Beziehungen zwischen Klassen

1 Assoziation

1.1 Definition

Eine Assoziation ist eine allgemeine Beziehung zwischen einem oder mehreren Objekten.

Bemerkung:
Eine Assoziation ist eine Beziehung zwischen Objekten, nicht zwischen Klassen. Es ist jedoch üblich von einer Assoziation zwischen Klassen zu sprechen, obwohl streng genommen die Objekte dieser Klassen gemeint sind.

1.2 Darstellung in UML

1.2.1 Kurzbeschreibung

Eine Assoziation kann einen **Assoziationsnamen** haben (Beispiel unten: "besitzt").
Mit einem ausgefüllten Dreieck kann die **Leserichtung** dieser Assoziation angegeben werden.
Diese 2 Eigenschaften sind optional, müssen also nicht im UML-Diagramm angegeben werden.

Mit einer **Rolle** (Beispiel unten: auto1, fahrer1) wird das Attribut (also der Name des assoziierten Objekts) bezeichnet. Man kann den Rollennamen entweder an das Pfeilende oder in die Klasse schreiben.

Die **Kardinalität** (Multiplizität) gibt an, wie viele Objekte aus der einen Klasse (Beispiel unten: 1..n, d.h. zwischen 1 und n Objekten) mit wie vielen Objekten der anderen Klasse (Beispiel unten: 1..m, d.h. zwischen 1 und m Objekten) in Beziehung stehen.

Mengenangaben:

0	:	keins
1	:	genau ein
0..n	:	0,1,2,...,n
1..n	:	1,2,...,n
4..20	:	4,5,...,20
1,5,7	:	1,5,7
*	:	beliebig viele
0..*	:	beliebig viele (wie *)
n	:	n

Die **Navigation** (Navigationsrichtung) gibt die Richtung (durch eine Pfeillinie angegeben) an, in der man von einer Klasse zur anderen navigieren kann (wie man die andere Klasse “erreichen“ kann).
Es gibt mehrere Möglichkeiten:
a) Assoziationen, die genau auf einer Seite navigierbar sind, heißen unidirektional und werden durch einen Pfeil am Linienende eingetragen.
b) Assoziationen, die auf beiden Seiten navigierbar sind, heißen bidirektionale Assoziationen. Sie werden ohne Pfeile an den Linienenden dargestellt. (Manchmal sieht man aber auch an allen zwei Linienenden Pfeile).
c) Ein Kreuz am Pfeilende bedeutet eine nicht erlaubte Navigationsrichtung.

1.2.2 Beispiel

Ein Auto kann mehrere Fahrer haben.

1.2.2.1 Mögliche Darstellung

Die Klasse Auto besitzt das Attribut dieFahrer[] und die Klasse Fahrer besitzt das Attribut dasAuto. Die Assoziation ist bidirektional.

Bemerkung:

Bei einer unidirektionalen Assoziation (Vom Auto zum Fahrer) sieht das Beispiel oben wie folgt aus:

1.2.2.2 Mögliche Realisierung der Assoziation

```
public class MainAssoziationUNI_1zuN_3 {
  public static void main(String[] args){
    Fahrer[] dieFahrer = new Fahrer[2];
    dieFahrer[0]=new Fahrer("Maier");
    dieFahrer[1]=new Fahrer("Müller");
    // Verlinkung: myCar --> dieFahrer
    Auto myCar = new Auto ("B-KI 11", dieFahrer);
    System.out.println("Name 1. Fahrer =" +myCar.getFahrer(0).getName());
    System.out.println("Name 2. Fahrer =" +myCar.getFahrer(1).getName());
  }
}

class Auto{
  private String kfzZeichen;
  private Fahrer[] seineFahrer;

  public Auto(String pKfzZeichen, Fahrer[] pSeineFahrer){
    kfzZeichen=pKfzZeichen;
    seineFahrer = pSeineFahrer;
  }

  public void setKfzZeichen(String pKfzZeichen){
    kfzZeichen=pKfzZeichen;
  }

  public void setSeineFahrer(Fahrer[] pSeineFahrer){
    seineFahrer = pSeineFahrer;
  }

  public void setFahrer(int i, Fahrer pDerFahrer){
    seineFahrer[i] = pDerFahrer;
  }

  public Fahrer getFahrer(int i){
    return(seineFahrer[i]);
  }
}

class Fahrer{
  private String name;

  public Fahrer(String pName){
    name = pName;
  }

  public void setName(String pName){
    name = pName;
  }

  public String getName(){
    return(name);
  }
}
```

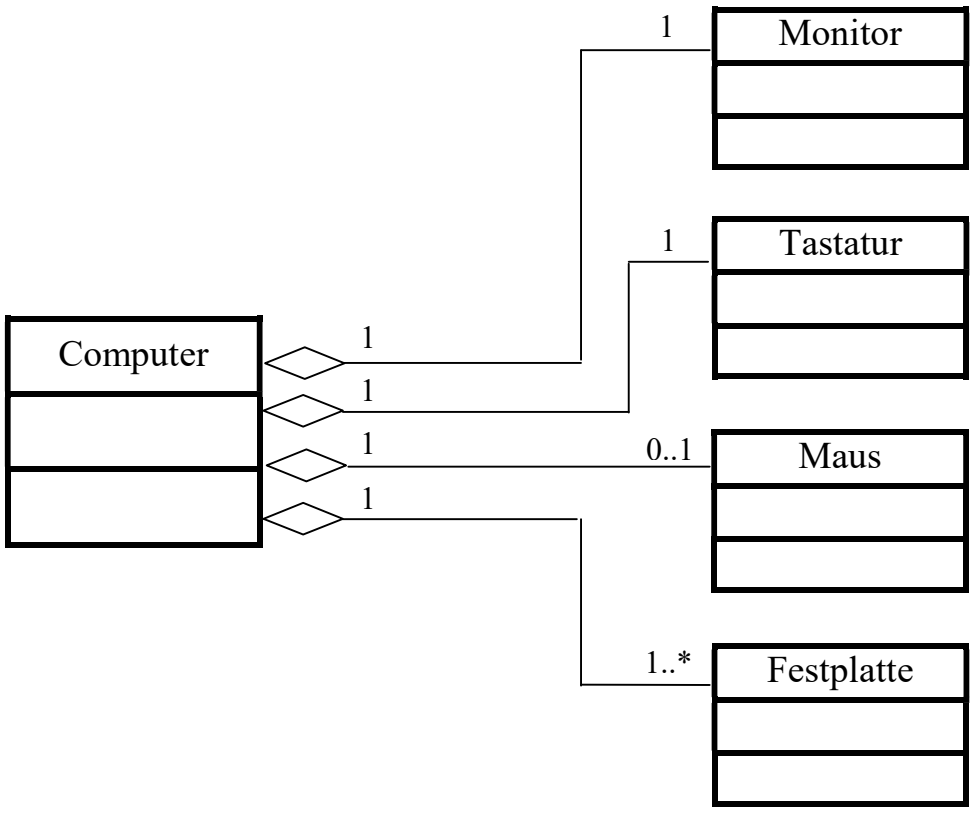
2 Aggregation

2.1 Definition

Eine Aggregation ist eine spezielle Assoziation und bezeichnet eine "Teil-Ganzes" oder "ist Teil von" Beziehung.
Stirbt das "Ganz-Element", lebt das "Teil-Element" noch weiter.
Ein Objekt besteht aus einem oder mehreren anderen Objekten. In ihr wird die **hat** (**besteht aus**) - Beziehung realisiert. Das Ganz-Element dieser Beziehung besteht aus den Teil-Elementen.

2.2 Darstellung in UML

Beispiel:
Ein Computer besteht aus mehreren Bestandteilen.
Wird der Computer ausgemustert, sind seine Bestandteile (Monitor, Tastatur, usw.) immer noch vorhanden.



weiteres Beispiel:
1)
Ganz-Teil: Ehepaar
Teil-Element(e): Person1, Person2
Wird die Ehe geschieden, existieren die 2 Personen immer noch.

2.3 Realisierung der Aggregation

Die Teil-Elemente leben noch, wenn das Ganz-Element stirbt.

2.3.1.1 Beispiel (ein Vorschlag)

```
public class MainAggregation10 {
  public static void main(String[] args) {
    Teil1 t1;
    Teil2 t2;
    t1 = new Teil1("Schraube");
    t2 = new Teil2(10);
    Gesamt g;
    g = new Gesamt(t1, t2);
    g = null;
    System.out.println("t1-Name="+t1.getName());
    System.out.println("t2-Gewicht="+t2.getGewicht());
  }
}

class Teil1{
  private String name;

  public Teil1(String pName){
    name = pName;
  }

  public String getName(){
    return name;
  }
}

class Teil2{
  private int gewicht;

  public Teil2(int pGewicht){
    gewicht = pGewicht;
  }

  public int getGewicht(){
    return gewicht;
  }
}

class Gesamt{
  private Teil1 t1;
  private Teil2 t2;

  public Gesamt(Teil1 pTeil1, Teil2 pTeil2){
    t1 = pTeil1;
    t2 = pTeil2;
  }
}
```

2.3.1.2 Unterschied (programmtechnisch) Assoziation - Aggregation

Der (programmtechnische) Unterschied zwischen Assoziation und Aggregation ist eher philosophischer Natur. Bei der Umsetzung in Java ist in jedem Fall die Klasse "Teil-Element" (bzw. assoziierte Klasse) ein Attribut der Klasse "Ganz-Element" (bzw. Klasse von der der Assoziationspfeil wegführt).

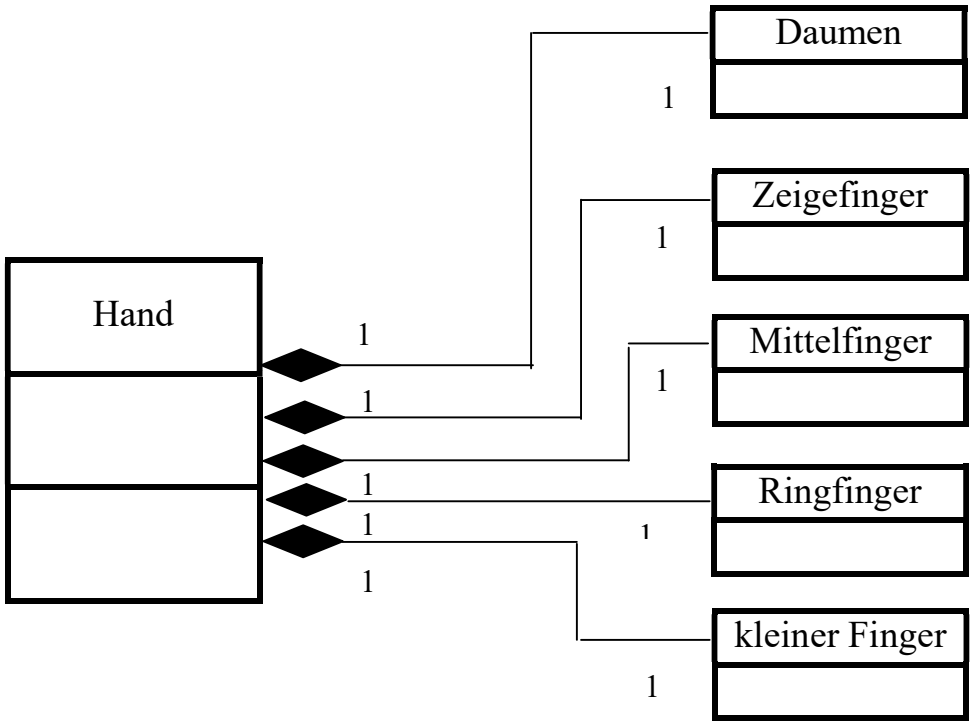
3 Komposition

3.1 Definition

Eine Komposition ist eine ganze enge Aggregation. Wenn der Ganz-Teil stirbt, sterben auch die Teil-Elemente (wird z.B. ein Haus abgerissen, werden auch die Räume zerstört).

3.2 Darstellung in UML

Beispiel:
Eine Hand besteht aus 4 Fingern und dem Daumen. Stirbt die Hand (wird sie z.B. bei einem Unfall mit der Kettensäge abgetrennt), sterben die Finger und der Daumen ab.



weitere Beispiele:
1)
Ganz-Teil: Wohnung
Teil-Element(e): Küche, Bad, Wohnzimmer, Schlafzimmer
Wird die Wohnung abgerissen, existieren die Einzelzimmer nicht mehr.

2)
Ganz-Teil: Auto
Teil-Element(e): Kofferraum
Wird das Auto verschrottet, gibt es auch keinen Kofferraum mehr.

3.3 Realisierung der Komposition

Das Objekt, das aus mehreren anderen Objekten besteht, enthält diese jeweils als Attribut. Wenn der Ganz-Teil stirbt, sterben auch die Teil-Elemente.

3.3.1.1 Beispiel(ein Vorschlag)

```
public class MainKomposition10 {
  public static void main(String[] args) {
    Gesamt g;
    g = new Gesamt("Schraube", 10);
    g = null;  // Ganz-Teil stirbt
    // Hier gibt es Laufzeitfehler
    System.out.println("t1-Name="+g.getTeil1().getName());
    System.out.println("t2-Gew.="+g.getTeil2().getGewicht());
  }
}

class Teil1{
  private String name;

  public Teil1(String pName){
    name = pName;
  }

  public String getName(){
    return name;
  }
}

class Teil2{
  private int gewicht;

  public Teil2(int pGewicht){
    gewicht = pGewicht;
  }

   public int getGewicht(){
     return gewicht;
   }
}

class Gesamt{
  private Teil1 t1;
  private Teil2 t2;

  public Gesamt(String pName, int pGewicht){
    t1 = new Teil1(pName);
    t2 = new Teil2(pGewicht);
  }

  public Teil1 getTeil1(){
    return t1;
  }

  public Teil2 getTeil2(){
    return t2;
  }
}
```