Vererbung

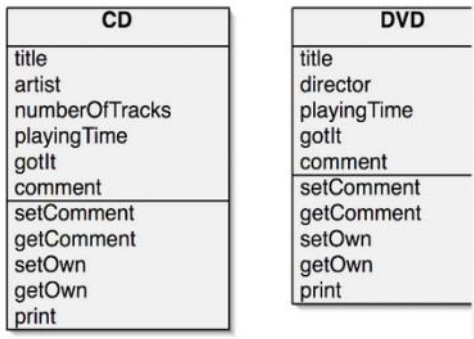
Sonntag, 12. Januar 2020 21:48

Vererbung¶

Vererbung setzt sich mit folgenden Konzepten auseinander:

- Basisklasse / Abgeleitete KLasse
- Einfachvererbung
- Überschreiben von Methoden

Beim Erstellen von Klassen wird man häufig Abhängigkeiten/Gemeinsamkeiten zwischen Klassen erkennen. Irgendwie gehören sie zusammen aber doch auch wieder nicht. Die OOA löst diese Problematik durch das Konzept der **Vererbung**. Es bedeutet, dass Klassen Attribute und Fähigkeiten anderer Klassen übernehmen und gleichzeitig erweitern können. Häufig wird man auch erst im Laufe der Analyse eine bestehende Klasse in mehrere Klassen unterteilen wollen.



Aufgabe: Wie könnte man die oben dargestellten Klassen besser organisieren ?

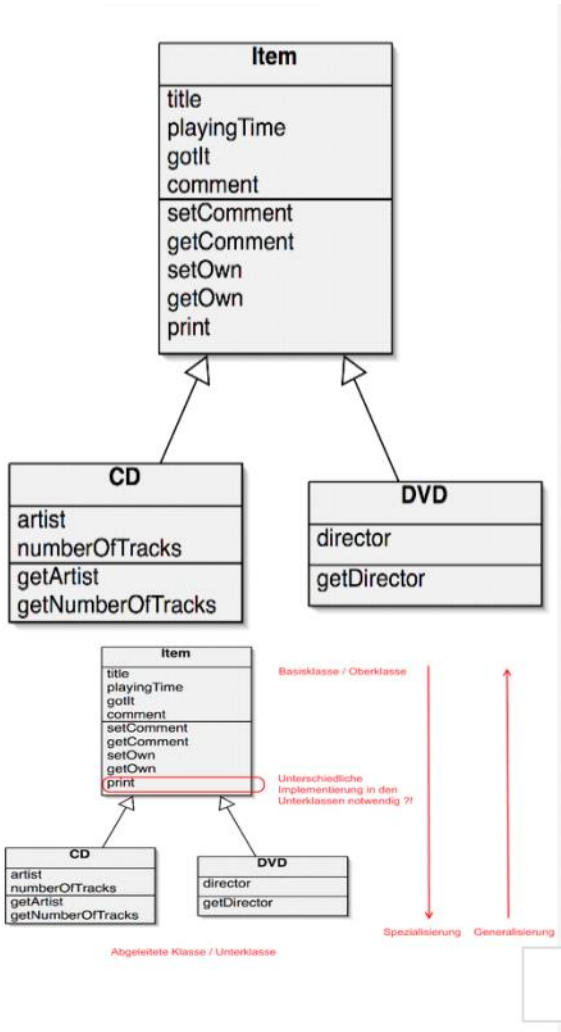
Die Klasse CD und DVD verfügen über gemeinsame Attribute und Methoden. Sie werden in einer Oberklasse **Item** ausgelagert. Zusätzliche Informationen und Fähigkeiten bleiben in den **spezialisierten Unterklassen**.

Im Zusammenhang mit Vererbung existieren einige Begriffe:

- **Basisklasse**: Die Klasse, die alle Gemeinsamkeiten aufnimmt (in unserem Fall Item)
- **Abgeleitete Klasse / Unterklasse**: Die Klassen, die spezielle Attribute und Methoden aufnehmen (in unserem Fall CD und DVD)
- **Generalisierung**: Untersuchung des Vererbungsaspektes von den speziellen Klassen zu den allgemeinen Klassen
- **Spezialisierung**: Untersuchung des Vererbungsaspektes von den allgemeinen Klassen zu den speziellen Klassen

Es sind folgende Dinge zu beachten:

- Informationen werden in Ober- und Unterklassen gehalten. Damit wird Redundanz vermieden.
- Die Unterklassen brauchen aber Zugriffsmöglichkeiten auf Attribute und Methoden der Oberklasse
- Die Unterklassen **halten** über den Vererbungsmechanismus **den Vertrag der Oberklasse**. **Konkrete Instanzen der Unterklassen können überall dort benutzt werden, wo Instanzen der Oberklasse erwartet werden.** In einem solchen Falle ist dann aber lediglich der Zugriff auf die Schnittstelle der Oberklasse möglich. Die Unterklassen sind somit vom gleichen Typ wie die Oberklasse. Eine DVD ist damit auch ein Item.
- Die Oberklasse hat im Allgemeinen keine Kenntnis darüber, dass es abgeleitete Unterklassen gibt.



Vererbung in PHP

Bei der Implementierung des obigen Beispiels ist zu beachten, dass

- insgesamt 3 Klassen erzeugt werden müssen
- die Unterklassen Zugriff auf Attribute und Methoden der Oberklasse haben müssen und dementsprechend Zugriffsmodifizier (internal/protected) gewählt werden müssen
- gewisse Methoden in den Unterklassen überschrieben werden müssen.

Implementierung der Oberklasse

```php
<?php
class items{
   private $title;
   private $playingTime;
   protected $comment;
   private $gotIt;

   public function __construct(string $t, int $p, string $c, bool $g)
   {
      $this->title = $t;
      $this->playingTime = $p;
      $this->comment = $c;
      $this->gotIt = $g;
      }

   public function getTitle():string
   {
      return $this->title;
   }

   public function setComment(string $c)
   {
      $this->comment = $c;
   }

   public function print():string
   {
      return $this->getTitle() . ", " . $this->comment;
   }
}
```

Hide

Hide

Implementierung der Unterklasse CD

Eine erste, naive Implementierung sieht wie folgt aus:

```php
<?php
class cd extends items
{
   private $artist;
   private $numberOfTracks;

   public function __construct(string $a, int $n)
   {
      $this->artist = $a;
      $this->numberOfTracks = $n;
   }

   public function setComment2(string $c)
   {
      $this->comment = $c;
   }
```

Hide

Dies führt aber zu folgenden Problemen:

```
3   abstract class items{
4
5       private $title;
6       private $playingTime;
7       private $comment;
8       private $gotIt;
9
10
```

```
11
12      public function __construct(string $t, int $p, string $c, bool $g)
```

Ausnahme aufgetreten.
ArgumentCountError: Too few arguments to function items::__construct(), 0 passe
exactly 4 expected

Hide

Konstruktorproblematik

Beim Erzeugen einer Unterklasse wird es aufgrund der Vererbung auch notwendig, eine Instanz der Oberklasse zu erzeugen. Da diese jedoch in unserer Implementierung keinen parameterlosen Konstruktor zulässt, erzeugt PHP einen Fehler. Hätten wir einen parameterlosen Konstruktor, hätte der Compiler keinen Fehler gemeldet.

Daran ist prinzipiell auch nichts Verwerfliches, da wir in der Oberklasse über den Konstruktor das Setzen von Zuständen implementiert haben, die für unser Verhalten wichtg sind (Titel und Spielzeit). Auch eine CD sollte über diese Informationen verfügen. Der Konstruktor von CD muss deshalb diese Informationen an den Konstruktor der Oberklasse Item weiterreichen.

Dies erfolgt durch das Weiterleiten im Konstruktoraufruf:

```php
<?php
class cd extends items
{
   private $artist;
   private $numberOfTracks;

   public function __construct(string $a, int $n, string $t, int $p, bool $g, string $c )
   {
      $this->artist = $a;
      $this->numberOfTracks = $n;
      parent::__construct($t, $p, $c, $g);
   }
```

parent

spricht Oberklasse von

seiten der Unterklasse an

Hide

Zugriff auf Attribute und Methoden der Oberklasse

Durch die Vererbung wird zunächst der Zugriff auf öffentliche Methoden und Attribute gewährt. Private Attribute und Methoden sind durch das Prinzip der Kapselung weiter geschützt.

Will man den Unterklassen Zugriff auf diese Elemente gewähren, so müssen die Zugrifsmodifizierer der Oberklasse geändert werden.

In PHP steht dafür folgender Modifizierer zur Verfügung: **protected**

In unserem Beispiel könnten die bisher als private deklarierten Attribute der Oberklasse mit dem Schlüsselwort **protected** ersetzt werden.

protected

Erlaubt Zugriff von der Unterklasse

auf die Oberklasse

```php
class items {

   protected $title;
   protected $playingTime;
   protected $comment;
   protected $gotIt;
```

Überschreiben von Methoden der Oberklasse

Die Methode print() der Klasse Item gibt zur Zeit den Titel und den Kommentar aus.

```php
<?php>
class items

public function print():string
{
   return $this->getTitle() . ", " . $this->comment;
}
```

Dieses Verhalten kann eventuell nicht das sein, was Instanzen der Klasse CD möchten. Doch wie kann die Ausgabe für Objekte der Klasse CD geändert werden, ohne eine andere Methode benutzen zu müssen. Um es allgemeiner auszudrücken:

Wie kann eine Unterklasse ein anderes Verhalten als das der Oberklasse implementieren?

Die Lösung besteht im Neudefinieren der Methode in den Unterklassen.

```php
<?php
class cd

   public function print():string
   {
      return parent::print() . ", " . $this->artist . ", " .
      $this->numberOfTracks;
   }
```

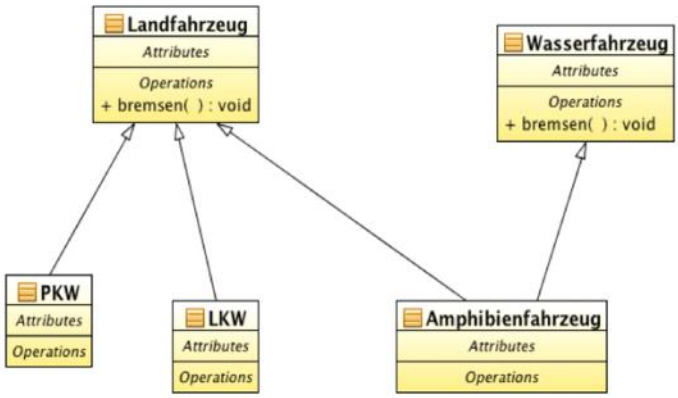
parent

Zugriff auf Methode/Attribute der Oberklasse

Hinweis: Andere Programmiersprachen benutzen dafür Konstrukte wie virtual und override

Mehrfachvererbung

Mehrfachvererbung erweitert die grundsätzliche Vererbung um die Fähigkeit, von mehreren Oberklassen gleichzeitig ableiten zu können. Damit ist aber auch bereits die Problematik vorgegeben.



Welche Methode bremsen() sollen Amphibienfahrzeuge ausführen ??

Aufgrund dieser Nebeneffekte haben sich die meisten Programmiersprachen vom Konzept der Mehrfachvererbung abgewendet, ohne jedoch die prinzipiellen Vorteile der Mehrfachvererbung mit Hilfe von Interfaces nicht zu implementieren.

Aufgaben zu Vererbung

Sonntag, 12. Januar 2020 21:59