Orientación a objetos en PHP

En php es posible la programación orientada a objetos; ello conlleva la definición de clases, interfaces,…y el resto de los elementos y características propias de este tipo de programación.

La programación orientada a objetos favorece la modularización y el encapsulamiento, lo cual favorece la reutilización de código y facilita el mantenimiento de una aplicación.

1. Clases en PHP.

Una clase está formada por propiedades (variables) y métodos (funciones) y constantes.

El nombre de una clase debe comenzar con una letra o un guión bajo y una serie letras, número y guiones.

Ej:

```php
<?php

class Alumno {
    const CICLO="2º DAW";
    private $nom;
    private $edad;
    function __construct ($nombre,$e)
    {
        $this->nom=$nombre;
        $this->edad=$e;
    }
    function getNombre()
    {
        return $this->nom;
    }
    function setNombre($nombre)
    {
        $this->nom=$nombre;
    }
    function getEdad()
    {
        return $this->edad;
    }
    function setEdad ($e)
    {
        $this->edad->$e;
    }
    function __toString()
    {
        return "Alumno:".$this->nom." ".$this->edad;
    }
} //fin de la clase Alumno

//crear un alumno
$alum=new Alumno("Ana","20");
// mostrar los datos del alumno
echo $alum;


?>
```

$this es utilizable cuando ha sido instanciada una clase (creado un objeto de la clase); con $this se puede hacer referencia al objeto dentro de los métodos.

New se utiliza para instanciar una clase, (como en Java); si se ha definido un método constructor, se utilizará el método constructor para la creación e inicialización del objeto (en el ejemplo, la función ***__construct()***). También se

puede crear un método destructor para que finalizado el script en el cual se definieron los objetos, dejen de existir. (__***destruct()***). La ejecución de este método es automática al finalizar la ejecución del script.

Antes de instanciar una clase, ésta debe ser definida. Con **instanceof** es posible saber si un objeto es una instancia de una clase (ej: var_dump($alum instanceof Alumno);).

Para el manejo de las clases en php existen **multitud de funciones predefinidas**. Una de ellas es __**toString()** que permite establecer cómo visualizar un objeto de una clase tratado como un string.

Actividad 1: Crea una clase con:
- los datos de un alumno (matrícula, Nombre y Apellidos, calificación del módulo).
- Método para instanciar la clase.
- Método para visualizar sus datos si está aprobado.

2. Herencia de clases.

Al igual que en Java se utiliza *extends* para crear nuevas clases que derivan de otras, heredando sus métodos y características y añadiendo otros nuevos o sobreescribirlos. (si al sobreescribirlos interesa mantener también el método de la clase padre, incluir *parent::nombre_metodo_clasepadre*)

Por ejemplo,

class becado extends alumno

{

private $importebeca;

}, de tal forma que becado tendrá además de las características y métodos de alumno, un atributo nuevo, $importebeca.

Siempre definir primero la clase padre y después la hija.

PHP admite herencia múltiple (por ejemplo, *class clase3 extends clase2 extends clase1*)

Actividad 2: Agrega la nueva clase a tu código y un nuevo constructor.

3. Modificadores

Los modificadores posibles en php, tanto de los métodos como de los atributos, son:

- Public: en este caso el método o el atributo puede ser utilizado fuera de la clase. (*por defecto; en las funciones no suele ponerse pero en los atributos sí. Lo normal es que sean public los métodos).*
- Private: sólo puede ser utilizado dentro de la clase.(*lo normal es que sean privados los atributos*)
- Protected: por la clase y las clases descendientes. (*Se definen los atributos y los métodos de la clase padre para que puedan ser accesibles por la clase hija*).

Los modificadores posibles en php, tanto para las clases como para los métodos, son:

- Final: un método no podría ser sobreescrito por clases descendentes.
- Abstract: las clases o los métodos no pueden ser utilizados si no son previamente heredados.

4. Propiedades y métodos estáticos.

Cada vez que se instancia una clase (se crea un objeto), se reserva espacio para sus propiedades y sus métodos (definiendo con ellos el comportamiento de cada objeto); es decir, cada objeto tiene una copia de las propiedades y de los métodos definidos en la clase.

Si fuesen **estáticos**, se considera que son propiedades o métodos de la clase y los objetos no tienen una copia de ellos; en este caso son "compartidos" entre los objetos. Para definirlos hay que utilizar **static** antes del nombre del método o variable.

Para acceder a un método o a una variable (propiedad) estática dentro de la propia clase, hay que utilizar el operador **self::** (*self::nombremetod o nombreatributo*); y para utilizarlo fuera de la propia clase, **nombre_clase::** (*nombreclase::nombremétodo o nombre atributo*).

Una propiedad o método estático es **por naturaleza público y no pertenece al objeto** sino a la clase y por ello puede ser utilizado en cualquier script. Utilizando el modificador privado en su definición se impide su acceso fuera de la clase.

Actividad 3: Siguiendo con el ejemplo de nuestra clase alumno añadir una variable (propiedad) estática denominada tasa y darle el valor importe del sobre de matrícula (2 euros). Comprobar su acceso.

Actividad 4: Siguiendo con el ejemplo de nuestra clase alumno crear un método estático que calcule el importe de la matrícula, teniendo en cuenta que si es realizada a partir del 1-06-2020, el importe será 0 y sino será igual a la tasa de la matrícula. Comprobar su acceso.

Para un ejemplo de método estático, sería el cálculo del importe de matrícula, en el cual se utilizar el atributo estático $tasa. Por ejemplo:

Static importematricula()

{ if (date("m-d-y")>"01-06-20") $importe=0 else $importe=self::$tasa;}

5. Arrays y objetos.

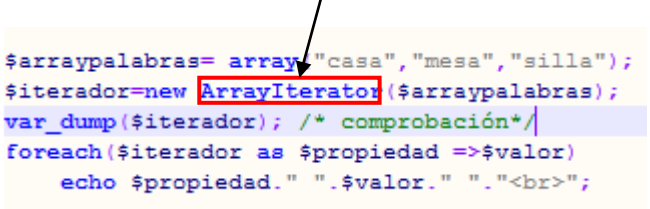
- **Un array de objetos.**

```php
//crear un array de alumnos

$arrayAlumnos=array();
$arrayAlumnos[0]=new Alumno("Maria", 19,8.5);
$arrayAlumnos[1]=new Alumno("David",21,7.3);
for ($i=0; $i<2;$i++)
{
    echo $arrayAlumnos[$i]->getCURSO();
    echo $arrayAlumnos[$i]->getNombre();
    echo $arrayAlumnos[$i]->getEdad();
    echo $arrayAlumnos[$i]->getCali();
```

- **Un array tratado como objeto**

En muchas ocasiones, tal es el caso del acceso a una BD, será necesario tratar un array (fila o filas de una tabla relacional) como un objeto. Para ello, PHP dispone de la **librería SPL** (Standard PHP Library, https://www.php.net/manual/es/book.spl.php).



```php
$arraypalabras= array("casa","mesa","silla");
$iterador=new ArrayIterator($arraypalabras);
var_dump($iterador); /* comprobación*/
foreach($iterador as $propiedad =>$valor)
    echo $propiedad." ".$valor." "."<br>";
```