WEB-programozás II (1. előadás)
OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

A félévközi tanulmányi követelmények:

- Egy 40 pontos elméleti zárthelyi dolgozat a 11. héten előadáson.
- Két 30 pontos zárthelyi gyakorlati dolgozat a 6. és a 12. héten a laborokban.
- Pótlási lehetőség a 13. héten előadáson (elméleti) és laborokban (gyakorlati).

Értékelés:

A sikeres félévhez szükséges:

- Az elméleti dolgozatban legalább 20 pont elérése.
- A gyakorlati dolgozatokban legalább 30 pont elérése.

A gyakorlati jegy a dolgozatokban és az otthonra kiadott feladatban elért pontok összege alapján, a TVSz-nek megfelelően kerül meghatározásra.

WEB-programozás II (1. előadás)
OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Osztályok PHP5-ben: (Az User osztály definíciója)

```
<?php
/*** User.php: az User osztály definíciója ***/

class User {
   private $name;
   private $birthday;

   public function __construct($name, $birthday) {
     $this->name = $name;
     $this->birthday = $birthday;
   }

   public function hello( ) {
     return "Szia ".$this->name."!";
   }

   public function goodbye( ) {
     return "Viszlát ".$this->name."!";
   }

   public function age() {
     $ts = strtotime($this->birthday);
     if(! $ts || $ts === -1) {
       return "Ismeretlen";
     }
     else {
       return date("Y", time( ) - $ts) - 1970;
     }
   }
   public function out( ) {
     echo "<b>Név:</b> ".$this->name." <b>Szül.:</b> ".
          $this->birthday;
   }

   public static function comp($a, $b) {
     return $a->name > $b->name;
   }

   public static function reverseComp($a, $b) {
     return $a->name < $b->name;
   }
}
?>
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Az User osztály tesztelése:

A tesztelő php szkript:

```
<?php
/*** UserTest.php: az User osztály tesztelése ***/
include_once "User.php";

$user = new User("Nagy László", "1986-02-10");
echo $user->hello()."<br />";
echo "Te vagy ".$user->age()." éves.<br />";
echo $user->goodbye()."<br />";
echo "<br />";
$user->out();
?>
```

A szkript végrehajtásának az eredménye:
(Végrehajtás időpontja: 2019.09.09)

Szia Nagy László!
Te vagy 33 éves.
Viszlát Nagy László!

Név: Nagy László **Szül.:** 1986-02-10

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Objektumok:

A felszínen az objektumok nagyjából úgy festenek, mint egy társításos tömb (asszociatív tömb), amihez rajta műveleteket végző függvények gyűjteménye tartozik. Ugyanakkor rendelkeznek néhány további fontos tulajdonsággal, mégpedig a következőkkel:

- **Öröklés** - Az öröklés annak képessége, hogy már meglevő osztályokból új osztályokat származtathatunk, és örökölhetjük vagy felülírhatjuk azok tulajdonságait és tagfüggvényeit.

- **Egységbe zárás** - Az egységbe zárás (betokozás, enkapszuláció) annak képessége, hogy elrejthetjük az adatokat az osztály felhasználói elől.

- **Többalakúság** (polimorfizmus) - Amikor két osztály ugyanazokat a külső tagfüggvényeket valósítja meg, felcserélhetőnek kell lenniük a függvényekben, mindig a megfelelő tagfüggvény kerül végrehajtásra.

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Öröklés PHP 5-ben: (A RegisteredUser osztály definíciója)

```
<?php
/*** RegisteredUser.php: a RegisteredUser osztály definíciója ***/

 include_once "User.php";

 class RegisteredUser extends User {

   private $shortname;
   private $password;

   public function __construct($name, $birthday,
                               $shortname, $password) {
     parent::__construct($name, $birthday);
     $this->shortname = $shortname;
     $this->password = sha1($password);
   }

   public function authenticate($suppliedPassword) {
     if($this->password ===
         sha1($suppliedPassword)) {
       return true;
     }
     else {
       return false;
     }
   }

   public function out() {
     parent::out();
     echo " <b>Rövid név:</b> ".
        $this->shortname." <b>Kódolt jelszó:</b> ".
        $this->password;
   }
}
?>
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

A RegisteredUser osztály tesztelése:

A tesztelő php szkript:

```
<?php
/*** RegisteredUserTest.php: a RegisteredUser osztály tesztelése ***/

include_once "RegisteredUser.php";
$registeredUser = new RegisteredUser("Nagy László", "1986-02-10",
"nlaszlo", "teszt");
echo $registeredUser->hello()."<br />";
if($registeredUser->authenticate("teszt")) {
  echo "Te vagy ".$registeredUser->age()." éves.<br />";
}
else {
  echo "Rossz jelszó!<br />";
}
if($registeredUser->authenticate("más")) {
  echo "Te vagy ".$registeredUser->age()." éves.<br />";
}
else {
  echo "Rossz jelszó!<br />";
}
echo $registeredUser->goodbye()."<br />";
echo "<br />";
$registeredUser->out();
?>
```

A szkript végrehajtásának az eredméneye:

(Végrehajtás időpontja: 2010.02.17)

Szia Nagy László!
Te vagy 24 éves.
Rossz jelszó!
Viszlát Nagy László!

Név: Nagy László **Szül.:** 1986-02-10 **Rövid név:** nlaszlo
Kódolt jelszó:
34228a532093278fcdc65c3a1338482e8bdc44f6

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Egységbe zárás

A PHP5 a **public** (nyilvános), **protected** (védett) és **private** (privát) tagváltozók és tagfüggvények bevezetésével adatrejtési képességeket ad a nyelvhez. Ezt már alkalmaztuk korábbi példáinkon. Ezekre általában a *PPP* néven hivatkoznak. Szabványos jelentésük pedig a következő:

- **Public** - A nyilvános tagváltozókat és tagfüggvényeket az osztályt felhasználó bármilyen kód közvetlenül elérheti.

- **Protected** - A védett tagváltozók és tagfüggvények nem érhetők el közvetlenül az osztály felhasználói által, csak egy, az osztálytól öröklő alosztályon belül.

- **Private** - A privát tagváltozók és tagfüggvények csak azon az osztályon belül hozzáférhetők, amelyben meghatározták őket. Ez azt jelenti, hogy az osztályt bővítő gyermekekből nem hívhatók meg.

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Statikus tulajdonságok és tagfüggvények

A PHP-ben a tagfüggvényeket és tagváltozókat statikusként is bevezethetjük:

- A ***statikus tagfüggvények*** egy osztályhoz kötődnek, nem pedig annak egy példányához (vagyis egy objektumhoz), ezért osztálymetódusoknak is nevezik őket. Meghívásuk az ***OsztályNév: : tagfüggvény*()** formában történik. A statikus tagfüggvényekben a **$this** nem érhető el.

- A ***statikus tagváltozók*** az osztályhoz, és nem annak egy példányához kapcsolódó osztályváltozók. Ez azt jelenti, hogy módosításuk hatással van az osztály valamennyi példányára. A statikus tagváltozókat a **static** kulcsszóval vezetjük be, és az ***OsztályNév: : $tagváltozó*** formában érjük el.

Az osztálynév helyett a **self** és a **parent** kulcsszavakat is használhatjuk a statikus tagfüggvények és tagváltozók elérésére, amelyek az aktuális osztályra, illetve annak szülőjére mutatnak .

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

```php
<?php
/*** Users.php: az Users osztály definíciója ***/
  include_once "User.php";
  include_once "RegisteredUser.php";

  class Users {

    private static $counterGroups = 0;
    private static $userGroups = Array();

    private $counter = 0;
    private $users = Array();
    private $groupName;

    public static function groupsNumber() {
      return self::$counterGroups;
    }
    public static function getGroupByIndex($i) {
      return self::$userGroups[$i];
    }
    public static function listGroups() {
      for($i = 0; $i < self::$counterGroups; $i++) {
        echo self::$userGroups[$i]->groupName."<br />";
      }
    }
    public function __construct($groupName) {
      self::$userGroups[self::$counterGroups++] = $this;
      $this->groupName = $groupName;
    }
    public function getGroupName() {
      return $this->groupName;
    }
    public function addUser($user) {
      $this->users[$this->counter++] = $user;
    }
    public function orderUsers() {
      usort($this->users, Array("User","comp"));
    }
    public function reverseOrderUsers() {
      usort($this->users, Array("User","reverseComp"));
    }
    public function listUsers() {
      for($i = 0; $i < $this->counter; $i++) {
        $this->users[$i]->out();
        echo " <b>Osztály:</b> ".get_class($this->users[$i])."<br />";
      }
    }

  }
?>
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

```php
<?php
/*** UsersTest.php: az Users osztály tesztelése ***/
  include_once "Users.php";

$group = new Users("Első csoport");
$user1 = new User("Elso 1", "1970-10-11");
$user2 = new RegisteredUser("Masodik 1", "1981-02-12", "Mas",
"Mas01");
$user3 = new User("Harmadik 1", "1987-10-10");
$user4 = new RegisteredUser("Negyedik 1", "1975-08-09", "Negy",
"Negy01");
$group->addUser($user1);
$group->addUser($user2);
$group->addUser($user3);
$group->addUser($user4);

$group = new Users("Második csoport");
$user1 = new User("Elso 2", "1970-10-11");
$user2 = new RegisteredUser("Masodik 2", "1969-10-15", "Mas",
"Mas02");
$user3 = new User("Harmadik 2", "1687-18-18");
$user4 = new RegisteredUser("Negyedik 2", "1978-12-19", "Negy",
"Negy02");
$user5 = new RegisteredUser("Ötödik 2", "1958-12-13", "Öt", "Öt02");
$group->addUser($user1);
$group->addUser($user2);
$group->addUser($user3);
$group->addUser($user4);
$group->addUser($user5);

echo "<b>Csoportok:</b><br /><br />";
Users::listGroups();
echo "<br /><br />";

for($i = 0; $i < Users::groupsNumber(); $i++) {
  $group = Users::getGroupByIndex($i);
  echo "<b>".$group->getGroupName().":</b>";
  echo "<br /><br />";
  $group->listUsers();
  echo "<br /><br />";
  echo "<b>Rendezett (Asc) ".
        $group->getGroupName().":</b>";
  echo "<br /><br />";
  $group->orderUsers();
  $group->listUsers();
  echo "<br /><br />";
  echo "<b>Rendezett (Desc) ".
         $group->getGroupName().":</b>";
  echo "<br /><br />";
  $group->reverseOrderUsers();
  $group->listUsers();
  echo "<br /><br />";
}
?>
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Az Users osztály tesztelésének az eredménye:

Csoportok:

Első csoport
Második csoport

Első csoport:

Név: Elso 1 **Szül.:** 1970-10-11 **Osztály:** User
Név: Masodik 1 **Szül.:** 1981-02-12 **Rövid név:**Mas **Kódolt jelszó:** 369e24f319d833084acd3f64f13ca63e5b7c74f8 **Osztály:** RegisteredUser
Név: Harmadik 1 **Szül.:** 1987-10-10 **Osztály:** User
Név: Negyedik 1 **Szül.:** 1975-08-09 **Rövid név:** Negy **Kódolt jelszó:** 2f611b88f81715b4cfedc549ca000763dd8943b1 **Osztály:** RegisteredUser

Rendezett (Asc) Első csoport:

Név: Elso 1 **Szül.:** 1970-10-11 **Osztály:** User
Név: Harmadik 1 **Szül.:** 1987-10-10 **Osztály:** User
Név: Masodik 1 **Szül.:** 1981-02-12 **Rövid név:** Mas **Kódolt jelszó:** 369e24f319d833084acd3f64f13ca63e5b7c74f8 **Osztály:** RegisteredUser
Név: Negyedik 1 **Szül.:** 1975-08-09 **Rövid név:** Negy **Kódolt jelszó:** 2f611b88f81715b4cfedc549ca000763dd8943b1 **Osztály:** RegisteredUser

Rendezett (Desc) Első csoport:

Név: Negyedik 1 **Szül.:** 1975-08-09 **Rövid név:** Negy **Kódolt jelszó:** 2f611b88f81715b4cfedc549ca000763dd8943b1 **Osztály:** RegisteredUser
Név: Masodik 1 **Szül.:** 1981-02-12 **Rövid név:** Mas **Kódolt jelszó:** 369e24f319d833084acd3f64f13ca63e5b7c74f8 **Osztály:** RegisteredUser
Név: Harmadik 1 **Szül.:** 1987-10-10 **Osztály:** User
Név: Elso 1 **Szül.:** 1970-10-11 **Osztály:** User

Második csoport:

Név: Elso 2 **Szül.:** 1970-10-11 **Osztály:** User
Név: Masodik 2 **Szül.:** 1969-10-15 **Rövid név:** Mas **Kódolt jelszó:** 9036d00c5efcebbeb472e4c57d0fac36222ae1e0 **Osztály:** RegisteredUser
Név: Harmadik 2 **Szül.:** 1687-18-18 **Osztály:** User
Név: Negyedik 2 **Szül.:** 1978-12-19 **Rövid név:** Negy **Kódolt jelszó:** 92edb067579d5fab76d6b7e238f19f4d8133c04e **Osztály:** RegisteredUser
Név: Ötödik 2 **Szül.:** 1958-12-13 **Rövid név:** Öt **Kódolt jelszó:** 1f3d3c3fee168e8b854404f611b84e2dda010755 **Osztály:** RegisteredUser

Rendezett (Asc) Második csoport:

Név: Elso 2 **Szül.:** 1970-10-11 **Osztály:** User
Név: Harmadik 2 **Szül.:** 1687-18-18 **Osztály:** User
Név: Masodik 2 **Szül.:** 1969-10-15 **Rövid név:** Mas **Kódolt jelszó:** 9036d00c5efcebbeb472e4c57d0fac36222ae1e0 **Osztály:** RegisteredUser
Név: Negyedik 2 **Szül.:** 1978-12-19 **Rövid név:** Negy **Kódolt jelszó:** 92edb067579d5fab76d6b7e238f19f4d8133c04e **Osztály:** RegisteredUser
Név: Ötödik 2 **Szül.:** 1958-12-13 **Rövid név:** Öt **Kódolt jelszó:** 1f3d3c3fee168e8b854404f611b84e2dda010755 **Osztály:** RegisteredUser

Rendezett (Desc) Második csoport:

Név: Ötödik 2 **Szül.:** 1958-12-13 **Rövid név:** Öt **Kódolt jelszó:** 1f3d3c3fee168e8b854404f611b84e2dda010755 **Osztály:** RegisteredUser
Név: Negyedik 2 **Szül.:** 1978-12-19 **Rövid név:** Negy **Kódolt jelszó:** 92edb067579d5fab76d6b7e238f19f4d8133c04e **Osztály:** RegisteredUser
Név: Masodik 2 **Szül.:** 1969-10-15 **Rövid név:** Mas **Kódolt jelszó:** 9036d00c5efcebbeb472e4c57d0fac36222ae1e0 **Osztály:** RegisteredUser
Név: Harmadik 2 **Szül.:** 1687-18-18 **Osztály:** User
Név: Elso 2 **Szül.:** 1970-10-11 **Osztály:** User

Típusspecifikáció

```
public function addUser(User $user) {
    $this->users[$this->counter++] = $user;
}
```

Ilyenkor az **addUser** metódus hívása egy olyan paraméterrel, amely nem felel meg a specifikált típusnak szintaktikus hibát eredményez.

A típusspecifikáció a PHP5-től nem csak tagfüggvényekben lehetséges, hanem bármelyik függvényben is.

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

- **get_class** beépített függvény: visszaadja a paraméterként kapott objektum osztályának a nevét

- **instanceof** beépített operator: visszaad egy logikai értéket, amely kifejezi, hogy az első operandusként megadott objektum a második operandusként megadott osztálynak egy példánya vagy sem.

Példák:

Legyen a $a tartalma az User osztály egy példánya, akkor:

- **get_class**($a) eredménye: ’User’

- $a **instanceof** User eredménye: igaz (true)

- $a **instanceof** Users eredménye: hamis (false)

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Értékadás

A PHP5-ben az objektumok létrehozásakor egy leírót (handler) kapunk az objektumhoz, ami fogalmilag megegyezik a C++ hivatkozásaival (reference). Ha a következő kódot a PHP5-ben hajtjuk végre, az objektumból csak egy példány keletkezik, másolatok nem.

```
<?php
    $firstObject = new MyClass();
    $secondObject = $firstObject;
?>
```

$secondObject és $firstObject ugyanaz az objektum, az egyikükön végzett módosítások érvényesek lesznek a másikon is. Ha azt szeretnénk, hogy $secondObject ténylegesen a $firstObject objektum egy másolata legyen, akkor használni kell a **clone** utasítást (beépített metódust):

```
<?php
    $firstObject = new MyClass();
    $secondObject = clone $firstObject;
?>
```

$secondObject egy új objektum, amely a $firstObject pontos másolata, az egyikükön végzett módosítások nem lesznek érvényesek lesznek.

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

A __clone metódus

Egyes osztályok esetében a mélymásolást végző beépített __clone () tagfüggvény nem biztos, hogy megfelel a céljainknak, ezért a PHP megengedi annak felülbírálását.

```
<?php
class MyObject {
        public $firstVar = 10;
        public $secondVar = 20;
        function __clone() {
                $this->secondVar = 0;
        }
}

$firstObject = new MyObject();
$secondObject = clone $firstObject;

var_dump($firstObject);
var_dump($secondObject);
?>
```

A szkript kimenete:

```
object(myObject)#1 (2) {
["firstVar"]=>
int(10)
["secondVar"]=>
int(20)
}
object(myObject)#2 (2) {
["firstVar"]=>
int(10)
["firstVar"]=>
int(0)
}
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Ha a következő példában a definiált TestClass osztályban az alapértelmezett __clone () tagfüggvényt használnánk, az azonosító (az id tulajdonság) is lemásolódna, ezért inkább írjuk át az osztályt, valahogy így:

```
class TestClass {
    public static $counter = 0;
    public $id;
    public $other = "other";
    public function  __construct() {
      $this->id = self::$counter++;
    }
    public function __clone() {
      $this->other = $this->other. "modified";
      $this->id = self::$counter++;
    }
}
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Konstruktőrök és destruktőrök:

```
<?php
class SimpleClass {
  function __construct($param) {
        echo "Created a new instance of SimpleClass!";
  }
  function __destruct() {
        echo "Destroyed this instance of SimpleClass";
  }
}

$myInstance = new SimpleClass("value");
unset($myInstance);
?>
```

A konstruktőrök hasznosak a tagváltozók kezdőértékének a megadására. A konstruktőrök és destruktőrök együttes használata nagyon hasznos sok más esetben. Egy klasszikus példa egy adatbázist kezelő osztály, amikor a konstruktőr létrehozza az adatbázis kapcsolatot, és a destruktőr lezárja azt. Hasonló példa lehet egy fájlt kezelő osztály.

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Osztály konstansok:

```
<?php
    class ConstExample {

      private $myvar;
      public $readme;
      const MY_CONSTANT = 10;

      public function showConstant() {
            echo "The value is: ".self::MY_CONSTANT;
      }
    }

    $inst = new ConstExample;
    $inst->showConstant();
    echo "The value is: ".ConstExample::MY_CONSTANT;
    ?>
```

A fenti példában figyeljük meg hogyan elérhető az osztály konstans az osztályon belül, ill. az osztályon kívül.

Az osztály konstansok örökölhetők a szülő osztályoktól, és felülírhatók.

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Absztrakt osztályok és absztrakt metódusok:

Az absztrakt osztályok olyan szuperosztályok, amelyek definiálják tőle származtatott osztályok közös tulajdonságait. Az absztrakt osztályok nem példányosíthatók, azaz nem lehet létrehozni objektumokat egy absztrakt osztályból.

```
<?php
        abstract class Number {
                private $value;
                abstract public function value();
                public function reset() {
                        $this->value = NULL;
                }
        }

        class Integer extends Number {
                private $value;
                public function value() {
                        return (int)$this->value;
                }
        }

        $num = new Integer; /* Rendben van */
        $num2 = new Number; /* Hibát ad */
?>
```

Egy absztrakt osztály leszármazottjának nem kell megvalósítania a szülő osztály minden absztrakt metódusát, de abban az esetben a gyerek osztálynak is absztraktnak kell lennie.

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Interfészek

Az interfészek biztosítják az osztály funkcionalitását. Az interfészek definiálnak metódusokat, amelyeket meg kell valósítaniuk minden olyan osztálynak, amely az interfészt valósítja meg.

```
<?php
        interface printable {
                public function printme();
        }
?>

<?php
        class Integer implements printable {
                private $value;

                public function value() {
                        return (int)$this->value;
                }

                public function printme() {
                        echo (int)$this->value;
                }
        }
?>
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

```
<?php
interface printable {
        public function printme();
}

abstract class Number {
        private $value;
        abstract public function value();
        public function reset() {
                $this->value = NULL;
        }
}

class Integer extends Number implements printable {
        private $value;
        function __construct($value) {
                $this->value = $value;
        }
        public function value() {
                return (int)$this->value;
        }
        public function printme() {
                echo (int)$this->value;
        }
}

/* A következő függvény paraméterének meg kell
valósítania a printable interfészt */

function printNumber(printable $myObject) {
        $myObject->printme();
}
$inst = new Integer(10);
printNumber($inst);
?>
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Több interfész megvalósítása:

```
<?php
interface printable {
        public function printme();
}

interface Inumber {
        public function reset();
}

class Integer implements printable, Inumber {
        private $value;
        function __construct($value) {
                $this->value = $value;
        }
        public function printme() {
                echo (int)$this->value;
        }
        public function reset() {
                $this->value = NULL;
        }
        public function value() {
                return (int)$this->value;
        }
}

function resetNumber(Inumber $obj) {
        $obj->reset();
}

function printNumber(printable $obj) {
        $obj->printme();
}

$inst = new Integer(10);
printNumber($inst);
resetNumber($inst);
?>
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

Final Osztályok és Metódusok

Egy final osztály olyan osztály, amelyből újabb osztályokat nem lehet leszármaztatni, azaz nem lehet egyetlen osztály szülő osztálya sem.

Egy final metódus olyan metódus, amelyet nem lehet felülírni a leszármaztatott osztályokban.

```
<?php
final class NoExtending {
      public function myFunction() {
        /* Function logic */
      }
}

class restrictedExtending {
      final public function anotherFunc() {
        /* Function logic */
      }
}

class myChild extends restrictedExtending {
      public function thirdFunction() {
        /* Function logic */
      }
}
?>
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

A PHP osztályai bizonyos tagfüggvénynek különleges visszahívható (callback) függvények, amelyek bizonyos eseményeket kezelnek. Már ismerjük a következőket: __construct, __destructor és __clone.

Az osztályok ezen kívül három további ilyen függvényt használnak: A __get, a __set és a __call, amelyek a tagváltozók és metódusok hívásának módját szabályozzák.

function __get($varname) { }

E tagfüggvény meghívására akkor kerül sor, amikor olvasás céljából próbálunk hozzáférni egy meghatározatlan tulajdonsághoz.

function __set($varname, $value) { }

E tagfüggvény meghívására akkor kerül sor, amikor írás céljából próbálunk hozzáférni egy meghatározatlan tulajdonsághoz.

A következő példában az osztály valamennyi tulajdonságának értékét egy statikus tömbben tárolja. Amikor írás vagy olvasás céljából hozzáférünk valamelyik tulajdonsághoz, a __get és a __set elérés kezelők ebben a statikus tömbben keresnek, nem pedig az objektum belső tulajdonságtáblájában.

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

```
class Singleton {
        private static $props = array();
        public function __construct () {}
        public function __get($name)
        {
                if (array_key_exists($name,  self::$props))
                {
                        return self::$props[$name];
                }
        }
        public function __set($name, $value) {
                self::$props[$name] = $value;
        }
}

$a = new Singleton;
$b = new Singleton;
$a->property = "hello world";
print $b->property;
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

__call metódus:

A PHP a __call () metóduson keresztül a tagfüggvény-túlterhelést is támogatja. Ez azt jelenti, hogy ha meghívjuk egy objektum valamelyik tagfüggvényét és az nem létezik, helyette a __call () hívódik meg. A szolgáltatást általában arra használjuk, hogy védekezzünk a meghatározatlan tagfüggvények ellen.

A következő példában egy osztály __call () metódusának egy olyan megvalósítását láthatjuk, amely egyszerűen kiírja a hívni próbált tagfüggvény nevét, illetve az osztálynak átadott argumentumokat:

```
class Test {
    public function __call($funcname, $args) {
        print "Nem létező ". $funcname.
              " metódus hívása.<br />";
        print "A hívás paraméterei:<br />";
        print_r($args);
    }
}

$obj = new Test;
$obj->hello("Gergely") ; // Nem létező metódus hívása
```

Kimenet:

```
Nem létező hello metódus hívása.
A hívás paraméterei:
Array ( [0] => Gergely )
```

WEB-programozás II (1. előadás)

OBJEKTUM ORIENTÁLT PROGRAMOZÁS PHP-BEN

A webes alkalmazások fejlesztése során sok PHP forrásokban szabadon használható osztálykönyvtár áll rendelkezésre. Példák:

• PHPExcel – Excel fájlok beolvasására és létrehozására használható osztálykönyvtár. (https://github.com/PHPOffice/PhpSpreadsheet)

• tcpdf – PDF formátumú fájlok létrehozására használható osztálykönyvtár. (https://github.com/tecnickcom/tcpdf)

• PHPMailer – E-mailek elküldésére használható osztálykönyvtár. (https://github.com/PHPMailer/PHPMailer)

Javasolt weboldal PHP osztálykönyvtárak keresésére:

http://www.phpclasses.org/