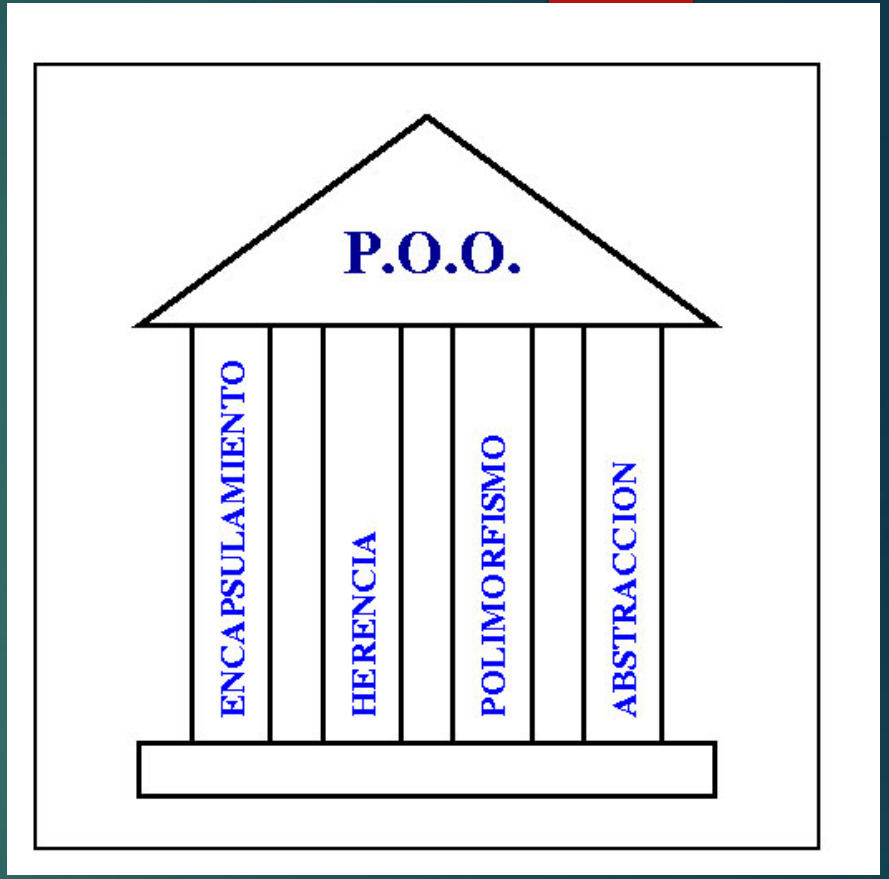
PROGRAMACIÓN ORIENTADA A OBJETOS EN PHP

VALERIA ALEXANDRA ESQUIVEL DOMÍNGUEZ

MATRICULA: 1530205

- La POO es un paradigma de programación (o técnica de programación) que utiliza objetos e interacciones en el diseño de un sistema.
- se compone de 4 pilares y diferentes características



CARACTERÍSTICAS CONCEPTUALES DE LA POO

- **Abstracción**

Aislación de un elemento de su contexto. Define las características esenciales de un objeto.

- **Encapsulamiento**

Reúne al mismo nivel de abstracción, a todos los elementos que puedan considerarse pertenecientes a una misma entidad.

- **Modularidad**

Característica que permite dividir una aplicación en varias partes más pequeñas (denominadas módulos), independientes unas de otras.

- **Ocultación (aislamiento)**

Los objetos están aislados del exterior, protegiendo a sus propiedades para no ser modificadas por aquellos que no tengan derecho a acceder a las mismas.

- **Polimorfismo**

Es la capacidad que da a diferentes objetos, la posibilidad de contar con métodos, propiedades y atributos de igual nombre, sin que los de un objeto interfieran con el de otro.

- **Herencia**

Es la relación existente entre dos o más clases, donde una es la principal (padre) y otras son secundarias y dependen (heredan) de ellas (clases "hijas"), donde a la vez, los objetos heredan las características de los objetos de los cuales heredan.

- **Recolección de basura**

Es la técnica que consiste en destruir aquellos objetos cuando ya no son necesarios, liberándolos de la memoria.

ELEMENTOS PRINCIPALES DE POO

CLASES:

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

OBJETOS:
Es una entidad provista de métodos o mensajes a los cuales responde (comportamiento); atributos con valores concretos (estado); y propiedades (identidad).

```php
<?php

class   Persona   {
#   Propiedades
#   Métodos
}

?>
```

```php
$persona = new Persona();
/*
El objeto, ahora, es $persona, que se ha
creado siguiendo el modelo de la clase Persona
*/
```

MÉTODOS:

Es el algoritmo asociado a un objeto que indica la capacidad de lo que éste puede hacer.

```
class Persona {
# Propiedades
# Métodos

    function caminar() {
        #...Código a ejecutar
    }

}
```

PROPIEDADES Y ATRIBUTOS:
Las propiedades y atributos, son variables que contienen datos asociados a un objeto.

```
class Persona {
# Propiedades
# Métodos

    public $nombre;
    public $edad;
    public $altura;

    function caminar() {
        #...Código a ejecutar
    }

}
```

EJEMPLO DE CÓDIGO IMPERATIVO O ESPAGUETI VS POO

```php
<?php

$automovil1 = array(
    'marca' => 'Toyota',
    'modelo' => 'Corolla'
    );
$automovil2 = array(
    'marca' => 'Volkswagen',
    'modelo' => 'Jetta'
    );

function imprimir($automovil){
    echo "<p>Hola! soy un $automovil[marca] modelo $automovil[modelo]</p>";
}

imprimir($automovil1);
imprimir($automovil2);

?>
```

CÓDIGO IMPERATIVO

```php
<?php
class Automovil
{
    public $marca;
    public $modelo;

    public function imprimir() {
        echo "<p>Hola! soy un $this->marca modelo $this->modelo</p>";
    }
}

$a = new Automovil();
$a->marca = "Toyota";
$a->modelo = "Corolla";
$a->imprimir();

$b = new Automovil();
$b->marca = 'Volkswagen';
$b->modelo = 'Jetta';
$b->imprimir();
```

POO

HERENCIA DE CLASES

- Los objetos pueden heredar propiedades y métodos de otros objetos. Para ello, PHP permite la "extensión" (herencia) de clases, cuya característica representa la relación existente entre diferentes objetos. Para definir una clase como extensión de una clase "padre" se utiliza la palabra clave extends.

```php
<?php

class clasePadre {
#...

}
class claseHija extends clasePadre {

/* esta clase hereda todos los métodos y propiedades de
la clase madre NombreDeMiClaseMadre
*/

}

?>
```

Clases abstractas

- Las clases abstractas son aquellas que no necesitan ser instanciadas pero sin embargo, serán heredadas en algún momento. Se definen anteponiendo la palabra clave abstract:

```php
<?php

abstract class claseAbstracta {
#...
}

?>
```

Este tipo de clases, será la que contenga métodos abstractos y generalmente, su finalidad, es la de declarar clases "genéricas" que necesitan ser declaradas pero a las cuales, no se puede otorgar una definición precisa (No se pueden instanciar), de eso, se encargarán las clases que la hereden).

Clases finales

- Las clases finales no pueden ser heredadas por otra. Se definen anteponiendo la palabra clave final.

```php
<?php

final class claseFinal {
#esta clase no podrá ser heredada
}

?>
```

- Esto pasaría si se hace una herencia de una clase final:

```php
<?php

final class Automovil{

}

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

}
```

OBJETOS E INSTANCIAS

- Para instanciar una clase, solo es necesario utilizar la palabra clave new. El objeto será creado, asignando esta instancia a una variable (la cual, adoptará la forma de objeto).

```php
<?php

# declaro la clase
class Persona {
#...
}
# creo el objeto instanciando la clase
$persona = new Persona();


?>
```

ATRIBUTOS O PROPIEDADES

- Las propiedades representan ciertas características del objeto en sí mismo. Se definen anteponiendo la palabra clave var al nombre de la variable (propiedad). No es necesario utilizar la palabra reservada var para la definición de la variable, pues PHP la reconoce por defecto:

```php
<?php

class Persona {
    var $nombre;
    var $edad;
    var $genero;
}


?>
```

- Las propiedades pueden gozar de diferentes características, como por ejemplo, la visibilidad: pueden ser públicas, privadas o protegidas.

Propiedades públicas:
Las propiedades públicas se definen anteponiendo la palabra clave public al nombre de la variable.
Éstas, pueden ser accedidas desde cualquier parte de la aplicación, sin restricción.

```
 8
 9  class Persona {
10      public $nombre;
11      public $genero;
12  }
```

Propiedades privadas:
Las propiedades privadas se definen anteponiendo la palabra clave private al nombre de la variable.
Éstas solo pueden ser accedidas por la clase que las definió.

```
14  class Persona {
15      public $nombre;
16      public $genero;
17      private $edad;
18  }
```

Propiedades protegidas:
Las propiedades protegidas pueden ser accedidas por la propia clase que la definió, así como por las clases que la heredan, pero no, desde otras partes de la aplicación. Éstas, se definen anteponiendo la palabra clave protected al nombre de la variable:

```
20  class Persona {
21      public $nombre;
22      public $genero;
23      private $edad;
24      protected $pasaporte;
25  }
```

Propiedades estáticas:
Las propiedades estáticas representan una característica de "variabilidad" de sus datos, de gran importancia en PHP. Una propiedad declarada como estática, puede ser accedida sin necesidad de instanciar un objeto y su valor es estático Ésta, se define anteponiendo la palabra clave static al nombre de la variable:

```
class Persona{
    public $nombre = "Yhoan";//Valor por defecto Yhoan
    public $apellido;
    public static $tipoSangre = 'A+';
```

ACCEDIENDO A LAS PROPIEDADES DE UN OBJETO

- Se accede a una propiedad no estática dentro de la clase, utilizando la pseudo-variable *$this*. Cuando la variable es estática se accede a ella mediante el operador de resolución de ámbito., doble dos puntos (::) anteponiendo la palabara clave *self* o *parent*

```php
class Persona{
    public $nombre = "Yhoan";//Valor por defecto Yhoan
    public $apellido;
    public static $tipoSangre = 'A+';

    public function hola()
    {
        echo $this->nombre."<br>";
        echo self::$tipoSangre."<br>";

        //Esto seria para la clase que extiende
        // echo parent::$tipoSangre."<br>";

    }
}
```

Acceso a variables desde el exterior de la clase

- Se accede a una propiedad no estática con la siguiente sintaxis:
  $objeto->variable

```php
// creo el objeto instanciando la clase
$p2 = new Persona();
// accedo a la variable NO estática
$p2->nombre = "Andres";
echo $p2->nombre."<br>";
```

CONSTANTES:

- A diferencia de las propiedades estáticas que pueden ser declaradas dentro de una clase, las constantes solo pueden tener una visibilidad pública y no deben ser creadas dentro de las clases. El acceso a constantes es exactamente igual que al de otras propiedades.

```php
<?php

//Estos valores no puedes ser modificados

define('MENSAJE','<h1>Bienvenidos al Curso</h1>');
echo MENSAJE;

const MI_CONSTANTE = 'Este es el valor estatico de mi constante';
echo MI_CONSTANTE;

?>
```

MÉTODOS PHP

- La única diferencia entre método y función, es que llamamos método a las funciones de una clase (en la POO), mientras que llamamos funciones, a los algoritmos de la programación estructurada.
- La forma de declarar un método es:

```php
<?php

# declaro la clase

class Persona {

    #propiedades
    #métodos
    function donar_sangre($tipoSangre) {
        #...
    }
}


?>
```

Nota: Al igual que cualquier otra función en PHP, los métodos recibirán los parámetros necesarios indicando aquellos requeridos, dentro de los paréntesis, también los métodos pueden ser públicos, privados, protegidos o estáticos

MÉTODOS MÁGICOS EN PHP

- El Método Mágico __construct():
- El método __construct() es aquel que será invocado de manera automática, al instanciar un objeto. Su función es la de ejecutar cualquier inicialización que el objeto necesite antes de ser utilizado.

```php
<?php
# declaro la clase

class Producto {
    #defino algunas propiedades
    public $nombre;
    public $precio;
    protected $estado;

    #defino el método set_estado_producto()
    protected function setEstado($estado) {
        $this->estado = $estado;
    }

    public function ver()
    {
        echo "Hola soy el producto: ".$this->nombre."<br/>";
        echo "Tengo un precio de: ".$this->precio."<br/>";
        echo "Me encuentro con un estado de: ".$this->estado."<br/>";
    }

    # constructor de la clase
    function __construct() {
        $this->setEstado('en uso');
    }
}

$_producto = new Producto();
$_producto->nombre = "Leche Colanta";
$_producto->precio = 2000;
$_producto->ver();

?>
```

- El método mágico __destruct():
- El método __destruct() es el encargado de liberar de la memoria, al objeto cuando ya no es referenciado. Se puede aprovechar este método, para realizar otras tareas que se estimen necesarias al momento de destruir un objeto.

```php
<?php
# declaro la clase

class Producto {
    #defino algunas propiedades
    public $nombre;
    public $precio;
    protected $estado;

    #defino el método set_estado_producto()
    protected function setEstado($estado) {
        $this->estado = $estado;
    }

    public function ver()
    {
        echo "Hola soy el producto: ".$this->nombre."<br/>";
        echo "Tengo un precio de: ".$this->precio."<br/>";
        echo "Me encuentro con un estado de: ".$this->estado."<br/>";
    }

    # constructor de la clase
    function __construct() {
        $this->setEstado('en uso');
    }

    # destructor de la clase
    function __destruct() {
        $this->setEstado('liberado');
        echo "El objeto ha sido destruido <br/>";
        echo "El objeto ha sido: ".$this->estado."<br/>";
    }
}

$_producto = new Producto();
$_producto->nombre = "Leche Colanta";
$_producto->precio = 2000;
$_producto->ver();


?>
```

INTERFACES

- Una interfaz es un conjunto de métodos abstractos y de constantes cuya funcionalidad es la de determinar el funcionamiento de una clase, es decir, funciona como un molde o como una plantilla. Al ser sus métodos abstractos estos no tiene funcionalidad alguna, sólo se definen su tipo, argumento y tipo de retorno.
- Para implementar una interface es necesario que la clase que quiera hacer uso de sus métodos utilice la palabra reservada implements. La clase que la implemente, de igual modo debe sobrescribir los métodos y añadir su funcionalidad

La construcción de una Interfaz es la siguiente:

```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');

?>
```