www.uneweb.com

PHP: Programación orientada a objeto

La mayoría de los lenguajes de programación modernos son orientados a objetos o en su defecto se aproximan mucho a éstos permitiendo algunas de sus características como es el caso de PHP. La programación OO principalmente hace uso de clases, objetos, relaciones, instancias, propiedades y métodos.

Objetos y clases

Cuando hablamos de software OO los objetos casi siempre son elementos físicos, como puede ser un cliente, proveedor, etc. o elementos conceptuales que existen en el entorno software, por ejemplo un objeto encargado del mantenimiento de archivos. El objetivo es representar a éstos elementos de la vida real y a los conceptuales como unidades de software.

La programación OO esta pensada para construir objetos que contienen atributos y operaciones de manera que cubran nuestras necesidades. Los atributos son variables que contienen información del estado de un objeto. Y las operaciones también conocidas como métodos, funciones y acciones realizan modificaciones del propio objeto o realizan alguna acción externa a éste.

Una de las principales ventajas de la programación OO es el concepto de encapsulación, conocido también como protección de datos, mediante el cual solo se pueden modificar los datos de un objeto accediendo a través de sus métodos u operaciones (interfaz del objeto). La funcionalidad de un objeto esta sujeta a los datos que este maneja, una ventaja de usar objetos es que podemos modificar la funcionalidad de éste, añadir mejoras o corregir errores sin necesidad de cambiar su interfaz. Ya que en caso contrario un proyecto estaría sujeto a un mayor número de fallos y los cambios serían más costosos.

En algunas áreas de la programación de aplicaciones Web el uso de la programación OO está desestimada, usándose una metodología estructurada basada en funciones, esto es debido a que determinados proyectos no son lo suficientemente extensos como para aplicarles una metodología OO.

En la programación OO los objetos son únicos y son instancias a una clase determinada. En principio se define la clase con los atributos y métodos correspondientes y luego se crea el objeto que esta basado en una determinada clase (esto se conoce como instancia).

Cómo crear clases, atributos y operaciones en PHP

Clase: Una clase 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).

```
$persona = new Persona ();

/* El objeto, ahora, es $persona, que se ha creado siguiendo el modelo de la clase Persona */
```

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

```
function caminar () {
        #...
}
```

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

Persona es la representación de una clase (la abstracción de Juan, Pedro, Ana y María), cuyo comportamiento puede ser caminar, correr, estudiar, leer, etc. Puede estar en estado despierto, dormido, etc. Sus características (propiedades) pueden ser el color de ojos, color de pelo, su estado civil, etc.

```
class Persona {
        # Propiedades
        # Métodos
        }
```

Hasta ahora hemos hablado de las clases de una forma conceptual, a continuación veremos como se crean, para crear una clase en PHP usaremos la palabra reservada class.

La estructura mínima de una clase es la siguiente:

```
class NombreClase {

}
```

Para que una clase sea útil, necesita atributos y operaciones. Una clase puede tener sus propias constantes, variables (llamadas "propiedades"), y funciones (llamados "métodos").

Ejemplo #1 Definición de una clase sencilla

```
<?php

class ClaseSencilla {

        // Declaración de una propiedad

        public $var = 'un valor predeterminado';

        // Declaración de un método

        public function mostrarVar() {

        echo $this->var;
        }
}
?>
```

Podemos crear métodos declarando funciones dentro de la definición de la clase, el siguiente código crea una clase llamada NombreClase con dos operaciones que no hacen nada. A metodo1 no le pasamos ningún parámetro y a método 2 le pasamos dos parámetros.

```
class NombreClase {

        function metodo1() {
        }
```

```
        function metodo2($param1, $param2) {
        }
    }
```

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 objet**o.**

Declaración de 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 a class:

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

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.

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

Constantes de clases

Es posible definir valores constantes en función de cada clase manteniéndola invariable. Las constantes se diferencian de las variables comunes en que no utilizan el símbolo $ al declararlas o emplearlas. La visibilidad predeterminada de las constantes de clase es public.

El valor debe ser una expresión constante, no (por ejemplo) una variable, una propiedad o una llamada a una función.

```
<?php
class MiClase
{
    const CONSTANTE = 'valor constante';

    function mostrarConstante() {
        echo  self::CONSTANTE . "\n";
    }
}

echo MiClase::CONSTANTE . "\n";
```

```php
$nombreclase = "MiClase";
echo $nombreclase::CONSTANTE . "\n"; // A partir de PHP 5.3.0

$clase = new MiClase();
$clase->mostrarConstante();

echo $clase::CONSTANTE."\n"; // A partir de PHP 5.3.0
?>
```

Qué es el constructor de una clase

Las clases soportan un tipo de función especial que se conoce como constructor. El constructor es llamado cuando se crea el objeto. Normalmente se utiliza para inicializar tareas como: asignación de valores a determinados atributos, crear nuevos objetos necesarios para el correcto funcionamiento del objeto, etc.

El constructor se declara de la misma forma que los métodos, lo único que debemos tener en cuenta es que debe tener el mismo nombre que la clase. A continuación veremos cómo se declara el constructor de una clase:

```php
<?php
class BaseClass {
   function __construct() {
       print "En el constructor BaseClass\n";
   }
}

class SubClass extends BaseClass {
   function __construct() {
       parent::__construct();
       print "En el constructor SubClass\n";
   }
}

class OtherSubClass extends BaseClass {
    // heredando el constructor BaseClass
}
?>
```

Cómo usar objetos, instanciar una clase

Después de haber declarado una clase, ya podemos usarla creando un objeto que es una instancia a esa clase como ya se ha mencionado anteriormente. Es muy importante que esto quede claro, los objetos son instancias a una clase, por lo tanto cada objeto es único.

En php creamos un objeto usando la palabra reservada new. También debemos indicar a que clase va a instanciar el objeto que creemos y pasarle los parámetros (si los requiere) al constructor, en el siguiente código vamos a ver un ejemplo de esto, para ello tomamos como referencia la clase anterior NombreClase:

```
$a = new NombreClase( );
$a -> NombreMetodo("Primero");
```

```
$b = new NombreClase( );
$b -> NombreMetodo( );
```

Cómo usar los atributos de una clase

Para la versión PHP 5.0 una clase tiene la siguiente característica, tiene un puntero especial al que podemos referenciar como $this. Si nuestra clase tiene un atributo llamado $atributo, podemos hacer referencia a este desde nuestra clase (métodos) de la siguiente forma $this->atributo, a continuación podemos ver el código de ejemplo del acceso a un atributo de clase desde un método de la propia clase.

```
class NombreClase {

    var $atributo;

    function metodo($param) {
     $this->atributo = $param;
    echo $this->atributo;
     }
}
```

A partir de la versión de PHP 5.3.0

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 sólo 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+';

    $a = new NombreClase();
    $a->atributo = "valor";
    echo $a->atributo;
```

Cómo usar los métodos de una clase

Los métodos de una clase se pueden llamar de la misma forma que se llaman a los atributos, supongamos que tenemos la siguiente clase:

```
    class nombreClase {

        function metodoa() {
                return "Has llamado al método A";
                 }

        function metodob($parametro1) {
                return "Método B llamado con parámetro: ".$paramtro1;
                }
        }
    /*Creamos un objeto de la case nombreClase con el nombre $obj de la siguiente forma:  */

    $obj = new nombreClase();
```

Los métodos de un objeto, son llamados de la misma forma que se llaman a funciones normales, pasándoles como es el caso de metodob el parámetro que necesiten. Como los métodos son funciones que pertenecen a un objeto, deberemos indicar en la llamada el objeto que los incluye, la forma de llamar a un método es idéntica a como llamamos a los atributos de un objeto.

```
    $obj->metodoa();
    $obj->metodob("parámetro que pasamos");
```

Si nuestras operaciones, devuelven algún valor, lo capturamos asignándoselo a una variable, podemos ver el siguiente ejemplo

```
    $txt_retorno = $obj->metodob("Hola");
        $txt_retorno contendrá la cadena de texto:
    "Metodo B llamado con parámetro: Hola"
```

Qué es la Herencia en PHP y como implementarla

Como su nombre indica el concepto de herencia se aplica cuando creamos una clase, que va a heredar los métodos y atributos de una ya definida, entonces la clase que hemos creado es una subclase. Para que una clase sea subclase de otra ya creada deberemos usar la palabra reservada extends en el siguiente código podremos ver como creamos una clase llamada SubClaseA que heredará los métodos y atributos de una clase definida con anterioridad llamada ClaseA.

```
class ClaseA. {

        #...
}
```

```
class SubClaseA extends ClaseA {

        /* esta clase hereda todos los métodos y propiedades de la clase madre
        ClaseA */
}
```

Si creamos un objeto de la clase SubClaseA este heredará todos los métodos de la clase ClaseA, por lo tanto el siguiente código es válido:

```
$x = new SubClaseA();
$x->operacion1();
$x->atributo1 = 100;
$x->operacion2();
$x->atributo2 = 200;
```

Como podemos observar aunque declaremos un objeto de la clase SubClaseA, al ser una clase extendida de ClaseA podemos hacer uso de todos los métodos y atributos definidos en ClaseA como si estuvieran contenidos en SubClaseA.

Debemos tener en cuenta que la herencia solo trabaja en una dirección, la subclase o clase hija hereda las características de su clase padre o superclase, pero la clase padre no posee las características de la hija. Para el caso anterior ClaseA no tendría atributo2 ni metodo2();

Ejercicio # 1

Realiza una Clase llamada cuadrado que contenga un atributo $num el método calcularCuadrado que haga la operación de obtener el cuadrado del atributo $num, que imprima por pantalla el resultado.

```
<?php
        class cuadrado {
        // Estos son ATRIBUTOS de los

                public $num;

        // Este es el METODO para calcular

                function calcularCuadrado() {
                        return ($this->num * $this->num);
                }
        }

        // Creamos el Objeto
        $operacion = new cuadrado();
        //Asignamos un atributo
        $operacion->num = 3;
        // Invocamos un método
        echo $operacion->calcularCuadrado();
?>
```

Ejercicio # 2

Realiza una Clase llamada imagen que contenga dos atributos $src y $border, crea un método constructor llamado cargarimagen que haga la operación de obtenerlos valores de los atributos $src,$borde, y otro llamado Imprimir que permita mostrar por pantalla el resultado.

```
<?php
class imagen {
// Estos son ATRIBUTOS de los objetos
            protected $src;
            protectd $border;
 // Esta función es el CONSTRUCTOR
            public function cargarimagen($src,$border) {
            $this->src=$src;
            $this->border=$border;
            }
// Esta función es un METODO
            public function Imprimir() {
            echo " <img src=";
            echo $this->src;
            echo " border=";
            echo $this->border;
            echo " >";
            }
}
// Creamos el Objeto
$logo = new Imagen();
// Invocamos el método
$logo->Imprimir("imagen/msn2.jpg",8);
?>
```

Ejercicio # 3

Ahora planteamos un problema utilizando herencia en PHP. Supongamos que necesitamos implementar dos clases que llamaremos Aritmética y Operación.

Cada clase tiene como atributo $a, $b y $resultado. Los métodos a definir son valores que inicializan los atributos $a y $b. La clase Operaciones hereda de aritmética sus atributos (que en el caso del método "Sumar" suma los dos atributos y en el caso de la clase "Restar" hace la diferencia entre $a y $b, y otro método mostrarResultado. Si analizamos ambas clases encontramos que muchos atributos y métodos son idénticos. En estos casos es bueno definir una clase padre que agrupe dichos atributos y responsabilidades comunes

```
<?php
class Aritmetica {
// Estos son ATRIBUTOS de los objetos
protected $a;
protected $b;
        protected $resultado;

// Este es el CONSTRUCTOR

    public function valores($a, $b) {
        $this->a = $a;
        $this->b = $b;
    }

    public function ImprimirResultados() {
```

```
        echo $this-> resultado.'<br>';

    }
}


class Operacion extends Aritmetica{

    // Este es el METODO para sumar
    public function sumar() {
        $this-> resultado=$this->a + $this->b;
    }

    // Este es el METODO para restar
    public function restar() {
        $this-> resultado=$this->a - $this->b;
    }

    // Este es el METODO para multiplicar
    public function multiplicar() {
        $this-> resultado=$this->a * $this->b;
    }

    // Este es el METODO para dividir
    public function dividir() {
        $this-> resultado=$this->a / $this->b;
    }
}


// Creamos el Objeto
$operacion = new Operacion;
// Invocamos un método
$operacion->valores(8,3);
// Ya con los valores definidos invocamos tantos metodos se requieran
$operacion->sumar();
echo 'El resultado de la suma es: ';
$operacion->imprimirResultados();

$operacion->multiplicar();
echo 'El resultado de la multiplicacion es: ';
$operacion->imprimirResultados();

$operacion->restar();
echo 'El resultado de la resta es: ';
$operacion->imprimirResultados();

?>
```

Ejercicio # 4

Realiza una Clase llamada cuadrado que contenga un atributo $num el método calcularCuadrado que haga la operación de obtener el cuadrado del atributo $num, y una subclase llamada calcularcubo que obtenga el valor de la operación cubo para luego imprimir por pantalla el resultado.

```
<?
// Superclase
class cuadrado {
// Estos son ATRIBUTOS de los objetos
        public $num;
// Este es el METODO para calcular el cuadrado
          function calcularCuadrado() {
                  return ($this->num * $this->num);
                  }
          }

//Subclase
class cubo extends cuadrado{
// Este es el METODO para calcular el Cubo
        function calcularCubo() {
          return ($this->calcularCuadrado() * $this->num);
        }
}

// Creamos el Objeto
$objeto = new cubo();
//Asignamos un atributo
$objeto->num = 3;
// Invocamos un método
echo $objeto->calcularCubo();
?>
```

Ejercicio # 5: Conexiones

Primero creamos el archivo donde almacenaremos los datos de nuestro host, database, username y password de mysql, lo llamaremos conexion.php

```
<?php
class conexion{
$hostname = "localhost"; //IP del servidor
$database; //Nombre de la base de datos
$username = "root";//usuario mysql
$password = "root";//password de usuario mysql

static public function conectar()
{
 $conexion = mysqli_connect($hostname, $username, $password,$database) or die("problemas") ;
    mysqli_set_charset($conexion,'utf8');

 return $conexion;

  if ($conexion) {
    echo'Conexion Sastifactoria';
 }else {
    die('Error de conexion'.mysqli_connect_error);
 }
 ;
```

Si la conexion se utilizara en algún otro archivo php, debe ser incluido en ese archivo el cual contiene la clase e instanciar la misma, un ejemplo de esto seria:

```
<?php
include_once("conexion.php");
$conexion=conexion::conectar();
$sql="select * from tabla";
$consulta=mysqli_query($sql,$conexion); ?>
```

Ciclo: `<?php while (list($nombre,$apellido) = mysqli_fetch_array($consulta)){?>`

Mostrar: `<?php print $nombre ?>` **Cerrar ciclo:** `<?php }?>`

Ejercicio nº 6: Combos desplegables enlazados:

```
-- Estructura de tabla para la tabla
`ciudad` CREATE TABLE `ciudad` (
 `id_ciudad` int(3) NOT NULL,
 `ciudad` varchar(20) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- Volcar la base de datos para la tabla `ciudad`
INSERT INTO `ciudad` VALUES (1, 'caracas');
INSERT INTO `ciudad` VALUES (2, 'maracay');
```

```
-- Estructura de tabla para la tabla `parroquia`
CREATE TABLE `parroquia` (
 `id_parroquia` int(3) NOT NULL,
 `id_ciudad` int(3) NOT NULL,
 `parroquia` varchar(20) NOT NULL
) ENGINE=MyISAM DEFAULT CHARSET=utf8;

-- Volcar la base de datos para la tabla `parroquia`
INSERT INTO `parroquia` VALUES (1, 1, 'caricuao');
INSERT INTO `parroquia` VALUES (2, 1, '23 de enero');
INSERT INTO `parroquia` VALUES (3, 2, 'las delicias');
INSERT INTO `parroquia` VALUES (4, 2, 'centro');
```

```
<?php
// datos de conexion a la BD.
require_once("conexion.php");
$conn=conexion::$conexio();
 // Obtener el $id_padre..
 $id_ciudad=$_POST['id_ciudad'];
// Inicio Formulario .. PHP_SELF enviamos a si mismo (a este script).
 echo "<form action=\"".$_SERVER['PHP_SELF']."\" method=\"POST\">\n\n";
// Formar Select "Padre".
 echo "<select name=\"id_ciudad\" onChange=\"this.form.submit()\">\n";
 echo "<option value=\"\"> Seleccione un Item </option>\n";

$SQLconsulta_ciudad="SELECT * FROM ciudad";

 $consulta_padre = mysqli_query($SQLconsulta_ciudad,$conn) or die(mysqli_error());
while ($registro_ciudad=mysqli_fetch_assoc($consulta_padre)){
 // Se mira si el ID del registro es el mismo q el $id_ciudad q recibimos si hemos cambiado el select parroquia.  // Se
selecciona en consecuencia (selected) la opción elegida.

 if ($id_ciudad == $registro_ciudad['id_ciudad']){
 echo "<option value=\"".$registro_ciudad['id_ciudad']."\" selected>".$registro_ciudad['ciudad']."</option>\n";  } else {
 echo "<option value=\"".$registro_ciudad['id_ciudad']."\">".$registro_ciudad['ciudad']."</option>\n";  }
 }
 echo "</select>\n\n";
mysqli_free_result($consulta_padre); // Liberar memoria usada por consulta.
// Formar Select "Parroquia"
 echo "<select name=\"id_parroquia\">\n";
// Si $id_ciudad no tiene valor (caso de que no se ha seleccionado ningua opcion del select hijo  //
se muestra el mensaje de "seleccione un item" (del select padre).

 if (!empty($id_ciudad)){
$SQLconsulta_parroquia="SELECT * FROM parroquia WHERE id_ciudad='$id_ciudad'";
$consulta_parroquia = mysqli_query($SQLconsulta_parroquia,$conn) or die(mysqli_error());  // se
mira el total de registros de la consulta .. si es 0 se muestra mensaje en el select ..  if
(mysqli_num_rows($consulta_parroquia) != 0){
 While ($registro_parroquia=mysqli_fetch_assoc($consulta_parroquia)){
 echo "<option
value=\"".$registro_parroquia['id_parroquia']."\">".$registro_parroquia['parroquia']."</option>\n";  }
 } else {
 echo "<option value=\"\"> No hay registros para este Item </option>";
```

```
}
} else {
echo "<option value=\"\"> <-- Seleccione un Item </option>";
}mysqli_free_result($consulta_parroquia); // Liberar memoria usada por consulta.
?>
```

Ejercicio # 7: Gráfico de Línea **Colocar librería gráfica jpgraph**	**Ejercicio # 8: Gráfico de Torta** **Colocar librería gráfica jpgraph**

<?php
require_once("conexion.php");
$conn=conexion::$conexio();
 	
include("jpgraph/jpgraph.php");
include("jpgraph/jpgraph_line.php");
 	
$sql="select * from grafico";
$consulta=@mysqli_query($sql,$conn);
 	
$xdata= array();
$ydata= array();
 	
while($row =
mysqli_fetch_array($consulta,
MYSQL_ASSOC)){
array_push ($xdata,$row["mes"]);
array_push ($ydata,$row["monto"]);
}
 	
$graph = new Graph(450,350,"auto");
$graph->SetScale("textlin");
$graph->xaxis->SetTickLabels($xdata);
	<?php
	require_once("conexion.php");
	$conn=conexion::$conexio();
	
	include ("jpgraph/jpgraph.php");
	include ("jpgraph/jpgraph_pie.php");
	include ("jpgraph/jpgraph_pie3d.php");
	
	$sql="select * from grafico";
	$consulta=@mysqli_query($sql,$conn);
	
	$xdata= array();
	$ydata= array();
	
	while($row = mysqli_fetch_array($consulta,
	MYSQL_ASSOC)){
	array_push ($xdata,$row["mes"]);
	array_push ($ydata,$row["monto"]);
	}
	$graph = new PieGraph(470,350,"auto");
	$graph->SetShadow();
	$graph->img->SetMargin(5,5,5,5);

$lineplot1 = new LinePlot($ydata); $lineplot1->SetColor("red"); $lineplot1->mark- >SetType(MARK_FILLEDCIRCLE); $lineplot1->mark->SetFillColor("red"); $lineplot1->mark->SetWidth(2); $graph->img->SetMargin(55,20,20,55); $graph->title->Set("Montos Indice de Morosidad"); $graph->xaxis->title->Set("Meses"); $graph->ygrid- >SetFill(true,'#EFEFEF@0.5','#F9BB64@0.5') ; $graph->SetShadow(); $graph->Add($lineplot1); $graph->Stroke(); mysqli_close($conexion); ?>	$graph->title->Set("Indice de Morosidad"); $graph->title->SetColor("darkblue"); $graph->legend->Pos(0.05,0.2); $p1 = new PiePlot3d($ydata); $p1->SetTheme("sand"); $p1->SetCenter(0.4); $p1->SetSize(80); $p1->SetAngle(60); $p1->SetStartAngle(60); $p1->value->SetColor("navy"); $p1->SetEdge("navy"); $p1->SetLegends(array("Ene","Feb","Mar","Abr")); $graph->Add($p1); $graph->Stroke(); ?>

Ejercicio # 9: Gráfico de barra: **Colocar librería gráfica jpgraph**	**Ejercicio # 10: Imprimir archivos PDF** **Colocar librería de reportes en pdf**

<?php
require_once("conexion.php");
$conn=conexion::$conexio();
include("jpgraph/jpgraph.php");
include ("jpgraph/jpgraph_bar.php");
$sql="select * from grafico";
$consulta=@mysqli_query($sql,$conn);
$xdata= array();
$ydata= array();
 	
while($row = mysqli_fetch_array($consulta,
MYSQL_ASSOC)){
array_push ($xdata,$row["mes"]);
array_push ($ydata,$row["monto"]);
}
 	
$months = $gDateLocale->GetShortMonth();
$graph = new Graph(470,350);
$graph->SetScale("textlin");
$graph->SetMarginColor('white');
$graph->SetMargin(55,20,20,55);
$graph->SetBox();
$graph->SetFrame(false);
$graph->ygrid-
>SetFill(true,'#DDDDDD@0.5','#BBBBBB@0.5')
; $graph->ygrid->SetLineStyle('dashed');
$graph->ygrid->SetColor('gray');
$graph->xgrid->Show();
$graph->xgrid->SetLineStyle('dashed');
$graph->xgrid->SetColor('gray');
$graph->img->SetMargin(55,20,20,55);
$graph->title->Set("Indice de Morosidad");
$graph->xaxis->title->Set("Meses");
$graph->yaxis->title->Set("Indice");
$bplot = new BarPlot($ydata);
$bplot->SetWidth(0.6);
$fcol='#440000';
$tcol='#FF9090';
$bplot->SetLegend("Mora");
$bplot-
>SetFillGradient($fcol,$tcol,GRAD_LEFT_REFLECTION);
$bplot->SetWeight(0);
$graph->Add($bplot);
$graph->Stroke();
?>
	<?php
	include('class.ezpdf.php');
	$pdf =& new Cezpdf('a4');
	$pdf->selectFont('fonts/Courier.afm');
	$pdf->ezText("titulo\n",20);
	$pdf->ezText("prueba\n\n",12);
	$pdf->ezText("Fecha: ".date("d/m/Y"),10);
	$pdf->ezText("Hora: ".date("H:i:s"),10);
	$pdf->ezStream();
	?>

Ejercicio # 11: Imprimir archivo pdf
Colocar librería de reportes en pdf

```
<?php
error_reporting(7); //esto debe ir en la primera linea
include ('class.ezpdf.php');

require_once("conexion.php");
$conn=conexion::$conexio();

$xclave=$_GET['clave'];

$sql="select * from cliente";
$rs=mysqli_query($sql,$conn);
$row=mysqli_fetch_assoc($rs);

//titulos para la tabla
$titulo[0]='Codigo';
$titulo[1]='Nombre del Cliente';
$titulo[2]='Telefono';

do{
$registro[$titulo[0]]=$row['codigo'];
$registro[$titulo[1]]=$row['nombre'];
$registro[$titulo[2]]=$row['telefono'];

$tabla[]=$registro;
}while($row=mysqli_fetch_assoc($rs));

//tamaño de la hoja carta y letra helvetica
$pdf=new Cezpdf('letter','landspace');
$pdf->selectFont('./fonts/Courier-Bold.afm');

//para colocar la imagen
$imagen=ImageCreatefromjpeg('imagen/msn2.jpg');
$pdf->addImage($imagen,10,$pdf->y-30,50,0);

//para bajar 15 pixeles y colocar el titulo del reporte y luego 35 pixeles después del titulo

$pdf->ezsetdy(-10);
$pdf->ezText("Distribuidora Vimportet",12,array('justification'=>'center'));
$pdf->ezsetdy(-5);

$pdf->ezsetdy(-10);
$pdf->ezText("Listado de Clientes",12,array('justification'=>'center'));
$pdf->ezsetdy(-10);

$pdf->selectFont('./fonts/Helvetica.afm');
$pdf->ezTable($tabla); //$tabla es la tabla que se lleno en el do while
$pdf->ezStream();
?>
```