Programación Orientada a Objetos con PHP (POO)

Introducción

PHP tiene la característica de permitir programar con las siguientes metodologías:

Programación Lineal: Es cuando desarrollamos todo el código disponiendo instrucciones PHP alternando con HTML.

Programación Estructurada: Es cuando planteamos funciones que agrupan actividades a desarrollar y luego dentro de la página llamamos a dichas funciones que pueden estar dentro del mismo archivo o en una librería separada.

Programación Orientada a Objetos: Es cuando planteamos clases y definimos objetos de las mismas, disponible en PHP desde su versión 5.

Escuela Politécnica Nacional – Centro de Educación Continua CEC-EPN
Realizado por: Ing. David Badillo Bernal – dbadillo@ecualinux.com – **Cel:** 099016846

Una Clase

Es la estructura de un objeto, es decir, la definición de todos los elementos de que está hecho un objeto.
Una clase se compone de dos partes:

- **Atributos.-** Son los datos que se refieren al estado de un objeto.

- **Métodos.-** Son funciones que pueden aplicarse a objetos.

Un Objeto

Es la instancia de una una clase. *Por ejemplo:*

Supongamos tenemos clase persona, la instancia (objeto) de esta clase, puede ser: Antonio Valencia, Jefferson Pérez etc...

Realizado por: Ing. David Badillo Bernal – dbadillo@ecualinux.com – **Cel:** 099016846

Características de la POO

Existes muchas características de la POO, sin embargo existen un consenso al definir las características principales, para considerar que un lenguaje es o no orientado a objetos, estas son:

- **Abstracción.-** Cada objeto en el sistema, sirve como modelo de un "agente" abstracto. La abstracción es básicamente la capacidad de separar los elementos de un sistema (al menos mentalmente), para poder verlos en forma singular. Como cuando describimos el cuerpo humano y decimos: cabeza, troncos, brazos, piernas... etc.

- **Ocultamiento / Encapsulación.-** El objetivo es el restringir el acceso a los métodos y propiedades de una clase.

  Niveles de Acceso:

  Público: funciones de toda clase pueden acceder a los datos o métodos de una clase que se define con el nivel de acceso público. Este es el nivel de protección de datos más bajo.

Protegido: El acceso a los datos está permitido a las clases heredadas, es decir, las funciones miembro de esa clase y todas las subclases.

Privado: el acceso a los datos está restringido a los métodos de esa clase en particular. Este es nivel más alto de protección de datos.

- **Herencia.-** Es la capacidad que tienen las clases de derivar las propiedades y métodos de otra.

- **Polimorfismo.-** es la capacidad que tienen los objetos de una clase de responder al mismo mensaje o evento en función de los parámetros utilizados durante su invocación. Un objeto polimórfico es una entidad que puede contener valores diferentes durante la ejecución del programa.

Ventajas y Desventajas de la POO

Algunas ventajas de la POO

- Re utilización de código
- Relacionar el sistema al mundo real
- Agilita el desarrollo de software
- Facilita el trabajo en equipo
- Facilita el mantenimiento del software
- Construcción de prototipos
- Desarrollo más flexible

Algunas desventajas de la POO.

- Dificultad en la comprensión de los conceptos POO
- Tamaño de las aplicaciones

Escuela Politécnica Nacional – Centro de Educación Continua CEC-EPN
Realizado por: Ing. David Badillo Bernal – dbadillo@ecualinux.com – **Cel:** 099016846

Declaración de una clase

```
<?php
class transporte {

        private $id;
        private $tipo;
        private $capacidad_carga;
        private $capacidad_personas;
        private $codigo;

        public function get_form_transporte(){
                $retval = '
                        <form name="trasporte" action="" method="POST">
                                <table align="center" border=0>
                                        <tr>
                                                <th colspan="2">Manejo de Transporte</td>
                                        </tr>
                                        <tr>
                                                <th align="left">Tipo:</th>
                                                <td>' .$this->_get_tipo_transporte() . '</td>
                                        </tr>
                                        <tr>
                                                <th align="left">Capacidad de Carga:</th>
                                                <td><input type="text"
name="capacidad_carga" size=4></td>
                                        </tr>
                                        <tr>
                                                <th align="left">Capacidad de Personas:</th>
                                                <td><input type="text"
name="capacidad_personas" size=4></td>
                                        </tr>
                                        <tr>
                                                <th align="left">Código:</th>
                                                <td><input type="text" name="codigo"
size=4></td>
                                        </tr>
```

```
                                        <tr>
                                                <th colspan="2"><input type="submit"
name="guardar" value="Guardar"></th>
                                        </tr>
                                </table>
                        </form>';
                        return $retval;
        }

        private function _get_tipo_transporte(){

        $tipo = array("Aéreo","Terrestre","Marítimo");
        $cont = count($tipo);

        $retval = '
                <select name="tipo_transporte">';
                for ($i=0;$i<$cont;$i++){
                        $retval .= '<option value="' . $i . '">' . $tipo[$i] .  '</option>';
                }
        $retval .= '    </select>';

        return $retval;

        }

}
?>
```

- Como pudimos ver se declara la clase con la palabra clave class
- Tanto las propiedades como los métodos pueden ser: públicos (public), privados (private), protegidos (protected).
- Para llamar a una propiedad o método de la misma clase, se usa $this.

Instancicación de Clases (Objetos)

```
<?php

include("include/class.transporte.inc.php");

$trasporte = new transporte();

echo $trasporte->get_form_transporte();

?>
```

Para instanciar una clase, debemos utilizar ***new*** seguido del nombre de la clase, esto lo guardamos en una variable **Tipo Objeto.**

Posteriormente para llamar a una función o propiedad de una clase, debemos utilizar el nombre de la variable objeto, seguido de **->** y el nombre de la propiedad u método.

echo $trasporte->get_form_transporte();

Escuela Politécnica Nacional – Centro de Educación Continua CEC-EPN
Realizado por: Ing. David Badillo Bernal – dbadillo@ecualinux.com – **Cel:** 099016846

Método Constructor

- Tiene el mismo nombre de la clase
- El constructor es el primer método que se ejecuta cuando se instancia una clase.
- El constructor se llama automáticamente
- El constructor no puede retornan un dato
- El constructor puede recibir parámetros al momento de instanciar la clase.

Ejemplo:

```
function transporte($id){
        echo "entra a la clase constructor";

        $this->id = $id;
        $this->tipo = 0;
        $this-> capacidad_carga = "";
        $this->capacidad_personas = "";
        $this->codigo = "";

    }
```

Herencia

Una de las propiedades fundamentales de la POO es la Herencia, la cual permite heredar (transferir) propiedades o métodos de una clase (parent) a otra hija. Tomando como ejemplo la clase definida anteriormente con el nombre de transporte, podemos crear una clase llamada vehículo la cual herede las propiedades y métodos de su clase padre.
A continuación un ejemplo de la clase hija vehículo.

```
<?php
include("class.transporte.inc.php");

class vehiculo extends transporte{

		private $id;
		private $marca;
		private $color;

		function vehiculo(){
			$this->id = 0;
			$this->marca = "";
			$this->color = "";
		}

		public function get_form_vehiculo(){
		$retval = '
			<form name="vehiculo" action="" method="POST">
				<table align="center" border=0>
					<tr>
						<th colspan="2">Manejo de Vehículos</td>
```

```
                                        </tr>
                                        <tr>
                                                <th align="left">Tipo de Transporte:</th>
                                                <td>' . parent::_get_tipo_transporte() . '</td>
                                        </tr>
                                        <tr>
                                                <th align="left">Marca:</th>
                                                <td>' . $this->_get_marca() . '</td>
                                        </tr>
                                        <tr>
                                                <th align="left">Color:</th>
                                                <td>' . $this->_get_color() . '</td>
                                        </tr>
                                        <tr>
                                                <th colspan="2"><input type="submit"
name="guardar" value="Guardar"></th>
                                        </tr>
                                </table>
                        </form>';
                        return $retval;
        }

        private function _get_marca(){

        $marca = array("Chévrolet","Ford","Fiat","Scoda","Toyota","Renault");
        $cont = count($marca);

        $retval = '
                <select name="marca">';
                for ($i=0;$i<$cont;$i++){
                        $retval .= '<option value="' . $i . '">' . $marca[$i] .  '</option>';
                }
        $retval .= '    </select>';

        return $retval;

        }
```

```
        private function _get_color(){

        $color = array("Negro","Blanco","Gris","Plata","Verde","Rojo","Azul");
        $cont = count($color);

        $retval = '
                <select name="color">';
                for ($i=0;$i<$cont;$i++){
                        $retval .= '<option value="' . $i . '">' . $color[$i] .  '</option>';
                }
        $retval .= '    </select>';

        return $retval;

        }
}
?>
```

En este caso la clase trasporte ya no es necesario que sea instanciada, ya que podemos llamarla una vez que instanciemos como un objeto a la clase vehículo.

```
<?php
include("include/class.vehiculo.inc.php");

$vehiculo = new vehiculo();

echo $vehiculo->get_form_vehiculo();
echo $vehiculo->get_form_transporte();

?>
```

Clases abstractas

Son aquellas que representan un novel alto de abstracción y estas no sirven para ser instanciadas. Como tenemos el objeto podemos decir que la clase trasporte puede ser tranquilamente una clase abstracta, para definir una clase como tal solo debemos anteponer el nombre de **abstract**

```
abstract  class transporte {

    private $id;
    private $tipo;
    private $capacidad_carga;
    private $capacidad_personas;
    private $codigo;

    function transporte($id){
        echo "entra a la clase constructor";

        $this->id = $id;
        $this->tipo = 0;
        $this-> capacidad_carga = "";
        $this->capacidad_personas = "";
        $this->codigo = "";
    }
```

Tarea:

1.- Crea una clase CRectángulo. La clase tiene dos datos miembro, largo y ancho. La clase cuenta con métodos que calculan el perímetro (obtenPerímetro) y el área (obtenÁrea) del rectángulo, así como métodos para obtener y poner los valores correspondientes a los atributos largo y ancho.

2.- Crear una clase CCuadrado. La clase debe heredar de la clase CRectángulo. Bajo un método de la clase debe verificar que el largo y ancho son iguales e invocar a las funciones de la clase padre.

Realizado por: Ing. David Badillo Bernal – dbadillo@ecualinux.com – **Cel:** 099016846