Woche 3 – Lernpaket Informatik – GK 11

Beziehungen (Assoziationen) zwischen Objekten verschiedener Klassen

Um eine bestimmte Situation möglichst realitätsnah mit einem objektorientierten Programm abzubilden, werden meist mehrere Objekte verschiedenster Klassen benötigt. Diese Objekte können miteinander interagieren und kommunizieren und somit zueinander in einer bestimmten Beziehung stehen.

Generell werden drei verschiedene Beziehungsarten unterschieden:

1. HAT – Beziehung
    a. Aggregation (existenzunabhängig)
    b. Komposition (existenzabhängig)
2. KENNT – Beziehung
3. IST – Beziehung (Vererbung)

Um Beziehungsstrukturen graphisch darzustellen, nutzt man meistens sogenannte **UML-Diagramme**. In diesen werden die Klassenkarten der beteiligten Klassen durch verschiedene Verbindungen zueinander in Beziehung gesetzt. Die Art der Verbindung ist für jeden Beziehungstyp eindeutig.

(Hinweis: Gelegentlich wird der besseren Übersicht wegen nur der Klassenname und nicht die gesamte Klassenkarte einer Klasse im UML-Diagramm aufgeführt.)

1. HAT – Beziehung

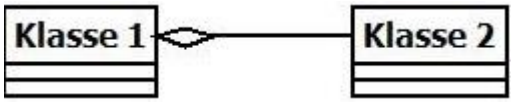
Objekte der `Klasse 1` bestehen aus Objekten der `Klasse 2`

→ In `Klasse 1` gibt es Attribute vom Typ der `Klasse 2`

a. Aggregation

Die beteiligten Objekte führen ein Eigenleben. Das „untergeordnete“ Objekt der `Klasse 2` kann auch ohne das „übergeordnete“ Objekt der `Klasse 1` existieren (existenzunabhängig).

Darstellung einer Aggregation in einem UML – Diagramm



Bei der Implementierung wird eine Aggregation dadurch realisiert, dass die `Klasse 1` ein Attribut vom Typ der `Klasse 2` erhält, welches im Konstruktor als Parameter mitübergeben wird.

b. Komposition

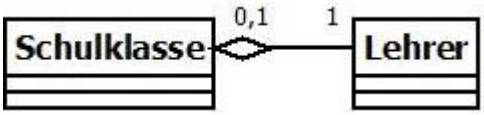
Ein Objekt der „übergeordneten“ `Klasse 1` ist auch für die Erzeugung eines anderen Objekts der „untergeordneten“ `Klasse 2` zuständig. Das erzeugte Objekt der „untergeordneten“ `Klasse 2` ist ohne das übergeordnete Objekt gar nicht vorhanden (existenzabhängig).

Darstellung einer Komposition in einem UML – Diagramm

Bei der Implementierung wird eine Komposition dadurch realisiert, dass die `Klasse 1` ein Attribut vom Typ der `Klasse 2` erhält, welches im Konstruktor neu erzeugt wird (durch Konstruktoraufruf der `Klasse 2`).

Beispiele:

a. Aggregation



Jede Schulklasse sollte einen Klassenlehrer haben. Um dies zu realisieren, wird eine Aggregations-Beziehung zwischen den Klassen `Schulklasse` und `Lehrer` festgelegt.
Ein Objekt der Klasse `Schulklasse` soll daher ein Attribut `klassenlehrer` vom Typ `Lehrer` besitzen. Es ist dabei jedoch klar, dass der Lehrer unabhängig von der Schulklasse existiert. Sollte die Schulklasse nichtmehr existieren, bleibt der Lehrer dennoch bestehend.

Die Zahlen (**Multiplizitäten**) an der Beziehungsverbindung geben jeweils an, wie viele Objekte der beiden Klassen an der Beziehung beteiligt sind.

Hier: Jede Schulklasse hat genau einen Klassenlehrer. Jeder Lehrer kann entweder für keine oder genau eine Klasse der Klassenlehrer sein.

b. Komposition

Bankkunden besitzen bei ihrer Bank ein Konto. Um dies zu realisieren, wird eine Kompositions-Beziehung zwischen den Klassen `Kunde` und `Konto` festgelegt.
Ein Objekt der Klasse `Kunde` soll daher ein Attribut `girokonto` vom Typ `Konto` besitzen. Hier ist aus dem Kontext ersichtlich, dass das Konto nicht unabhängig von seinem Kunden existiert. Sollte ein Bankkunde nichtmehr existieren, bleibt auch sein Konto nicht bestehend.

Die Multiplizitäten lassen darauf schließen, dass ein Bankkunde nur genau ein Konto besitzen kann und jedes Konto gehört nur zu genau einem Kunden.

Aufgabe 0

Versuch die dargestellten Inhalte zu verstehen und hefte sie zu deinen Unterrichtsunterlagen.

Aufgabe 1

Überlege dir jeweils ein eigenes Beispiel für eine Aggregations- und eine Kompositionsbeziehung.

Aufgabe 2

Implementiere die Aggregationsbeziehung zwischen den Klassen `Schulklasse` und `Lehrer` in deinem bestehenden Projekt `OOP` nach der folgenden Anleitung.

a) Ergänze in der Klasse `Schulklasse` das Attribut `klassenlehrer` vom Datentyp `Lehrer`.
b) Ergänze im Konstruktor der Klasse `Schulklasse` den Parameter `Lehrer kl` sodass der entsprechende Attributwert `klassenlehrer` gesetzt werden kann.
c) Vergewissere dich davon, dass im Projektfenster bei BlueJ nun ebenfalls eine Beziehungsverbindung zwischen den Klassen `Schulklasse` und `Lehrer` dargestellt wird.
d) Schreibe eine Get- und eine Set-Methode für das Attribut `klassenlehrer` in der Klasse `Schulklasse`.
e) Teste deine Veränderungen, indem du zunächst per Rechtsklick ein Objekt der Klasse `Lehrer` und danach ein Objekt der Klasse `Schulklasse` erzeugst. Der Konstruktor der Klasse `Schulklasse` sollte dich nun dazu auffordern, ein bereits bestehendes Objekt der Klasse `Lehrer` als Klassenlehrer zu übergeben.