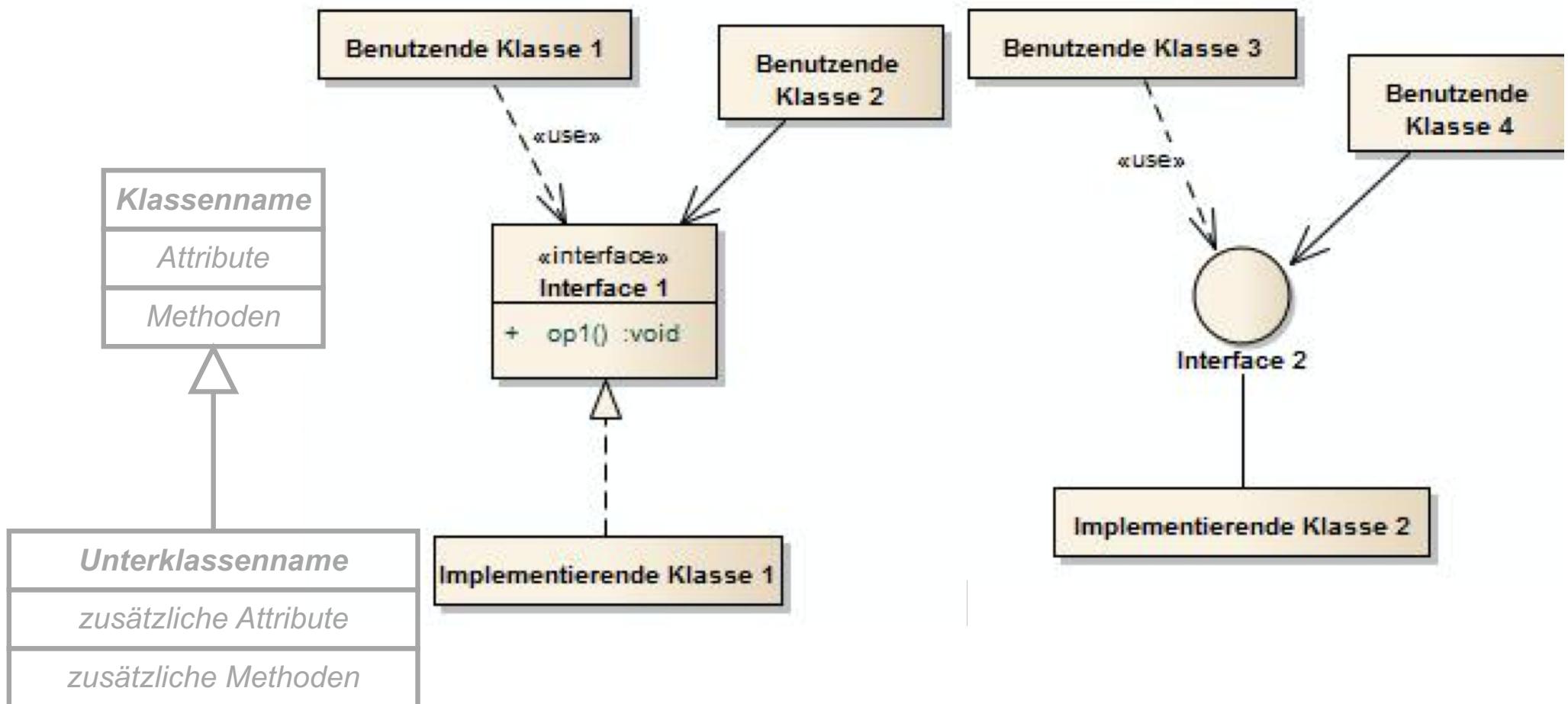


Ererbte Attribute und Methoden
werden in der UML
nicht aufgeführt

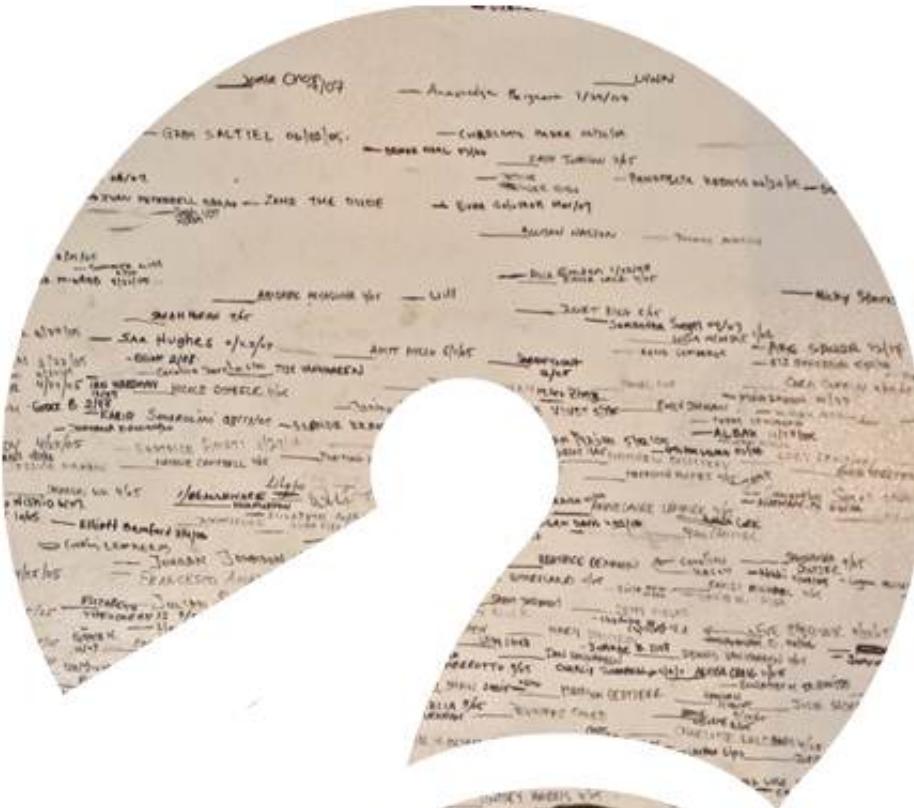
Beispiel



Darstellung eines Interface' in der UML



F R A G E N



photography: woodleywonderworks
<http://www.flickr.com/photos/wwworks/2350106729>
art work: Peter Kaiser

Überlegungen für das Design

- Wir überlegen uns, *welche Klassen* (zunächst *ohne Methoden und Felder!*) wir vermutlich benötigen werden, *zum Beispiel*:
 - eine Klasse für die GUI
 - eine Klasse für die eigentliche Verwaltung
 - eventuell weitere Hilfsklassen (die wir aber noch nicht kennen)
- Wir überlegen uns, *wie* diese Klassen *zusammenarbeiten* (genauer: welche Methoden und Felder sie dafür brauchen) und zeichnen das in ein Sequenzdiagramm ein (sowie gleichzeitig in ein Klassendiagramm)

Heuristik für das objektorientierte Design

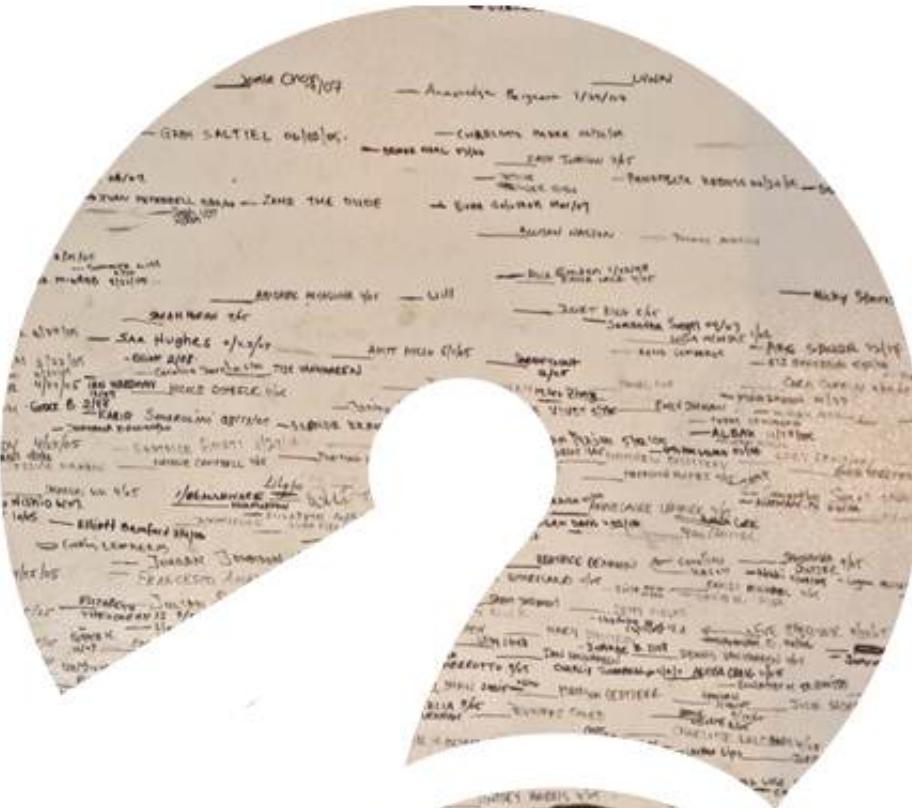
Idee

Jede Art von Objekt in der Realität wird durch eine Klasse im Klassendiagramm repräsentiert

Heuristik

1. Identifizierte Klassen und Attribute
anhand der Substantive aus der Problembeschreibung
 2. Identifizierte Beziehungen zwischen den Klassen
 3. Identifizierte Operationen (Methoden)
anhand der Verben aus der Problembeschreibung
(das machen wir noch genauer!)
 4. Überlege, ob Abstraktionen (Interfaces, Oberklassen)
sinnvoll eingesetzt werden können / sollten
- Das ist eine grobe Vereinfachung!!

F R A G E N



photography: woodleywonderworks
<http://www.flickr.com/photos/wwworks/2350106729>
art work: Peter Kaiser