

UD-03a: Orientación a Objetos con PHP

Desarrollo Web en Entorno Servidor

Curso 2020/2021

Características de la POO

- La programación orientada a objetos (POO, o OOP en lenguaje inglés), es una metodología de programación basada en objetos, es decir, estructuras que contienen datos y un código que los maneja.
- La estructura de los objetos se define en las clases. En ellas se encuentra el código que define las características y el comportamiento de los objetos que formarán parte de dicha clase.
- Una clase es algo similar a un modelo, una estructura, una plantilla, un molde o una maqueta, que nos va a permitir generar objetos (instancias de la clase basadas en su estructura).
- Por tanto, un objeto es la realización de una clase cuando le damos valores concretos, valores definidos.

Características de la POO

- Entre los **miembros de una clase** puede haber:
 - Atributos o propiedades. Almacenan información acerca de las características del objeto al que pertenecen. Su valor puede ser distinto para cada uno de los objetos de la misma clase.
 - Métodos. Son los miembros de la clase que determinan las acciones (comportamiento) que puede desarrollar un objeto de esta clase. Un método es como una función. Puede recibir parámetros y devolver valores.
- Podremos decir ahora que un **objeto tiene** un estado definido, es decir, que tiene atributos con valores y unos comportamientos asociados o métodos ejecutables.

Características de la POO

- Los pilares fundamentales de la POO son:
 - **Herencia**. Es el proceso de crear una clase a partir de otra, heredando su comportamiento y características y pudiendo redefinirlos.
 - **Abstracción**. Hace referencia a que cada clase oculta en su interior las peculiaridades de su implementación y presenta al exterior una serie de métodos cuyo comportamiento está bien definido. Visto desde el exterior, cada objeto es un ente abstracto que realiza un trabajo.
 - **Polimorfismo**. Un mismo método puede tener comportamientos distintos en función del objeto con que se utilice y/o los parámetros que utilice.
 - **Encapsulación**. En la POO se juntan en un mismo lugar los datos y el código que los manipula.

Características de la POO

- Las ventajas más importantes que aporta la POO son:
 - **Modularidad**. Permite dividir los programas en partes o módulos más pequeños que son independientes unos de otros, aunque pueden comunicarse entre ellos.
 - **Extensibilidad**. Si se desean añadir nuevas características a una aplicación, la POO facilita esta tarea de dos formas: añadiendo nuevos métodos al código, o creando nuevos objetos que extiendan el comportamiento de los ya existentes.
 - **Mantenimiento**. Los programas desarrollados utilizando POO son más sencillos de mantener, debido a la modularidad. También ayuda seguir ciertas convenciones al escribirlos, por ejemplo, escribir cada clase en un fichero propio.

Orientación a objetos en PHP

Clases

Se declaran con `class`. Primero propiedades o atributos, y luego métodos.

```
class Producto
{
    private $codigo;
    public $nombre;
    public $PVP;

    public function muestra()
    {
        print "<p>" . $this->codigo . "</p>";
    }
}
```

*\$codigo es **private**.*
Esto implica que no se puede acceder a él desde fuera de esta clase.
No se puede mostrar directamente.

Es posible indicarles un valor por defecto en la declaración de la clase.

Orientación a objetos en PHP

Crear objetos y acceder a atributos

Para crear instancias de una clase (objetos) se utiliza la palabra ***new***:

```
$pr = new Producto();
```

Aunque claro, antes debemos haber declarado la clase.

```
require('producto.php');
```

Para acceder desde un objeto a sus atributos o métodos se usa el operador flecha:

```
$p->nombre = 'Samsung Galaxy S';  
$p->muestra();
```

Orientación a objetos en PHP

Archivo: persona.php

Modificadores de Acceso: public, private, protected.

<?php

```
class Persona
{
```

```
    public $nombre;
    public $apellido1;
    public $apellido2;
    public $fnac;
```

Empieza por __

```
    public function __construct($nombre, $apellido1, $apellido2, $fnac) {
        $this-> nombre = $nombre;
        $this-> apellido1 = $apellido1;
        $this-> apellido2 = $apellido2;
        $this-> fnac = $fnac;
    }
```

Un constructor es una función que crea una instancia de una clase. Es un mecanismo de inicialización de objetos. Si no ponemos un constructor, PHP genera uno por defecto.

```
    public function caminar(){
        echo "Estoy caminando";
    }
    public function saltar(){
        echo "Estoy saltando";
    }
```

Pasa una persona a formato cadena.

```
    public function __toString() {
        return "Nombre completo: " . $this->nombre . " " . $this->apellido1 . " " . $this->apellido2 . ", nacido/a el: " . $this->fnac . ".";
    }
```

?>

Orientación a objetos en PHP

Archivo: ejemplo1.php


<?php

```
require_once ("persona.php");
```

```
$antonio = new Persona("Antonio", "Perez", "Mas", "02-09-1973");  
$tomas = new Persona("Tomas", "Agullo", "Mastino", "22-05-1973");  
$luisa = new Persona("Luisa", "Garcia", "Influe", "13-01-1977");
```

```
echo $antonio . "<br>";  
echo $tomas . "<br>";  
echo $luisa . "<br>";
```

?>



Nombre:Antonio,Apellido1:Perez,Apellido2:Mas,FN:02-09-1973.
Nombre:Tomas,Apellido1:Agullo,Apellido2:Mastino,FN:22-05-1973.
Nombre:Luisa,Apellido1:Garcia,Apellido2:Influe,FN:13-01-1977.

Orientación a objetos en PHP

Al declarar un atributo, indicamos su visibilidad. Los principales niveles son:

- ❖ **public**. Hace que la variable/función se pueda acceder desde cualquier lugar, como por ejemplo otras clases y otras instancias de esa misma clase.
- ❖ **private**. Los atributos declarados como privados sólo pueden ser accedidos y modificados por los métodos definidos en la clase, no directamente por los objetos de la misma.
- ❖ **protected**. Hace que la variable/función se puede acceder desde la clase que las define y también desde cualquier otra clase que herede de ella.

```
class Producto
{
    private $codigo;

    public function muestra()
    {
        print "<p>" . $this->codigo . "</p>";
    }
}
```

Orientación a objetos en PHP

El objeto `$this`

Cuando desde un objeto se invoca un método de la clase, a éste se le pasa siempre una referencia al objeto que hizo la llamada. Esta referencia se almacena en la variable `$this`.

```
echo "<p>" . $this->codigo . "</p>";
```

Orientación a objetos en PHP

getters y setters básicos

```
private $codigo;

public function setCodigo($codigo)
{
    $this->codigo = $codigo;
}

public function getCodigo()
{
    return $this->codigo;
}
```

Orientación a objetos en PHP

Constantes de clase

En una clase también se pueden definir constantes. Usaremos la palabra *const*.

Características de las constantes:

- No pueden cambiar su valor
- No usan el carácter \$
- Está asociado a la clase. Es decir, no existe una copia del mismo en cada objeto. Por tanto, para acceder a las constantes de una clase, se debe utilizar el nombre de la clase y el operador ::, llamado operador de resolución de ámbito (se usa para acceder a elementos de la clase).

```
class Cuenta
{
    const USUARIO = 'dwes';
    ...
}

echo Cuenta::USUARIO;
```

Orientación a objetos en PHP

Atributos estáticos

Los atributos estáticos, también llamados de clase, se usan para guardar información general sobre la misma, como el número de objetos que se han instanciado.

Cuando se declara un objeto, no se realiza una copia de la variable estática, sino que todas las instancias de la clase comparten la misma variable estática.

```
class Autobus {  
    public static $precio = 1.20;  
}
```

La forma de acceder a ellas es igual que con las constantes.

```
if(Autobus::$precio < 1.50){  
    //todo bien  
}
```

Orientación a objetos en PHP

El método estático es un método que pertenece a la clase y no al objeto.

Sólo puede llamar a otros métodos estáticos.

Se puede acceder directamente por el nombre de la clase y no necesita ningún objeto.

```
Producto::nuevoProducto();
```

Si el método o atributo es público, se debe acceder con el nombre de la clase y el operador de resolución de ámbito.

Si es privado, sólo se podrá acceder a él desde los métodos de la propia clase, utilizando la palabra `self`. De la misma forma que `$this` hace referencia al objeto actual, `self` hace referencia a la clase actual.

```
self::$num_productos++;
```

Orientación a objetos en PHP

Constructor

Puedes definir métodos constructores que se ejecutan cuando se crea el objeto. El constructor de una clase debe llamarse **__construct** (se trata de otro método mágico). Se pueden utilizar, por ejemplo, para asignar valores a atributos.

```
class Producto
{
    private static $num_productos = 0;
    private $codigo;

    public function __construct()
    {
        self::$num_productos++;
    }
    ...
}
```


Orientación a objetos en PHP

El constructor de una clase puede tener parámetros, que deberán pasarse cuando se crea el objeto. Sin embargo, sólo puede haber un constructor en cada clase, dado que el lenguaje no soporta la sobrecarga de métodos.

```
class Producto
{
    private static $num_productos = 0;
    private $codigo;
    public function __construct($codigo)
    {
        $this->codigo = $codigo;
        self::$num_productos++;
    }
}

$p = new Producto('GALAXYS');
```

Orientación a objetos en PHP

También es posible definir un **destructor** y debe llamarse `__destruct` (otro método mágico). Permite definir acciones que se ejecutarán cuando se elimine el objeto.

```
class Producto
{
    private static $num_productos = 0;
    private $codigo;

    public function __construct()
    {
        self::$num_productos++;
    }

    public function __destruct()
    {
        self::$num_productos--;
    }
}
```

EJERCICIOS



- ❖ Crea una clase llamada **Contacto** con los siguientes atributos: **id**, **nombre** y **teléfono**. Debes implementar el **constructor**, los **getters**, los **setters** y el **método de conversión a cadena** que devuelva la concatenación del nombre con el teléfono entre paréntesis. Guarda esta clase en un fichero llamado **Contacto.php**. Finalmente, crea una página llamada **ud03ej01.php** que pruebe todas las funcionalidades.
- ❖ Haciendo uso de la clase Contacto, crea una clase llamada **Agenda** que contenga un *array* de objetos *Contacto*. Esta nueva clase debe tener un método para **añadir** un contacto a la Agenda (recibiendo como argumento el objeto Contacto) y otro para **eliminar** un contacto de la misma (recibiendo como argumento sólo su id). También debes crear un método para convertir el objeto Agenda a cadena, de manera que en HTML se muestre cada contacto de la agenda en cada fila de una tabla. Guarda esta clase en un fichero llamado **Agenda.php**. Finalmente, crea una página llamada **ud03ej02.php**. En ella crea una agenda. Añade tres contactos. Muestra el segundo de ellos. Después, elimínalo y muestra la agenda.

(Contacto: 002 Ángel 965222222)

idRegistro	Nombre	Teléfono
001	Raunak	965111111
003	Jesús	965333333

← Tu página debe mostrar algo como esto.

Se pueden recorrer las propiedades de un objeto de datos usando foreach. Lo siguiente recorre las propiedades del empleado del mes.

```
<?php
$edm = $compañía->empleadoDelMes;
foreach ($edm as $nombre => $valor) {
    echo "$nombre: $valor\n";
}
?>
```

Para imprimir la agenda como una tabla puede que necesites recorrer los atributos de cada contacto.

lo que imprimirá:

nombre: Jane Doe

NS: E0003