 (POO)Programación Orientada a Objetos.

**¿Por qué utilizar POO?**

Php es un lenguaje de programación estructurado con extensiones de orientación a objetos.

En php una función declarada en una página php podrá ser declarada inmediatamente después de su declaración.

Como primera aproximación, uno podría plantearse programar en cada archivo php todas las funciones necesarias que vayan a usarse en el documento. En proyectos grandes esto supone un alto costo de mantenimiento debido a la complejidad de seguimiento del código y a la duplicación del mismo código en distintas partes, en caso de querer aplicar una mejora en una función, en caso de detectarse un error y el código que lo corrige se deberán revisarse todos los scripts php para comprobar si contienen la función que hay que corregir **PHP PERMITE DEFINIR CLASES, QUE POSEEN VARIABLES (TAMBIEN LLAMDAS PROPIEDADES ) Y METODOS (TAMBIEN LLAMADAS FUNCIONES ).**

<?php

class Matemáticas (

//Declaración de las variables internas

$error=””;

//Funciones

//sumar los valores

function suma($valor1,$valor2)

{

if(is\_numeric($valor1) && is\_numeric($valor2))

{

$this->error=””;

return $valor1 + $valor2;

}

else

{

$this->error=” Un valor no es numérico”;

return “NaN”

}

}

//Restar los valores

function restar ($valor1, $valor2)

{

if(is\_numeric($valor1) && is\_numeric($valor2))

{

$this->error=””;

return $valor1 - $valor2;

}

else

{

$this->error=” Un valor no es numérico”;

return “NaN”

}

}

function obtener error () {

return $error;

]

}

?>

Las clases permiten organizar mejor el código, ampliar nuestra productividad y eficiencia mediante la reutilización de código, la multiplicidad y la versatilidad

**Organización del código**

En php (POO) cada clase se programa dentro de un único fichero.

Posteriormente a la creación del documento con la definición de la clase, pueden crearse instancias de la clase llamadas objetos

<?php

class Texto (

//Declaración de las variables internas

$cadena=””;

//Funciones

//Obtener el valor iniciar de la cadena

function entrar\_cadena($variable) {

$this->cadena = $variable;

}

//Elimina los espacios existentes al inicio y fin de cadena

function limpiar\_espacios () {

return tris($this->cadena);

}

//se inserta una cadena de caracteres delante o detras de la cadena o trabajos

function insertar\_delante($textoDelante) {

$this->cadena = $textoDelante . $this->cadena;

}

function insertar\_detras($textoDetras){

$this->cadena = $this->cadena . $textoDetras. ;

}

//Escapa los caracteres especiales para evitar errores en SQL

function preparar\_SQL () {

return addslashes($this->cadena);

}

//Escribe en formato HTML los caracteres especiales, tales como los acentos

function preparar\_HTML () {

return htmlspecialchars($this->cadena);

}

function mostrar\_cadena () {

$this->cadena;

}

}

taller

-Crear dos archivos dichos archivos deben de tener la programación orientada a objetos y el uso de la programación orientada a objetos.

-Solicitar tres cadenas o tres variables una de tipo entero (INT), una de tipo doublé y una de tipo string

-Crear un método que defina qué tipo de variable es

-Crear una función que me extraiga y me muestre por consola la letra (a)

-Crear un método que te muestre por consola el número 3, tomar como referencia las variables que indica el usuario

Reutilización de código

La creación de un portal dinámico en php suele componerse de páginas de presentación, formularios, obtención de información existente dentro de bases de datos, tratamientos de datos obtenidos desde formularios, portales externos etc.

Hay funcionalidades comunes a la mayoría de los proyectos, como la variación de variables o del acceso a bases de datos, por lo que el programador tiende a reutilizar código implementado en proyectos anteriores, con lo que incrementa así su eficiencia y evita posibles errores de programación.

Estamos acostumbrados a reutilizar código de terceras personas, como por ejemplo la api de php con funciones tales como (REQUIRE ECHO MYSQLY)

MULTIPLICIDAD

como ya hemos detallado, una clase es un **prototipo** que define variables y funciones que pueden realizarse sobre estas variables **una vez definida una clase,, podemos instanciar tantos objetos de la clase como sean necesarios, teniendo todos ellos las características de la clase es decir sus variables y funciones.**

entendimiento del concepto =

dentro de una aplicación web en la que se debe hacer un traspaso de dinero entre dos cuentas, existe una cuenta a la que se hará un ingreso de dinero (cuenta de destino B) que provienen de una cuenta a la que se le deberá sustraer dicho saldo (cuenta de origen A).

un traspaso de un saldo supone una consulta de saldo en la cuenta origen y una actualización en la misma cuenta para restar el saldo que se va a traspasar. Por parte del beneficiario, implica una consulta de saldo actual para poder realizar una actualización que lo incrementan con el traspaso

¿PREGUNTA?

desarrollar el código necesario e implementación del sistema de información con POO que me permita realizar las acciones planteadas.

Implementar una (bd).

**HERENCIA**

**Modificadores y diferencias de clases**

Los modificadores definen las restricciones de acceso a propiedades y métodos, y la herencia de estos es posible dependiendo del modificador.

las clases, propiedades y métodos se definen mediante los modificadores **public, protected, private, final, abstract seguido de una declaración de variable normal:**

**1-PUBLIC: la propiedad o método podrá usarse en cualquier parte del escrito.**

**2-PRIVATE: la propiedad o método sólo podrá usarse en la clase a la que pertenece**

**3-PROTECTED: la propiedad o método se podrá usar por la clase a la que pertenece y por sus descendientes.**

**4-FINAL: la clase o método no puede ser sobre escrito en clases descendientes.**

**5-ABSTRACT: la clase o método no puede ser usada directamente a de ser heredada primero para usarse**

primero es necesario saber que es la herencia de clase una clase puede heredar los métodos y propiedades de otra clase usando la palabra **EXTENDS.** no es posible extender a múltiples clases, solo se puede heredar de una clase, los métodos y propiedades heredados pueden ser pobre escritos declarándose de nuevo con el mismo nombre que tiene en la clase padre

class Perro

{

public $nombre = “Rudolf”;

public function ladrar () {

print “Guau!”,

}

}

class Bulldog extends Perro {

public function ladrar () {

print “Woof!”;

}

}

$cachorro = new Bulldog (); // instancia de la clase hija

$cachorro -> nombre = “Jeffrey” ;// heredamos la propiedad padre $nombre y le asignamos

echo $cachorro -> nombre;

$cachorro -> ladrar ();

1-PÚBLIC: Las propiedades y métodos públicos son accesibles desde cualquier parte del escrito, siendo este modificado el más fácil de usar**. en php 4 las variables se declaraban con la palabra reservada var.** y eran public, pero esa forma de denominarlas está obsoleta y puede generar algunos warning

class Perro

{

public $nombre;

public function ladrar () {

print “Guau!”,

}

}

class Bulldog extends Perro {

public function ladrar () {

print “Woof!”;

}

}

$cachorro = new Bulldog ();

$cachorro -> nombre = “Bunbuns”

echo $cachorro -> nombre; // devuelve bunbuns

por defectos todas son public, no es necesario especificar public pero es mejor hacerlo para que sea más legible. con los métodos puedes no utilizar ninguna palabra y escribir directamente functions. con las propiedades en cambio necesitas una palabra por lo menos sino daría un error de sintaxis; ya que si no hay manera de saber que propiedades tiene una clase. nunca hay que utilizar var

2-PRIVATE: el problema de las propiedades y métodos public es que se permite llamar a lso métodos y establecer las propiedades desde cualquier lado del script. En el ejemplo anterior en ves de crear un objeto bulldog creamos un objeto perro e intentamos establecer nombre nos dará un fatal error **cannot acess private proprety**

class Perro

{

public $nombre;

public function ladrar () {

print “Guau!”,

}

}

$cachorro = new Perro ();

$cachorro -> nombre = “BunBuns” Fatal error

No se podrá ni mostrar ni modificar el nombre. Para hacerlo se utiliza getters y setters, methods public para poder acceder a estas propiedades y métodos private

Desde otras partes del script

class Perro

{

public $nombre;

public function ladrar () {

print “Guau!”,

}

}

Public function setNombre($nombre) {

$this -> nombre = $nombre;

}

Public function getNombre () {

return $this -> nombre;

}

}

$cachorro = new Perro ();

$cachorro -> setNombre = (“Chicken”);

echo $cachorro -> getNombre; // devuelve Chicken

3-PROTECTED: las propiedades y métodos marcados como protected son accesibles a través de la clase donde se crean y sus descendientes.

class Perro

{

protected $nombre;

protected function ladrar () {

print “Guau!”,

}

}

Class Bulldog extends Perro {

Public function ladrarBulldog () {

// podemos acceder a ladrar () de la clase perro

Return $this ->ladrar ();

}

Public function setNombre($nombre) {

$this -> nombre = $nombre;

}

Public function getNombre () {

print $this -> nombre;

}

}

4-FINAL: la palabra final se utiliza para declarar que un método o clase no puede ser sobrescrito por una clase hija

class Perro

{

public $nombre;

final function ladrar () {

print “Guau!”,

}

}

Class Bulldog extends Perro {

// dará error

Public function ladrarBulldog () {

{

Print “Woof!”,

}

}

$cachorro = new Bulldog ();

$cachorro -> ladrar (); //cannot override final method

5-ABSTRACT: el modificador abstract indica que una clase o método no puede instanciarse y solo puede heredarse, trasladamos su funcionamiento obligatorio a sus clses hijas

TALLER

Realizar un CRUD en php con poo con las siguientes condiciones:

1-Usar framework Bootstrap

2-las hojas de estilos (CSS) no debe tener más de 10 líneas.

3-Definir un archivo llamado function para hacer set y get de los distintos métodos

4-Definir un archivo para mis clases

5-Realizar la programación con herencia

6-Hacer uso de base de datos

Programación Orientada a Objetos 2

Método Constructor y destructor

Anteriormente Creamos una clase, definíamos sus atributos o propiedades y creamos nuestros métodos o funciones para posteriormente instanciar el objeto o clase y hacer uso de sus métodos, setear sus propiedades y realizar operaciones CRUD, también aprendimos a definir tipos de variables como lo eran privadas, protegidas, públicas y estáticas y a su vez métodos o funciones también privadas o protegidas de la siguiente manera.

Anteriormente definíamos una clase Persona de la siguiente manera:

class Persona {

}

En la programación orientada a objetos nosotros tenemos la posibilidad de crear un método llamado constructor, que diferencia de los otros métodos no es invocable(Instanciado), sino que se dispara en el preciso momento en el que nosotros creamos el objeto ósea cuando hacemos la instancia de la clase:

$persona = new Persona();

Esto es muy útil ya que permitirá en dicho método constructor definir todas aquellas características que tiene un objeto al ser creado. El método constructor se escribe como un método mágico, Por tanto nosotros no podemos ponerle cualquier nombre al método sino que tendrá que ser **\_\_construct.**

class Persona{

        public $nombre;

        public $apellido;

        public $edad;

        public function \_\_construct($nombre, $apellido, $edad){

            $this->nombre = $nombre;

            $this->apellido = $apellido;

            $this->edad = $edad;

        }

7

        public function saludar(){

            return 'Hola, soy' . $this->nombre . ' ' . $this->apellido . ' y tengo' . $this->edad . ' año';

        }

    }

Como verán en el anterior ejemplo nosotros teníamos que, luego de crear nuestro objeto persona, acceder desde el mismo para definir los valores de las propiedades: $nombre, $apellido y $edad. Pero ahora en este caso el constructor será el encargado de definir los valores de estas propiedades, por tanto, nosotros al crear el objeto estamos obligados a setear los valores de estas propiedades.

Ahora que sabemos de que se trata un método constructor podemos aprender a usar un método **destructor.** Los métodos destructores, son lo contrario a un método constructor. Los constructores se disparan cuando el objeto se crea, mientras que los destructores se disparan en cuanto se borra de memora. Si bien estos métodos no suelen usarse a menudo como si pasan con los constructores.

Primero que nada para destruir un objeto tenemos que usar la función de PHP unset() que sirve para borra cualquier tipo de variable de memoria, incluso un objeto. Esto puede resultar útil para liberar memoria en nuestro script.

Diseñar o maquetar Loguin responsive donde a mano izquierda exista una imagen que este animada, no se pueden utilizar videos y a mano derecha debe de esta el Loguin, cuando el dispositivo sea móvil debe desaparecer y solo quedar el formulario, manipular la sesión de usuario en una vista home a través de un botón donde me destruya las variables de sesión sin usar el sesión\_destroy ni sesión\_starthp