INTRODUCERE OOP IN PHP

1. INTRODUCERE

Pentru a intelege scopul in care vei utiliza **Programarea Orientata pe Obiecte**(**Object Oriented Programming** in engleza) imagineaza-ti o cutie, in acea cutie sunt variabile si functii cu un scop asemanator, cu o legatura stransa. Folosim OOP pentru a stabilii o structura mai buna aplicatiilor noastre.
Un alt avantaj al OOP este ca nu mai este nevoie sa folosim **global** pentru a face ***variabile globale in functii***, **le vom apela pe cele ale clasei** cu

$this->nume_variabila

Sa dam exemplu unui blog cu sistem de utilizatori si cu comentarii, putem avea urmatoarele ***clase***:

- O clasa care se ocupa de Useri, cu functii specifice pentru logare, inregistrare s.a.m.d.
- O clasa pentru paginare, care va pagina articolele
- O clasa pentru comentarii, cu functii specifice comentariilor
- O clasa pentru MySQL, pentru a securiza query-urile mysql si a oferii functii specifice MySQL pentru terminarea mai rapida a proiectului

Cred ca ati prins ideea. Intr-o clasa putem avea functii (numite **metode** pentru a face diferenta dintre functiile simple si functiile dintr-o clasa), putem avea variabile (numite **proprietati** pentru a face diferenta dintre variabilele simple si variabilele dintr-o clasa).
Un exemplu de obiect este MySQLi, cand lucrati cu baze de date si le manilpulati cu obiectul facut din clasa mysqli.
Functiile si metodele pot fi:

- **public** - **valabila oriunde**
- **private** - **valabila doar in clasa respectiva**
- **protected** - valabila in clasa respectiva si in clasele facute/mostenite din ea

1.1 Ce este o clasa?

O clasa este o structura cu functii si variabile predefinte.

1.2 Ce este un obiect?

Un obiect este o instanta a clasei, o forma a ei.

1.3 Clasa si obiecte

Dintr-o clasa putem crea obiecte nelimitate. Toate obiectele pornesc cu functiile si variabilele predefinite din clasa, pe parcurs obiectele pot fi modificate. O clasa este ca un parinte al obiectelor, obiectele pornesc din start la fel ca clasa dar pe parcurs le putem modifica.

2. PRACTICA

Dupa atata teorie este necesar sa si practicam, altfel nu o sa tinem minte nimic.

2.1 Creare clasa

O clasa o cream asemanator unei functii, doar ca avem doua diferente. In loc de **function** folosim **class** si nu folosim paranteze.

Se recomanda ca clasele sa fie salvate in fisiere cu numele asemanator lor si intr-un director specific doar pentru clase, pentru o mai buna organizare. Numele fisierului trebuie sa fie de forma **class.nume_clasa.php**

Noi in exemplul urmator vom crea o clasa cu numele **exemplu**, o vom salva in directorul specific claselor si anume **clase** iar fisierul se va numii **class.exemplu.php**

Cod **class.exemplu.php**:

```
1  <?php
2  # fisier: clase/class.exemplu.php
3
4  // Cream clasa cu numele exemplu
5  class exemplu
6  {
7     # Acesta este interiorul clasei
8     # Aici putem declara metode(functii) si proprietati(variabile)
9  }
   ?>
```

2.2 Proprietati, metode si obiecte

In exemplul anterior am creat o clasa goala iar acum vei invata sa creezi si metode si proprietati in clase.

In exemplul urmator vom crea clasa exemplu(exact clasa creata mai devreme) dar in interiorul ei vom declara si proprietatea(variabila) publica cu numele **site** si functia publica cu numele **limbaj**.

Nu uita:

- Creaza directorul cu numele **clase** in caz ca nu exista deja
- In director clase vei introduce fisierul **class.exemplu.php** cu codul specific de mai jos
- In directorul cu un director mai sus decat **clase** vom crea fisierul **test-clasa.php**

Cod /clase/**class.exemplu.php**

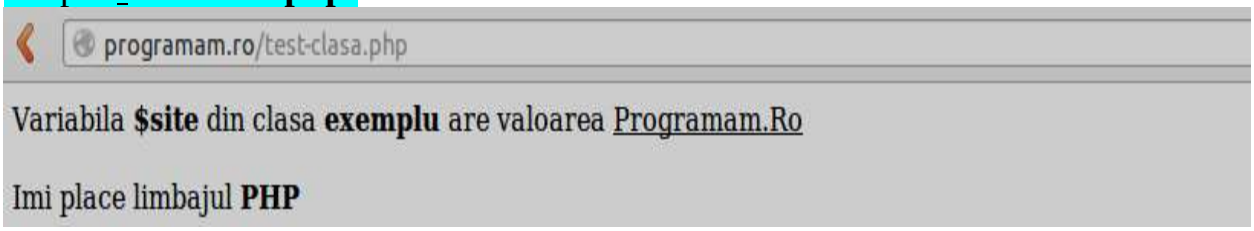
```
1  <?php
2  # fisier: /clase/class.exemplu.php
3
4  // Cream clasa cu numele exemplu
5  class exemplu
6  {
7     public $site = 'Programam.Ro';
8
9     public function limbaj()
10    {
11       echo 'Imi place limbajul <b>PHP</b><br/>';
12    }
13 }
   ?>
```

Cod /**test-clasa.php**:

```
1   <?php
2   # fisier: /test-clasa.php
3
4   // Includem fisierul cu clasa
5   include 'clase/class.exemplu.php';
6
7   // Cream o instant/exemplar a/l clasei
8   $obiect = new exemplu;
9
10  # Observati cum este apelata o variabila, fara semnul $
11  echo 'Variabila <b>$site</b> din clasa <b>exemplu</b>
12  are valoarea <u>'.$obiect->site.'</u><br/><br/>';
13
14  # Apelam functia limbaj din instanta $obiect a clasei exemplu
15  $obiect->limbaj();
    ?>
```

Observati ca numele variabilelei **$site** se apeleaza fara semnul **$**, deci daca avem variabila **site** o apelam cu **$obiect->site** si nu cu **$obiect->$site**. Instanta clasei(obiectul) si variabilele/functiile sunt despartite cu ajutorul -> un minus si semnul >

Output /**test-clasa.php**:



Noi am folosit in acest exemplu doar ***proprietati si metode publice***, daca una ar fi privata iar noi am apela-o cu ajutorul instantei(obiectului) ar genera o eroare asemanatoare cu:
Fatal error: Cannot access private property exemplu::$site
Proprietatile private **apartin doar clasei, nu si obiectului.**

3. METODA CONSTRUCTOR

Programarea orientata pe obiecte mai are si un avantaj, metodele magice. **Metoda constructor** este o metoda magica care se apeleaza odata ce instanta clasei ce este creata.
Cum cream metoda constructor? Pai cream o metoda(functie) in interiorul clasei cu numele **__construct**(doua caractere _) iar aceea este metoda constructor care va fi apelata odata ce este creata o instanta a clasei.
Metoda constructor mai are un avantaj, si anume **ca putem adauga parametrii la o clasa**. Clasa va fi apelata ca o functie doar că cu **new** in fata iar dupa numele ei

vor venii paranteze in care vor fi parametrii. **De mentionat ca metoda constructor este intotdeauna publica**.

Mai jos vom vedea un exemplu in care cream o clasa care are metoda constructor si este apelata cu un parametru(poate avea oricati parametrii, chiar si unul) iar noi cream doua obiecte din acea clasa.

Cod **/clase/class.exemplu.php**:

```
<?php
# fisier: clase/class.exemplu.php

// Cream clasa cu numele exemplu
class exemplu
{

   function __construct($nume)
   {
      echo "<br/>A fost creata o noua instanta cu
parametru <b>$nume</b> cu valoarea <u>{$nume}</u><br/>";
   }
}
?>
```

Cod **/test-clasa.php**:

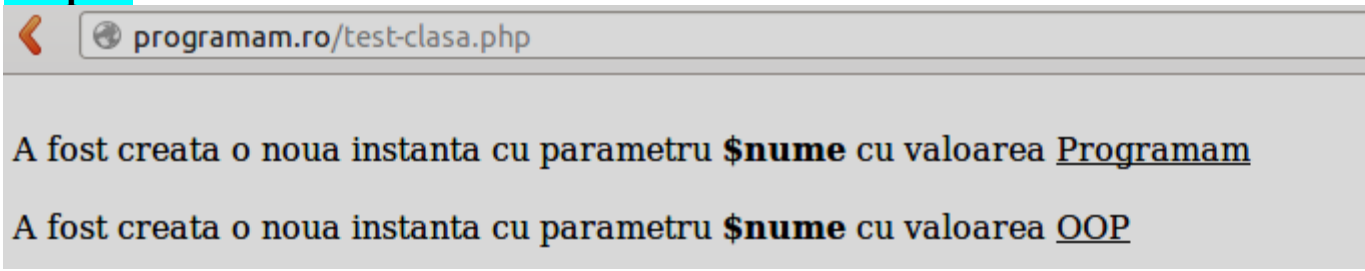
```
<?php
# fisier: /test-clasa.php

// Includem fisierul cu clasa
include 'clase/class.exemplu.php';

// Cream o instanta a clasei
$obiect = new exemplu('Programam');

$obiect2 = new exemplu('OOP');
?>
```

Output:



4. METODA DESTRUCTOR

O alta metoda magica este **metoda destructor**, ea este apelata cand obiectul este distrus (la incheierea executiei scriptului sau din script cu functia **unset**).

Metoda este asemanatoare ca nume cu metoda constructor deoarece numele metodei trebuie sa fie **__destruct** (cu doua caractere _).

In exemplul urmator vom crea doua obiecte, unul il vom distruge cu **unset** iar altul il vom lasa sa fie distrus cand scriptul isi incheie executia.

Cod /clase/**class.exemplu.php**:

```
    <?php
    # fisier: /clase/class.exemplu.php
1
2   // Cream clasa cu numele exemplu
3   class exemplu
4   {
5
6       function __construct($nume)
7       {
8           $this->nume = $nume; // salvam variabila in clasa
9           echo "<br/>A fost creata o noua instanta a clasei cu parametru <b>$nume</b
10          cu valoarea <u>{$nume}</u><br/>";
11      }
12
13      function __destruct(){
14          echo "<br/>Obiectul deschis cu parametru <b>{$this->nume}</b>
15          a fost distrus.<br/>";
16      }
17
    }
    ?>
```

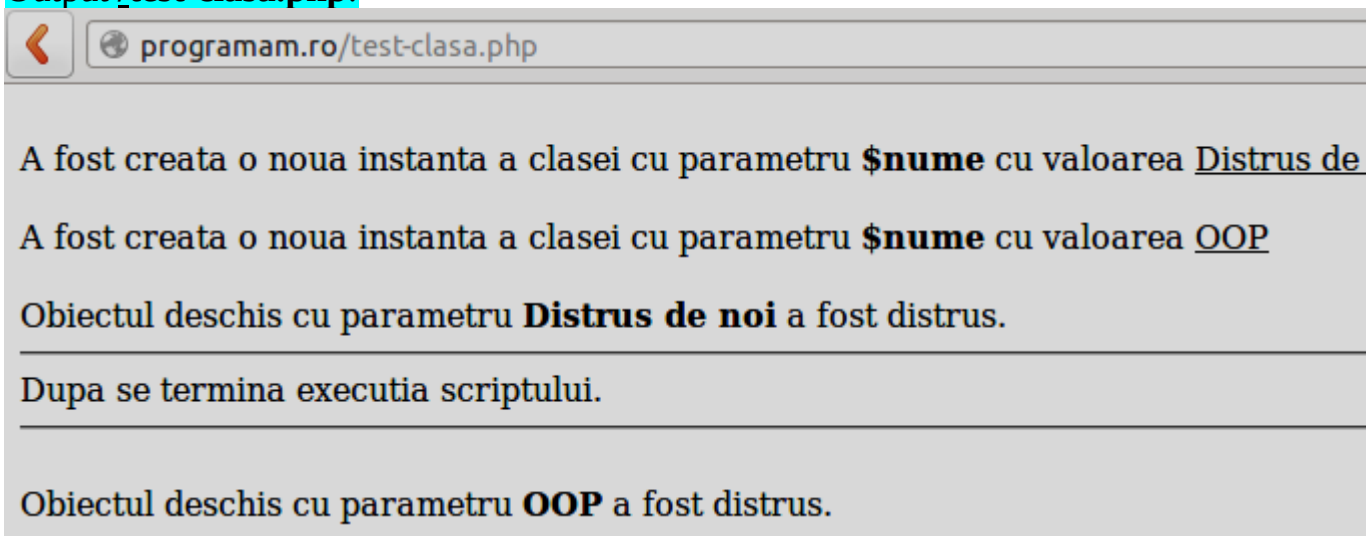
Observăm ca proprietatile clasei la apelam cu **$this->proprietate**, la fel si functiile.

Cod **/test-clasa.php**:

```
1       <?php
2       # fisier: /test-clasa.php
3
4       // Includem fisierul cu clasa
5       include 'clase/class.exemplu.php';
6
7       // Cream o instanta a clasei
8       $obiect = new exemplu('Distrus de noi');
9
10      // Cream alt obiect
11      $obiect2 = new exemplu('OOP');
```

```
12
13    unset($obiect); // distrugem primul obiect creat
14
15    echo '<hr/>Dupa se termina executia scriptului.<hr/>';
16    ?>
```

Output **/test-clasa.php**:



https://programare.tech/blog/articole/php-orientat-spre-obiecte-pentru-incepatori

PHP ORIENTAT SPRE OBIECTE, PENTRU INCEPATORI

2012-05-14 | Marian Acasa | PHP - Incepatori

Deoarece incepatorilor le este greu sa inteleaga conceptul de orientare spre obiecte o sa fac un tutorial in care o sa explic doar cateva lucruri de baza, astfel inceptatori sa nu se complice, urmand sa scriu un tutorial pentru avansati mai incolo.

Ca sa va dati seama la ce se refera programarea pe obiecte gandti-va la o masina,aveti acces in ea doar prin portiere, o porniti doar cu o cheie, o accelerati si o franati doar prin pedale, asta este si conceptul poo, sa creati o clasa iar datele membre sa le puteti modifica doar prin intermediul unei functii din interiorul clase(pedale,cotact,portiera).

Cel mai bine o sa intelegeti oop daca o sa va dau un exemplu clar de clasa. O clasa se defineste prin cuvantul cheie php "class" urmat de numele functiei, variabilele membre ale clase se declara prin cuvantul cheie "var" (in php 5 nu mai este nevoie), metodele (functiile) se declara prin cuvantul cheie

"function", atat variabilele cat si metodele pot fi ***private,publice sau protejate***. Daca sunt publice ele pot fi accesate in exteriorul unei clase daca sunt private pot fi accesate doar in interiorul clase. ***Referirea la o data membra a clasei,*** metoda, variabila se face prin pointerul **$this->nume** , dar sa nu mai lungim vorba si sa scriem cod.

```
1.
2.  class masina{
3.   //datele membre ale clasei
4.        private $motor;
5.        private $putere;
6.        private $masina;
7.
8.   //constructorul clasei
9.    public function masina(){
10.
11.       $this->motor=1.0;
12.       $this->putere='20cp';
13.       $this->masina='golf';
14. }
15. //metoda ce schimba tipul motorului
16. public function modifica_motor($motor_){
17.         $this->motor=$motor_;
18.
19. }
20. //metoda de schimbare a puteri motorului
21. public function modifica_putere($putere_){
22.        $this->putere=$putere_;
23.
24. }
25. //metoda de schimbare a tipului masini
26. public function modifica_masina($masina_){
27.       $this->masina=$masina_;
28. }
29.
30. //metoda afisare a masini,motorului si puterea
31. public function toString(){
32.        echo 'Masina: '.$this->masina.'
33.                <br> Motor: '.$this->motor.'
34.                <br> Putere:'.$this->putere.'<br>';
35. }
36.}
```

Am construit **clasa masina**, ce este de mentionat pentru fiecare clasa este o functie speciala numita ***constructor***, care poarta numele clasei sau incepand cu versiunea 5 ***__construct().*** Metoda constructor se apeleaza in timpul definiri obiectului clasei.

```
1.
2. //declaram obiectul clasei
3. $ob=new masina();
4. //afisam masina dupa ce am apelat constructorul
5. $ob->toString();
6. //setam o noua masina
7. $ob->modifica_masina("Passat");
8. //setam un nou motor
9. $ob->modifica_motor("2.5");
10.//setam o noua putere pentru masina
11.$ob->modifica_putere("180cp");
12.//afisam noua masina
13.$ob->toString();
```

Rezultat:

```
1.
2. Masina: golf
3. Motor: 1
4. Putere:20cp
5.
6. Masina: Passat
7. Motor: 2.5
8. Putere:180cp
```

Dupa cum se observa mai sus obiectul se declara prin cuvantul cheie "**new**" urmat de numele clasei. ca sa apelam o metoda din clasa **folosim obiectul urmat de operatorul "->" urmat de numele metodei**.
Daca o sa incercati sa modificati sau sa afisati o data membra care are tipul "privat" o sa constatati ca primiti eroare.
Cam atat despre oop in acest tutorial, urmeaza mostenirea, tratarea exceptiilor si altele, pana atunci bafta

http://www.php1984.com/2011/12/10/poo-in-php-introducere/
POO IN PHP – INTRODUCERE
Voi incepe introducerea POO in PHP cu un mic exemplu. POO inseamna programare orientata pe obiecte. Pentru a lucra cu obiecte avem nevoie de clase.

Clasele sunt alcatuite din proprietati si metode. **O proprietate este o variabila, iar o metoda este o functie in interiorul clasei.**
Ex: Implementare clasa Phone care are 3 proprietati: brand, model, price (pret).

```php
<?php

class Phone {

    protected $brand;
    protected $model;
    protected $price;

    public function __construct($_brand, $_model, $_price) {
        $this->brand = $_brand;
        $this->model = $_model;
        $this->price = $_price
    }

    // Metode get,
    // aceste metode sunt folosite pentru a afisa o propietate
    public getBrand(){
      return $this->brand;
    }

    public getModel(){
      return $this->model;
    }

    public getPrice(){
      return $this->price;
    }

    // Metode set,
    // aceste metodae sunt folosite pentru a asigna o valoare pentru o proprietate.

    public setBrand($_brand){
      $this->brand = $_brand;
    }

    public setModel($_model){
      $this->model = $_model;
    }

```

```
41    public setPrice($_price){
42      $this->price = $_price;
43    }
44
45  }
46
    ?>
```

Ex2: Cream un obiect de tip Phone.

```
1   <?php
2
3   $my_phone = new Phone();
4
5   echo $my_phone->getBrand();
6   echo '<br />';
7   echo $my_phone->getModel();
8   echo '<br />';
9   echo $my_phone->getPrice();
10  echo '<br />';
11
12
13  //schimbam proprietatile obiectului
14  $my_phone->setBrand('Samsung');
15  $my_phone->setModel('Galaxy S2');
16  $my_phone->setPrice('300 euro');
17
18
19
20  //afisam din nou toate proprietatile.
21  echo $my_phone->getBrand();
22  echo '<br />';
23  echo $my_phone->getModel();
24  echo '<br />';
25  echo $my_phone->getPrice();
26  echo '<br />';
27
28  ?>
```

Sfaturi
==========

1. implementarea clasei se salveaza intr-un fisier separat si se va include in fisierul php unde vom avea nevoie.
2. Afisearea proprietatilor se poate pune intr-o functie pentru a nu duplica codul.

PS: Urmeaza si partea a 2-a .

EXEMPLU DE UTILIZARE $this->

EXEMPLU CE-I $this->????

```
1.
2.  <?php
3.  class A{
4.      private $a = 1;
5.
6.      public function getA()
7.  {
8.          return $this->a;
9.  }
10.
11.     public function someMethod()
12.{
13.         return $this->getA();
14.}
    }
    // TERMINARE DESCRIERE CLASA
15.$obj = new A();
16.echo $obj->someMethod();
1.
    class Dog {
2.    private $age;
3.    private $name;
4.
5.    public function __construct($name = '', $age = 0){
6.      $this->age = $age;
7.      $this->name = $name;
8.    }
9.    public function getName(){
10.     return $this->name;
11.   }
12.
13.   public function getAge(){
14.     return $this->age;
15.   }
16.}
17.
```

```
18.$sharik = new Dog('Шарик', 3);
19.$pusya = new Dog('Пуся', 5);
20.
21.$sharic->getName();
22.$pusya->getName();
```

$sharic->getName() вернёт "Шарик"
$pusya->getName() вернёт "Пуся"

для шарика $this - это шарик, для пуси $this - это пуся.
означает текущий объект, через который можно внутри обратиться к имени и возрасту.

```
<?php
class produs
{
		public $denumireProdus = "Cactus";
		public $numeProducator = "Flora SA";
		public $dataProducerii = "12/12/2014";
				public $num=5;
		public $pretUnitar = 90;
				public $zero=0;
				public $one=1;
				Public $two=2;
				public $array = array("foo", "bar", "hallo", "world");
function afiseazaDate()
{
return "{$this->denumireProdus}, {$this->numeProducator}, {$this-
>dataProducerii}, {$this->pretUnitar}";
}
function square($num)
{
   return $num * $num;
}
function small_numbers()
{
 return $this->array;
}
}
$produs1 = new produs();
$produs1->denumireProdus = 'Begonie';
$produs1->numeProducator = 'Plante SA';
$produs1->dataProducerii = '01/01/2015';
$produs1->pretUnitar = 70;
$produs1->num=6;
```

```
// accesarea metodei
echo "Descriere produs: <br>".$produs1->afiseazaDate()."</br><br />";
echo $produs1->square($produs1->num);
echo "<br>";
// utilizare foreachi
foreach ($produs1->array as $key => $value)
{
// $arr[3] будет перезаписываться значениями $arr при каждой итерации
цикла
    echo "{$key} => {$value} "."<br/>";
 //   print_r($array);
}
echo "<br>";
print_r($produs1->array);
echo "<br/>";
list ($zero, $one, $two) = $produs1->array;
// Составить список всех переменных
echo "<br>"."Drink is $zero and $one makes it special $two."."<br/>";
// Составить список только некоторых из них
list($zero, , $two) = $produs1->array;
echo "<br>"."$zero has $two."."<br/>";
// Или только третья
list( , , $two) = $produs1->array;
echo "<br>"."I need $two!";
?>
```

REZULTATELE

Descriere produs:
Begonie, Plante SA, 01/01/2015, 70

36
0 => foo
1 => bar
2 => hallo
3 => world

Array ([0] => foo [1] => bar [2] => hallo [3] => world)

Drink is foo and bar makes it special hallo.

foo has hallo.

I need hallo!

Exemplu de clasa

```
<?php
class a_1
{
   public $nume;
   private $Private_name;
   protected $Protected_name;
   public function salutare()
   {
      echo "<br>".$this->nume."Salut!</br>";
   }
   public function pocasel($a)
   {
      $this->nume = $a;
      echo "<br>".$this->nume."Pocasel!</br>";
   }
}
?>
```

Care desi se cheama din fisier de tip html
Fisierul de tip html trebuie sa fie memorizat ca fisier php..

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
<HEAD>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>CLASA</title>
 </head>
 <body>
<?php
include_once 'classes/class.a_1.php';
var_dump(include_once 'classes/class.a_1.php');
$config = include_once('classes/class.a_1.php');
var_dump($config);
print_r($config);
$obj=new a_1;
$obj->salutare();
$obj->nume = "Misca";
```

```
$obj->salutare();
$obj->pocasel("Chistol");
$obj->salutare();
echo "trece";
?>
</body>
</html>
```

REZULTATELE

bool(true) bool(true) 1
Salut!

MiscaSalut!

ChistolPocasel!

ChistolSalut!
trece
DIN SCRIPTUL CENTRAL NAME=2
Apel prin instanta ->Din interiorul clasei, din metoda ei Nume = 2

CLASA. LUCRUL CU CONSTRUCTORUL

```
<?
class a_2
{
   private $nume;

   function __construct( $a="sa vedem cum lucreaza clasa cu constructorul" )
   {
      $this->nume = $a;
   }
   function Salut()
   {
      echo "<H1>".$this->nume."! Salut!</H1>";
   }
}
?>

<html>
<head>
   <title>Clasa cu constructor!</title>
</head>
<body>

<?
```

```
include_once "classes/class.a_2.php";
$incl=include_once "classes/class.a_2.php";
var_dump($incl);
print_r($incl);
$obj0 = new a_2();
$obj1 = new a_2("Ionica");
$obj2 = new a_2("Valiusa");
$obj0->Salut();
$obj1->Salut();
$obj2->Salut();
?>

</body>
</html>
```

REZULTATUL

```
bool(true) 1
sa vedem cum lucreaza clasa cu constructorul! Salut!
Ionica! Salut!
Valiusa! Salut!
```

Сложив все, изложенное выше, можно создать более осмысленный класс. Например, класс, который будет располагать данные в виде таблицы с поименованными столбцами.

```
<html>
<head>
   <title>Utilizare clasa Tabel</title>
</head>
<body>

<?php
include_once "classes/class.tabel.php";
$incl=include_once "classes/class.tabel.php";
var_dump($incl);
print_r($incl);
$test = new Tabel (array("a","b","c"));
$test->addRow(array("a"=>1,"b"=>3,"c"=>2));
$test->addRow(array("b"=>1,"a"=>3));
$test->addRow(array("c"=>1,"b"=>3,"a"=>4));
$test->output();
?>

</body>
</html>

<?php
```

```
class Tabel
{
   private $headers =array();
   private $data = array();
   function Tabel ( $headers )
   {
      $this->headers = $headers;
   }
   function addRow ( $row )
   {
      $tmp = array();
      foreach ( $this->headers as $header )
      {
         if ( ! isset( $row[$header] )) $row[$header] = "";
         $tmp[] = $row[$header];
      }
      array_push ( $this->data, $tmp );
   }
   function output ()
   {
      echo "<PRE><B>";
      foreach ( $this->headers as $header ) echo "$header  ";
      echo "</B><BR>";
      foreach ( $this->data as $y )
      {
         foreach ( $y as $x ) echo "$x  ";
         echo "<BR>";
      }
      echo "</PRE>";
   }
}
?>
```

Rezultatul NUMAR__22.PHP

bool(true) 1

a	b	c
1	3	2
3	1	
4	3	1

EXEMPLE DE CLASE MOSTENITE NUMAR__23

```
<html>
<head>
   <title>CLASE MOSTENITE </title>
</head>
<body>
<?php
class a_3
{
```

```
   public $nume = "Ioana";
   function Salutare()
   {
      echo "<H1>".$this->nume."! Salutare!</H1>";
   }
}
class a_4 extends a_3
{
   function Salutare()
   {
      echo "<H1>".$this->nume."! OOOO...Ne-am intilnit!</H1>";
   }
}
$obj = new a_4();
$obj->Salutare();
?>
</body>
</html>
```

REZULTATUL

Ioana! OOOO...Ne-am intilnit!

Метод **Привет** переопределен для производного класса. Свойство **имя** наследуется от родительского.

Итак, производный класс может наследовать, переопределять и дополнять свойства и методы другого класса.

В следующем примере **NUMBER_24.PHP** создан класс **HTMLTable**, основанный на классе **Table** из примера 3. Новый класс формирует данные, сохраненные методом **addRow** родительского класса, и выводит их в HTML-таблицу. Свойства **$cellpadding** и **$bgcolor** дают возможность изменять соответствующие аргументы, при этом переменной **$cellpadding** присваивается значение по умолчанию, равное 2.

```
<html>
<head>
   <title>Clase Tabele si HTMLTabele</title>
</head>
<body>

<?php

include_once "classes/class.tabele.php";
```

```
include_once "classes/class.tabele.php";
$incl=include_once "classes/class.tabele.php";
var_dump($incl);
print_r($incl);

$test = new HTMLTabele ( array("a","b","c"), "#00FFFF" );
$test->setCellpadding ( 5 );
$test->addRow(array("a"=>1,"b"=>3,"c"=>2));
$test->addRow(array("b"=>1,"a"=>3));
$test->addRow(array("c"=>1,"b"=>3,"a"=>4));
$test->output();

?>
</body>
</html>

<?php
class Tabele
{
   public $headers = array();
   public $data = array();
   function Tabele( $headers )
   {
                        $this->headers = $headers;
                       }
   function addRow ( $row )
   {
      $tmp = array();
      foreach ( $this->headers as $header )
      {
         if ( ! isset( $row[$header] )) $row[$header] = "";
         $tmp[] = $row[$header];
      }
      array_push ( $this->data, $tmp );
   }
   function output ()
   {
      echo "<PRE><B>";
      foreach ( $this->headers as $header ) echo "$header  ";
      echo "</B><BR>";
      foreach ( $this->data as $y )
   {
         foreach ( $y as $x ) echo "$x  ";
         echo "<BR>";
   }
      echo "</PRE>";
   }
}
class HTMLTabele extends Tabele
{
   public $cellpadding = "7";
   public $bgcolor;
```

```
    function HTMLTabele ( $headers, $bg="#CDCDCD" )
    {
       Tabele::Tabele( $headers );
       $this->bgcolor = $bg;
    }
    function setCellpadding ( $padding )
    {
       $this->cellpadding = $padding;
    }
    function output ()
    {
       echo "<table cellpadding='".$this->cellpadding."'><tr>";
       foreach ( $this->headers as $header )
          echo "<th bgcolor='".$this->bgcolor."'>".$header;
       foreach ( $this->data as $y )
       {
          echo "<tr>";
          foreach ( $y as $x )
             echo "<td bgcolor='".$this->bgcolor."'>$x";
       }
       echo "</table>";
    }
}
?>
```

REZULTATELE

bool(true) 1

a	b	c
1	3	2
3	1	
4	3	1