MATERIA: TECNOLOGÍAS Y APLICACIONES WEB

PROFESOR: M.S.I. MARIO HUMBERTO RODRÍGUEZ CHÁVEZ

ALUMNO: MIGUEL ANGEL RUIZ GARCÍA

PROGRAMACIÓN ORIENTADA A OBJETOS EN PHP

PROGRAMACIÓN ORIENTADA A OBJETOS EN PHP

La POO es una técnica de programación que utiliza objetos e interacciones en el diseño de un sistema. Este paradigma tiene se compone de 4 pilares, y diferentes características.

CARACTERÍSTICAS CONCEPTUALES DE LA POO

Abstracción: Aisla el objeto del contexto, lo que permite definir las características esenciales del mismo.
Encapsulamiento: Reúne los elementos que puedan pertenecer a una entidad.
Modularidad: Permite dividir la aplicación en partes más pequeñas, independientes entre sí.
Ocultación: Aisla los objetos del exterior, y no permite modificar el contenido de ellos a menos que tengan los permisos.
Polimorfismo: Que métodos con el mismo nombre puedan realizar distintas cosas.
Herencia: Relación entre dos o más clases, dónde existe una principal (clase Padre), que esta hereda sus atributos y métodos a las clases secundarias (clases Hijas).
Recolección de basura: Técnica que destruye todos aquellos objetos que no son necesarios.

CLASES O CLASES CONCRETAS

Una clase es un modelo que se usa para crear objetos un mismo comportamiento, estado e identidad.

Objeto
Es una entidad que tiene atributos, valores concretos y propiedades, todas estas basadas en una clase que les da esa forma.

Método
Es un algoritmo asociado a un objeto, que le permite realizar algo.

Propiedades y atributos
Contienen datos asociados a un objeto.

POO BÁSICA

Para crear una clase se pone la palabra reservada “class” seguido del nombre de la clase, luego llaves y todo lo que este dentro de estas es el contenido de la clase.

La clase está construida principalmente por 3 bloques:
- Constantes
- Variables
- Métodos

```php
<?php

class clase{
    // BLOQUE DE CONSTANTES
    const CONSTANTE = "Una constante";

    // BLOQUE DE VARIABLES
    public $variable = "Un variable";

    // BLOQUE DE MÉTODOS
    public function metodo(){
        echo $this->variable;
    }
}
```

Para instanciar un objeto de una clase, primero se crea una variable a la cual se le da la propiedades de la clase poniendo “new” seguido del nombre de la clase;

```php
$clase1 = new clase();
$clase1->variable = "Variable modificada";
$clase1->metodo();
```

Un objeto es moldeado a una clase, cuando se le asigna la clase obtiene sus características, y métodos.

Herencia

La herencia permite pasar los métodos y características de una clase a otra. Y se hace de la siguiente forma.

```php
<?php

class Padre{
    #...
}

class Hijo extends Padre{
    /*
        Esta clase hereda todos los metodos
        y propiedades de "Padre"
    */
}
```

Clases Abstractas

Una clase abstracta sirve para crear una clase genérica, donde los métodos que incluye puedan ser rehechos de distintas formas dependiendo de las necesidades de las clases que hereden. Una de las principales diferencias es que no se pueden instanciar, sólo las clases que hereden de esta.

```php
abstract class claseAbstracta{
    #...
}
```

Clases finales

PHP desde su versión 5.1 implementa las clases finales, estas clases no pueden heredarse. Y se definen anteponiendo “final”.

```php
<?php

final class Automovil{

}

//Fatal error:
//Class Camion may not inherit from final class (Automovil)
class Camion extends Automovil{

}
```

Niveles de acceso

En la POO existen varios niveles de acceso los cuales son los siguientes:

-Nivel Público: Se usa con la palabra reservada “public” antes de crear la variable/método. Cualquiera puede acceder a esta variable o método.

-Nivel privado: Se usa con la palabra reservada “privado” antes de crear la variable/método. Nadie más que la propia clase puede acceder a estas características.

-Nivel protegido: Se usa con la palabra reservada “public” antes de crear la variable/método. Sólo las clases que hereden de esta, y ella misma, pueden acceder a estas características.

Métodos

Todo lo mencionado anteriormente se puede aplicar con los métodos. Los cuales son funciones pero que pertenecen a una clase, y solo pueden ser invocados mediante dicha clase.

Pero a diferencia de las variables, con los métodos también se pueden hacer abstractos, como con la clase completa, pero el efecto que tiene en los métodos es que sólo se declara el nombre y lo que puede recibir, no hay nada de implementación. Esa se hará en la clase que herede este método.

Ejemplo:

Clase Padre

```php
abstract class servicio
{
    abstract public function saludar($nombre);

    public function carta($nombre, $almuerzo)
    {
        echo $this->saludar($nombre);
        echo "Le puedo ofrecer: <br/>";

        foreach ($almuerzo as $clave => $valor) {
            echo $clave.": ".$valor."<br/>";
        }
    }

}
```

Clase Hija:

```php
class mesero extends servicio
{
    public function saludar($nombre)
    {
        return "Buenas tardes señor(a): ".$nombre."<br/>";
    }

}
```

Métodos mágicos en PHP
PHP incluye unos autodenominados “métodos mágicos”, los cuales tienen funcionalidades ya predefinidas por PHP, las cuales pueden facilitar diversas tareas y de esta forma ahorrarnos código.

Entre los mencionados métodos mágicos, se destacan los siguientes:

__construct()
Este método lo que hace es realizar una función de manera automática al momento de crear la instancia. La función principal es realizar algún proceso que la clase requiera antes de que sea usada.

__destruct()
Este método es muy similar al anterior, sólo que este realiza alguna acción antes de que el objeto sea destruido. La acción realizada será especificada por el programador.

Otros métodos mágicos
PHP nos ofrece otros métodos mágicos tales como __call, __callStatic, __get, __set,__isset, __unset, __sleep, __wakeup, __toString, __invoke, __set_state y__clone.

Interfaces
Las interfaces son conjuntos de métodos abstractos y de constantes cuya funcionalidad es la de determinar el funcionamientos de una clase, es decir, funciona como una plantilla. Al ser métodos abstractos estos no tienen funcionalidad solo se definen su tipos, argumentos y tipo de retorno.

```php
<?php

interface Filtro{
  public function filtrar($elem);
}

class Filtro_Vocales implements Filtro{
  public function filtrar($cadena){
    return preg_replace('/[aeiou]/', '', $cadena);
  }
}

$a = new Filtro_Vocales();
echo $a->filtrar('Dinosaurio');

?>
```