Κλάσεις και αντικείμενα

Στην PHP 5 η υλοποίηση των κλάσεων και των αντικειμένων αναθεωρήθηκε ριζικά σε σχέση με την PHP 4.

Στα χαρακτηριστικά που υποστηρίζει η PHP 5 προστέθηκαν η κληρονομικότητα (inheritance), οι αφηρημένες (abstract) και οι τελικές (final) κλάσεις και μέθοδοι, οι διεπαφές (interfaces) κ.ά.

Η δήλωση μιας κλάσης ξεκινάει με τη λέξη-κλειδί class, ακολουθεί το όνομα της κλάσης και στη συνέχεια προσδιορίζονται οι ιδιότητες και οι μέθοδοι της κλάσης μέσα σε άγκιστρα.

Το όνομα της κλάσης πρέπει να ξεκινάει από γράμμα ή «_» (underscore) και να ακολουθείται από οποιοδήποτε πλήθος γραμμάτων, ψηφίων και «_» (underscores).

```
<?php
class WebDevelopmentAcademy
{
	public $property = 5;
	public myMethod()
	{
		return $this->var;
	}
}
?>
```

$this

Η ψευδο-μεταβλητή $this είναι διαθέσιμη στις μεθόδους που καλούνται μέσα από ένα αντικείμενο (στιγμιότυπο) μιας κλάσης και αποτελεί αναφορά στο αντικείμενο που καλεί τη μέθοδο.

```
<?php
class SimpleClass
{
    // property declaration
    public $var = 'a default value';

    // method declaration
    public function displayVar() {
        echo $this->var;
    }
}
?>
```

Δημιουργία αντικειμένων

Η δημιουργία αντικειμένων πραγματοποιείται με τη λέξη-κλειδί new ακολουθούμενη από το όνομα της κλάσης που θέλουμε να χρησιμοποιήσουμε για το αντικείμενό μας.

```
<?php

class SimpleClass
{
    public $var = 'a default value';

    public function displayVar() {
        echo $this->var;
    }
}

$instance = new SimpleClass();

// This can also be done with a variable:
$className = 'SimpleClass';
$instance = new $className(); // new SimpleClass()
?>
```

Κατασκευαστές

Οι κλάσεις που διαθέτουν τη μέθοδο __construct την εκτελούν κατά τη δημιουργία ενός νέου αντικειμένου της κλάσης. Η μέθοδος __construct ονομάζεται κατασκευαστής και χρησιμοποιείται για την αρχικοποίηση των ιδιοτήτων του αντικειμένου της κλάσης.

Οι κατασκευαστές των κληρονομηθέντων κλάσεων δεν καλούνται αυτόματα. Για να καλέσουμε τον κατασκευαστή μιας κλάσης-γονέα χρησιμοποιούμε τη λέξη-κλειδί parent.

```
<?php
class BaseClass {
   function __construct() {
       print "In BaseClass constructor\n";
   }
}

class SubClass extends BaseClass {
   function __construct() {
       parent::__construct();
       print "In SubClass constructor\n";
   }
}
?>
```

Καταστροφείς

Οι κλάσεις που διαθέτουν τη μέθοδο __destruct την εκτελούν όταν δεν υπάρχουν πλέον σε χρήση αναφορές προς το αντικείμενο ή κατά τη διαδικασία τερματισμού της εκτέλεσης του κώδικα. Η μέθοδος __destruct ονομάζεται καταστροφέας.

Οι καταστροφείς των κληρονομηθέντων κλάσεων δεν καλούνται αυτόματα. Για να καλέσουμε τον καταστροφέα μιας κλάσης-γονέα χρησιμοποιούμε τη λέξη-κλειδί parent.

```
<?php
class BaseClass {
   function __destruct() {
       print "In BaseClass destructor\n";
   }
}

class SubClass extends BaseClass {
   function __destruct() {
       parent::__destruct();
       print "In SubClass destructor\n";
   }
}
?>
```

Κληρονομικότητα

Μια κλάση μπορεί να κληρονομήσει τις ιδιότητες και τις μεθόδους άλλης κλάσης με χρήση της λέξης-κλειδιού extends στον ορισμό της.

Μέσω του μηχανισμού extends δεν υποστηρίζεται πολλαπλή κληρονομικότητα.

Οι ιδιότητες και οι μέθοδοι που κληρονομήθηκαν μπορούν να παρακαμφθούν (override) σε περίπτωση που ξαναοριστούν με εξαίρεση τις τελικές μεθόδους της κλάσης-γονέα.

Η πρόσβαση στις μεθόδους της κλάσης γονέα γίνεται με τη λέξη-κλειδί parent.

Η παράκαμψη (overriding) μεθόδων απαιτεί ίδια υπογραφή για την ίδια μέθοδο σε κάθε κλάση με μόνη εξαίρεση τον κατασκευαστή.

Κληρονομικότητα

```php
<?php

class SimpleClass
{
    public $var = 'a default value';

    public function displayVar() {
        echo $this->var;
    }
}

?>
```

```php
<?php
class ExtendClass extends SimpleClass
{
    // Redefine the parent method
    function displayVar()
    {
        echo "Extending class\n";
        parent::displayVar();
    }
}

$extended = new ExtendClass();
$extended->displayVar();
?>
```

Ιδιότητες κλάσης

Οι μεταβλητές-μέλη της κλάσης ονομάζονται ιδιότητες της κλάσης. Ορίζονται ανάλογα με την απαιτούμενη ορατότητα με μια από τις λέξεις-κλειδιά public, protected και private ακολουθούμενες από το όνομα της μεταβλητής.
Ο ορισμός μιας ιδιότητας μπορεί να συνοδεύεται με την αρχικοποίησή της με μια σταθερή τιμή.

Μέσα στις μεθόδους των κλάσεων η προσπέλαση των μη-στατικών ιδιοτήτων πραγματοποιείται με τον τελεστή προσπέλασης μέλους -> πχ $this->property
Για τις στατικές ιδιότητες της κλάσης χρησιμοποιείται ο τελεστής :: πχ self::$property

```
<?php
class SimpleClass
{
	public static $var1 = 5;
	public $var2 = 10;

	public function test()
	{
		return $this->var2 + self::$var1;
	}
}
?>
```

Μέθοδοι κλάσης

Οι συναρτήσεις-μέλη της κλάσης ονομάζονται μέθοδοι της κλάσης. Ορίζονται ανάλογα με την απαιτούμενη ορατότητα με μια από τις λέξεις-κλειδιά public, protected και private ακολουθούμενες από τον ορισμό της μεθόδου ακολουθώντας τους ίδιους κανόνες με τον ορισμό μιας συνάρτησης.

Η κλήση των μη-στατικών μεθόδων πραγματοποιείται με τον τελεστή προσπέλασης μέλους -> πχ $this->method(), ενώ για τις στατικές μεθόδους της κλάσης χρησιμοποιείται ο τελεστής :: πχ self::staticMethod()

```
<?php
class SimpleClass
{
	public static function myStaticFunction() { }
	public function test() {}
}

SimpleClass::myStaticFunction();
$a = new SimpleClass();
$a->test();

?>
```

Ορατότητα μελών κλάσης

Η ορατότητα των μελών μιας κλάσης καθορίζεται από τις λέξεις-κλειδιά public, protected και private.

Όταν το μέλος είναι public επιτρέπεται η προσπέλασή του από οποιοδήποτε σημείο του κώδικα.

Όταν το μέλος είναι protected επιτρέπεται η προσπέλασή του από οποιοδήποτε σημείο της κλάσης που ορίζεται, σε κλάσεις που κληρονομεί αυτή η κλάση και στις κλάσεις που κληρονομείται.

Όταν το μέλος είναι private επιτρέπεται η προσπέλασή του μόνο από την κλάση στην οποία ορίζεται.

Ορατότητα ιδιοτήτων κλάσης

```php
<?php
class MyClass
{
    public $public = 'Public';
    protected $protected = 'Protected';
    private $private = 'Private';

    function printHello()
    {
        echo $this->public;
        echo $this->protected;
        echo $this->private;
    }
}

$obj = new MyClass();
echo $obj->public; // Works
echo $obj->protected; // Fatal Error
echo $obj->private; // Fatal Error
$obj->printHello(); // Shows Public, Protected and Private
?>
```

Ορατότητα ιδιοτήτων κλάσης

```
<?php
class MyClass2 extends MyClass
{
    // We can redeclare the public and protected properties, but not private
    public $public = 'Public2';
    protected $protected = 'Protected2';

    function printHello()
    {
        echo $this->public;
        echo $this->protected;
        echo $this->private;
    }
}

$obj2 = new MyClass2();
echo $obj2->public; // Works
echo $obj2->protected; // Fatal Error
echo $obj2->private; // Undefined
$obj2->printHello(); // Shows Public2, Protected2, Undefined
?>
```

Ορατότητα μεθόδων κλάσης

```
<?php
class MyClass
{
    public function __construct() { }
    public function MyPublic() { }
    protected function MyProtected() { }
    private function MyPrivate() { }
    function Foo()
    {
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate();
    }
}

$myclass = new MyClass;
$myclass->MyPublic(); // Works
$myclass->MyProtected(); // Fatal Error
$myclass->MyPrivate(); // Fatal Error
$myclass->Foo(); // Public, Protected and Private work
?>
```

Ορατότητα μεθόδων κλάσης

```
<?php
class MyClass2 extends MyClass
{
    // This is public
    function Foo2()
    {
        $this->MyPublic();
        $this->MyProtected();
        $this->MyPrivate(); // Fatal Error
    }
}

$myclass2 = new MyClass2;
$myclass2->MyPublic(); // Works
$myclass2->Foo2(); // Public and Protected work, not Private
?>
```

Ορατότητα μεθόδων κλάσης

```
class Bar
{
    public function test() {
        $this->testPrivate();
        $this->testPublic();
    }

    public function testPublic() {
        echo "Bar::testPublic\n";
    }

    private function testPrivate() {
        echo "Bar::testPrivate\n";
    }
}
```

```
class Foo extends Bar
{
    public function testPublic() {
        echo "Foo::testPublic\n";
    }

    private function testPrivate() {
        echo "Foo::testPrivate\n";
    }
}

$myFoo = new Foo();
$myFoo->test(); // Bar::testPrivate
                // Foo::testPublic
```

Σταθερές κλάσης

Η PHP 5 προσφέρει τη δυνατότητα ορισμού σταθερών σε επίπεδο κλάσης.

Οι σταθερές ορίζονται με τη λέξη-κλειδί const ακολουθούμενη από το όνομα της σταθεράς (συνήθως με κεφαλαία και χωρίς $) και την εκχώρηση της τιμής της.

```
<?php
class MyClass
{
    const CONSTANT = 'constant value';

    function showConstant() {
        echo  self::CONSTANT . "\n";
    }
}

echo MyClass::CONSTANT . "\n";

$class = new MyClass();
echo $class::CONSTANT."\n";
?>
```

Περισσότερα για τις κλάσεις

http://php.net/manual/en/language.oop5.php

Χρήση πολλαπλών αρχείων πηγαίου κώδικα

Για να οργανώσουμε καλύτερα σε επίπεδο αρχείων ένα έργο συχνά απαιτείται η διάσπαση ενός αρχείου πηγαίου κώδικα σε μικρότερα.

Για να χρησιμοποιήσουμε σε ένα αρχείο της PHP ένα άλλο αρχείο πηγαίου κώδικα χρησιμοποιούμε την οδηγία include ακολουθούμενη από την τοποθεσία του αρχείου που θέλουμε να συμπεριλάβουμε στον κώδικά μας.

index.php

```
<!DOCTYPE html>
<head>
<title>Sample page</title>
</head>
<body>
<?php include 'menu.php'; ?>

</body>
</html>
```

menu.php

```
<a href=”#”>Link 1</a>
<a href=”#”>Link 2</a>
<a href=”#”>Link 3</a>
```

Χρήση πολλαπλών αρχείων πηγαίου κώδικα

index.php

```
<html>
<body>

<h1>Welcome to my home page!</h1>
<?php
include 'vars.php';
echo "I have a $color $car.";
?>

</body>
</html>
```

vars.php

```
<?php
$color='red';
$car='BMW';
?>
```