Web Development II

Department of Software and Computing Systems

Universitat d'Alacant
Universidad de Alicante

PHP Object Oriented
Classes and objects

Sergio Luján Mora
Jaume Aragonés Ferrero
Department of Software and Computing Systems

Contents

- Introduction
- Class definition
- Object definition
- Constructor and destructor
- Inheritance
- Visibility
- Abstract, Final, static
- More OO features

Introduction

- http://www.php.net/manual/en/language.oop.php
- http://www.php.net/manual/en/language.oop5.php
- PHP offers principal features of OO Paradigm to the programmer:
  - Encapsulation,
  - Simple inheritance,
  - Constructor and destructor methods
  - Member privacity (visibility)
  - Interfaces
  - Overloading
  - Abstraction
  - etc.

Defining classes

- How to declare a class:

```
class myClass
{
  public $attribute1, $attribute2, …;
  function __Construct($arg1, $arg2, …) {…}
  function Method1(…) {…}
  function Method2(…) {…}
  ...
}
```

- Class attributes must be declared explicitly depending on its visibility.
  - It is the unique case where an identifier is due to declare.
  - It is possible to use ‘var’ in order to maintain compatibility with PHP4.
- A class only can have one constructor

Defining objects

- Creating an object

```
$myObject = new myClass(…);
```

- How to access to class members, operator arrow: -> (hyphen, right angle bracket)
  - Outside the class: `$myObject->attribute/method`
  - Inside the class: `$this->attribute/method`
- Scope resolution operator ::
  - allows access to static, constant, and overridden members or methods of a class.
  - `Self::` allows to access to own static members and methods.
  - `parent::` allows to access to superclass static members and methods.

Constructor & destructor

- Class Constructor and Destructor have its own name:
  - Constructor: `__construct()`
  - Destructor: `__destruct()`
- In previous PHP 5 versions:
  - The constructor was a member method with the same name of its class.
  - There were not class destructors.
- Destructor will be invoked if:
  - All references of an object disappear.
  - An object is explicitly destroyed.

Example

```
<?
  class Person
  {
    var $name, $surname;

    function __Construct($n, $a)
    {
      $this->name = $n;
      $this->surname = $a;
    }
  }

    $p1 = new Person(‘Homer', ‘Simpson');
    $p2 = new Person(‘Peter', ‘Griffin');

    echo "$p1->surname, $p1->name<br />";
    echo "$p2->surname, $p2->name<br />";
?>
```

Inheritance

- Simple inheritance can be realised by means of `extends keyword`.
- Simple inheritance means a class can only inherit from just one superclass (or base class)
- ATTENTION: base class constructor is not called automatically from the subclass constructor. You must invoke it explicitly.
- Inheritance systeax:

```
class SubClass extends BaseClass
{
 ...
}
```

- It is possible to overwrite all base class methods, whenever they are not 'final'.
- We can access to base class methods or attributes using: `parent::`

Constructor & destructor: inheritance

- Child classes do not call base class constructor or destructor automatically
  → They have to be invoked explicitly:
  - `parent::__construct()`
  - `parent::__destruct()`

Inheritance Example

```
<?
   ...
  class Client extends Person
  {
    var $code;

    function __Construct($n, $a, $c)
    {
        parent::__Construct($n, $a);
        $this->code = $c;
    }
  }

  $c1 = new Client(‘Bender', ‘Rodriguez', 123);
  echo "$c1->surmane, $c1->name: $c1->code<br />";
?>
```


One more example

```
<?php
   class myClass {
      function __construct() {
          print “Parent class Constructor\n";
      }
   }

   class childClass extends myClass {
      function __construct() {
          parent::__construct();
          print “Child class Constructor\n";
      }
   }

   $oParent = new myClass();
   $oChild = new childClass();
?>
```

Visibility

- The visibility of a property or method can be defined by prefixing the declaration with the keywords: public, protected or private.
- There are three levels of visibility:
  - `public`: Public declared items can be accessed everywhere.
    `protected`: Protected limits access to inherited and parent classes (and to the class that defines the item).
  - `private`: Private limits visibility only to the class that defines the item.
- Class attributes have to be declared with their visibility.
  - If they are declared with ‘var’ they will be ‘public’.
- By default, methods declared without visibility will be ‘public’.

abstract classes and methods

- `abstract:`
  - It is not allowed to create an instance of a class that has been defined as abstract
  - Any class that contains at least one abstract method must also be abstract.
  - Methods defined as abstract simply declare the method's signature they cannot define the implementation.
  - When inheriting from an abstract class, all methods marked abstract in the parent's class declaration must be defined by the child

Final and Static

- `final:`
  - A class defined final cannot be extended.
  - Prevents child classes from overriding a method by prefixing the definition with final.
- `Static:`
  - Todo método estático puede ser invocado sin tener que instanciar un objeto de esa clase.
  - `$This` no está accesible desde un miembro estático.

More OO features

- Interface: allows to specify like a class templates.
- Method overloading: provides means to dynamically "create" members and methods
- Magic methods: You cannot have functions with these names in any of your classes unless you want the magic functionality associated with them.
- Object cloning: Creating a copy of an object with the same data and status.
  - `Clone` and `__clone`: spetial methods.
- Object comparison operator: ===

Exercises

- Define a class ‘cAuthor’ with three attributes: name, surname and country.
- Define a class ‘cBook’ with three attributes: title, number of pages and and object ‘cAuthor’.
  - Create all necessary methods: constructor, destructor, gets and sets.
  - Program each class in a single file.
- Code a new PHP script including previous files and declare an object ‘cBook’ with data and finally print its attribute values.