UNIVERSIDAD DE SONSONATE

Implantación de Sistemas

Guía 2 –Programación Orientada a Objetos en PHP

Instructor: Ing. David A. Rodríguez

26/08/2017

Facultad de Ingeniería y Ciencias Naturales

Contenido

ELEMENTOS DE LA PROGRAMACIÓN ORIENTADA A OBJETOS.

La programación orientada a objetos o POO (OOP según sus siglas en inglés) es un paradigma de programación que usa los objetos en sus interacciones, para diseñar aplicaciones y programas informáticos, además está compuesta por una serie de elementos que se detallan a continuación.

- **Clase:** Una es un modelo que se utiliza para crear objetos que comparten un mismo comportamiento, estado e identidad.
- **Objeto:** Es una entidad provista de métodos o mensajes a los cuales responde (comportamiento); atributos con valores concretos (estado); y propiedades (identidad).
- **Método:** Es el algoritmo asociado a un objeto que indica la capacidad de lo que este puede hacer.
- **Evento y Mensaje:** Un evento es un suceso en el sistema mientras que un mensaje es la comunicación del suceso dirigida al objeto.
- **Propiedades y Atributos:** Las propiedades y atributos, son variables que contienen datos asociados a un objeto.

CARACTERISTICAS DE LA PROGRAMACION ORIENTADA A OBJETOS.

La programación orientada a objeto posee cuatro características fundamentales las cuales son las siguientes:

- ***Abstracción:*** Visión de un problema que extrae la información esencial para un determinado propósito, omitiendo los detalles menos significativos.
- ***Encapsulamiento:*** Es un agrupamiento bajo un mismo nombre de la información y las operaciones que acceden a ella.
- ***Polimorfismo:*** Consiste en utilizar un mismo símbolo para fines distintos.
- ***Herencia:*** Permite especializar o refinar una clase, y/o generalizar los conceptos de otra clase ya existentes.

En muchas ocasiones se menciona que la programación orientada a objetos posee más características, pero sus características fundamentales son las descriptas anteriormente, pero como agregado podemos listar las siguientes:

- ***Modularidad:*** Características que permite dividir una aplicación en varias partes más pequeñas denominadas módulos, las cuales son independientes una de otras.
- ***Ocultación:*** 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.
- ***Recolección de Basura:*** Es la técnica que consiste en destruir aquellos objetos cuando ya no son necesarios, librándolos de la memoria.

PROGRAMACION ORIENTADA A OBJETOS EN PHP.

¿Qué es PHP?

Es un lenguaje de código abierto muy popular especialmente adecuado para desarrollo web y que puede ser incrustado en HTML.

Ejemplo:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
  "http://www.w3.org/TR/html4/loose.dtd">
<html>
  <head>
    <title>Example</title>
  </head>
  <body>

    <?php
      echo "Hola, ¡soy un script PHP!";
    ?>

  </body>
</html>
```

En la actualidad PHP ha tenido una evolución con sus últimas versiones de las cuales ha permitido que dicho lenguaje, se convierta no solamente en un lenguaje que permita crear páginas web dinámicas, si no poder desarrollar sistemas a través del paradigma Orientado a Objetos.

CLASES Y OBJETOS EN PHP.

DEFINICION DE CLASES EN PHP.

Según lo describe el manual de PHP, una clase en PHP es *"una colección de variables y funciones que trabajan con estas. Las variables se definen utilizando **var** y las funciones utilizando **function**"*. Aplicando lo mencionado anteriormente, la definición de una clase sería de la siguiente manera:

Ejemplo:

```
class NombreClase{

        #....

}
```

DEFINICION DE CLASES ABSTRACTAS EN PHP.

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 a class.***

Ejemplo:

```
abstract class NombreClase{

        #....

}
```

Este tipo de clase contendrán métodos abstractos, y su finalidad es crear clases genéricas que necesitan ser declaradas pero a las cuales, no se puede otorgar una definición precisa ya que esto se encarga la clase que herede de ella.

DEFINICION DE CLASES FINALES EN PHP.

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

Ejemplo:

```
final class NombreClase{

        #....

}
```

HERENCIA DE CLASES.

Los objetos pueden heredar propiedades y métodos de otros objetos. Para ello, PHP permite la herencia de clases, cuyas características representa la relación existente entre diferentes objetos. Para definir una clase como extensión de una clase Padre se utiliza la clave ***extends.***

```
class NombreDeMiClasePadre{
        #...
}
class NombreDeMiClaseHija extends NombreDeMiClaseMadre {
        /* esta clase hereda todos los métodos y propiedades de
           la clase Padre NombreDeMiClasePadre
        */
}
```

DEFINICION DE OBJETOS EN PHP.

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

Para instanciar una clase, solo es necesario utilizar la palabra clave **new.** El objeto será creado, asignando esta instancia a una variable. Un ejemplo de la aplicación de esta palabra reservada es la siguiente:

Ejemplo:

```
class Persona{
        #...
}

#Creamos el objeto instanciando
$person = new Persona();
```

PROPIEDADES Y METODOS DE CLASES EN PHP.

Las propiedades representan ciertas características del objeto en sí mismo. Se definen anteponiendo la palabra clave ***var*** al nombre de la variable (propiedad):

Ejemplo:

```
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. 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.

Ejemplo:

```
class Persona {

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

PROPIEDADES PRIVADAS.

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

Ejemplo:

```
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:

Ejemplo:

```
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:

Ejemplo:

```
class PersonaAPositivo extends Persona {

        public static $tipo_sangre ='A+';

}
```

ACCEDIENDO A LAS PROPIEDADES 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:

Ejemplo:

```
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:

Ejemplo:

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

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.

Ejemplo:

```
# 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:

Ejemplo:

```
Clase::$variable_estática
```

METODOS EN PHP.

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:

Ejemplo:

```
# 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:

Ejemplo:

```
# declaro la clase
class Persona {
        #propiedades
        #métodos
        function donar_sangre() {
                #...
        }
}
```

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.

Ejemplo:

```
static functiona() { }
protect ed functionb() { }
private functionc() { }
# etc...
```

METODOS MAGICOS EN PHP.

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');
        }
}
```

EL MÉTODO MÁGICO __DESTRUCT()

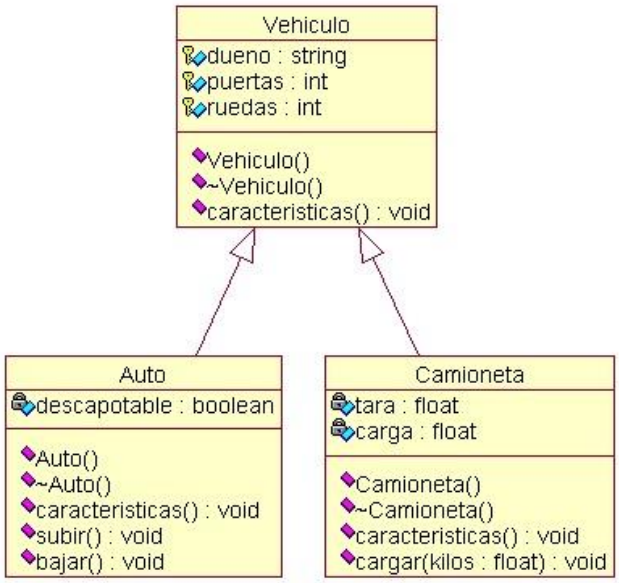
El método ***__destruct()*** es el encargado de liberar de la memoria, al objeto cuando ya no es referenciado. Se puede aprovechar este método, para realizar otras tareas que estimen necesarias al momento de destruir un objeto.

```
# declaro la clase
class Producto {
        #defino algunas propiedades

        # constructor de la clase

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

EJERCICIOS.

A continuación se le presentaran una serie de diagramas de clases de los cuales se pide elaborar su código equivalente para PHP, poniendo en práctica lo antes descrito en esta guía.

1. Codifique el siguiente diagrama utilizando los elementos de la Programación Orientada a Objetos.



2. Realice la codificación del siguiente diagrama, utilizando los elementos de POO.