5. Polymorphie

Die Polymorphie ist eine direkte Folge der Spezialisierung von Klassen. Jede abgeleitet Klasse weist die Merkmale der Oberklasse auf und ist somit auch vom Typ der Oberklasse.

Beispiel 6.5:

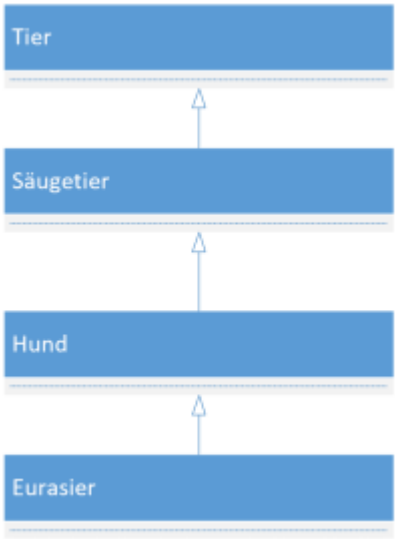

Abb 6.8: Spezialisierung einer Klasse
Für die Klasse `Eurasier` gilt die Aussage:
Eurasier IST EIN Hund IST EIN Säugetier IST EIN Tier.
Somit ist klar, dass ein Objekt vom Typ `Eurasier` immer auch ein Objekt vom Typ `Hund`, vom Typ `Säugetier` und vom Typ `Tier` ist.
Das Objekt Eurasier ist somit **polymorph**, was „vielkörperig“ bedeutet.

Schauen wir uns die Auswirkungen der Polymorphie etwas vertieft an.
Dazu geben wir jeder Klasse aus Beispiel 6.5 eine Fähigkeit mit auf den Weg und visualisieren dann das Objekt der Klasse `Eurasier`.

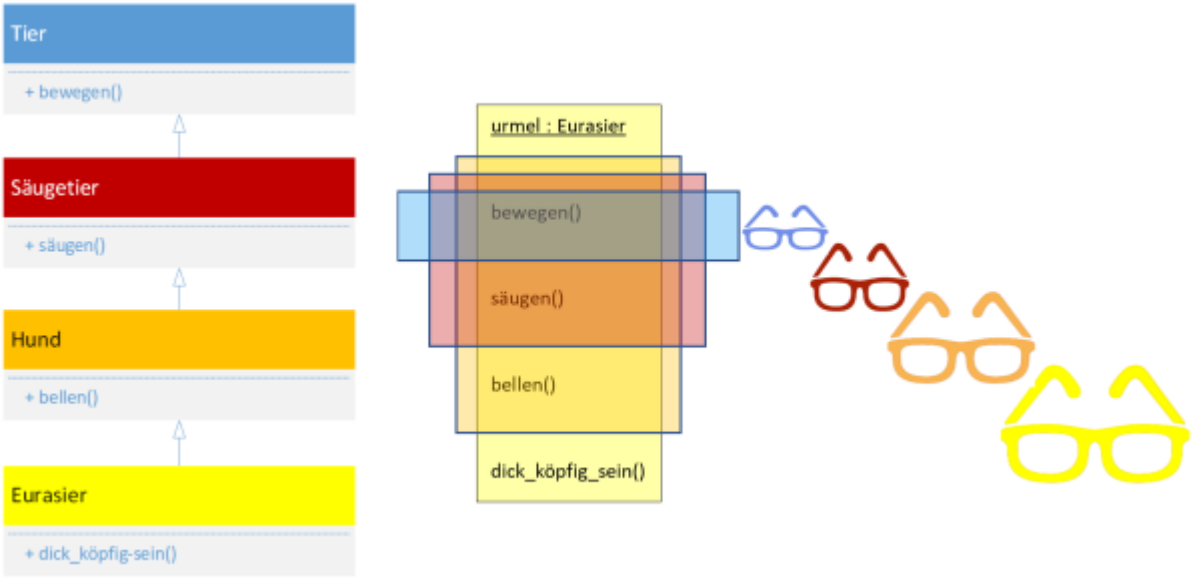

Abb. 6.9: Sichtbarkeit von Methoden in der Vererbungshierarchie

Je nach Sicht - hier durch die farbigen Brillen symbolisiert - sind mehr oder weniger Funktionen (Methoden) verfügbar. Aber was heisst das nun genau?
Wenn das Objekt `urmel` (ein Eurasier) aus der Sicht eines `Eurasier`-Objekts betrachtet wird, sind alle 4 Methoden im Zugriff und können aufgerufen werden. Wird aber z.B. die Sicht `Säugetier` verwendet, dann sind auch nur die beiden Methoden `säugen` und `bewegen` verfügbar.
Aber warum braucht es das?

Auswirkung der Polymorphie

Die Auswirkung der Polymorphie lässt sich am besten an einem Beispiel mit „unseren“ Tieren erläutern.

Beispiel 6.6:
Wir spielen dazu mal Landwirt. Eine Aufgabe ist es, alle Tiere zu bewegen. Unsere Tiergruppe umfasst eine ganz bunte Mischung.

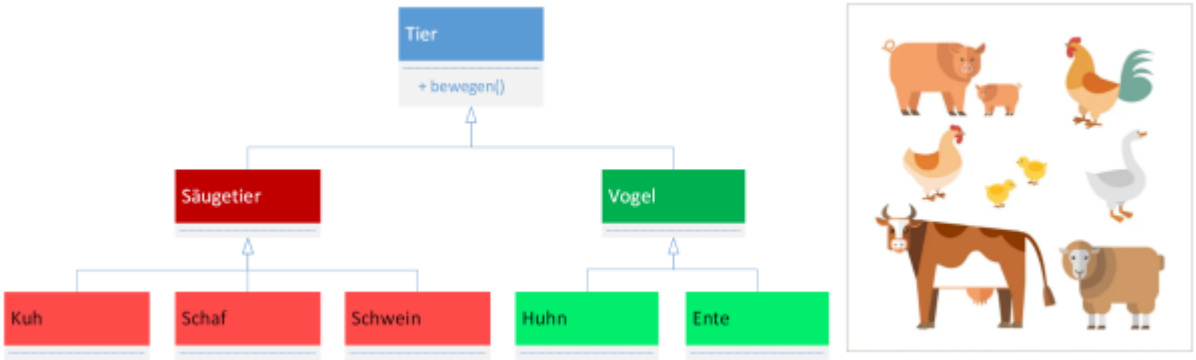

Abb 6.10: Vererbungshierarchie „Tiere“

Wenn wir nun alle Tiere bewegen wollen, dann müsste das ohne Nutzung der Polymorphie in etwa wie folgt geschehen:

```
+-----------------------------------------+
¦ für eine_kuh in alle_kühe               ¦
¦    +-----------------------------------+
¦    ¦ eine_kuh.bewegen()                 ¦
+----+-----------------------------------+
¦ für eine_ente in alle_enten             ¦
¦    +-----------------------------------+
¦    ¦ eine_ente.bewegen()                ¦
+----+-----------------------------------+
¦ ....                                    ¦
```

Das ist natürlich recht umständlich. Und wenn neue Tiere dazu kommen, z.B. Lamas, muss der ganze Code entsprechend ergänzt werden. Hier kommt nun die Polymorphie zum Einsatz. Denn alle unsere Hoftiere erben ja von der Klasse `Tier`. Wenn wir unsere Hoftiere also mit der enstprechenden „Brille“ betrachten, dann sind das eben alles Tiere und nicht mehr Hühner, Kühe oder Schweine. Und der Code reduziert sich entsprechend.

```
+----------------------------------------+
¦ für ein_tier in alle_tiere             ¦
¦    +-----------------------------------+
¦    ¦ ein_tier.bewegen()                ¦
+----+-----------------------------------+
```

Und das wärs dann auch schon. Kommen jetzt neue Tierarten dazu, ändert das an diesem Stück Code nichts. Toll oder?
