EXKURS UML

Beziehungen zwischen Objekten und Typen

Christian Schirmer

Beziehungen zwischen Objekten und Typen

- Assoziation
- Aggregation
- Komposition
- Has-A sowie Is-A Beziehungen

ASSOZIATION

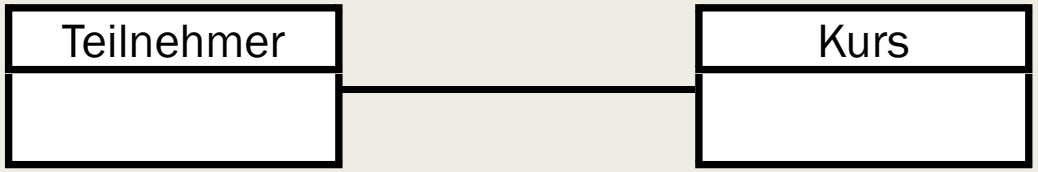
Assoziation

- Eine Assoziation bezeichnet eine Beziehung zwischen Klassen.
  - *Eine Assoziation in einfachsten falle, meldet keine Richtung und oder Art der Beziehung.*
  - *Am Häufigsten wird die Binäre Assoziation verwenden, womit die Beziehung zwischen 2 Klassen beschrieben wird.*
  - *Die UML bietet viele zusätzliche Notationen womit eine Assoziation genauer beschrieben werden kann.*
  - *Es gibt ebenso Unäre wie auch n-äre Assoziationen.*

- Ich empfehle für die Praxis sich in UML tiefer einzuarbeiten, da sie für die Kommunikation und Planung mittlerweile unverzichtbar geworden ist.

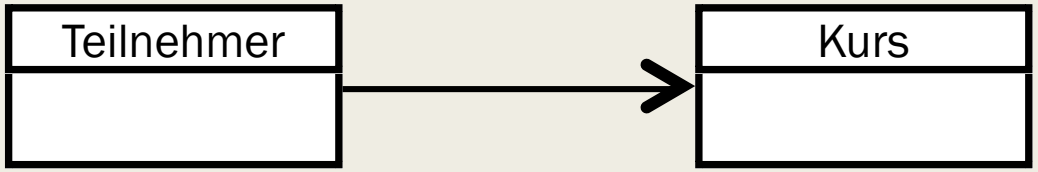
Assoziation im Klassendiagramm

- Die ist eine ungerichtete Assoziation zwischen den Klassen Teilnehmer und Kurs.
- Es wird keine Aussagen getroffen außer das die beiden Klassen irgendwie miteinander bekannt sind.
- Die Modellierung überlässt es den Quellcode Entwickler zu entscheiden, wer mit wem.
  - *Was im übrigen keine gute Idee ist. Ein Gute Planung sollte möglichst wenig Spielraum lassen.*



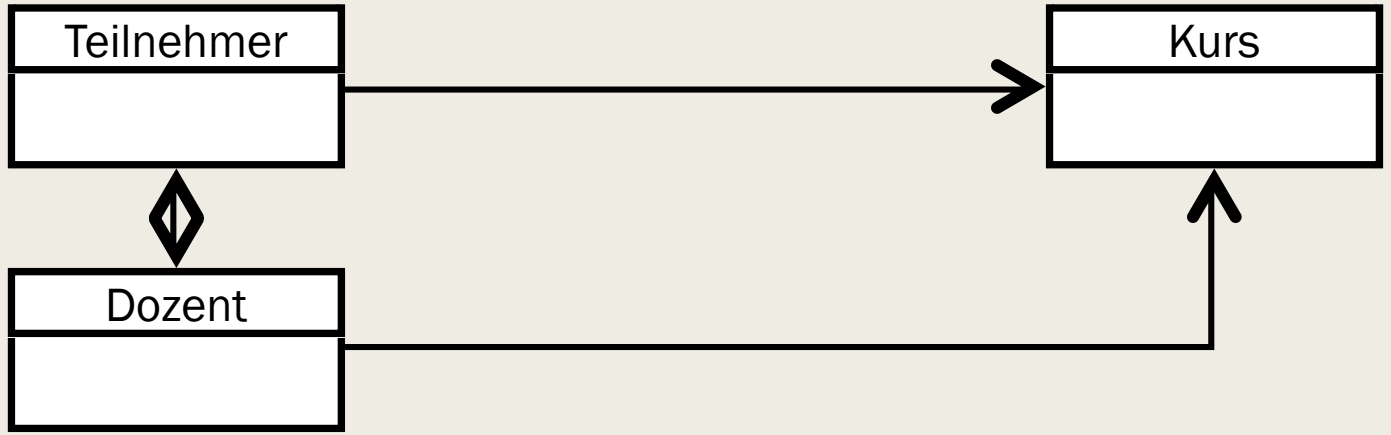
Assoziation im Klassendiagramm

- Eine gerichtete Assoziation hingegen, gibt eine Aussage wer kennt wem.
  - *Hier im Beispiel nun, kennt der Teilnehmer den Kurs aber der Kurs nicht den Teilnehmer.*
  - *Wie auch Sie Ihren Kurs kennen – Java Development Expert OCP oder so ähnlich.*



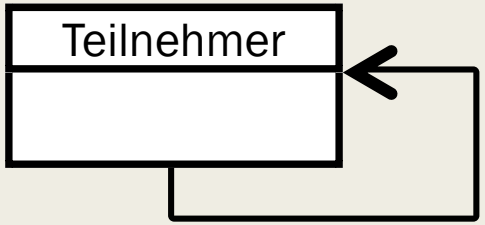
Assoziation im Klassendiagramm

- Eine Bidirektional gerichtete Assoziation wie hier zwischen Teilnehmer und Dozent erlaubt ein gegenseitiges Aufrufen von Operationen.
  - *Der Dozent kann den Teilnehmer Fragen stellen*
  - *Der Teilnehmer seinerseits kann dem Dozenten Fragen stellen*
  - *Der Dozent kennt ebenfalls den Kurs, um dessen Inhalte vermitteln zu können.*



Assoziation im Klassendiagramm

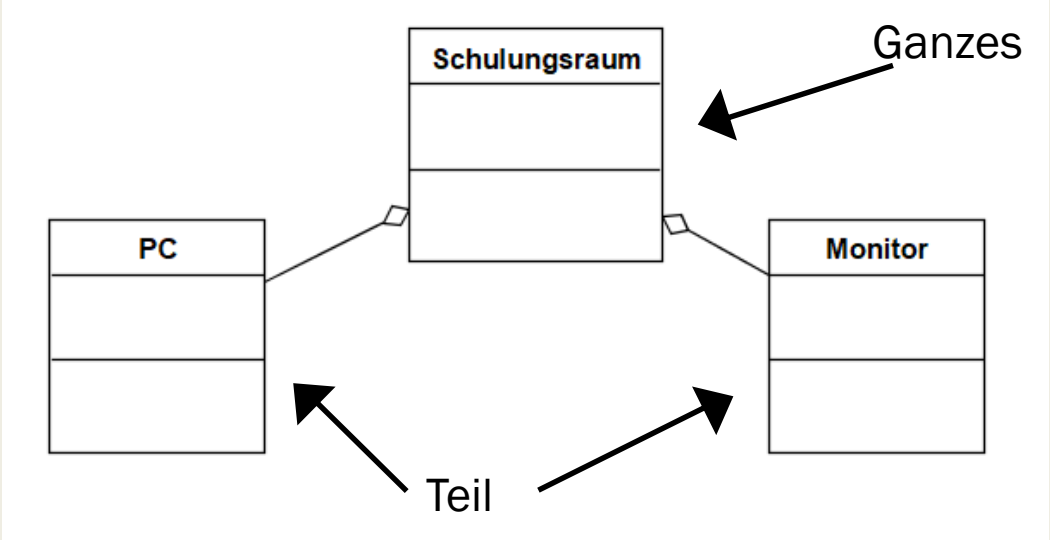
- Eine Unäre Beziehung, wie hier dargestellt bedeutet:
  - *Der Teilnehmer kann mit sich Selbst bzw. mit anderen Teilnehmern interagieren.*
  - *Unär bezieht sich auf die Klasse, wie auch das Klassendiagramm, nicht auf ein Konkretes Objekt bzw. eine Instanz einer Klasse.*



AGGREGATION

Aggregation

- Die Aggregation stellt eine spezielle Ausprägung der binären Assoziation dar. Sie beschreibt eine Lose Ganze-Teile-Beziehung. Eine Definition einer Aggregation mit mehr als 2 Enden gibt es in der UML nicht.
- Ein Schulungsraum besteht inhaltlich aus PCs und Monitoren.
- Die Lose Verbindung die die Aggregation darstellt besagt das die Einzelnen Teile wie auch das Ganze unabhängig voneinander bestehen kann.



Aggregation

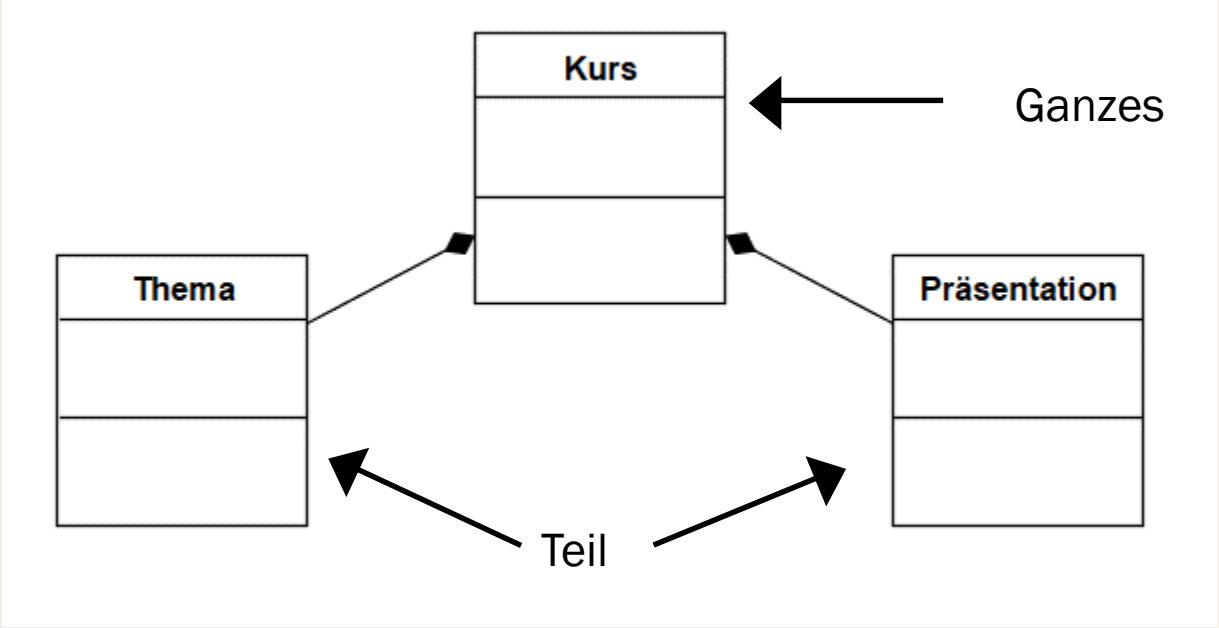
- In Java bedeutet das
  - *Die Klasse Schulungsraum besteht auch ohne PCs und Monitore, wenn die Listen leer sind.*
  - *Die Monitore und die PCs existieren auch außerhalb der Klasse Schulungsraum.*

```
public class Schulungsraum {

    public ArrayList<PC> listePC;
    public ArrayList<Monitor> listeMonitor;

    public Schulungsraum() {
        listePC = new ArrayList();
        listeMonitor = new ArrayList();
    }
```

KOMPOSITION

Komposition

- Die Komposition stelle feste Form der Assoziation dar, womit ebenfalls eine Ganzes-Teile-Beziehung notiert wird. Allerdings sind die Verbindungen zwischen den Teilen und dem Ganzen als untrennbar definiert.
  - *Die Komposition wird verwendet, wenn man eine Existenznotwendigkeit der Teile zum Ganzen definieren will. Ohne dem Ganzen existieren die einzelnen Teile auch nicht.*



Komposition

- Im Gegensatz zur Aggregation, wo die Teile eigenständig erzeugt wurde, ist nun das Ganze für die Erzeugung und auch Löschung verantwortlich.
  - *Das heißt hier erstellt der Kurs die Themen und Präsentationen. Diese können über den Kurs abgefragt werden. Eine GetThema bzw. GetPräsentation liefert dann entsprechend eines der Attribute nach außen.*
  - *Bei der **Aggregation** werden **Referenzen auf Objekte** als Teile gespeichert. Wird das Ganze gelöscht, gehen nur die Referenzen verloren. Die einzelnen Teile blieben weiter bestehen.*
  - *Bei der **Komposition** werden die **Objekte selbst** erschaffen und gespeichert. Wird das Ganze, in diesem Fall der Kurs , gelöscht, bedeutet dies auch das Ende aller Objekte (Themen wie auch Präsentationen), die er in sich speichert.*

Komposition

- In Java bedeutet das
  - *Kurs erzeugt seine Themen anhand der Übergebenen Informationen.*
  - *Beim auslesen der Themen wird eine Werte-Kopie übergeben.*

```java
public class Kurs implements IKurs {

    private ArrayList<Thema> themen;

    public Kurs() {
        themen = new ArrayList<>();
    }

    /**
     * Beim Hinzufügen geben wir lediglich eine Information zu den neu zu
     * erzeugenden Teil der Komposition mit.
     *
     * @param nr
     */
    public void addThema(int nr) {
        themen.add(new Thema(nr));
    }

    /**
     * Es wird nur eine Kopie herausgegeben. Womit die Komposition ihre Aussage
     * behält, wenn das Ganze gelöscht wird, werden auch die Teile gelöscht.
     *
     * @return
     */
    public Thema getThema() {
        Thema neuesThema = new Thema(0);
        neuesThema.themenNr = themen.get(themen.size() - 1).themenNr;
        return neuesThema;
    }
```

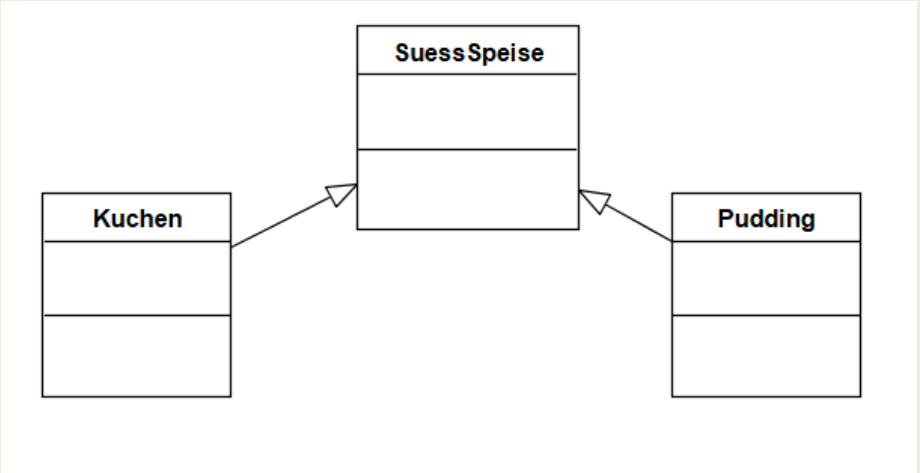
HAS-A UND IS-A

Has-A

- Has-A (hat ein/e) Beziehungen, bedeuten das die Klasse ein Attribute der anderen Klasse besitzt. Die Komposition wie auch die Aggregation bezeichnen eine Has-A Beziehung.
- Eine Has-A Beziehung ist unabhängig davon ob es sich um ein Einzelnes Attribut oder um eine ganze Liste handelt.
  - *Die untenstehende Abbildung zeigt eine Has-A Beziehung zwischen Kurs und Thema. Kurs hat ein oder mehrere Themen*

IS-A

- Die IS-A (ist ein/e) Beziehung kommt in der Vererbung in der Objekt Orientierten Programmierung daher. Da geht es um Basis- (Eltern-) und Sub-(Kind-)klassen.
  - *Subklassen sind von den Basisklassen abgeleitet. Bei einer IS-A Prüfung wird die Vererbungshierachie nach oben abgefragt.*
  - *Kuchen wie auch Pudding sind abgeleitet von SuessSpeise. Somit würden beide Klassen die Prüfung nach „ist ein/e“ SuessSpeise bestehen.*
  - *Umgekehrt ist eine SuessSpeise nicht unbedingt ein Pudding oder Kuchen*



IS-A

- Die IS-A (ist ein/e) Beziehung kommt ob in der Vererbung in der Objekt Orientierten Programmierung daher. Da geht es um Basis- (Eltern-) und Sub-(Kind-)klassen.
  - *Ein Interface stellt eine besondere art Abstrakter Klasse dar.*
  - *Nun würden alle 3 Klassen – SuessSpeise, Pudding und Kuchen – eine Prüfung „ist ein/e“ Essbar bestehen. - instanceof*
  - *SuessSpeise ist weiterhin nicht unbedingt eine Pudding oder Kuchen.*