**Introducción**

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. A partir de PHP 5, el modelo de objetos ha sido reescrito para permitir un mejor rendimiento y con más características. Este fue un cambio importante a partir de PHP 4. PHP 5 tiene un modelo de objetos completo.

Entre las características de PHP 5 están la inclusión de la visibilidad, las clases abstractas y clases y métodos finales, métodos mágicos adicionales, interfaces, clonación y tipos sugeridos.

PHP trata los objetos de la misma manera como referencias o manejadores, lo que significa que cada variable contiene una referencia de objeto en lugar de una copia de todo el objeto.

**Índice**

## Conceptos fundamentales

## Clase y Objeto

## Propiedades de una clase

## Herencia

## Practica

## Ejercicio

## Conceptos fundamentales

* **Objeto**

Un objeto es una entidad (tangible o intangible) caracterizada por sus atributos propios y cuyo comportamiento está determinado por las acciones o funciones que pueden modificarlo, así como también las acciones que requiere de otros objetos. Un objeto tiene identidad e inteligencia y constituye una unidad que oculta tanto datos como la descripción de su manipulación. Puede ser definido como una encapsulación y una abstracción: una encapsulación de atributos y servicios, y una abstracción del mundo real.

 Los servicios ofrecidos por los objetos son de dos tipos:

* Los datos, que llamamos **atributos**.
* Las acciones o funciones, que llamamos **métodos**.
* **Clase**

La clase puede definirse como la agrupación o colección de objetos que comparten una estructura común y un comportamiento común.

 Es una plantilla que contiene la descripción general de una colección de objetos. Consta de atributos y métodos que resumen las características y el comportamiento comunes de un conjunto de objetos.

 Todo objeto (también llamado instancia de una clase), pertenece a alguna clase. Mientras un objeto es una entidad concreta que existe en el tiempo y en el espacio, una clase representa solo una abstracción.

 Todos aquellos objetos que pertenecen a la misma clase son descritos o comparten el mismo conjunto de atributos y métodos. Todos los objetos de una clase tienen el mismo formato y comportamiento, son diferentes únicamente en los valores que contienen sus atributos. Todos ellos responden a los mismos mensajes.

* **Relación entre Clase y Objeto**

Algorítmicamente, las clases son descripciones netamente estáticas o plantillas que describen objetos. Su rol es definir nuevos tipos conformados por atributos y operaciones.

 Por el contrario, los objetos son instancias particulares de una clase. Las clases son una especie de molde de fábrica, en base al cual son construidos los objetos. Durante la ejecución de un programa sólo existen los objetos, no las clases.

* **Atributo**

Son los datos o variables que caracterizan al objeto y cuyos valores en un momento dado indican su estado.

Un atributo es una característica de un objeto. Mediante los atributos se define información oculta dentro de un objeto, la cual es manipulada solamente por los métodos definidos sobre dicho objeto.

 Los modos de acceso son:

* *Público:* Atributos (o Métodos) que son accesibles fuera de la clase. Pueden ser llamados por cualquier clase, aun si no está relacionada con ella.
* *Privado:* Atributos (o Métodos) que sólo son accesibles dentro de la implementación de la clase.
* *Protegido:* Atributos (o Métodos) que son accesibles para la propia clase y sus clases hijas (subclases).
* **Método**

 Son las operaciones (acciones o funciones) que se aplican sobre los objetos y que permiten crearlos, cambiar su estado o consultar el valor de sus atributos.

 Los métodos constituyen la secuencia de acciones que implementan las operaciones sobre los objetos. La implementación de los métodos no es visible fuera de objeto.

* **Constructor**

 Cada objeto o instancia de una clase debe ser creada explícitamente a través de un método u operación especial denominado Constructor. Los atributos de un objeto toman valores iniciales dados por el constructor.

* **Mensaje**

 Es la petición de un objeto a otro para solicitar la ejecución de alguno de sus métodos o para obtener el valor de un atributo público.

El Enfoque Orientado a Objeto se basa en cuatro principios que constituyen la base de todo desarrollo orientado a objetos. Estos principios son: la Abstracción, el Encapsulamiento, la Modularidad y la Herencia.

Otros elementos a destacar (aunque no fundamentales) en el EOO son: Polimorfismo, Enlace dinámico (o *binding*), Concurrencia y Persistencia.

* **Abstracción**

Es el principio de ignorar aquellos aspectos de un fenómeno observado que no son relevantes, con el objetivo de concentrarse en aquellos que si lo son. Una abstracción denota las características esenciales de un objeto (datos y operaciones), que lo distingue de otras clases de objetos. Decidir el conjunto correcto de abstracciones de un determinado dominio, es el problema central del diseño orientado a objetos.

* **Encapsulamiento (Ocultamiento de Información)**

Es la propiedad que permite ocultar al mundo exterior la representación interna del objeto. Esto quiere decir que el objeto puede ser utilizado, pero los datos esenciales del mismo no son conocidos fuera de él.

La idea central del encapsulamiento es esconder los detalles y mostrar lo relevante. Permite el ocultamiento de la información separando el aspecto correspondiente a la especificación de la implementación; de esta forma, distingue el "qué hacer" del "cómo hacer". La especificación es visible al usuario, mientras que la implementación se le oculta.

El encapsulamiento en un sistema orientado a objeto se representa en cada clase u objeto, definiendo sus atributos y métodos con los siguientes modos de acceso:

* *Público (+)* Atributos o Métodos que son accesibles fuera de la clase. Pueden ser llamados por cualquier clase, aun si no está relacionada con ella.
* *Privado (-)* Atributos o Métodos que solo son accesibles dentro de la implementación de la clase.
* *Protegido (#):* Atributos o Métodos que son accesibles para la propia clase y sus clases hijas (subclases).
* **Modularidad**

Es la propiedad que permite tener independencia entre las diferentes partes de un sistema. La modularidad consiste en dividir un programa en módulos o partes, que pueden ser compilados separadamente, pero que tienen conexiones con otros módulos. En un mismo módulo se suele colocar clases y objetos que guarden una estrecha relación. El sentido de modularidad está muy relacionado con el ocultamiento de información.

* **Herencia**

Es el proceso mediante el cual un objeto de una clase adquiere propiedades definidas en otra clase que lo preceda en una jerarquía de clasificaciones. Permite la definición de un nuevo objeto a partir de otros, agregando las diferencias entre ellos (Programación Diferencial), evitando repetición de código y permitiendo la reusabilidad.

Las clases heredan los datos y métodos de la superclase. Un método heredado puede ser sustituido por uno propio si ambos tienen el mismo nombre.

* **Polimorfismo**

 Es una propiedad que permite que un método tenga múltiples implementaciones, que se seleccionan en base al tipo objeto indicado al solicitar la ejecución del método.

 El polimorfismo operacional o Sobrecarga operacional permite aplicar operaciones con igual nombre a diferentes clases o están relacionados en términos de inclusión. En este tipo de polimorfismo, los métodos son interpretados en el contexto del objeto particular, ya que los métodos con nombres comunes son implementados de diferente manera dependiendo de cada clase.

1. **Clase y Objeto**

**CLASS**

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. Podemos crear atributos como si de variables se trataran, con la palabra reservada var

|  |
| --- |
| class NombreClase {var $atributo1;var $atributo2;} |

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 metodo2 le pasamos dos parámetros.

|  |
| --- |
| class NombreClase { function metodo1() { } function metodo2($param1, $param2) { }} |

1. **Ejemplo :**

|  |
| --- |
| <?phpclass ClaseEjemplo{ // Declaración de la propiedad public $var = 'a default value'; // Declaración del método public function displayVar() { echo $this->var; }}?> |

1. **Ejemplo:**

|  |
| --- |
| <?phpclass A{ function EjemploA() { if (isset($this)) { echo '$this está definida ('; echo get\_class($this); echo ")\n"; } else { echo "\$this no está definida.\n"; } }}class B{ function EjemploB() { // Nota: la siguiente línea arrojará un Warning si E\_STRICT está habilitada. A::EjemploA(); }}$a = new A();$a->EjemploA();// Nota: la siguiente línea arrojará un Warning si E\_STRICT está habilitada.A::EjemploA();$b = new B();$b->EjemploB();// Nota: la siguiente línea arrojará un Warning si E\_STRICT está habilitada.B::EjemploB();?> |

**Nota:**

**get\_class** : Devuelve el nombre de la clase de un objeto

**NEW**

Para crear una instancia de una clase, la palabra clave new debe ser usada. Un objeto siempre se creará a menos que el objeto tenga un constructor que arroje una excepción en caso de error. Las clases deberían ser definidas antes de la instanciación (y en algunos casos esto es un requerimiento).

Si un string que contiene el nombre de una clase se usa con new, una nueva instancia de esa clase será creada. Si la clase está en un espacio de nombres, su nombre completo debe ser usado cuando se hace esto.

1. **Ejemplo:**

|  |
| --- |
| <?phpclass ClaseEjemplo{ // Declaración de la propiedad public $var = 'a default value'; // Declaración del método public function displayVar() { echo $this->var; }}$instance = new ClaseEjemplo();// Esto también se puede hacer con variables:$className = 'ClaseEjemplo';$instance = new $className(); // ClaseEjemplo()?> |

**Objeto**

Cuando se asigna una instancia de una clase ya creada a una nueva variable, ésta última accederá a la misma instancia como al objeto que le fue asignado. Esta conducta es la misma cuando se pasan instancias a una función. Una copia de un objeto ya creado se puede lograr a través de la clonación de la misma.

1. **Ejemplo**

|  |
| --- |
| <?phpclass ClaseEjemplo{ // Declaración de la propiedad public $var = 'a default value'; // Declaración del método public function displayVar() { echo $this->var; }}$instance = new ClaseEjemplo();$assigned = $instance;$reference =& $instance;$instance->var = '$assigned tendrá este valor';$instance = null; // $instance y $reference se transforman en nullvar\_dump($instance);var\_dump($reference);var\_dump($assigned);?> |

**Nota:**

**var\_dump :** Vuelca información sobre una variable

**Constructor**

1. **Ejemplo**

|  |
| --- |
| <?phpclass BaseClass { function \_\_construct() { echo "En el constructor BaseClass <br>"; }}class SubClass extends BaseClass { function \_\_construct() { parent::\_\_construct(); echo "En el constructor SubClass <br>"; }}class OtherSubClass extends BaseClass { // heredando el constructor BaseClass}// En el constructor BaseClass$obj = new BaseClass();// En el constructor BaseClass// En el constructor SubClass$obj = new SubClass();// In BaseClass constructor$obj = new OtherSubClass();?> |

**Nota:**

void **\_\_construct** : PHP 5 permite a los desarrolladores declarar métodos constructores para las clases. Aquellas que tengan un método constructor lo invocarán en cada nuevo objeto creado, lo que lo hace idóneo para cualquier inicialización que el objeto pueda necesitar antes de ser usado.

## Propiedades de una clase

Las propiedades de las clases son los atributos del objeto como por ejemplo: el tamaño de una persona , el peso de una mesa. En php hay 3 niveles de acceso tanto para las propiedades como para los metodos los cuales son:

**Public:** Cuando le damos este nivel de acceso a un atributo, este puede ser accedido desde cualquier parte, así sea dentro de su misma clase como desde otra.

**Private:** Cuando se le da el nivel de acceso private estamos quitándole la libertad de acceso a las demás que intenten acceder solo puede ser accedida desde su propia clase donde este fue definido.

**Protected:** Cuando le damos el nivel de acceso protegido sigue restringiendo la libertad de acceso a los demás que no sean de la clase donde se esta declarando o de sus subclases.

|  |
| --- |
| <?phpclass Propiedades { public $a=1;private $b=2;protected $c=3;public function VerB() {return ($this->b);}public function VerC() { return ($this->b); }}$Ejemplo = new Propiedades ();echo "Valor public ".$Ejemplo->a."<br>";echo "Valor private mediante un metodo ".$Ejemplo->VerB()."<br>";echo "Valor protected mediante un metodo ".$Ejemplo->VerC()."<br>";//ERRORecho "Valor private ".$Ejemplo->b;//ERRORecho "Valor protected ".$Ejemplo->c;?> |

1. **Ejemplo**
2. **Herencia**

La herencia es un principio de programación bien establecido y PHP hace uso de él en su modelado de objetos. Este principio afectará la manera en que muchas clases y objetos se relacionan unas con otras.

Por ejemplo, cuando se extiende una clase, la subclase hereda todos los métodos públicos y protegidos de la clase padre. A menos que una clase sobrescriba esos métodos, mantendrán su funcionalidad original.

Esto es útil para la definición y abstracción de la funcionalidad y permite la implementación de funcionalidad adicional en objetos similares sin la necesidad de reimplementar toda la funcionalidad compartida.

1. **Ejemplo**

|  |
| --- |
| **<?php****class foo****{** **public function printItem($string)****{** **echo 'Foo: ' . $string . PHP\_EOL .'<br>';** **}** **public function printPHP()** **{** **echo 'PHP is great.' . PHP\_EOL .'<br>';** **}****}****class bar extends foo****{****public function printItem($string)****{** **echo 'Bar: ' . $string . PHP\_EOL .'<br>';** **}****}****$foo = new foo();****$bar = new bar();****$foo->printItem('baz'); // Salida: 'Foo: baz'****$foo->printPHP(); // Salida: 'PHP is great'** **$bar->printItem('baz'); // Salida: 'Bar: baz'****$bar->printPHP(); // Salida: 'PHP is great'****?>** |

1. **Practica**

|  |  |
| --- | --- |
| **Ejercicio # 1** | **Ejercicio # 2** |
| <?phpclass cuadrado { // Estos son ATRIBUTOS de los objetos var $num=10; // Este es el METODO para calcular function calcularCuadrado() {  return ($this->num \* $this->num); } }// Creamos el Objeto$objeto = new cuadrado();//Asignamos un atributo//$objeto->num = 3;// Invocamos un métodoecho $objeto->calcularCuadrado();?> | <?php class imagen {// Estos son ATRIBUTOS de los objetos var $src;  var $border;  // Esta función es el CONSTRUCTOR function imagen($src,$border) {  $this->src=$src;  $this->border=$border;  } // Esta función es un METODO function Imprimir() {  echo " <img src=";  echo $this->src;  echo " border=";  echo $this->border;  echo " >";  }} // Creamos el Objeto$logo = new Imagen("imagen/msn2.jpg",8); // Invocamos el método$logo->Imprimir(); ?> |

|  |  |
| --- | --- |
| **Ejercicio # 3** | **Ejercicio # 4** |
| <? class Aritmetica { // Estos son ATRIBUTOS de los objetos var $a; var $b;// Este es el CONSTRUCTOR function Aritmetica($a, $b) {  $this->a = $a; $this->b = $b; }// Este es el METODO para sumar function sumar() {  return ($this->a + $this->b); }// Este es el METODO para restar function restar() {  return ($this->a - $this->b); } // Este es el METODO para multiplicar function multiplicar() {  return ($this->a \* $this->b); }// Este es el METODO para dividir function dividir() {  return ($this->a / $this->b); }}// Creamos el Objeto$objeto = new Aritmetica(5, 8);// Invocamos un métodoecho $objeto->sumar();?> | <? // Superclaseclass cuadrado { // Estos son ATRIBUTOS de los objetos var $num; // Este es el METODO para calcular el cuadrado function calcularCuadrado() {  return ($this->num \* $this->num); } }//Subclaseclass cubo extends cuadrado{ // Este es el METODO para calcular el Cubo function calcularCubo() {  return ($this->calcularCuadrado() \* $this->num); } }// Creamos el Objeto$objeto = new cubo();//Asignamos un atributo$objeto->num = 3;// Invocamos un métodoecho $objeto->calcularCubo();?> |

|  |  |
| --- | --- |
| **Ejercicio # 5 Grafico de Linea** | **Ejercicio # 5 Grafico de Barra** |
| <?php include\_once('jpgraph/jpgraph.php'); include\_once('jpgraph/jpgraph\_line.php'); $minimo = array(19, 20, 22, 24, 25, 25, 25, 25, 24, 24, 21, 20); $maximo = array(26, 27, 29, 31, 32, 32, 32, 32, 32, 31, 29, 27); $promedio = array(22.5, 23.5, 25.5, 27.5, 28.5, 28.5, 28.5, 28.5, 28.0, 27.5, 25.0, 23.5);  $graph = new Graph(650, 450); $graph->SetScale('textlin',18, 38); // Escala linear en x $graph->SetMargin(50,60,40,45); $graph->SetMarginColor('white'); $graph->title->Set('Temperatura en Veracruz, Ver. México'); $graph->xaxis->title->Set('Mes');  $gDateLocale->Set('spanish'); //Obtiene los meses del año $mes = $gDateLocale->GetShortMonth();  $graph->xaxis->SetTickLabels($mes); //$graph->xaxis->SetFont(FF\_ARIAL,FS\_BOLD,8); $graph->yaxis->title->Set('Grados centígrados'); //$graph->yaxis->SetFont(FF\_ARIAL,FS\_BOLD,8); $tminimo = new LinePlot($minimo); $tminimo->SetLegend('Mínimo'); $tminimo->SetColor('blue'); $tpromedio = new LinePlot($promedio); $tpromedio->SetLegend('Promedio'); $tpromedio->SetColor('green'); $tmaximo = new LinePlot($maximo); $tmaximo->SetColor('red'); $tmaximo->SetLegend('Máximo');  $graph->Add($tminimo); $graph->Add($tpromedio); $graph->Add($tmaximo); $graph->Stroke();?> | <?php  require\_once ('jpgraph/jpgraph.php'); require\_once ('jpgraph/jpgraph\_bar.php'); $psemestre = array(74, 75, 75, 75, 100, 75, 75, 75, 100); $cero = array(90, 36, 72, 90, 90, 90, 61, 61, 90); $carreras = array('Admin', 'Bioquímica','Electrónica', 'Electríca','Industrial','Mecánica','Mecatrónica','Química','Sistemas');  $grafica = new Graph(650, 450); // Crea la gráfica  $grafica->SetScale('textlin', 0, 120); //$grafica->SetShadow(); $grafica->SetMarginColor('white:1.1'); $grafica->legend->SetPos(0.5,0.97,'center','bottom'); $serie1 = new BarPlot($psemestre); // Crea las barras $serie1->SetLegend('1er. Semestre'); $serie1->value->Show(); $serie1->value->SetFormat('%0.0f'); $serie1->SetWidth(0.6); $serie1->SetFillGradient('navy:0.9','navy:1.85',GRAD\_LEFT\_REFLECTION); $serie2 = new BarPlot($cero); $serie2->SetLegend('Sem. Cero'); $serie2->value->Show(); $serie2->value->SetFormat('%0.0f'); $serie2->SetWidth(0.6); $serie2->SetFillGradient('red:0.95','red:1.85',GRAD\_RIGHT\_REFLECTION); $gbplot = new GroupBarPlot(array($serie1,$serie2)); // Create the grouped bar plot  $grafica->Add($gbplot); // Suma la serie a la gráfica $grafica->title->Set('Resultado oficial examen admisión I.T.V 2010'); $grafica->xaxis->title->Set('Carrera'); $grafica->xaxis->SetTickLabels($carreras); $grafica->yaxis->title->Set('Número de aceptados'); $grafica->Stroke(); // Despliega la gráfica?> |

|  |  |
| --- | --- |
| **Ejercicio # 6 Grafico de pastel** | **Ejercicio # 5 General PDF** |
| <?php // Ejemplo con gráfica de pastel con círculo central require\_once ("jpgraph/jpgraph.php"); require\_once ("jpgraph/jpgraph\_pie.php");  $datos = array(16, 30); // Datos  $grafica = new PieGraph(600,400,'auto'); // Crear Gráfico $grafica->SetFrame(true); // Despliega el borde $grafica->SetShadow(); $grafica->title->Set("Alumnos por genero Seguridad en Redes. Verano 2010. I.T.V"); $grafica->title->SetMargin(8); // Add a little bit more margin from the top  // Crea la gráfica $genero = new PiePlotC($datos); $genero->SetSize(0.35); $genero->value->SetColor('white'); $genero->value->Show(); $genero->midtitle->Set("Genero"); $genero->SetMidColor('yellow'); $genero->SetLabelType(PIE\_VALUE\_PER);  $lbl = array("Mujeres\n%.1f%%","Hombres\n%.1f%%"); $genero->SetLabels($lbl); $genero->SetShadow();  $genero->ExplodeAll(15); // Explode all slices 15 pixels $grafica->Add($genero); // Add plot to pie graph $grafica->Stroke(); //Despliega la gráfica?> | <?phpinclude('class.ezpdf.php'); $pdf = new Cezpdf('A4','landscape');$pdf->selectFont('fonts/Helvetica.afm'); $datacreator = array ( 'Title'=>'Mi Archivo PDF, creado en PHP', 'Subject'=>'PDF con otras opciones', 'Author'=>'Rut', 'Producer'=>'UNEWEB' );$pdf->addInfo($datacreator); $data[] = array('pais'=>'Peru', 'capital'=>'Lima');$data[] = array('pais'=>'Colombia', 'capital'=>'Bogota');$data[] = array('pais'=>'Chile', 'capital'=>'Santiago de Chile');$data[] = array('pais'=>'Brasil', 'capital'=>'Brasilia');$data[] = array('pais'=>'Ecuador', 'capital'=>'Quito');$data[] = array('pais'=>'Bolivia', 'capital'=>'La Paz');$data[] = array('pais'=>'Argentina', 'capital'=>'Buenos Aires');$data[] = array('pais'=>'Guyana', 'capital'=>'Georgetown');$data[] = array('pais'=>'Surinam', 'capital'=>'Paramaribo');$data[] = array('pais'=>'Uruguay', 'capital'=>'Montevideo');$data[] = array('pais'=>'Paraguay', 'capital'=>'Asuncion');$data[] = array('pais'=>'Venezuela', 'capital'=>'Caracas'); $titles = array('pais'=>'<b>Pais</b>', 'capital'=>'<b>Capital</b>'); $pdf->ezImage('Imagen.jpg',0, 300, 'none', 'center',array('color'=>'black')); $pdf->ezText("<b>PDF en PHP</b> ",16);$pdf->ezText("Paises Sudamericanos\n",12,array('justification'=>'center')); $pdf->ezTable($data,$titles,'',$options ); $pdf->ezText("\n\n\n",10);$pdf->ezText("<b>Fecha:</b> ".date("d/m/Y"),10);$pdf->ezText("<b>Hora:</b> ".date("H:i:s")."\n\n",10); $pdf->ezText("<b>Prueba</b>\n",16);ob\_end\_clean();$pdf->ezStream();?> |

|  |  |
| --- | --- |
| **Ejercicio # 8 Base Datos** | **Ejercicio # 9 Ejecución Base Datos** |
| <?php class oRG\_BDConector { var $servidor;  var $usuario;  var $password;  var $bd;  var $consulta;  var $enlace; var $resultado; function oRG\_BDConector($servidor,$usuario,$password,$bd)  { $this->servidor=$servidor; $this->usuario=$usuario; $this->password=$password; $this->bd=$bd; } function oConectar($enlace=NULL)  { $this->enlace=mysql\_connect($this->servidor,$this->usuario,$this->password); mysql\_select\_db($this->bd,$this->enlace); return $this->enlace;  } function oEjecutar($query)  { $this->consulta=mysql\_query($query, $this->enlace) or die('Mysql dijo: '.mysql\_error()); return $this->consulta; } function oUltimo() { return mysql\_insert\_id(); } function oNumreg()  { $this->total=mysql\_num\_rows($this->consulta); return $this->total; } function oNumregaf()  { $this->total=mysql\_affected\_rows($this->consulta); return $this->total; } function oDatosarray($consulta=NULL)  {  if($consulta) $this->consulta=$consulta; $this->resultado=mysql\_fetch\_array($this->consulta); return $this->resultado; } public function oLimpiaconsulta() {  mysql\_free\_result($this->consulta); } public function oCerrarconexion()  { mysql\_close($this->enlace); } public function oError() {  return $this -> e\_rror = mysql\_error();  }}?> | <?php include("BaseDatos.php");$BaseDatos = new oRG\_BDConector("localhost","root","","Ejemplo");$BaseDatos->oConectar();$query="Select nombre, cirif from usuario ";$Resultado = $BaseDatos->oEjecutar($query);while($row=mysql\_fetch\_array($Resultado)){ echo"Nombre: ".$row['nombre']."<br>";echo"Cedula: ".$row['cirif']."<br>";}?> |

1. **Ejercicios**

Realizar:

1. Un sistema web en PHP y Mysql para una escuela, donde cada maestro pueda cargar las notas de sus alumnos, consultar las notas , visualizar en un archivo con todas las notas cargadas y obtener la siguiente estadística:
2. ¿Cuántos alumnos reprobaron dicha clase?
3. ¿Cuantos alumnos Pasaron dicha clase?
4. ¿Cuantos alumnos sacaron 20-15 puntos, 15-10 puntos, 10-5 puntos y 5-0 puntos?
5. ¿Cuantos alumnos tiene la nota más alta?
6. ¿Cuantos alumnos tiene la nota más baja?

Cada maestro tienes que ingresar la información del la clase que dicta:

* Nombre Maestro
* Cedula Maestro
* Grado
* Sección
* Numero de Alumnos

Luego tiene que ingresar información de cada alumno y su nota

* Nombre Alumno
* Cedula Alumno
* Nota

Dicho sistema tiene que cumplir con el siguiente requerimiento:

1. Clase para Manejo de Base de Datos y que realice las siguientes acciones:
2. Conexión a base de datos.
3. Ejecución de Sentencias SQL.
4. Verificación de data Consultada o Afectada.
5. Manejo de formularios.
6. Cree y Maneje Formularios.
7. Cree y Maneje Campos.
8. Todos los cálculos que se realice deben estar en clases y métodos
9. Manejar grafico
10. Descargar un Archivo pdf