Beziehungen zwischen Klassen,

Entwurfs- und Implementationsdiagramme in 

Wie bereits bekannt, existieren in der OOP aus Attributen und Methoden bestehende Klassen, die die Grundlage für Objekte bilden, welche in Programmen erstellt werden können.

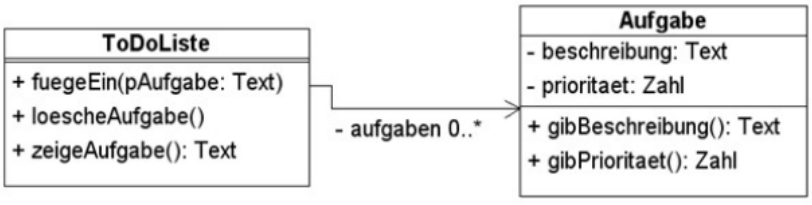
Damit derartige Objekte in gewissen Situationen untereinander Nachrichten austauschen können, müssen **Beziehungen** zwischen ihren Klassen modelliert werden. Derartige Klassenbeziehungen nennt man auch **(gerichtete) Assoziationen**.

Bei einer Assoziation kann man außerdem angeben, wie viele Objekte einer Klasse B in einer solchen Beziehung zu einem Objekt der Klasse A stehen bzw. stehen können – die entsprechende Zahl nennt man **Multiplizität**. Mögliche Multiplizitäten sind:

- 1 genau ein assoziiertes Objekt
- 0..1 kein oder ein assoziiertes Objekt
- 0..* beliebig viele assoziierte Objekte
- 1..* mindestens ein, beliebig viele assoziierte Objekte

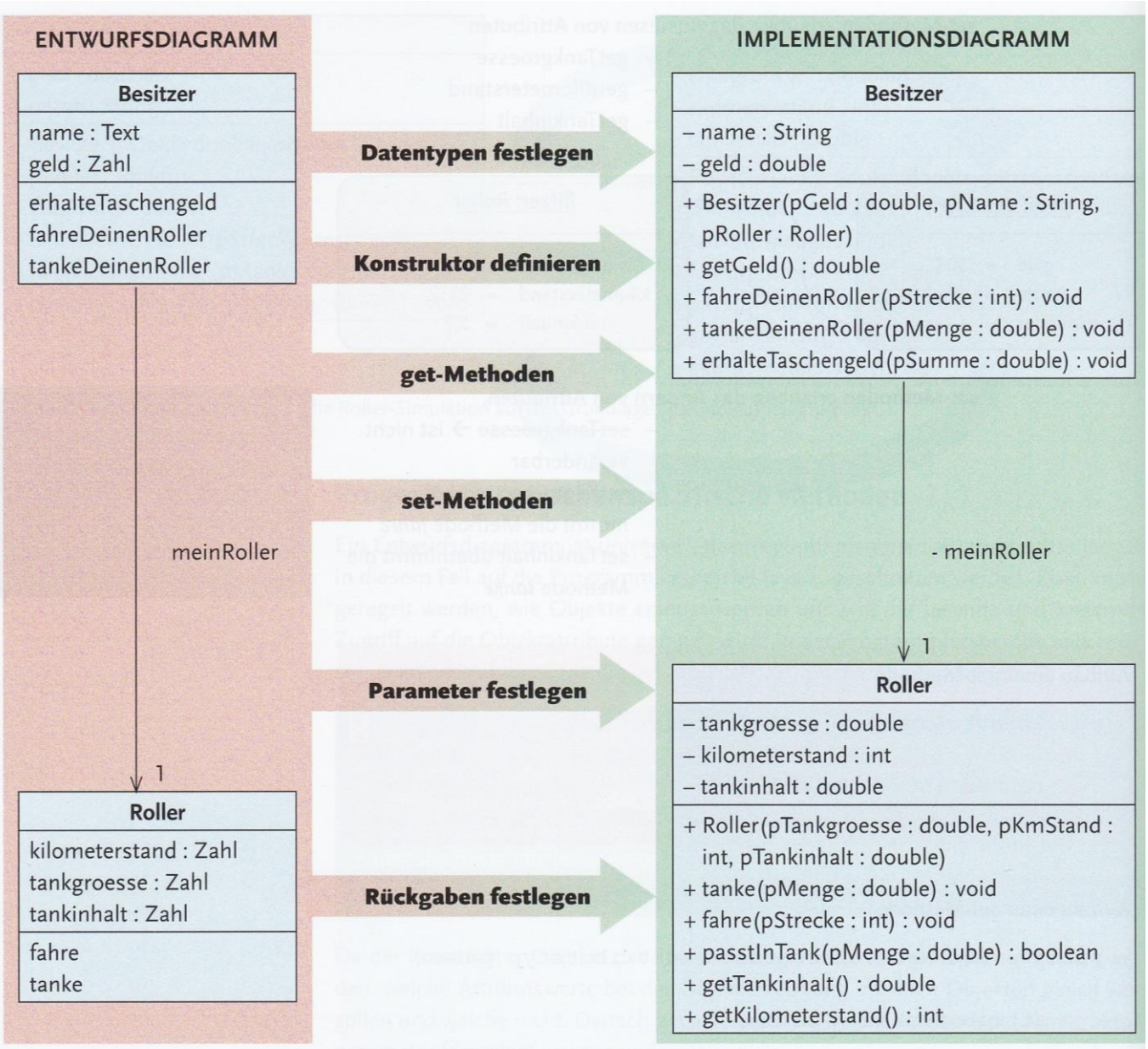
Um an einem Gesamtprojekt beteiligte Klassen und ihre Beziehungen mit Multiplizitäten modellieren zu können, gibt es das sogenannte **Entwurfsdiagramm**, das relativ einfach gehalten und auch von Laien nachvollziehbar ist.
Die Darstellung ist programmiersprachenunabhängig – es werden lediglich die Datentypen „Zahl, Text, Wahrheitswert und Datenansammlung<·>“ unterschieden. Bei der Datenansammlung steht in Klammer der Datentyp oder die Klassenbezeichnung der Elemente, die dort verwaltet werden. Anfragen werden durch den Datentyp des Rückgabewertes gekennzeichnet.
Assoziationen werden durch einen geöffneten Pfeil dargestellt, der die zugehörige Multiplizität neben sich trägt. Bezeichner der Attribute, durch die die Assoziationen realisiert werden, stehen an den Pfeilen.

einfaches Beispiel:



Ein derartiges Entwurfsdiagramm muss logischer Weise vor der Programmierung in ein programmiersprachenspezifisches **Implementationsdiagramm** überführt werden - wie das im Einzelnen funktioniert wird in der Abbildung auf der nächsten Seite beschrieben.

Jenes Implementationsdiagramm stellt letzten Endes die Grundlage für die Umsetzung der beteiligten Klassen in der jeweiligen Programmiersprache dar.



Die Überführung des Entwurfsdiagramms in ein Implementationsdiagramm für Java