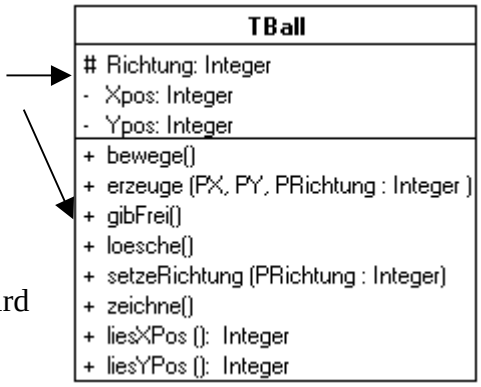
UML-Diagramme

Klassendiagramme:



Ein Klassendiagramm besteht aus dem Namen der Klasse sowie einem Bereich für die Attribute (Zustandsvariablen) und einem Bereich für die angebotenen Dienste (Methoden).
Falls statt einer Klasse ein Objekt dargestellt werden soll, wird der Klassenname unterstrichen.

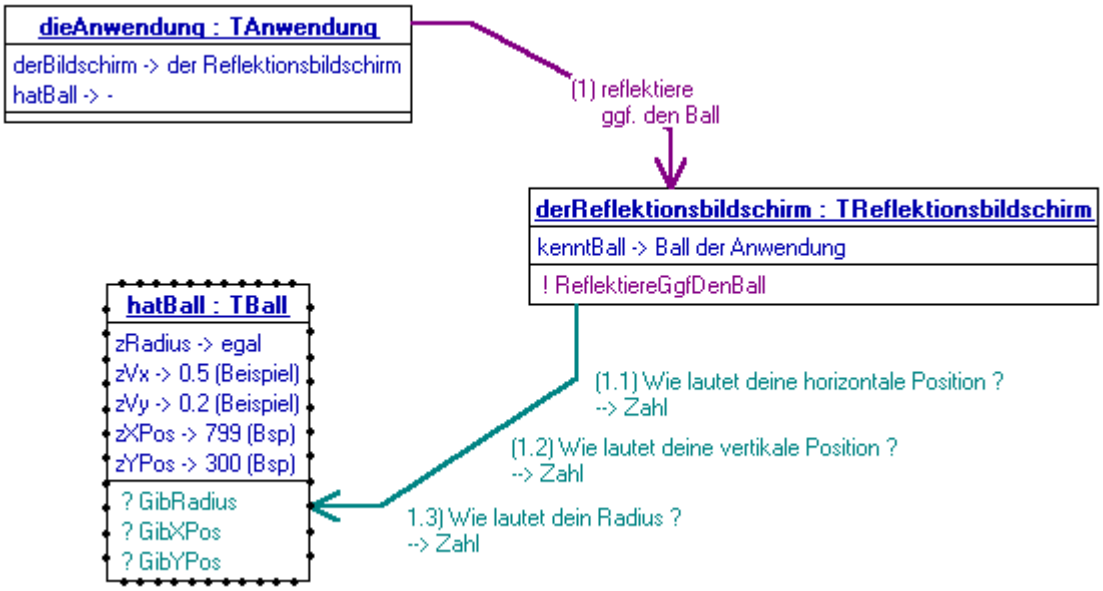
Der Sichtbarkeitsbereich für Attribute und Dienste wird durch "+" für **public** und "-" für **private** und # für **protected** gekennzeichnet. **Private** bedeutet, dass nur diese Klasse auf die Eigenschaft zugreifen kann. **Protected** bedeutet, dass Nachfolgeklassen (und deren Nachfolgeklassen usw.) auf die entsprechende Eigenschaft zugreifen können. Andere Klassen können auf dieses Attribut nicht zugreifen. **Public** bedeutet, dass alle Klassen auf diese Eigenschaft zugreifen können. Ein Zugriff ist natürlich nur möglich, wenn überhaupt Zugriff auf ein Objekt der entsprechenden Klasse möglich ist. Unter Delphi gibt es noch das Zugriffsrecht "++" für **published**, das dem Zugriffsrecht **public** entspricht und zusätzlich die Anzeige im Objektinspektor ermöglicht.

Attribute sollten immer **private** oder **protected** sein, auf sie sollte lesend und schreibend nur durch spezielle Dienste (Methoden) zugegriffen werden können.

Die Dienste/Methoden werden nach Aufträgen (realisiert durch Prozeduren) und Anfragen (realisiert durch Funktionen) unterschieden. Im Klassendiagramm sind Anfragen daran zu erkennen, dass der Parameterliste ein Resultattyp folgt. Aufträgen kann ein Rufzeichen, Anfragen ein Fragezeichen vorangestellt werden.
Dienste sind in der Regel öffentlich, also **public**. In der Entwurfsphase kann es als Vorbereitung der Programmierung jedoch durchaus sinnvoll sein, private Dienste für Objekte einer Klasse vorzusehen. Verwendet man Klassendiagramme zur Dokumentation, werden private Dienste häufig nicht aufgeführt.

Botschafts-Diagramme (Kollaborationsdiagramme)

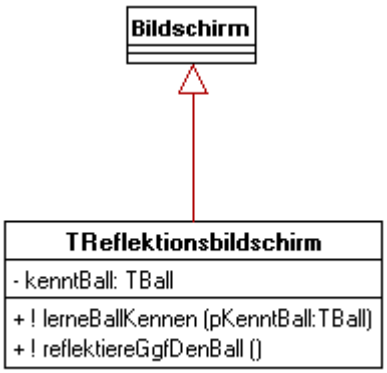
Will ein Objekt den Dienst eines anderen Objektes nutzen, muss es ihm zur Auftragserteilung **(unten violett)** oder zum Stellen einer Anfrage **(unten grün)** eine Botschaft senden. Das Senden einer Botschaft wird durch einen Pfeil vom Auftraggeber (Client) zum Auftragnehmer (Server) dargestellt. Werden Daten versandt, könne diese mit Richtungsangabe und Typ am Botschaftspfeil notiert werden. Parameter sind Daten, die vom Auftraggeber zum Auftragnehmer fließen. Die Antworten auf eine Anfrage fließen vom Auftragnehmer zum Auftraggeber.



Diese Diagramme sollten innerhalb von ***UMLed*** immer als Unterdiagramme eines Hauptdiagramms realisiert werden.

Beziehungsdiagramme:

IST-Beziehung (Verfeinerung bzw. umgekehrt Verallgemeinerung, Vererbung)



Durch einen Pfeil auf die Oberklasse wird die Ist-Beziehung dargestellt.

Gemeint ist im nebenstehenden Diagram, dass die menge der Reflektionsbildschirme in der menge der Bildschirme enthalten ist.

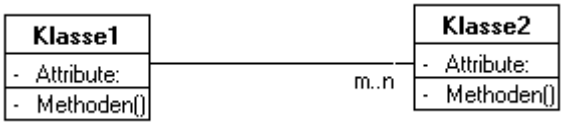
In der Unterklasse werden nur die zusätzlichen Attribute und Methoden aufgeführt.

Die Kennt-Beziehung (Verbindung, Assoziation)

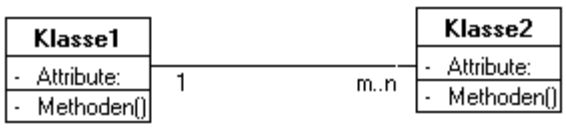
Will ein Objekt einen Dienst eines anderen Objektes in Anspruch nehmen, muss es dieses Objekt kennen.

Die Assoziation ist also eine Beziehung zwischen Objekten. Sie wird jedoch normalerweise mit Hilfe der Klassen dargestellt.

Die mögliche Anzahl der gekannten Objekte wird durch Zahlen angegeben. Ist die Anzahl nicht bekannt, verwendet man Buchstaben. So bedeutet die nebenstehende Angabe "m..n" beispielsweise, dass ein Objekt der Klasse1 mindestens **m** und höchstens **n** Objekte der Klasse2 kennt. Objekte der Klasse2 kennen die Objekte der Klasse 1 jedoch nicht !

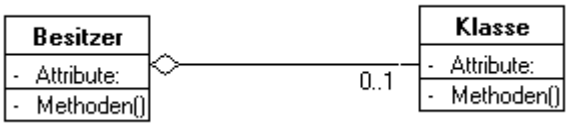


Handelt es sich um eine wechselseitige Kennt-Beziehung, wird dies durch entsprechende Zahlenangaben (Kardinalitäten) an der Verbindung dargestellt. Im nebenstehenden Beispiel kennen die Objekte der Klasse 2 jeweils genau ein Objekt der Klasse 1.



Die Hat-Beziehung (Zerlegung, Aggregation)

ist eigentlich eine spezielle kennt Beziehung. Der Besitzer hat (besitzt) Objekte einer anderen Klasse. Im Gegensatz zur Kennt Beziehung, bei der es sich um die Kontaktaufnahme zweier autonomer Objekte handelt, ist bei der HAT-Beziehung das besitzende Objekt für die Erzeugung, Verwaltung und das Löschen des anderen Objektes zuständig
Auch die HAT-Beziehung wird meistens mit Klassen dargestellt. Die Verbindungslinie zeigt eine (eigentlich gefüllte) Raute bei der besitzenden Klasse.



Können die Objekte, die der Besitzer hat, auch ohne diesen existieren, (schwache Aggregation), dann wird die Raute nicht ausgefüllt. Bsp: Jedes Auto hat einen Motor. Ein Motor kann aber auch ohne Auto existieren. Da im Unterricht die schwache Aggregation nicht vorkommt, arbeitet UMLed immer mit unausgefüllten Rauten.
Die Anzahl der Objekte, die der Besitzer hat, kann auch hier durch Zahlenangaben dargestellt werden. Ist die Anzahl nicht bekannt, verwendet man wie bei der Assoziation Buchstaben.