Beziehungen zwischen Klassen

Damit ein Objekt zu einem anderen Objekt eine Beziehung eingehen kann, muss natürlich auch vorher irgendwo festgelegt werden, dass dies diesem Objekt überhaupt erlaubt ist. Diese Festlegungen werden auf Klassenebene vorgenommen. Die wichtigste Unterscheidung zwischen den Beziehungen auf Objektebene und denen auf Klassenebene ist, dass viele Beziehungen auf Klassenebene angeben, dass die konkreten Objekte des durch diese Klasse definierten Datentyps prinzipiell in der Lage sind, Beziehungen zu Objekten einzugehen, die von dem durch die Klasse definierten Datentyp sind, mit der die erste Klasse eine Beziehung unterhält. *Dies heißt nicht, dass die konkreten Objekte solch eine Beziehung eingehen müssen.* Auf Klassenebene werden die prinzipiellen Eigenschaften und Verhaltensweisen definiert. Zu diesen prinzipiellen Eigenschaften gehören die Beziehungen.

Einige Beziehungen auf Klassenebene finden gar keine Entsprechung auf Objektebene. Ein Beispiel hierfür wäre die IST-Beziehung, die im Kurs aber erst später vertiefende Beachtung finden wird.

Jede verschiedene Art der Beziehung bekommt einen eigenen „Pfeiltyp“ spendiert. Zunächst wird die Darstellung der einzelnen Beziehungstypen vorgestellt, im Anschluss wird auf die Bedeutung eingegangen.

Zur Darstellung

Ein Diagramm, in dem Klassen sowie ihre Beziehungen zueinander dargestellt sind, heißt **Klassendiagramm**. Das Klassendiagramm ist der wichtigste Diagrammtyp, da es schon sehr konkret ist, wenn man von einer Problembeschreibung ausgeht, aber aus Sicht der konkreten Implementierung hinreichend weit entfernt ist, um den Überblick zu behalten und Strukturen zu visualisieren und zu erkennen. Damit sind Klassen das wichtigste Strukturierungselement objektorientierter Programmiersprachen.

Im Unterricht werden schwerpunktmäßig drei Beziehungstypen verwendet. Die HAT- und die KENNT-Beziehung sind vom Namen her schon bekannt und werden hier weiter konkretisiert. Die IST-Beziehung wird hier zwar der Vollständigkeit halber aufgeführt, wird aber erst zu einem späteren Zeitpunkt interessant und näher thematisiert.

Beziehung	Fachbegriff	Darstellung
KENNT	Assoziation	
HAT	Aggregation bzw. Komposition	
IST	Vererbung	

Zur Bedeutung

- Die Assoziation zwischen einer Klasse A und einer Klasse B sagt aus, dass Objekte vom Typ A mit Objekten von vom Typ B arbeiten dürfen. Dieses Arbeiten beinhaltet z. B. den Aufruf von Methoden, die das Objekt vom Typ B nach außen bereitstellt. Damit ein Objekt im Laufe eines Programms ein anderes kennt, muss das entsprechende Objekt vom Typ B dem Objekt vom Typ A bekannt gemacht werden. Es muss also der Wert des Beziehungsattributs, das „kenntB“ heißen könnte, gesetzt werden. Und zwar wird eine Referenz auf das Objekt vom Typ B gesetzt. Bei der Implementierung muss vorher die Benutzung der Klasse A in der Klasse B angegeben werden (Näheres dazu während der Implementierungsphasen).
- Die Aggregation zwischen einer Klasse A und einer Klasse B sagt aus, dass Objekte vom Typ A Objekte vom Typ B „besitzen“ können. Soll heißen, dass sie i. d. R. für die komplette Verwaltung der ihnen angehörenden Objekte verantwortlich sind. Nimmt man das Spiel Tennis, gibt es zwei Spieler, die jeweils einen Schläger haben. Während des Spiels wird ein Spieler niemandem erlauben, ihm den Schläger wegzunehmen. Der Spieler entscheidet auch darüber, wann er den Schläger abgibt.

Kardinalitäten

Ein Bestandteil von Klassendiagrammen sind die **Kardinalitäten** (Vielfachheiten). So kann man beispielsweise ausdrücken, dass ein Tennisspieler genau einen Schläger hat oder aber beliebig viele andere Spieler kennt.