UML Klassendiagramme

Eine gute Möglichkeit Klassen und ihre Beziehungen grafisch darzustellen sind UML Klassendiagramme. UML ist eine Abkürzung für "Unified Modeling Language".

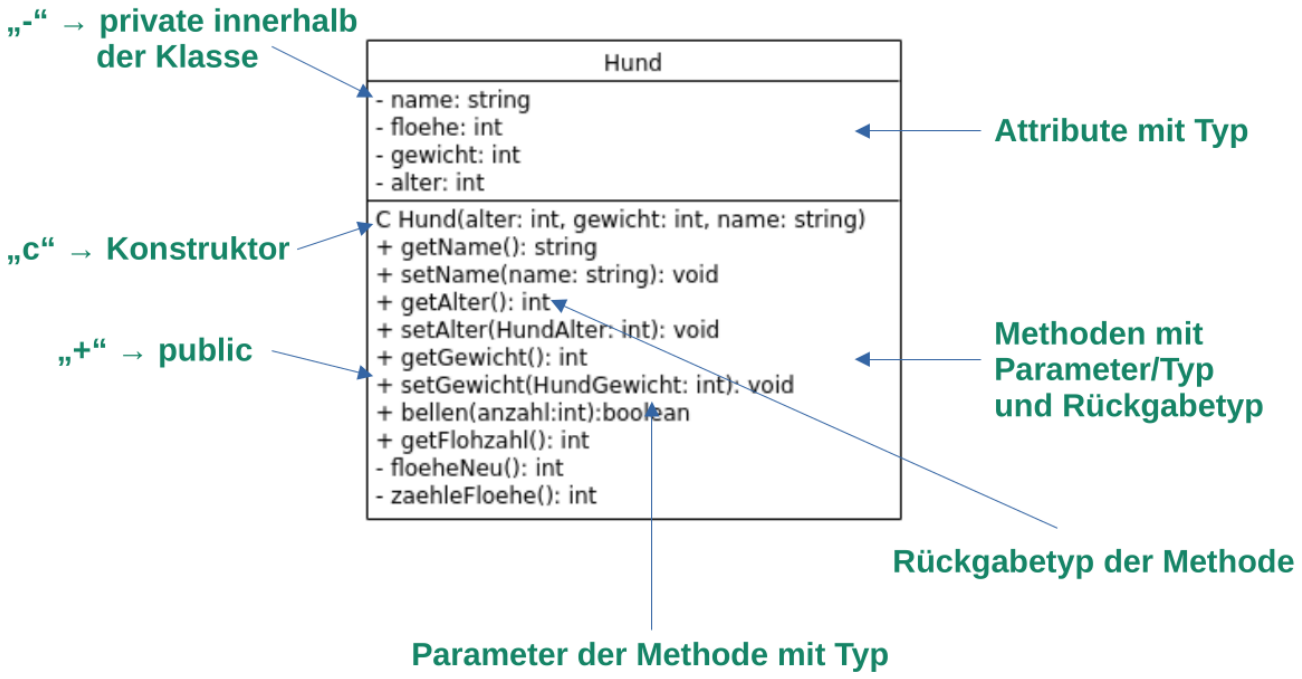
Klassen bestehen aus drei Teilen:

- Klassenname
- Eigenschaften (auch Attribute oder Instanzvariablen)
- Methoden

Aus einer Klasse können wir **Objekte** erzeugen, man spricht von "instanziieren". Ein bestimmtes Objekt ist eine Instanz einer Klasse. Aus der unten abgebildeten Klasse "Hund" kann man also eine Hunde-Objekt mit dem Namen "Higgs", dem Gewicht "20kg", und der Farbe "weiß" erzeugen.

- Die **Attribute** einer Klasse beschreiben den *Zustand* eines Objekts, wie z.B. Name und Gewicht eines Hundes.
- Die **Methoden** einer Klasse definieren das *Verhalten* eines Objekts - die Methoden geben dem Objekt Fähigkeiten, unser Hund kann beispielsweise bellen.

In einem UML Klassendiagramm werden die drei Bestandteile durch waagerechte Striche getrennt. Für das Hunde-Beispiel sieht das Klassendiagramm so aus:



- Im oberen Feld steht der Klassenname (Hund).
- Darunter folgen die Klassen-Attribute. Den Datentyp der Attribute schreibt man durch einen Doppelpunkt getrennt hinter den jeweiligen Attributs-Namen.
- Die Methoden werden mit ihrer Parameterliste und ihrem Rückgabewert im unteren Feld des UML Diagramms angegeben. Der Datentyp des Rückgabewerts einer Methode steht dabei hinter dem Doppelpunkt.

Vorzeichen?

Den Attributen können verschiedene Zeichen vorangestellt sein: -, +, C und (nicht im Beispiel) #.

Der Konstruktor wird mit einem C markiert.

Instanz-Variablen werden für gewöhnlich, um sie gegen Manipulation zu schützen, als "von außen nicht sichtbar", also als "privat" deklariert. Dieses Vorgehen heißt **Kapselung**.

Solche "privaten" Attribute und Methoden werden mit einem Minuszeichen - gekennzeichnet.

Attribute und Methoden, denen ein Pluszeichen + vorangestellt ist, sind als public (öffentlich) deklariert, auf diese kann man also von außen zugreifen.

Darüber hinaus gibt es Attribute und Methoden, die als protected definiert werden können, diese werden im UML Diagramm durch eine vorangestellte Raute # markiert. Als protected deklarierte Attribute und Methoden sind nur innerhalb der Klasse selbst und allen Unterklassen sichtbar..

Getter/Setter

Neben der Methode "bellen" enthält unsere Klasse vor allem "getter-" und "setter-"Methoden für die Attribute. Diese sind nötig, weil die Attribute als "private" deklariert wurden, und somit ein direkter Zugriff auf die Werte der Instanzvariablen nicht möglich ist. Um einem Hund nun einen neuen Namen zu geben, würde man ein Objekt der Klasse Hund erzeugen und könnte der Instanzvariablen dann mit der "setter"-Methode für den Namen einen neuen Wert zuweisen. Ebenso funktioniert das Auslesen der Werte der Instanzvariablen mit den "getter"-Methoden:

```
myDog = new Hund(5,36,"Emil")
myDog.getName() // "Emil"
myDog.setName("Higgs")
myDog.getName() // "Higgs"
myDog.name // FEHLER
myDog.name = "Higgs" // FEHLER
```

Klassenvariablen

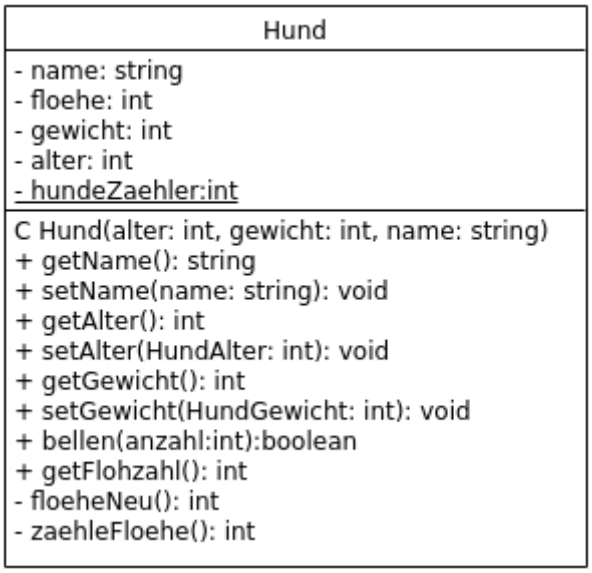
Alle bisherigen Attribute waren Instanzvariablen. Jede Instanz der Hunde-Klasse speichert die Werte ihrer Instanzvariablen in einem eigenen Speicherbereich, zwei Instanzen der Hunde-Klasse können also nicht gegenseitig auf die Werte ihrer Instanzvariablen zugreifen.

Wenn wir nun beispielsweise die Zahl der erzeugten Hunde-Objekte zählen wollen, haben wir ein Problem: Wir können zwar Flöhe auf jedem Hund zählen, haben aber keinen gemeinsamen Zähler für alle Objekte, den jede Instanz lesen und schreiben kann.

Wir benötigen also eine Integer-Variable, auf die alle Hunde-Instanzen zugreifen können. Eine solche

Variable heißt **Klassenvariable**, in Java wird sie mittels des Schlüsselworts `static` definiert.

In eine UML Diagramm werden Klassenvariablen mit Hilfe eines Unterstrichs gekennzeichnet:



Los! Lass uns das Klassendiagramm von oben um eine Klassenvariable, mit der wir die Anzahl der erzeugten Hunde zählen können ergänzen.

From:
https://info-bw.de/ -

Permanent link:
https://info-bw.de/faecher:informatik:oberstufe:java:objektorientierung:uml:start

Last update: **07.10.2023 09:17**