OOP Objektorientierte Programmierung in PHP - Part 2

Hallo liebe Community!
Dies ist mein erstes Tutorial also seit nicht zu streng mit der Kritik, über Verbesserungsvorschläge würde ich mich dennoch freuen!

Ich setze voraus, dass man weiß wie Funktionen geschrieben werden und dass man mit Variablen umgehen kann. Außerdem sollte man folgende Parts meines Tutorials gelesen haben:

- [wiki]OOP Objektorientierte Programmierung in PHP - Part 1[/wiki]

Ich werde weiterhin mit unseren Tollen Raumschiffen spielen xD
Was wenn die Raumschiffe nicht alle gleich aussehen, sondern der Benutzer beim Bau eines Raumschiffes eine Farbe angeben soll? Wie nicht anders zu erwarten gibt es auch hier eine Lösung. Jedes Objekt wird konstruiert. Und damit PHP weiß ob bei dem konstruieren eines Objekts irgendetwas beachtet werden muss (eine Variable initialisiert werden, etc.) gibt es eine Funktion die das Objekt konstruiert, den Konstruktor (construct).

Quellcode

```
<?php
class Raumschiff
{
private $leben;
private $farbe
public function __construct($farbe) //Die method __construct kann auch Parameter bekommen
{
$this->farbe = $farbe;
$this->leben = 100;
}
public function fliegen()
{
echo "I am crusin through space oda so";
}
public function getLeben()
{
return $this->leben;
}
public function setLeben($leben)
{
$this->leben = $leben;
}
public function getFarbe()
{
return "Das Raumschiff ist in ".$this->farbe." angestrichen";
}
}
$Schiff = new Raumschiff("sehr-dunklem-braun-das-wie-schwarz-aussieht-und-ein-raumschiff-im-weltall-fast-unsichtbar-macht");
echo $Schiff->getLeben();
echo $Schiff->getFarbe();
?>
```

Alles anzeigen

Beim erzeugen eines Objekts der Klasse Raumschiff wird der Konstruktor (__constuct) aufgerufen. Parameter können in den runden Klammern bei der Instanzierung eines Objekts übergeben werde.

Ein Standartwert für Parameter ist nätürlich auch möglich:

Quellcode

```
1. public function __construct($farbe = "schwarz")
2. {
3. $this->farbe = $farbe;
4. $this->leben = $leben;
5. }
```

Die nächste Frage, die man sich stellt, sollte an dieser Stelle im Tutorial sein, was passiert wenn ein Raumschiff mal abgeschossen wird? Stellt euch vor ihr seid ein süßer kleiner Jäger der in einer Flotte mit anderen Schiffen unterwegs ist. Nun wird einen knappen Kilometer neben euch ein riesiger Zerstörer abgeschossen und explodiert. In dem Fall sollte der kleine süße Jäger doch eigentlich auch zerstört werden.
Nun auf PHP bezogen: was passiert wenn ein Objekt destruiert (heißt das so?) wird. Für diesen Fall gibt es den sogenannten Destruktor (destruct). Das sieht dann so aus:

Quellcode

```
1. <?php
2. class Raumschiff
3. {
4. private $leben;
5. private $farbe
6. public function __construct($farbe)
7. {
8. $this->farbe = $farbe;
9. $this->leben = 100;
10. }
11. public function __destruct()
12. {
13. return "Keine Explosion, wegen Benzin Mangels!";
14. }
15. public function fliegen()
16. {
17. echo "Bald muss ich tanken ;(";
18. }
19. public function getLeben()
20. {
21. return $this->leben;
22. }
23. public function setLeben($leben)
24. {
25. $this->leben = $leben;
26. }
27. public function getFarbe()
28. {
29. echo "Das Raumschiff ist in ".$this->farbe." angestrichen";
30. }
31. }
32. $Schiff = new Raumschiff("neongrün");
33. echo $Schiff->fliegen();
34. echo $Schiff->getFarbe();
35. ?>
```

Alles anzeigen

In der Praxis habe ich einen destruct noch nie benutzt und deswegen ist mir auch leider kein sinvolles Beispiel für unsere Raumschiffe eingefallen.

Ein Anwendungsbeispiel wären Datenbank Klassen. Sie bauen verbindung zur Datenbank auf (meist im construct) und am Ende der Scripts muss die Verbindung wieder geschlossen werden. (thx Gambler)

Man sollte den Konstruktor einer Klasse public haben! Die einzige SINVOLLE Ausnahme bilden Singleton-Klassen, darauf komme ich bestimmt in einem Teil des Tutorials zurück.
construct und destruct haben NIE ein return statement!

#To be continued. Next part out now: [wiki]OOP Objektorientierte Programmierung in PHP - Part 3[/wiki]
n0x-f0x