- **Clase 3:** 22/08/2016

Programación orientada a Objetos (POO)

La POO está compuesta por una serie de elementos que se detallan a continuación.

Clase

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

```
classPersona {
	# Propiedades
	# Métodos
}
```

Objeto

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

```
/*
El objeto, ahora, es $persona1, que se ha creado siguiendo el modelo
de la clase Persona
*/
$persona1 = newPersona();
```

Método

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

```
functioncaminar() {
	#...
}
```

Propiedades y atributos

Las propiedades y atributos, son variables que contienen datos asociados a un objeto.

```
$nombre ='Juan';
$edad ='25 años';
$altura ='1,75 mts';
```

Características conceptuales de la POO

La POO debe guardar ciertas características que la identifican y diferencian de otros paradigmas de programación. Dichas características se describen a continuación.

- 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 (madre) 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.

POO en PHP 5

Definición de Clases

La definición básica de clases comienza con la palabra clave *class*, seguido por un nombre de clase, continuado por un par de llaves que encierran las definiciones de las propiedades y métodos pertenecientes a la clase. El nombre de clase puede ser cualquier etiqueta válida que no sea una palabra reservada de PHP. Un nombre válido de clase comienza con una letra o un guion bajo, seguido de la cantidad de letras, números o guiones bajos que sea.

```
Class NombreDeMiClase {
        #...
}
```

Reglas de Estilo sugeridas:

Utilizar *CamelCasepara* el nombre de las clases. La llave de apertura en la misma línea que el nombre de la clase, permite una mejor legibilidad del código.

Declaración de Clases abstractas

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

```
abstract class NombreDeMiClaseAbstracta {
        #...
}
```

Este tipo de clases, será la que contenga métodos abstractos (que veremos más adelante) 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 (de eso, se encargarán las clases que la hereden).

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 extención de una clase "madre" se utiliza la palabra clave extends.

```
Class NombreDeMiClaseMadre {
        #...
}

Class NombreDeMiClaseHija extends NombreDeMiClaseMadre {
        /* esta clase hereda todos los métodos y propiedades de
        la clase madre NombreDeMiClaseMadre
        */
```

}

Declaración de Clases finales En PHP

PHP 5 incorpora clases finales que no pueden ser heredadas por otra. Se definen anteponiendo la palabra clave final.

```
final class NombreDeMiClaseFinal {
        #esta clase no podrá ser heredada
}
```

Objetos en PHP 5

Una vez que las clases han sido declaradas, será necesario crear los objetos y utilizarlos, aunque hemos visto que algunas clases, como las clases abstractas son solo modelos para otras, y por lo tanto no necesitan instanciar al objeto.

Instanciar una clase

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). Lógicamente, la clase debe haber sido declarada antes de ser instanciada, como se muestra a continuación:

```
# declaro la clase
Class Persona {
       #...
}

# creo el objeto instanciando la clase
$persona= newPersona();
```

Reglas de Estilo sugeridas

Utilizar nombres de variables (objetos) descriptivos, siempre en letra minúscula, separando palabras por guiones bajos. Por ejemplo si el nombre de la clase es NombreDeMiClase como variable utilizar $nombre_de_mi_clase. Esto permitirá una mayor legibilidad del código.

Propiedades en PHP 5

Las propiedades representan ciertas características del objeto en sí mismo.

```
Class Persona {
       public $nombre;
       public $edad;
       public $genero;
}
```

Las propiedades pueden gozar de diferentes características, como por ejemplo, la visibilidad: pueden ser públicas, privadas o protegidas. Como veremos más adelante, la visibilidad de las propiedades, es aplicable también a la visibilidad de los métodos.

- 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.

```
Class Persona {
```

```
        public $nombre;
        public $genero;
}
```

- 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.

```
Class Persona {
        public $nombre;
        public $genero;
        private $edad;
}
```

- 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:

```
Class Persona {
        public $nombre;
        public $genero;
        private $edad;
        protected $pasaporte;
}
```

- Propiedades estáticas

Las propiedades estáticas representan una característica de “variabilidad” de sus datos, de gran importancia en PHP 5. Una propiedad declarada como estática, puede ser accedida sin necesidad de instanciar un objeto y su valor es estático (es decir, no puede variar ni ser modificado). Ésta, se define anteponiendo la palabra clave *static* al nombre de la variable:

```
class PersonaAPositivo extends Persona {
        public static $tipo_sangre ='A+';
}
```

Accediendo a las propiedad de un objeto

Para acceder a las propiedades de un objeto, existen varias maneras de hacerlo. Todas ellas, dependerán del ámbito desde el cual se las invoque así como de su condición y visibilidad.

Acceso a variables desde el ámbito de la clase

Se accede a una propiedad no estática dentro de la clase, utilizando la pseudo-variable $this siendo esta pseudo-variable una referencia al objeto mismo:

```
return $this->nombre;
```

Cuando la variable es estática, se accede a ella mediante el operador de resolución de ámbito, doble dos-puntos :: anteponiendo la palabra clave self o parent según si trata de una variable de la misma clase o de otra de la cual se ha heredado, respectivamente:

```
print self::$variable_estatica_de_esta_clase;
print parent::$variable_estatica_de_clase_madre;
```

Acceso a variables desde el exterior de la clase

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

Nótese además, que este acceso dependerá de la visibilidad de la variable. Por lo tanto, solo variables públicas pueden ser accedidas desde cualquier ámbito fuera de la clase o clases heredadas.

```
# creo el objeto instanciando la clase
$persona_a_positivo= new PersonaAPositivo();

# accedo a la variable NO estática
print $persona_a_positivo->nombre;
```

Para acceder a una propiedad pública y estática el objeto no necesita ser instanciado, permitiendo así, el acceso a dicha variable mediante la siguiente sintaxis:

Clase::$variable_estática

```
# accedo a la variable estática
print PersonaAPositivo::$tipo_sangre;
```

Métodos en PHP 5

Cabe recordar, para quienes vienen de la programación estructurada, que el método de una clase, es un algoritmo igual al de una función.

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.

Reglas de Estilo sugeridas

Utilizar nombres_de_funciones_descriptivos, en letra minúscula, separando palabras por guiones bajos, ayuda a comprender mejor el código fuente haciéndolo más intuitivo y legible.

La forma de declarar un método es anteponiendo la palabra clave *function* al nombre del método, seguido por un par paréntesis de apertura y cierre y llaves que encierren el algoritmo:

```
# declaro la clase
class Persona {
	#propiedades

	#métodos
	function donar_sangre() {
		#...
	}
}
```

Al igual que cualquier otra función en PHP, los métodos recibirán los parámetros necesarios indicando aquellos requeridos, dentro de los paréntesis:

```
# declaro la clase
class Persona {
	#propiedades

	#métodos
	function donar_sangre($destinatario) {
		#...
	}
}
```

Métodos públicos, privados, protegidos y estáticos

Los métodos, al igual que las propiedades, pueden ser públicos, privados, protegidos o estáticos. La forma de declarar su visibilidad tanto como las características de ésta, es exactamente la misma que para las propiedades.

```
static function a() { }

protected function b() { }

private function c() { }
```

```
# etc...
```

Métodos abstractos

A diferencia de las propiedades, los métodos, pueden ser abstractos como sucede con las clases.

El Manual Oficial de PHP, se refiere a los métodos abstractos, describiéndolos de la siguiente forma:

"Los métodos definidos como abstractos simplemente declaran la estructura del método, pero no pueden definir la implementación. Cuando se hereda de una clase abstracta, todos los métodos definidos como abstract en la definición de la clase parent deben ser redefinidos en la clase child; adicionalmente, estos métodos deben ser definidos con la misma visibilidad (o con una menos restrictiva). Por ejemplo, si el método abstracto está definido como protected, la implementación de la función puede ser redefinida como protected o public, pero nunca como private."

Para entender mejor los métodos abstractos, podríamos decir que a grandes rasgos, los métodos abstractos son aquellos que se declaran inicialmente en una clase abstracta, sin especificar el algoritmo que implementarán, es decir, que solo son declarados pero no contienen un "código" que especifique qué harán y cómo lo harán.

Métodos mágicos en PHP 5

PHP 5, nos trae una gran cantidad de auto-denominados "métodos mágicos". Estos métodos, otorgan una funcionalidad pre-definida por PHP, que pueden aportar valor a nuestras clases y ahorrarnos grandes cantidades de código. Lo que muchos programadores consideramos, ayuda a convertir a PHP en un lenguaje orientado a objetos, cada vez más robusto.

Entre los métodos mágicos, podemos encontrar los siguientes:

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.

```
# declaro la clase
class Producto {
       #defino algunas propiedades
       public $nombre;
       public $precio;
       protected $estado;
```

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

	# constructor de la clase
	function__construct() {
		$this->set_estado_producto('en uso');
	}
}
```

En el ejemplo anterior, el constructor de la clase se encarga de definir el estado del producto como “en uso”, antes de que el objeto (Producto) comience a utilizarse. Si se agregaran otros métodos, éstos, podrán hacer referencia al estado del producto, para determinar si ejecutar o no determinada función. Por ejemplo, no podría mostrarse a la venta un producto “en uso por el sistema”, ya que a éste, se le podría estar modificando el precio.

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.

```
# declaro la clase
class Producto {
	#defino algunas propiedades
	public $nombre;
	public $precio;
	protected $estado;

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

	# constructor de la clase
	function__construct() {
		$this->set_estado_producto('en uso');
	}
```

```
        # destructor de la clase
        function __destruct() {
                $this->set_estado_producto('liberado');
                print 'El objeto ha sido destruido';
        }
}
```

Algunas características del trabajo con POO en PHP 5

1. Nombres fijos para los constructores y destructores
   Los nombres predefinidos para los métodos constructores y destructores son __construct() y __destruct().
2. Acceso public, private y protected a propiedades y métodos
   Estos modificadores de acceso habituales de la POO sirven para definir qué métodos y propiedades de las clases son accesibles desde cada entorno.
3. Posibilidad de uso de interfaces
   Éstas se utilizan en la POO para definir un conjunto de métodos que implementa una clase. Una clase puede implementar varias interfaces o conjuntos de métodos.
4. Métodos y clases final
   Se puede indicar que un método es "final" o que la clase es "final", lo que se indica es que esa clase no permite ser heredada por otra clase.
5. Operador instanceof
   Se utiliza para saber si un objeto es una instancia de una clase determinada.
6. Atributos y métodos static
   Son las propiedades y funcionalidades a las que se puede acceder a partir del nombre de clase, sin necesidad de haber instanciado un objeto de dicha clase.
7. Clases y métodos abstractos
   Las clases abstractas no se pueden instanciar y los métodos abstractos no se pueden llamar. Ambos se utilizan más bien para ser heredados por otras clases, donde no tienen por qué ser declarados abstractos.
8. Constantes de clase
   Se pueden definir constantes dentro de la clase y luego acceder a ellas a través de la propia clase.
9. Funciones que especifican la clase que reciben por parámetro
   En caso que el objeto no sea de la clase correcta, se produce un error.
10. Función __autoload()
    La función __autoload() sirve para incluir el código de una clase que se necesite, y que no haya sido declarada todavía en el código que se está ejecutando.

Ejemplo de clase y de objeto:

```
class hombre{
	var $nombre;
	var $edad;

	function comer($comida){
		//aquí el código del método
	}
	function moverse($destino){
		//aquí el código del método
	}

	function estudiar($asignatura){
		//aquí el código del método
	}
}

/* implememementacion */
$pepe = new hombre();
$juan = new hombre();
```

Echo Tags: Desde la introducción de etiquetas cortas, también conocidas como “etiquetas de eco”, permiten imprimir el resultado de una expresión directamente a la salida de la secuencia de comandos. Con el lanzamiento de PHP 5.4, las etiquetas cortas y etiquetas de