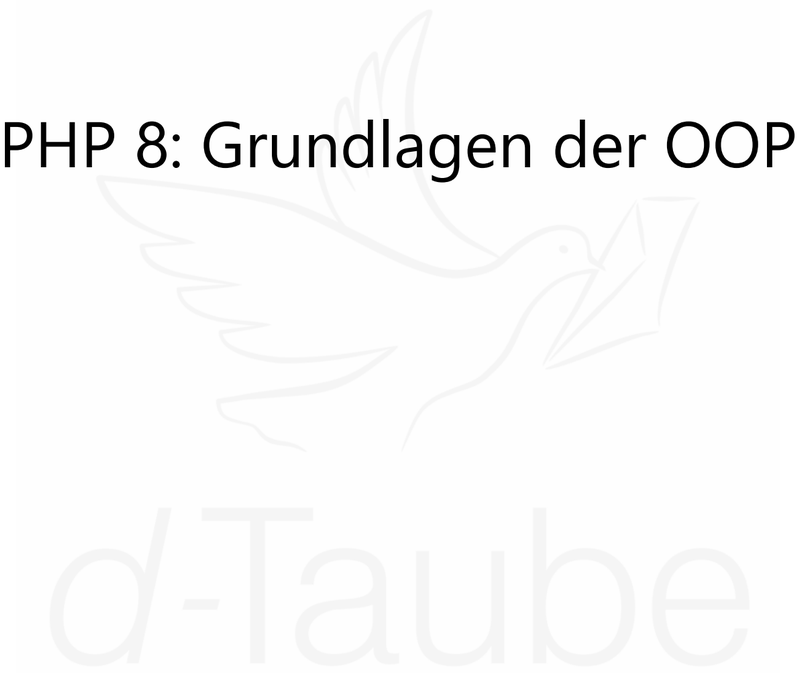
Grundlagen der PHP-Programmierung

PHP 8: Grundlagen der OOP



22.04.2024 bis 12.07.2024 - Oliver Vogt

Objektorientierte Programmierung mit PHP

Agenda Tag 01

Entwurfsmuster

Geschichte der Objektorientierung in PHP

OOP - Klassen und Objekte – Grundbegriffe:

- Was sind Objekte?
- Objekte in PHP
- Klassen in PHP
- Attribute
- Methoden
- Pseudovariable $this

Lernziele

Teilgebiet Einführung in die OOP

- Kennenlernen was ein Entwurfsmuster ist
- Kennenlernen Grundbegriffe der OOP
- Kennenlernen Objekte und Klassen in PHP
- Kennenlernen Attribute, Methoden und Pseudovariable $this

Kurzeinführung in Muster in der Softwareentwicklung

In der Softwareentwicklung wiederholen sich bestimmte Aufgaben und Probleme regelmäßig. Um das "Rad nicht neu erfinden" zu müssen, verwendet man häufig eine **Mustersprache**.

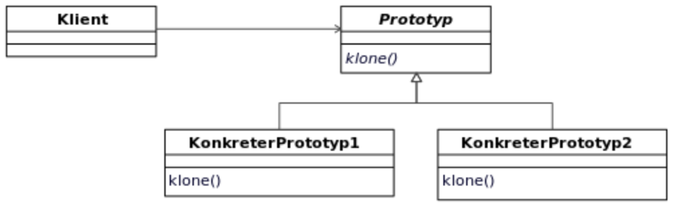
Mithilfe einer **Mustersprache**, die sich durch Referenzen auf andere Muster ergibt, können Entwickler auf einen **Katalog bewährter Lösungen** zurückgreifen.

Diese Lösungen werden durch **definierte Benennungen** (***Termini***) **klar kommuniziert**. Der Architekt Diese Lösungen werden durch definierte Benennungen (Termini) klar kommuniziert. Der Architekt Christopher Alexander prägte den Ansatz für Entwurfsmuster bereits 1964.

In der Softwareentwicklung wurden 1994 im Buch 'Design Patterns' von der sogenannten **Gang of Four** (*GoF*) 23 Entwurfsmuster spezifiziert, gefolgt von **Martin Fowler**'s Werk über Enterprise Application Architecture acht Jahre später. prägte den Ansatz für Entwurfsmuster bereits 1964.

Entwurfsmuster und Architekturmuster

- **Entwurfsmuster** sind bewährte Lösungsschablonen für wiederkehrende Probleme in der Softwarearchitektur und Softwareentwicklung. Sie dienen als wiederverwendbare Vorlagen in spezifischen Kontexten und erleichtern die Entwicklung durch Vorgabe von Strukturen für häufig auftretende Probleme.
- **Architekturmuster** hingegen sind auf einer höheren Ebene angesiedelt und bestimmen die grundlegende Organisation und Interaktion zwischen den Komponenten einer Anwendung.



Das Entwurfsmuster Prototyp mit der Unified Modeling Language (UML)
(c) wikipedia Jarling~commonswiki

Schematischer Aufbau des Pipes und Filter Musters
(c) wikipedia Kako

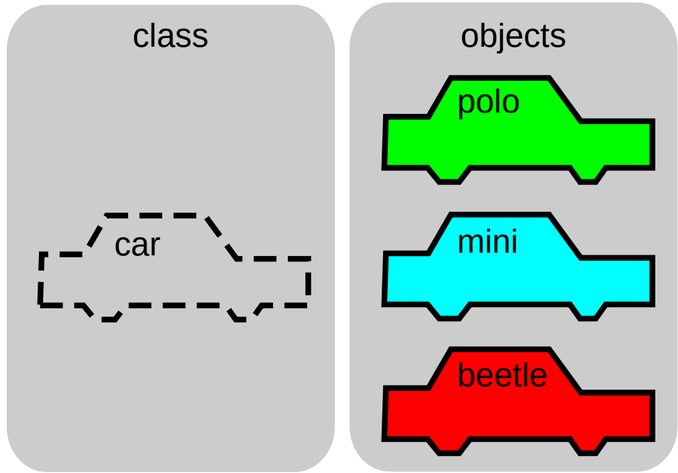
Während Entwurfsmuster sich auf konkrete Teilprobleme fokussieren, geben Architekturmuster den strukturellen Rahmen vor. Eine klare Abgrenzung zwischen beiden ist jedoch nicht immer gegeben und führt oft zu Diskussionen.

Objektorientierte Programmierung (OOP)

Die Objektorientierte Programmierung (OOP) hat in der Welt der Softwareentwicklung eine entscheidende Rolle gespielt, und auch PHP hat im Laufe der Zeit eine bemerkenswerte Entwicklung in Bezug auf OOP durchgemacht.

> Die **objektorientierte Programmierung** (kurz ***OOP***) ist ein auf dem Konzept der Objektorientierung basierendes Programmierparadigma. Die Grundidee besteht darin, die Architektur einer Software an den Grundstrukturen desjenigen Bereichs der Wirklichkeit auszurichten, der die gegebene Anwendung betrifft. Ein Modell dieser Strukturen wird in der Entwurfsphase aufgestellt. Es enthält Informationen über die auftretenden Objekte und deren Abstraktionen, ihre Typen. Die Umsetzung dieser Denkweise erfordert die Einführung verschiedener Konzepte, insbesondere Klassen, Vererbung, Polymorphie und spätes Binden (dynamisches Binden).

wikipedia



Geschichte der Objektorientierung in PHP [1/4]

Die Geschichte der Objektorientierung (OOP) in PHP spiegelt die Entwicklung von PHP selbst wider, von einem einfachen Skriptwerkzeug zu einer vollwertigen Programmiersprache.Sie lässt sich in mehrere Etappen unterteilen.

Frühe Jahre:

PHP wurde 1994 von Rasmus Lerdorf entwickelt und war ursprünglich als einfache Skriptsprache für die Webentwicklung konzipiert. In den Anfangsjahren fehlten PHP jedoch viele Funktionen der OOP, und die Programmierung erfolgte vorwiegend prozedural.

Diese frühe Version der Objektorientierung in PHP war jedoch ziemlich rudimentär und bot nur grundlegende Funktionen wie Klassen und Objekte, Eigenschaften und Methoden.

Geschichte der Objektorientierung in PHP [2/4]

PHP 4:

Mit der Veröffentlichung von PHP 4 im Jahr 2000 wurden erstmals grundlegende OOP-Features eingeführt. **Klassen**, **Vererbung**, und **polymorphe Funktionen** wurden unterstützt, was Entwicklern die Möglichkeit gab, ihre Anwendungen besser zu strukturieren und zu organisieren.

Diese Version führte auch Konzepte wie **Konstruktoren**, **Destruktoren**, und **Zugriffsmodifikatoren** (`public`, `private`, `protected`) ein. Allerdings gab es immer noch bedeutende Einschränkungen, wie zum Beispiel die fehlende Unterstützung für ***Ausnahmen*** (*exceptions*) und ***Interfaces***.

Dennoch war die OOP-Unterstützung in PHP 4 im Vergleich zu anderen Sprachen wie Java oder C++ eher rudimentär.

Geschichte der Objektorientierung in PHP [3/4]

PHP 5:

Der Wendepunkt in der Objektorientierung von PHP kam mit der Veröffentlichung von PHP 5 im Jahr 2004. Mit dieser Version wurde der ***Zend Engine 2*** eingeführt, der die OOP-Fähigkeiten von PHP erheblich verbesserte. Neue **OOP-Features** und die Einführung des **objektorientierten SPL** (*Standard PHP Library*) trugen dazu bei, die Sprache für die Entwicklung komplexer Anwendungen zu stärken.

Dies gelang u. a. durch die Einführung von **Ausnahmen**, **Magic Methods**, **Interfaces**, **Abstrakten Klassen** und der verbesserten Handhabung von **Objektreferenzen**.

PHP 5 etablierte auch das **Konzept der Sichtbarkeit** für Eigenschaften und Methoden und führte den **magischen Methodenmechanismus** ein, der spezielle Methoden wie `__get`, `__set` und `__call` umfasste.

Geschichte der Objektorientierung in PHP [4/4]

PHP 7 und PHP 8:

Mit **PHP 7**, das 2015 veröffentlicht wurde, wurden die OOP-Fähigkeiten weiter optimiert. Die Performance wurde erheblich verbessert, und neue Features wie **Scalar Type Declarations** und **Return Type Declarations** wurden eingeführt. Diese Verbesserungen trugen dazu bei, die Sprache effizienter und sicherer zu machen.

PHP 8, erschienen Ende 2020, brachte Features wie **Union Types**, **Named Arguments**, **Attributes** (Annotations in anderen Sprachen), und **Konstruktor-Property-Promotion**, die die Arbeit mit objektorientiertem Code weiter vereinfachten und flexibler machten.

Die Geschichte der Objektorientierung in PHP zeigt eine kontinuierliche Weiterentwicklung der Sprache, um den wachsenden Anforderungen der Entwicklergemeinschaft gerecht zu werden. Heutzutage ist die OOP ein integraler Bestandteil von PHP-Anwendungen, und viele moderne Frameworks und Bibliotheken nutzen diese Konzepte, um skalierbare und wartbare Codebasen zu erstellen. Entwickler können auf eine reiche OOP-Unterstützung in PHP zurückgreifen, um robuste und moderne Webanwendungen zu entwickeln.

Objektorientierte Programmierung - Einführung [1/3]

> Objektorientierte Programmierung (OOP) ist ein Konzept in der Softwareentwicklung, das auf der Idee basiert, die realen Welt mithilfe von "Objekten" in Code abzubilden. Um dies zu verstehen, ist es hilfreich, zunächst den Begriff "Objekt" im allgemeinen Sprachgebrauch zu betrachten.

Objekt im Allgemeinen

Ein **Objekt** in der alltäglichen Welt kann alles sein, was **physisch existiert** oder ein **klar definiertes Konzept** hat. Zum Beispiel ist ein ***Auto***, ein ***Haus***, ein ***Buch*** oder sogar ein ***Konzept wie ein Termin*** ein Objekt.

Jedes dieser Objekte hat **spezifische Eigenschaften** (wie ***Farbe***, ***Größe***, ***Form***) und **Verhaltensweisen** oder **Funktionen** (***ein Auto kann fahren***, ***ein Buch kann gelesen werden***).

© DALL·E 2024-01-01

Objektorientierte Programmierung - Einführung [2/3]

Übertragung auf die Objektorientierte Programmierung

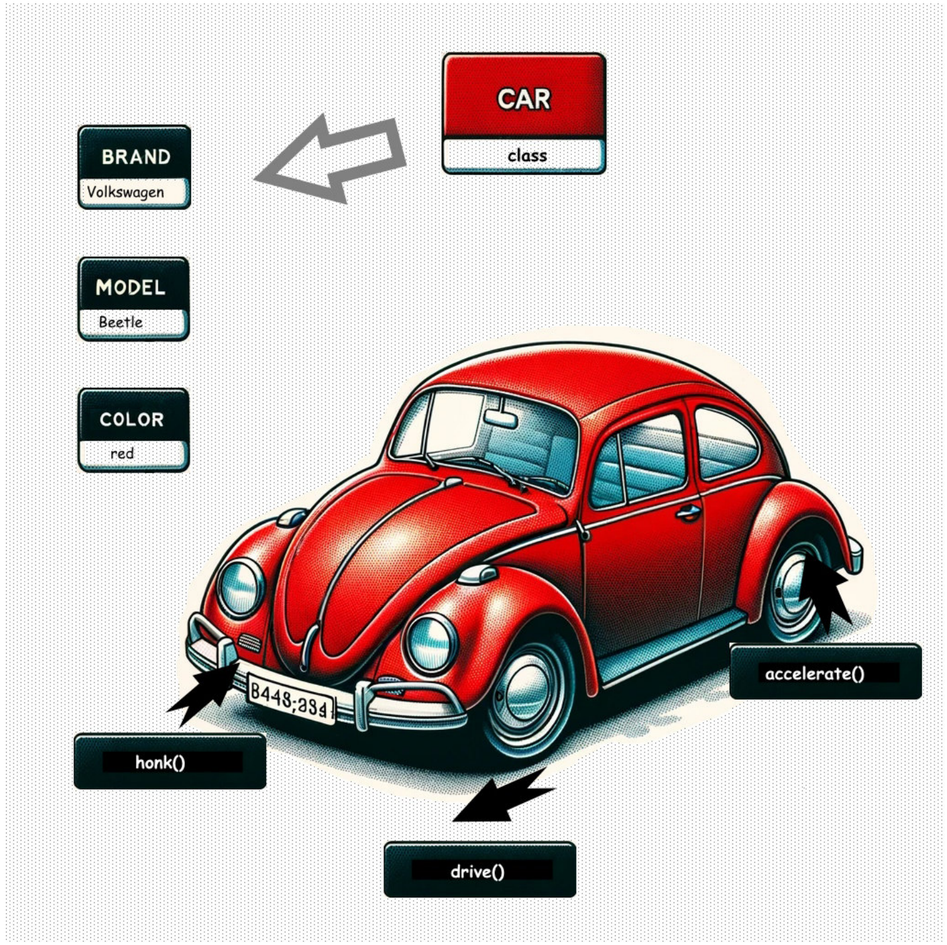
⇒ in der OOP wird dieser Gedanke auf die Programmierung übertragen. Ein **Objekt** in der OOP ist eine **Sammlung von Daten** (***Attributen***) und **Methoden** (***Funktionen***), die auf diese Daten operieren. Diese **Objekte** sind **Instanzen von Klassen**. Eine **Klasse** kann man sich wie eine **Schablone** oder einen **Bauplan** vorstellen, der definiert, welche **Eigenschaften** (Variablen) und **Verhaltensweisen** (Methoden) die Objekte haben werden.

Beispiel:

Stellen Sie sich vor, Sie programmieren ein Spiel, in dem Autos eine Rolle spielen. Hier könnte eine Klasse `Car` erstellt werden. Diese Klasse definiert, dass jedes Auto-Objekt Eigenschaften wie `Marke`, `Modell` und `Farbe` sowie Methoden wie `fahren`, `beschleunigen` oder `hupe` hat. Wenn im Spiel ein spezifisches Auto erstellt wird, sagen wir ein roter Volkswagen, dann ist das eine Instanz der Klasse `Auto` mit den spezifischen Attributen (`rote` Farbe, Marke `Volkswagen`).

Objektorientierte Programmierung - Einführung

[3/3]



© OV + DALL·E 2024-01-01

Klassendiagramm:

Car
+String brand +String model +String color
+drive() +accelerate() +honk()

⇒

Objektdiagramm:

CarInstance
String brand = "Volkswagen" String model = "Beetle" String color = "red"

Objekte in PHP

> PHP hat ein vollständiges Objektmodell.

Objekte in PHP:

- Objekte sind ein weiterer Datentyp, ähnlich wie Arrays.
- In PHP können nicht nur greifbare Gegenstände, sondern auch Lebewesen wie Menschen oder Tiere als Objekte dargestellt werden.

> PHP behandelt Objekte auf die gleiche Art wie Referenzen oder Handles, das bedeutet, dass jede Variable eine Objektreferenz statt einer Kopie des gesamten Objekts enthält.

> *Anmerkung:* In PHP enthält eine Objektvariable nicht mehr das Objekt als Wert, sondern nur einen Objektbezeichner, der bei Objektzugriffen die Identifizierung des eigentlichen Objektes ermöglicht. Wenn ein Objekt als Argument oder Rückgabewert übergeben oder einer anderen Variable zugewiesen wird, so sind die verschiedenen Variablen keine Aliase, sie enthalten vielmehr Kopien des Bezeichners, die auf dasselbe Objekt verweisen.

Klassen und Instanziierung

Klassen

- Klassen sind Prototypen oder Baupläne für Objekte.
- Eigene Klassen werden mit dem Schlüsselwort `class` definiert.

Instanziierung

- Erzeugung von Objekten aus einer Klasse mit dem Schlüsselwort `new`.
- Jedes Objekt erbt Attribute und Methoden seiner Klasse.

```php
<?php
$test = new stdClass();
echo gettype($test); // "object"
```

`stdClass`:

- Eine leere Standardklasse in PHP.
- Enthält weder Attribute noch Methoden.
- Geeignet für einfache Beispiele zur Instanziierung.

Objekte, Attribute und Methoden

Objekteigenschaften

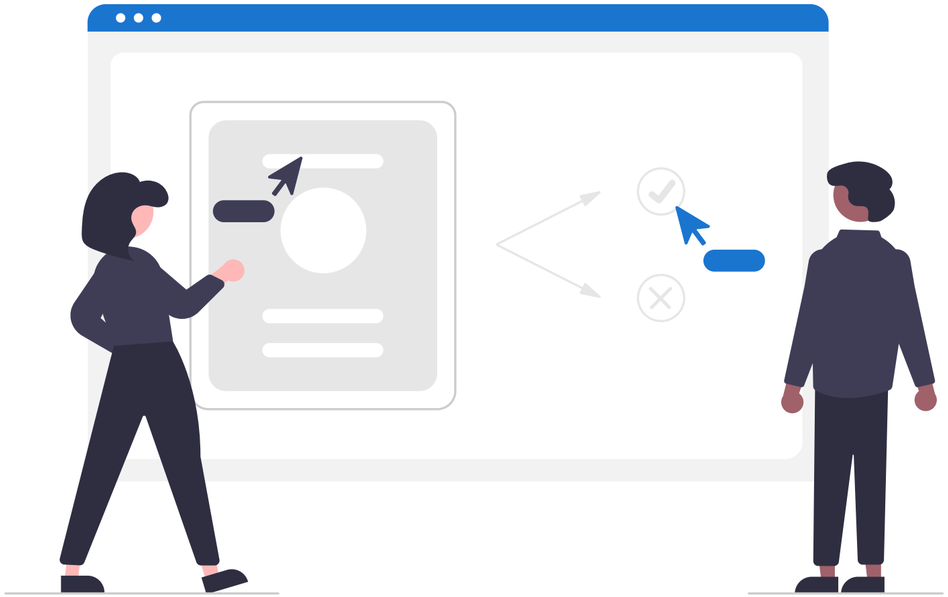
- Objekte können Daten in Form von Attributen speichern.
- Die Werte der Attribute können sich zwischen Objekten der gleichen Klasse unterscheiden.

Methoden

- Funktionen, die an Klassen gebunden sind.
- Können auf Objekten aufgerufen werden, die aus Klassen mit diesen Methoden erstellt wurden.
- Methoden sind in Objekten der gleichen Klasse identisch.

PraxisWorkshop - Objekte in PHP

⇒ Erzeugen einer Instanz aus der Klasse `stdClass`
⇒ Datentyp `object`
⇒ Wertegleichheit und Identität von Objekten



Workshopdateien:
→ oop01.php
→ oop02.php
→ oop03.php
→ oop04.php
→ oop05.php

Zusammenfassung Objekte in PHP

- haben den Datentyp `object`
- werden mit dem Schlüsselwort `new` erzeugt.
- können aus der allgemeinen Objektgruppe `stdClass` instanziiert werden.
- sind gleich sofern sich Objektgruppe und Eigenschaften (bzw. genauer deren Werte) nicht unterscheiden.
- sind identisch sofern sie die gleiche Identität haben.

Objektidentität in PHP

- Objekte haben den Datentyp `object`.
- Objekte sind gleich, wenn sie der gleichen Klasse angehören und ihre Attribute gleiche Werte haben.
- Objekte sind identisch, wenn sie dieselbe Identität haben.

Klassen erzeugen

- Klassen in PHP werden durch das Schlüsselwort `class` gefolgt von dem **Klassenbezeichner** deklariert.
- Danach folgt in geschweiften Klammern (`{}`) der **Klassenkörper**.

Beispiel:

```php
<?php

class MyClass {
 // ...
}
```

> *Hinweis:* bei einer Klassendefinition gibt es im Gegensatz zu Funktionsdefinitionen kein rundes Klammerpäarchen (`()`)
> ⇒ **das würde einen Fehler erzeugen!**

Objekt aus eigener Klasse instanziieren

- Die Instanziierung erfolgt wie bei der Standardklasse mittels des Schlüsselworts `new` und dem Klassennamen.

Beispiel:

```php
<?php

$myObject = new MyClass();
```

> *Hinweis:* bei der Instanziierung eines Objektes aus ein Klasse mittels des Schlüsselworts `new` folgt nach dem Klassenbezeichner ein rundes Klammerpäarchen (`()`).
> Dieses ist optional, sofern der Klassenkonstruktor keine Pflichtparameter definiert hat (dazu später mehr)

Über Klassen und Objekte

Zusammenfassung

- Objekte in PHP beschreiben nicht zwingend einen greifbaren Gegenstand der realen Welt.
- Klasse ist der Fachbegriff für die Gruppe bzw. den Bauplan eines Objektes.
- Eigene Klassen werden mit dem Schlüsselwort `class` definiert.

Konventionen: Klassennamen und Dateiorganisation

- Klassennamen werden immer in der Einzahl geschrieben und es ist üblich, den ersten Buchstaben eines Klassennamens großzuschreiben (**UpperCamelCase-Notation** - *Binnenmajuskel*).
- Es macht Sinn, die Definition jeder einzelnen Klasse in eine eigene Datei auszulagern um eine optimale Wiederverwendbarkeit zu ermöglichen.

Attribute

- Die **Eigenschaften** von Klassen bzw. daraus instanziierten Objekten bezeichnet man als ***Attribute***.
- Mittels des **Pfeiloperators** kann auf die Attribute zugegriffen (lesend oder schreibend).

Beispiel:

```php
<?php

$myObject = new MyClass();

$myObject->attribute1 = "Wert"; // Wert einem Attribut zuweisen
echo $myObject->attribute1; // Wert eines Attributs auslesen
```

> *Hinweis:* Dynamische Eigenschaften (Attribute) sind seit PHP 8.2.0 veraltet. Ausgabe im Beispiel:
> `Deprecated: Creation of dynamic property MyClass::$attribute1 is deprecated in ...`

Sichtbarkeit von Attributen

- Attribute können bereits in der Klassendefinition deklariert werden.
- Es empfiehlt sich, diese bereits mit einem Standardwert zu initialisieren und somit vorab quasi den Datentyp zu bestimmen.
- Die Schlüsselwörter `public` und `protected` legen den Gültigkeitsbereich der Attribute fest und steuern somit die Sichtbarkeit der Attribute.

Konzept der Sichtbarkeit

Die Schlüsselwörter `public` und `protected` sind wesentlich für das Konzept der Sichtbarkeit oder des Zugriffsschutzes innerhalb der objektorientierten Programmierung (OOP). Sie bestimmen, wie und von wo auf die Attribute (Eigenschaften) und Methoden einer Klasse zugegriffen werden kann.
Durch die Verwendung dieser Schlüsselwörter können Entwickler die Integrität und Sicherheit ihrer Objekte besser steuern, indem sie festlegen, wer Zugriff auf welche Eigenschaften und Methoden hat. Dies ist ein fundamentaler Aspekt der OOP, der hilft, robuste und sichere Software zu erstellen.

Schlüsselwörter `public`

- **Bedeutung**: Wenn ein Attribut oder eine Methode als `public` deklariert wird, bedeutet dies, dass es von überall her zugänglich ist. Das beinhaltet den Code innerhalb der Klasse selbst, den Code in abgeleiteten Klassen (Klassen, die von dieser Klasse erben) und den Code außerhalb der Klasse.
- **Verwendungszweck**: `public` wird normalerweise für solche Funktionen und Daten verwendet, die sicher von außerhalb der Klasse genutzt werden können, ohne das innere Funktionieren der Klasse zu stören oder Sicherheitsrisiken darzustellen.
- **Beispiel**: Ein öffentliches Attribut einer Klasse `Person` könnte `name` sein, da es in der Regel unproblematisch ist, wenn externe Codes auf den Namen einer Person zugreifen.

Schlüsselwörter `protected`

- **Bedeutung**: Ein `protected` Attribut oder eine Methode ist nur innerhalb der Klasse selbst und in Klassen, die von dieser Klasse erben, zugänglich. Es ist jedoch nicht zugänglich für den allgemeinen Code, der Instanzen dieser Klasse verwendet.
- **Verwendungszweck**: `protected` wird eingesetzt, um die interne Funktionsweise der Klasse zu kapseln und zu verbergen. Es gewährt einen gewissen Grad an Sicherheit, indem es verhindert, dass der Code außerhalb der Klasse oder nicht verwandter Klassen die internen Aspekte der Klasse manipuliert.
- **Beispiel**: Ein geschütztes Attribut in einer Klasse `Bankkonto` könnte `kontostand` sein, da der Kontostand nur von der Klasse `Bankkonto` und ihren abgeleiteten Klassen (z.B. `Sparkonto`) verwaltet werden sollte, um unerwünschte oder schädliche Zugriffe zu vermeiden.

Methoden

- Methoden sind Funktionen, die an Klassen gebunden sind.
- Methoden können auf Objekten aufgerufen werden, die aus einer Klasse erzeugt wurden, die diese Methode enthält.

Beispiel Methodendefinition:

```php
<?php
class MyClass {
    public function myMethod1($parmater1, $parameter2, $parameter3) {
      // Code der Methode 1
    }
    protected function myMethod2($parmater1, $parameter2, $parameter3) {
      // Code der Methode 2
    }
}
```

Aufruf von Methoden

- Methoden werden mittels des **Pfeiloperators** auf das Objekt der Klasse aufgerufen

Beispiel:

```php
<?php
class MyClass {
      // wie Beispiel zuvor
}

$object = new MyClass();

$object->myMethod1(/*Argumente*/); // ruft Methode 1 auf
$object->myMethod2(/*Argumente*/); // erzeugt einen Fehler - Methode 2 ist protected also
von außen nicht sichtbar bzw. zugreifbar
```

Begriffe Instanz und Objekt

Instanz:

- Eine "Instanz" bezieht sich spezifisch auf eine konkrete Realisierung einer Klasse.
- Wenn eine Klasse als eine Vorlage oder ein Bauplan angesehen wird, ist eine Instanz eine realisierte Version dieses Plans.
- Zum Beispiel, wenn `Auto` eine Klasse ist, dann wäre ein spezifisches Auto, das nach dieser Vorlage erstellt wurde (z.B. ein rotes Auto der Marke Toyota), eine Instanz dieser Klasse.
- Der Prozess der Erstellung einer Instanz aus einer Klasse wird "Instanziierung" genannt.

Objekt:

- Ein "Objekt" ist ein allgemeinerer Begriff, der jede Instanz einer Klasse bezeichnet.
- In OOP ist ein Objekt eine Sammlung von Daten und Methoden, die auf diese Daten operieren.
- Jedes Mal, wenn eine Instanz erstellt wird, wird ein Objekt erzeugt.
- In demselben Beispiel ist das rote Toyota-Auto nicht nur eine Instanz der Klasse `Auto`, sondern auch ein Objekt.

Zusammengefasst: Begriffe Instanz und Objekt

In der objektorientierten Programmierung (OOP), insbesondere in PHP, werden die Begriffe "Objekt" und "Instanz" oft synonym verwendet, aber sie haben leicht unterschiedliche Bedeutungen:

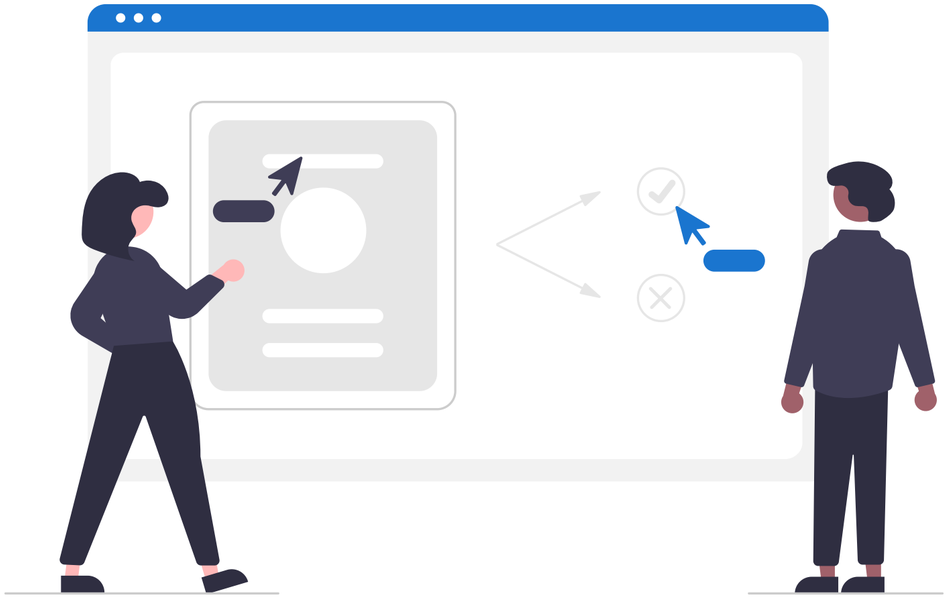
Bedeutung Instanz vs. Objekt:

Jede "**Instanz**" ist ein "**Objekt**", aber der Begriff "**Objekt**" wird in einem breiteren Sinne verwendet, um ***jede mögliche Realisierung*** einer Klasse in der OOP zu bezeichnen.

Der Begriff "**Instanz**" betont den Aspekt der ***Erzeugung eines spezifischen Objekts*** aus einer Klasse.

PraxisWorkshop - Objekte, Klassen und Instanziierung

⇒ Erzeugen einer eigenen Klassendefinition
⇒ Instanziierung - Erzeugung eines Objektes aus einer Klasse
⇒ Zugriff auf Attribute und Methoden von Objekten



Workshopdateien:

→ oop06.php
→ oop07.php
→ oop08.php
→ oop09.php
→ oop10.php
→ oop11.php
→ oop12.php
→ oop13.php
→ oop14.php

Zusammenfassung Objekte, Klassen und Instanziierung

- Jedes **Objekt** einer Klasse übernimmt bei der Instanziierung sämtliche **Attribute** und **Methoden** dieser Klasse.
- Die Klasse ist somit eine Art **Prototyp** oder **Bauplan**, in dem alles definiert wird, was später benötigt wird.
- Mit Hilfe dieses Bauplans können dann individualisierbare Varianten (die sogenannten Objekte) instanziiert werden.
- Die Attribute, etwa der Preis, können und werden sich zukünftig von Instanz zu Instanz unterscheiden.
- Der Begriff **Attribut** kann sowohl die Eigenschaft bezeichnen als auch den hierin abgelegten Wert.
- Zwei verschiedene Klassen können identische Bezeichner für Attribute und Methoden verwenden.
- Sie können zwei Methoden, die das Gleiche tun, also gleich benennen, sofern diese in unterschiedlichen Klassen definiert werden.
- Durch ihre Bindung an die Klasse bleiben sie trotzdem eindeutig.

Zugriff auf Attribute und Methoden innerhalb einer Klasse

```php
<?php
class MyClass {
    public $attr1 = "Wert1";
    protected $attr2 = "Wert2";

    public function myMethod1() {
      echo $attr1; // erzeugt einen Fehler
    }

    protected function myMethod2() {
      echo $attr1; // erzeugt einen Fehler
    }
}
```

Pseudovariable `$this`

> Die Pseudovariable `$this` ist verfügbar, falls eine Methode aus einem Objektkontext heraus aufgerufen wird. `$this` ist eine Referenz auf das aufrufende Objekt (üblicherweise das Objekt, zu dem die Methode gehört, aber möglicherweise ein anderes Objekt, falls die Methode statisch aus dem Kontext eines zusätzlichen Objektes aufgerufen wird).

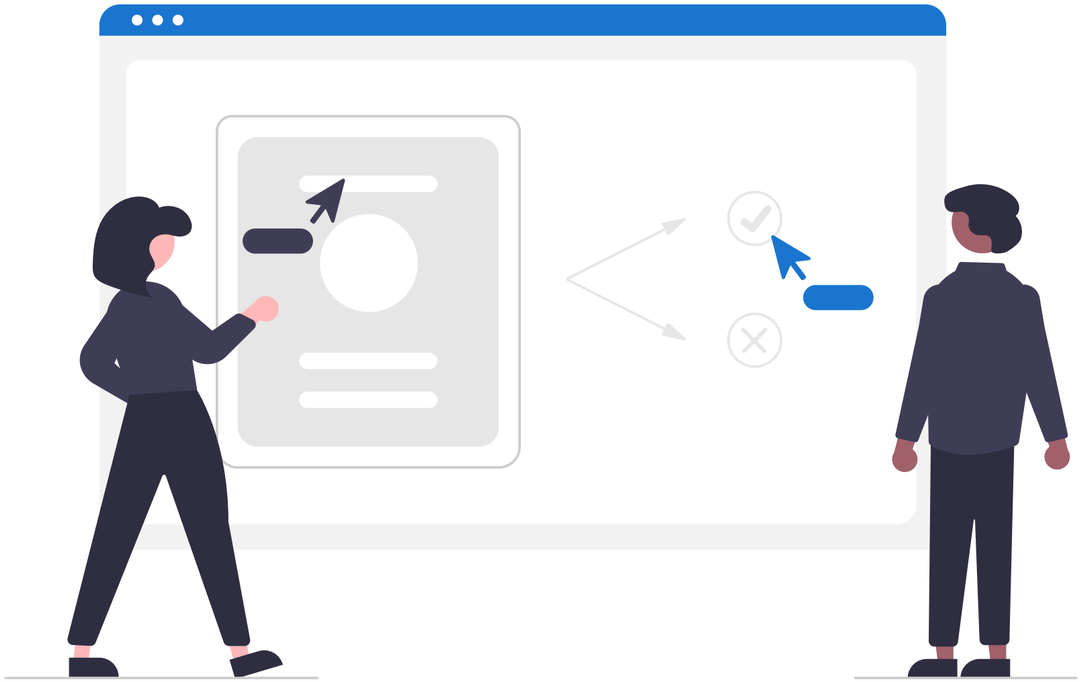
php.net

Die Variable `$this` in PHP

- Die Variable `$this` ermöglicht es innerhalb einer Klasse auf die Attribute und die Methoden der Klasse zuzugreifen ohne dass bereits ein Objekt aus der Klasse instanziiert wurde.
- `$this` stellt also eine Referenz auf die Klasse bzw. auf das später instanziierte Objekt dar.
- Wann immer Sie in einer Klasse auf ein Attribut oder eine Methode eines aus dieser Klasse instanziierten Objektes zugreifen wollen, müssen Sie die spezielle Variable `$this` verwenden.
- Also müssen Sie `$this->attribute` oder `$this->method()` schreiben.

PraxisWorkshop - Pseudovariable `$this`

⇒ Zugriff auf Attribute und Methoden innerhalb einer Klasse



Workshopdateien:

→ benutzer_version_1.php
→ benutzer_version_2.php
→ benutzer_version_3.php
→ benutzer_version_4.php

Namenskonventionen

1. **Klassen** - Klassennamen verwenden den **UpperCamelCase**, beginnen also mit einem Großbuchstaben. Bei zusammengesetzen Wörten wird jedes Wort großgeschrieben. Klassennamen sind i. R. Nomen und werden immer in Einzahl geschrieben.

2. **Attribute** - Attribute verwenden den **lowerCamelCase**, werden also kleingeschrieben. Bei zusammengesetzen Wörten wird ab dem 2-ten Wort jedes Wort großgeschrieben. Also analog der Namensgebung von Variablen.

3. **Methoden** - Für Methoden gelten dieselben Regeln wie für Attribute sowie für die Namensgebung von Funktionen, der Name sollte immer mit einem Verb beginnen.

„Erzähle mir und ich vergesse. Zeige mir und ich erinnere. Lass es mich tun und ich verstehe.“ *(Konfuzius, 551 bis 479 v. Chr.)*

Zu dieser Dokumentation / Lizenzbedingungen

Dieses Dokument ist eine begleitende Dokumentation zu den Workshops und wird fortlaufend kontextbezogen angepasst bzw. ergänzt

> *→ Living Document.*

Es hat ausdrücklich nicht den Anspruch eines Lehrbuchs, sondern soll lediglich als Orientierungshilfe für die Durchführung und Wiederholung der Workshops sowie als Einstiegshilfe für eigene weitere Recherchen verstanden werden.

Alle Inhalte dieses Dokuments, sofern nicht selbst erstellt oder nicht anders angegeben, stammen aus Quellen mit frei zugänglichen Inhalte, die unter der Creative Commons Attribution-ShareAlike-Lizenz veröffentlich oder speziell für die Erstellung von Unterrichts- bzw. Studienmaterial freigegeben sind.

> **Es sollen ausdrücklich diese Inhalte nicht als Eigene dargestellt und ausgegeben werden!**

Sollte trotz Überprüfung eine Urheberrechtsverletzung vorliegen, bitte ich um kurze Information, damit diese umgehend beseitigt werden kann.

> Dieses Dokument ist ausschließlich für Schulungszwecke erstellt und teilweise urheberrechtlich geschützt bzw. unterliegt tw. urheberrechtlichen Einschränkungen. Somit sind die Weitergabe sowie die Vervielfältigung dieses Dokuments als Ganzes, aber auch die Verwertung und Mitteilung seines Inhalts nicht erlaubt, soweit nicht ausdrücklich durch die angegebene Lizenz gestattet.
> *(s. a. ggf. Lizenzangaben im begleitenden Script/Handout)*