# Polymorphismus genauer

## Variablenpolymorphismus

**Polymorphismus** haben wir schon für Variablen kennengelernt: Eine Variable eines Supertyps kann auch Werte aller Subtypen halten - die Variable ist *polymorph*.

---

**(A1)**

Welche Typen können Werte haben, die in den folgenden Variablen gespeichert werden?

```
Fahrzeug f;
Roller r;
vierRaedrig v;
```

## Methodenpolymorphismus

### Problemstellung

Die Vererbungshierarchie unseres soziales Netzwerk mit Vererbung sieht gerade so aus:

Man sieht, dass die Methode zum Anzeigen eines Beitrags in der Klasse `Beitrag` definiert ist und an die Klassen `TextBeitrag` und `PhotoBeitrag` vererbt wird. Diese Methode weiß nichts über besondere Eigenschaften der Subklassen - Vererbung ist eine Einbahnstrasse. Das führt zum Problem, dass die Ausgabe aller Beiträge etwas so aussehen:

dabei werden die Besonderheiten der Beitragsarten nicht berücksichtigt - der `photoBeitrag` hat keine Bilddatei und keine Caption. Eigentlich sollte das nämlich so aussehen:

Die spontane Lösungsidee verschiebt die `display`-Methode in die Subklassen, so dass jede Subklasse eine eigene `display`-Methode hat, welche dann natürlich entsprechend der spezifischen

Eigenschaften implementiert sein könnte:

**Dieser Versuch ist zum Scheitern verurteilt:**

- Zugriff auf die privaten geeerbten Attribute aus Beitrag ist nicht möglich.
- Die Klasse `Newsfeed` benötigt eine `display`-Methode in Beitrag.

### Lösungsansatz: Überschreiben

- Superklasse und Subklasse definieren Methoden mit gleicher Signatur.
- Jede der Methoden hat Zugriff auf alle Attribute (Felder) ihrer jeweiligen Klasse.
- Der Check des statischen Typs der Superklasse ist erfüllt.
- Die Methode der Subklasse wird erst zur Laufzeit aufgerufen und überschreibt dabei die Version der Superklasse.

Es gibt also `display`-Methoden in der Superklasse und in den Subklassen (wenn nötig):

Fragen:

- Welche Rolle spielt die Version der Superklasse?
- Welche der `display`-Methoden wird denn zur Laufzeit tatsächlich aufgerufen?

## Methodenauswahl zur Laufzeit

### Ohne Vererbung

Keine Vererbung, kein Polymorphismus, kein Problem → die offensichtliche Methode wird ausgeführt:

```
v1.display()
```

### Vererbung, kein Überschreiben

```
v1.display()
```

Bei der Suche nach der Methode wird die Vererbungshierarchie von unten nach oben durchlaufen (beginnend beim dynamischen Typ), bis zum Treffer – diese Methode wird ausgeführt.

Die Methode der Superklasse wird also solange von den Versionen der Subklasse(n) „verdeckt“, solange es in den Subklassen Methoden mit gleicher Signatur gibt.

### Methodenwahl: Zusammenfassung

1. Auf die Variable wird zugegriffen
2. Das in der Variable gespeicherte Objekt wird aufgesucht
3. Die Klasse des Objekts wird bestimmt
4. Implementiert die Klasse die gesuchte Methode?
5. Wenn nicht, wird die Superklasse untersucht.
6. Schritte 4) und 5) werden solange wiederholt, bis die Methode gefunden wurde oder die Klassenhierarchie bis ganz nach oben durchlaufen wurde.
7. Die überschriebenen Methoden verdecken dabei stets die von der Superklasse geerbten Versionen.

### Vererbung und Überschreiben:

```
v1.display()
```

Bei der Suche nach der auszuführenden Methode wird die Vererbungshierarchie von unten nach oben durchlaufen (beginnend beim dynamischen Typ), **bis der erste Treffer gefunden wird** – diese Methode wird ausgeführt. Im Beispiel also die Methode, die in `photoBeitrag` implementiert ist.

## Material

[n/a: Keine Treffer]

From:
https://wiki.qg-moessingen.de/ - **QG Wiki**

Permanent link:
**https://wiki.qg-moessingen.de/faecher:informatik:oberstufe:modellierung:vererbung:polymorphismus:start?rev=1638216276**

Last update: **29.11.2021 21:04**