

Universiteti i Prishtinës


Fakulteti i Inxhinierisë Elektrike dhe Kompjuterike

Lënda: Programimi në Internet



Java 4: PHP Klasat dhe objektet – PHP në OO

Konceptet themelore të OOP:

Object	Properties	Methods
	<code>car.name = Fiat</code> <code>car.model = 500</code> <code>car.weight = 850kg</code> <code>car.color = white</code>	<code>car.start()</code> <code>car.drive()</code> <code>car.brake()</code>

- **Class:** This is a programmer-defined datatype, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.
- **Object:** An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.
- **Member Variable:** These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created.
- **Member function:** These are the function defined inside a class and are used to access object data.
- **INHERITANCE:** When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Parent class:** A class that is inherited from by another class. This is also called a base class or super class.
- **Child Class:** A class that inherits from another class. This is also called a subclass or derived class.
- **POLYMORPHISM:** This is an object oriented concept where same function can be used for different purposes. For example function name will remain same but it make take different number of arguments and can do different task.
- **Overloading:** a type of polymorphism in which some or all of operators have different implementations depending on the types of their arguments. Similarly functions can also be overloaded with different implementation.
- **DATA ABSTRACTION:** Any representation of data in which the implementation details are hidden (abstracted).
- **ENCAPSULATION:** refers to a concept where we encapsulate all the data and member functions together to form an object.

- **Constructor:** refers to a special type of function which will be called automatically whenever there is an object formation from a class.
- **Destructors:** refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope.

Ushtrimi 1: Të definohet një klasë në PHP, e cila së pari i definon vetitë, pastaj metodat dhe pas krijimit të objekteve i përdorë këto metoda.

```
<?php
```

```
class Ariu {  
    //definojme vetite  
  
    public $emri;  
    public $pesha;  
    public $mosha;  
    public $gjinia;  
    public $ngjyra;  
  
    //definojme metodat  
  
    public function ngrenia()  
    {  
        echo $this->emri. " eshte duke ngrene ...\n";  
    }  
  
    public function vrapimi()  
    {  
        echo $this->emri. " eshte duke vrapuar ...\n";  
    }  
  
    public function mbyt()  
    {  
        echo $this->emri. " eshte duke mbytur prene ...\n";  
    }  
  
    public function flen()  
    {  
        echo $this->emri. " eshte duke fjetur ...\n";  
    }  
}
```

```
$babai = new Ariu;
```

```
$babai->emri = "Ariu Baba";
$babai->mosha = 8;
$babai->gjinia = "Mashkull";
$babai->ngjyra = "i zi";
$babai->pesha = 300;

$nena = new Ariu;

$nena->emri = "Ariu Nene";
$nena->mosha = 7;
$nena->gjinia = "Femer";
$nena->ngjyra = "e bardhe";
$nena->pesha = 300;

$femiu = new Ariu;

$femiu->emri = "Ariu Femi";
$femiu->mosha = 2;
$femiu->gjinia = "Mashkull";
$femiu->ngjyra = "i zi";
$femiu->pesha = 180;

print ("<h2>Shembuj me PHP te OO</h2>");

$babai->mbyt();
echo "</br>";

$nena->ngrenia();
echo "</br>";

$femiu->ngrenia();
echo "</br>";

$nena->flen();
echo "</br>";

$babai->flen();
echo "</br>";

$femiu->flen();
echo "</br>";

?>
```

Ushtrimi 2: Të krijohet një klasë e cila përshin variabla të krijuara në të, dhe si të tilla këto variabla përfshihen në funksione, të cilat më pas thirren nga objektet e krijuara.

```
<?php
class Librat{
    /* Variablat */
    var $cmimi;
    var $titulli;
    /* Funksionet */
    function setcmimi($par){
        $this->cmimi = $par;
    }
    function getcmimi(){
        echo $this->cmimi . "<br/>";
    }
    function settitulli($par){
        $this->titulli = $par;
    }
    function gettitulli(){
        echo $this->titulli . " <br/>";
    }
}

//krijimi i objekteve
$fizika = new Librat;
$matematike = new Librat;
$kimia = new Librat;

//thirrja e funksioneve
$fizika->settitulli( "Fizika per shkollen e mesme" );
$kimia->settitulli( "Kimia e avancuar" );
$matematike->settitulli( "Algjebra" );

$fizika->setcmimi( 10 );
$kimia->setcmimi( 15 );
$matematike->setcmimi( 7 );

//tani thirren funk me vlerat qe i jane caktuar me heret

$fizika->gettitulli();
$kimia->gettitulli();
$matematike->gettitulli();
$fizika->getcmimi();
$kimia->getcmimi();
$matematike->getcmimi();?>
```

Ushtrimi 3: Të definohet një konstruktor, dhe pastaj të përdoren variablat brenda tij për të kryer funksionin e caktuar.

```
<?php
class Ariu {
    //definojme vetite

    public $emri;
    public $pesha;
    public $ngjyra;

    public function __construct()
    {
        $this->mosha = 0;
        $this->pesha = 100;
        $this->ngjyra = "i zi";
    }
}
?>

<?php
$femiu = new Ariu;

$femiu->emri = "Ariu Femi";

echo $femiu->emri." është ".$femiu->ngjyra." dhe peshon ". $femiu->pesha. " kg në lindje.";

?>
```

E kundërta e konstruktorit është destruktori. Këta ekzekutohen para se klasa të shkatërrohet, gjë e cila ndodh automatikisht kur të gjitha referencat në klase bëhen unset.

Ngjashëm emërohen edhe destrukturët `__destruct()`. Destruktorët nuk mund të kenë parametra.

Ushtrimi 4: Përdorimi i attributeve të klasës si dhe përdorimi i funksioneve get() dhe set().

```
class classname
{
    var $attribute;
    function operation($param)
    {
        $this->attribute = $param
        echo $this->attribute;
    }
}
```

Ky shembull nuk e kufizon qasjen në attribute, andaj ju mund ti qaseni atyre nga jashtë klasës :

```
class classname
{
    var $attribute;
}
$a = new classname();
$a->attribute = 'value';
echo $a->attribute;
```

```
class klasa
{
    var $attribute;
    function __get($name)
    {
        return $this->$name;
    }
    function __set ($name, $value)
    {
        $this->$name = $value;
    }
}

?>
```

```
<?php
class person
{
    var $emri, $mbiemri;
    function __get($name)
    {
        return $this->name;
    }
    function __set($name,$value)
    {
```

```
$this->name = $value;
}
}
$personi = new person();
$personi->emri = "Artan";
print $personi->emri;

?>
```

Ushtrimi 5: Kontrollimi i qasjes me modifikatorët **private** dhe **public**

- **public** scope to make that variable/function available from anywhere, other classes and instances of the object.
- **private** scope when you want your variable/function to be visible in its own class only.
- **protected** scope when you want to make your variable/function visible in all classes that extend current class including the parent class.

<?php

```
class Studenti
{
    public $NrIndeksit = "12345";
    var $emri= "Arta";
    private $mbiemri = "Gashi";
    protected $adresa = "Prishtine";

    public function printo()
    {
        echo $this->mbiemri;
    }
}

$objStudenti = new Studenti();
print $objStudenti->NrIndeksit;
print '<br />';
print $objStudenti->emri;
print '<br />';
//print $objStudenti->printo();
print $objStudenti->mbiemri;
print '<br />';
print $objStudenti->adresa;
print '<br />';
print '<br />';
```

?>

```
<?php
    class Studenti2
    {
        public function funk1()
        {
            return "Eshte thirrur funksioni funk1";
            echo "</br>";
        }
        function funk2()
        {
            return "Eshte thirrur funksioni funk2";
            echo "</br>";
        }

        private function funk3()
        {
            return "Eshte thirrur funksioni funk3";
            echo "</br>";
        }
    }
    $ObjStudenti = new Studenti2();
    print $ObjStudenti->funk1();
    print '</br>';
    print $ObjStudenti ->funk2();
    print '</br>';
    print $ObjStudenti->funk3();
    print '</br>';

?>
```

Ushtrimi 6: Të ipet një shembull ku shihet implementimi i trashëgimisë në PHP.

```
<?php
    class A
    {
        var $atributi_A;
        function funk_A()
        {
            print "<br/>". $this->atributi_A;
        }
    }
    class B extends A
    {
```



```
var $atributi_B;  
function funk_B()  
{  
    print "<br/>". $this->atributi_B;  
}  
}  
$Obj_B = new B();  
$Obj_B->atributi_A =2;  
$Obj_B->funk_A();  
$Obj_B->atributi_B=3;  
$Obj_B->funk_B();  
?>
```

b) Të bëhet Kontrollimi i dukshmërisë përgjatë trashëgimit me **private** dhe **protected**

```
<?php  
  
class A  
{  
    private function operation1()  
    {  
        echo "operation1 called";  
    }  
    protected function operation2()  
    {  
        echo "operation2 called";  
    }  
    public function operation3()  
    {  
        echo "operation3 called";  
    }  
}  
  
class B extends A  
{  
    function __construct()  
    {  
        $this->operation1();  
        print "<br/>";  
        $this->operation2();  
        print "<br/>";  
        $this->operation3();  
        print "<br/>";  
    }  
}
```

```
$b = new B;  
?>
```

Trashëgimia e shumëfishtë

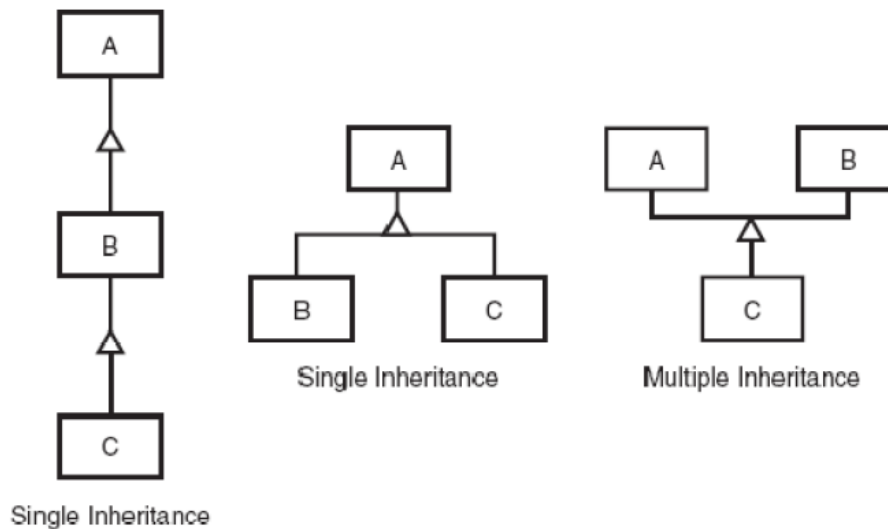


Figure 6.1 PHP does not support multiple inheritance.

Parandalimi i trashëgimit dhe mbishkruajtjes me final (diskutim)

```
class A  
{  
    var $attribute = 'default value';  
    final function operation()  
    {  
        echo 'Something<br />';  
        echo "The value of \$attribute is $this->attribute<br />";  
    }  
}
```

Mundeni gjithashtu të përdorni fjalën kyçe final që të parandaloni një klasë nga krijimet e të gjitha nënklasave. Të parandaloni klasën A nga ekzistenca e nënklasave, ju mund të shtoni këtë si në vazhdim:

```
final class A  
{...}
```

Nëse tentoni të trashëgoni nga A, do të merrni këtë error :
Fatal error: Class B may not inherit from final class (A)

Ushtrimi 7: Te shkruhet një klase që e paraqet mesazhin "Pershendetje, une jam Petriti!", ku "Petriti" është një vlerë e argumentit të një metode brenda klasës.

```
<?php
class user_message {
    public $message = 'Pershendetje, une jam ';

    public function introduce($name)
    {
        return $this->message.$name."<br />";
    }
}
$message = New user_message();
echo $message->introduce('Petriti!');
?>
```

b) Mbledhje, zbritje, pjesëtim, shumëzim.

```
<?php
class Kalkulatori {
    private $_fval, $_sval;
    public function __construct( $fval, $sval ) {
        $this->_fval = $fval;
        $this->_sval = $sval;
    }
    public function mledh() {
        return $this->_fval + $this->_sval;
    }
    public function zbrit() {
        return $this->_fval - $this->_sval;
    }
    public function shumezo() {
        return $this->_fval * $this->_sval;
    }
    public function pjesto() {
        return $this->_fval / $this->_sval;
    }
}
$mycalc = new Kalkulatori(12, 6);
echo $mycalc->mledh(); // Tregon 18
echo '<br />'.$mycalc->shumezo(); // Tregon 72
```

```
echo '<br />'. $mycalc->zbrit(); // Tregon 6  
echo '<br />'. $mycalc->pjesto(); // Tregon 2  
?>
```