Java Einführung
Umsetzung von Beziehungen zwischen Klassen

Kapitel 7

Inhalt

- Wiederholung: Klassendiagramm in UML

- Java-Umsetzung von
  - Generalisierung
  - Komposition
  - Assoziationen

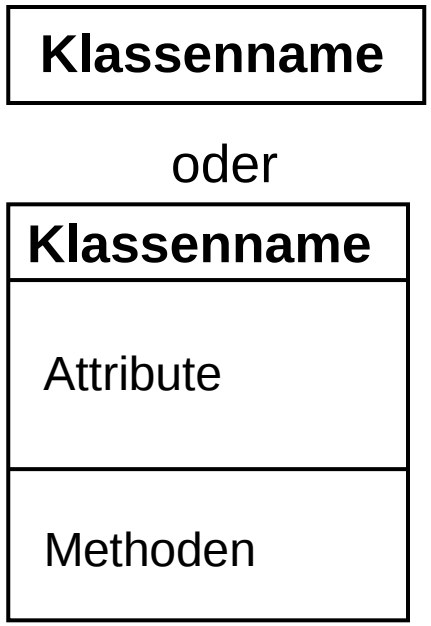
Das Klassendiagramm

- Zweck der Darstellung
  - Logischen Aufbau des Systems
  - Statischen Aspekte
  - Zusammenhänge und Beziehungen zwischen den Komponenten

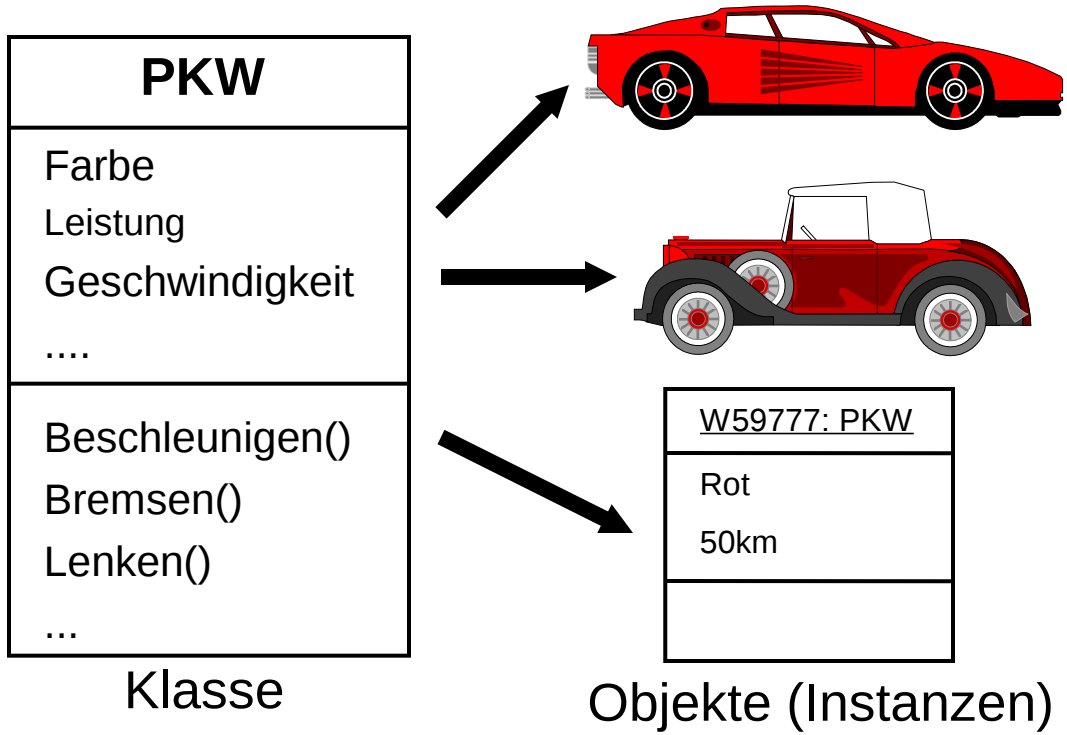
Klassen in UML

Klassen sind die Baupläne für Objekte (Instanzen)

Darstellung in UML



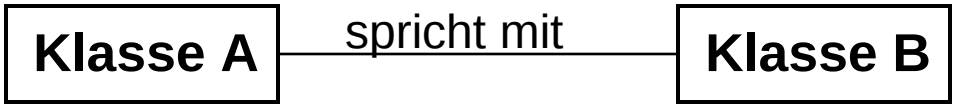
Beispiel einer Klasse in UML



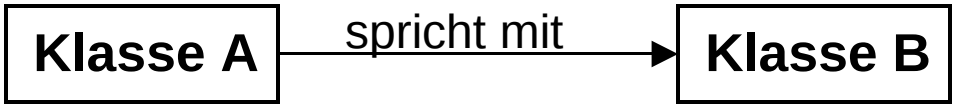
Elemente Klassendiagramm I

- Klassen

- Beziehungen (Paths/Associations): für die Übermittlung von Nachrichten (Messages = Aufruf von Methoden)



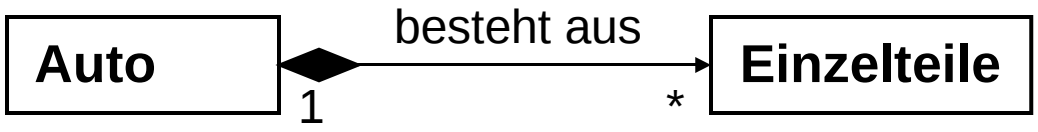
- Richtung der Beziehungen (Navigability): Nachrichten können nur in diese Richtung gehen



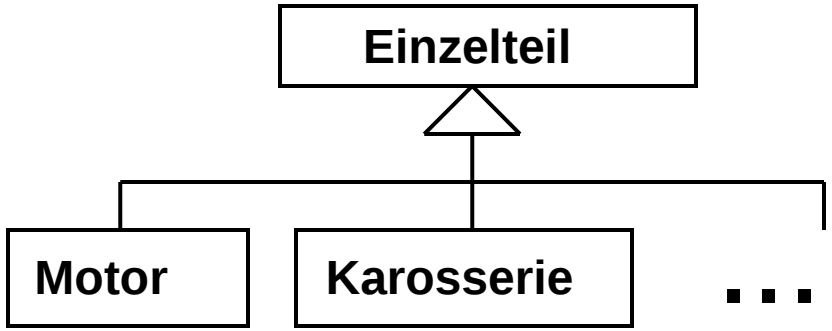
Elemente Klassendiagramm II

- Kardinalitäten (Multiplicity): Anzahl der möglichen Instanzen

- Aggregation (Aggregation/Composition): Besteht-aus Beziehung



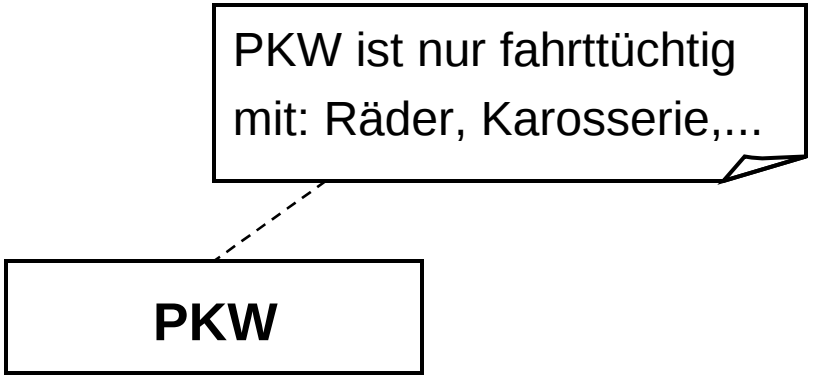
- Generalisierung (Generalization): is-a Beziehung -> Vererbung in Java



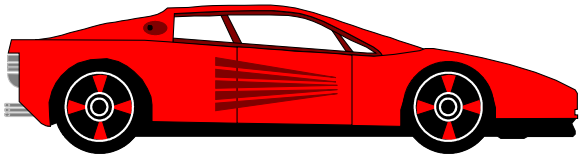
Elemente Klassendiagramm III

- Abhängigkeiten (Dependencies)

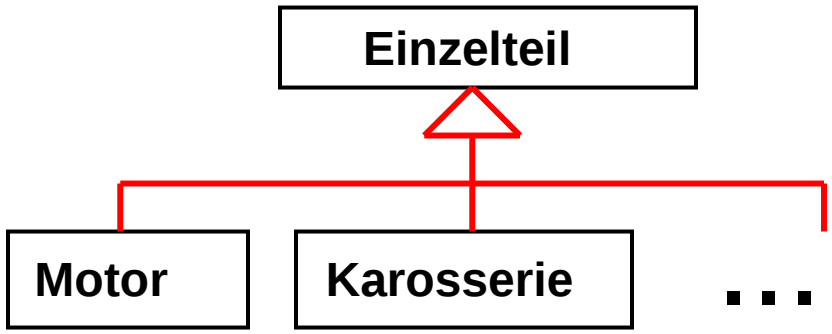
- Anmerkungen oder Einschränkung



Beispiel: Klassendiagramm



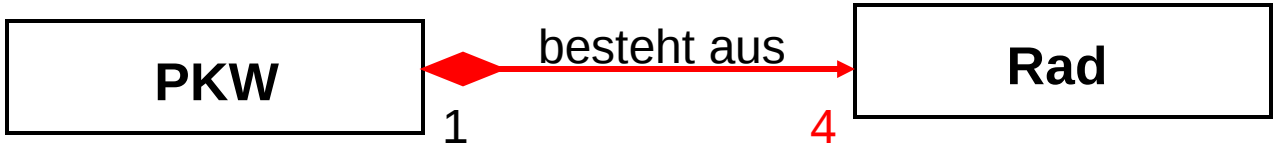
Umsetzung von Generalisierung



Vererbung

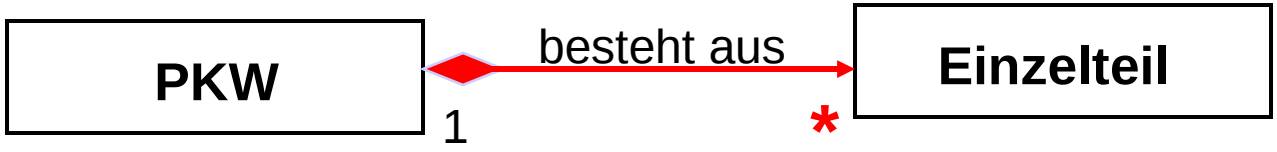
```
abstract class Einzelteil {
  // muss nicht unbedingt abstract sein!
  ...
}

class Motor extends Einzelteil {
  ...
}
```

Umsetzung von Komposition I




```
class Rad {...}

class PKW {
  Rad liVoRad, reVoRad, liHiRad, reHiRad; //Instanznamen

  PKW() {
     /* Erzeugung der Instanzen der Räder */
    liVoRad = new Rad();
    reVoRad = new Rad();
    liHiRad = new Rad();
    reHiRad = new Rad();
  }
}
```

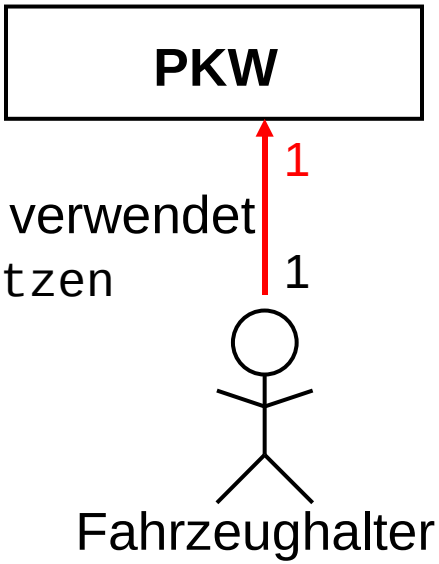
Umsetzung von Komposition II




```
class Einzelteil {...}

class PKW {
  final int ANZAHL=100; // besteht aus 100 Teilen
  Einzelteil[] dieEinzelteile; /* Implementierung als Array
  von Instanznamen (auch als Container moeglich) */

  PKW() {
    dieEinzelteile = new Einzelteil[ANZAHL]; /* Array
  erzeugen */
    for(int i=0;i<ANZAHL; i++) {
      dieEinzelteile[i] = new Einzelteil(); /* Erzeugung
  der Instanzen der Einzelteile */
   }
  }
}
```

Umsetzung einer Assoziation

```
class PKW {...}
class Fahrzeughalter {
  PKW meinPKW; // nur Instanzname
  ...
  void PKWkaufen(PKW einPKW) {
    meinPKW=einPKW; // Referenz auf die Inst. setzen
  }
}

class Michael {
public static void main(String[] args) {
  PKW pkw1 = new PKW(); // PKW erzeugen
  Fahrzeughalter michael = new Fahrzeughalter();
  michael.PWKkaufen(pkw1);
      // PKW an den Farzeughalter uebergeben
  }
}
```



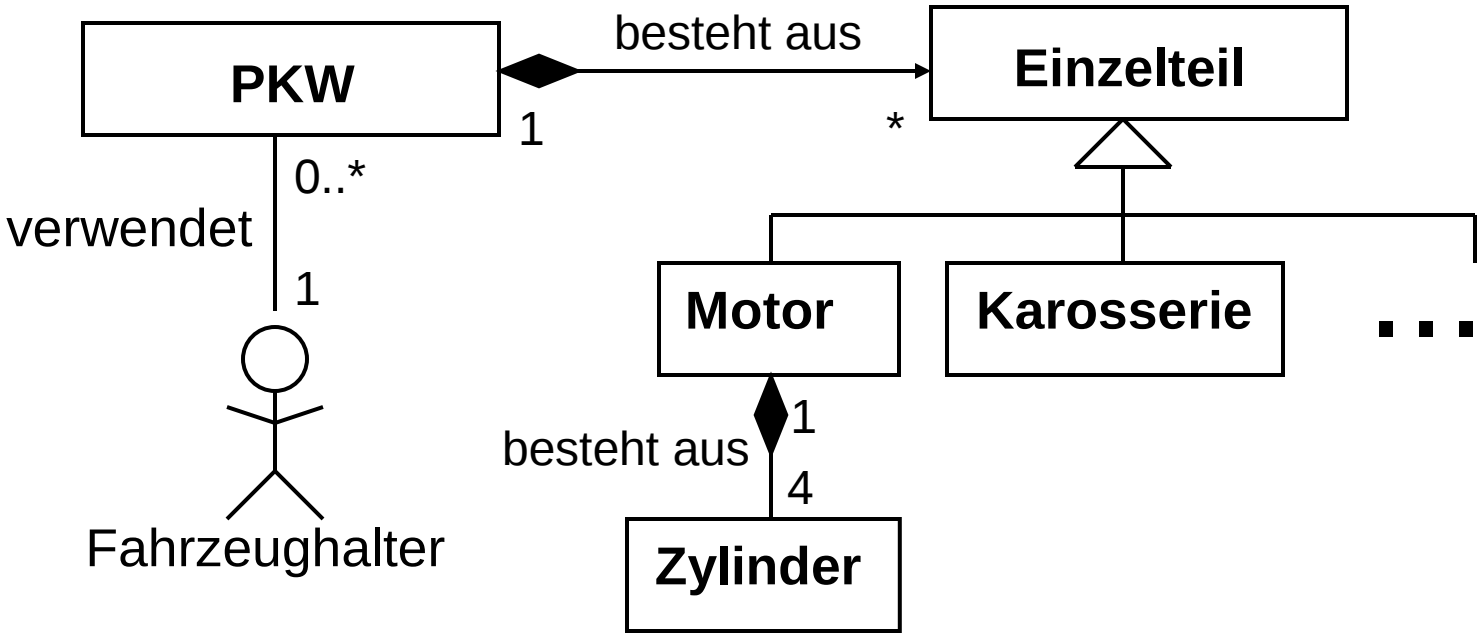
Umsetzung von komplexeren Assoziationen

- Assoziationen mit höherer (>1) Kardinalität
  - Es muss ein Container (z.B. Array, Liste) verwendet werden.

- Assoziationen mit Navigierbarkeit in beide Richtungen:
  - Jede Seite muss eine Referenz der anderen Seite haben.
    Achtung: Problem der Konsistenz!

Übung

- Implementieren Sie das gesamte Klassendiagramm. Überlegen Sie sich sinnvolle Attribute und Methoden sowie Einzelteile (mindestens 4).



Nach dieser Einheit sollten Sie wissen, ...

- wie Beziehungen in UML-Klassendiagrammen dargestellt werden
- welche Arten von Beziehungen es gibt
- wie diese Beziehungen in Java umgesetzt werden können.