	UNIVERSIDAD DON BOSCO **FACULTAD DE INGENIERÍA** **ESCUELA DE COMPUTACIÓN**
CICLO I-2021	**Lenguajes Interpretados en el Servidor** **Guía de práctica No. 8** **Programación Orientada a Objetos PHP**

I. OBJETIVOS

- Dar una visión general de la metodología de programación orientada a objetos.
- Estar en capacidad de diseñar clases y crear objetos a partir de aquellas.
- Hacer uso de constructores y destructores en el diseño de clases.
- Hacer uso del control de acceso a las propiedades y métodos de una clase.

II. INTRODUCCIÓN TEÓRICA

Programación Orientada a Objetos

El lenguaje PHP 5 ha sido rediseñado por completo para dar a los programadores todas las herramientas que un verdadero lenguaje orientado a objetos debe poseer.

La Programación Orientada a Objetos (POO) es un enfoque de programación en el que el diseño y desarrollo del software se fundamenta en la modelización de las características y comportamientos de elementos o sucesos reales o abstractos mediante el uso de clases y objetos. Una clase es una descripción genérica o plantilla de un determinado tipo de objetos. Los objetos, por su parte, se crean a partir de estas clases. De manera que cada vez que se crea un objeto, se dice que se está creando una instancia de la clase. Los objetos, por tanto, poseen dos características importantes, que son: el estado y el comportamiento. El estado se define mediante un conjunto de propiedades, en tanto que, el comportamiento se implementa mediante métodos. Puede considerarse al método como las operaciones que es posible llevar a cabo con las propiedades del objeto.

Creación de clases y objetos con PHP

Para crear una clase con PHP5 se utiliza la palabra reservada class, seguida por el nombre que se le asignará a la clase. Vea la siguiente sintaxis:

```
class nombre_clase {
//propiedades de la clase;
//métodos de la clase;
}
```

Las propiedades o atributos de la clase se declaran mediante el uso de variables a las cuales se les especifica un control de acceso mediante el uso de las palabras reservadas:

• public: este es el modificador de acceso predeterminado e indica que la propiedad o método será accesible desde cualquier punto del script.

• prívate: indica que el miembro de la clase se podrá acceder únicamente desde el interior de la clase.
• protected: significa que la propiedad o método, sólo será accesible desde el interior de la clase o desde sus clases derivadas.

Ejemplo:

```
class claseEjemplo {
//Propiedades
public $publicprop = 'Soy propiedad pública';
private $privateprop = 'Soy propiedad privada';
protected $protectedprop = 'Soy propiedad protegida';
//Métodos
function metodoEjemplo(){
echo $this->publicprop;
echo $this->privateprop;
echo $this->protectedprop;
}
}
//Instanciando un objeto de la clase
$obj1 = new claseEjemplo();
$obj1->publicprop = 'Soy pública';
$obj1->privateprop = 'Soy privada'; //Generará un error.
$obj1->protected = 'Soy protegida'; //Generará un error.
$obj1->metodoEjemplo();
```

Constructores y destructores

Un constructor es un método especial que es invocado de forma automática, cada vez que se crea una nueva instancia de la clase; es decir, cada vez que se crea un nuevo objeto a partir de la clase. El constructor por defecto realiza tareas de inicialización como establecer atributos con valores de inicio apropiados o crear otros objetos necesarios. Los constructores en PHP5 tienen un nombre especial que se muestra a continuación:

```
function __construct(){
//Establecer valores iniciales a propiedades;
}
En la versión 4, los constructores tenían el mismo nombre de la clase.
function nombreClase(){
//Establecer valores iniciales a propiedades;
}
```

Los destructores tienen el propósito de liberar recursos del servidor al terminar de ejecutar un script en el que se han creado objetos a partir de una clase. Además de esto, permiten implementar alguna funcionalidad concreta antes de que se destruya la clase. Un destructor en PHP5 se construye de la siguiente forma:

```
function __destruct(){
//Liberar recursos del sistema;
}
```

Creación de clases con PHP

La creación de clases en PHP se hace con la palabra reservada class. En el interior de la clase deben indicarse las propiedades o atributos para la clase. Esto en la práctica requiere de la especificación de variables que deben ser precedidas por alguno de los modificadores de acceso, como public (acceso público, que significa que puede accederse a la propiedad o método desde cualquier parte del script), private (acceso privado a la clase, únicamente desde la clase en la que está definida la propiedad) y protected (acceso protegido, que únicamente permitirá el acceso a la propiedad o método desde la clase misma o desde cualquier clase que se derive de ella).

Para comprender esto examinemos el siguiente código:

```
class persona {
//Propiedades
private $nombrecompleto;
//Métodos de la clase
function asignarNombre($nombre, $apellido){
$this->nombrecompleto = $nombre . " " . $apellido;
}
function decirNombre(){
return $this->nombre;
}
}
```

Si se crea la siguiente instancia de la clase persona:

```
$unapersona = new persona();
$unapersona->nombrecompleto = "Jorge Bustamante";
```

Al ejecutar el script obtendríamos un error, ya que la propiedad $nombrecompleto es privada. Por lo tanto, sólo puede accederse a ella desde alguno de los métodos definidos dentro de la clase.

Ahora bien, si realizamos un cambio en la declaración de la propiedad $nombrecompleto, colocando el modificador de acceso public, en lugar de, prívate. El mismo código anterior funcionaría correctamente.

```
class persona {
//Propiedades
public $nombrecompleto;
//Métodos de la clase
function asignarNombre($nombre, $apellido){
$this->nombrecompleto = $nombre . " " . $apellido;
}
function decirNombre(){
return $this->nombre;
}
}
```

Creación de la instancia de la clase y acceso a una de sus propiedades públicas:
$unapersona = new persona();
$unapersona->nombrecompleto = "Jorge Bustamante";

Herencia de clases

La herencia en Programación Orientada a Objetos es la relación que se da entre dos clases por la cual una de ellas, a la que se denominará clase hija, subclase o clase derivada, además de poseer sus propias propiedades y métodos (o incluso constantes), tiene a su disposición; o lo que es lo mismo, hereda, los miembros definidos en la otra, denominada clase padre o superclase.

También se puede ver la herencia como la capacidad que tiene una clase de extender su funcionalidad. Esto se debe a que una clase hija también puede volver a definir algunos o todos los métodos, propiedades y constantes de la clase padre para proporcionar una funcionalidad adicional o diferente a la clase.

Ejemplo:

```
class computer { //Esta es la superclase
//Propiedades de la superclase
private $password; //propiedad visible únicamente por esta clase
protected $userID; //propiedad visible por esta clase y sus clases derivadas
public $printer;
//Constructor de la superclase
function __construct(){
echo "Llamada al constructor del padre:<br>\n";
$this->userID = "estudiante";
$this->password = "Pa$$w0rd";
}
}
//Extendiendo la clase computer
class laptop extends computer{ //Subclase
//Propiedades de la clase hija
public $brand;
public $weight;
private $password = "newPa$$w0rd";
//Constructor de la clase hija
function __construct($brand, $weight){
parent::construct(); //Llamada al constructor de la clase padre
echo "Llamada al propio constructor de la clase hija
$this->brand = $brand;
$this->weight = $weight;
}
}
//Aplicación que utiliza la clase
$pc = new computer();
$portable = new laptop("Waio","6.0");
$pc->printer = "Lexmark 1100";
```

```
$portable->printer = "Epson Stylus 3i";
//echo "$portable->password<br>\n"; //Arrojará un error fatal
//echo "$pc->password<br>\n"; //Arrojará también un error fatal
echo "<pre>";
//Obtenemos las propiedades públicas disponibles
print_r(get_object_vars($pc));
print_r(get_object_vars($portable));
echo "</pre>";
```

III. MATERIALES Y EQUIPO

Para la realización de la guía de práctica se requerirá lo siguiente:

No.	Material	Cantidad
1	Guía de práctica #8: Programación orientada a objetos PHP	1
2	Computadora con WampServer instalado y funcionando correctamente	1
3	Editor PHP sublime Text o Eclipse PHP	1
4	Memoria USB o disco flexible	1

IV. PROCEDIMIENTO

Indicaciones: Asegúrese de digitar el código de los siguientes ejemplos que se presentan a continuación. Tenga en cuenta que el PHP Designer no es compilador solamente un editor. Por lo tanto, los errores de sintaxis los podrá observar únicamente hasta que se ejecute el script cargando la página en el navegador de su preferencia.

Ejemplo 1: El siguiente ejemplo muestra una aplicación orientada a objetos donde se define una clase auto con propiedades como la marca, el modelo, el color y una imagen del auto, un método constructor para inicializar las propiedades del objeto y un método más para mostrar la información de todos los autos existentes.

Archivo 1: auto.class.php

```php
<?php
//Definición de la clase
class auto   {
    //Propiedades de la clase auto
    private $marca;
    private $modelo;
    private $color;
    private $image;

    //Método constructor
    function __construct($marca='Honda', $modelo='Civic',
```

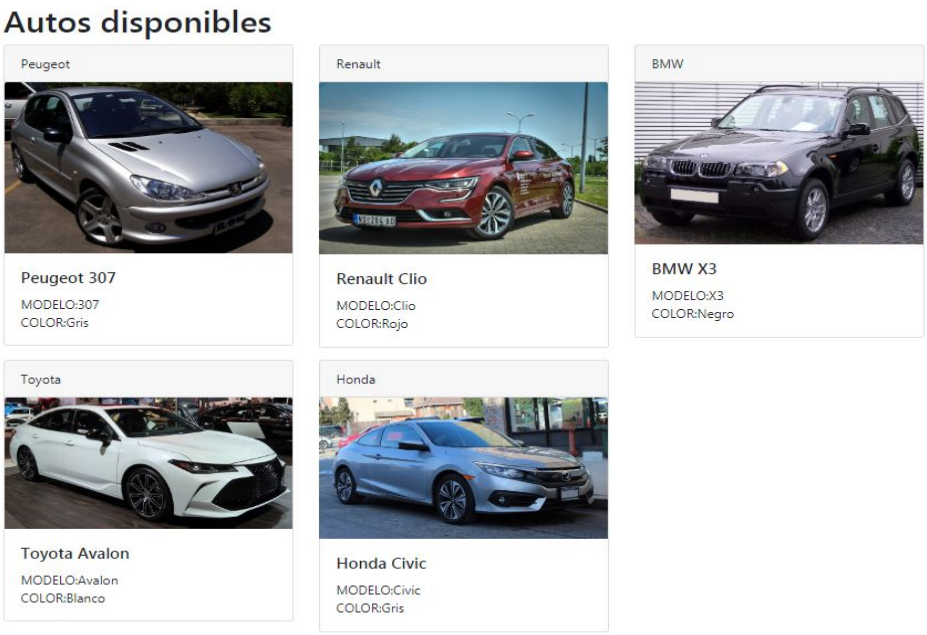
```
$color='Gris', $image='img/hondacivic.jpg'){
        //El constructor inicializada los valores de las
propiedades
        //del objeto con los valores recibido en los argumentos
        //del método constructor.
        $this->marca = $marca;
        $this->modelo = $modelo;
        $this->color = $color;
        $this->image = $image;
    }

    //Métodos de la clase
    function mostrar(){
        //El método mostrar() crea una tabla HTML donde se
muestran
        //los detalles del objeto auto, como la marca, una imagen,
        //el modelo y el color del auto.
        $tabla  = "<div class='col-4 mb-3'>";

        $tabla  .="<div class='card'>";
        $tabla  .="<div class='card-header'>". $this->marca
."</div>";
        $tabla  .="<img class='card-img-top' src='". $this->image
."' alt='Card image cap'>";
        $tabla  .="<div class='card-body'>";
        $tabla  .="<h5 class='card-title'>". $this->marca ." ".
$this->modelo ."</h5>";
        $tabla  .="<p class='card-text'> MODELO:". $this->modelo
."<br>";
        $tabla .= "COLOR:". $this->color ."</p>";
        $tabla  .="</div>";
        $tabla  .="</div>";
        $tabla  .="</div>";
        echo $tabla;
    }
}
?>
```

Archivo 2: autospoo.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/boots
trap.min.css"
```

```
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPG
FN9MuhOf23Q9Ifjh" crossorigin="anonymous">

    <title>Venta de autos</title>

    <!--[if lt IE 9]>
        <script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>
<body>

<div class="container">
<header>
    <h1>Autos disponibles</h1>
</header>
<div class="row">
<?php

  //Incluyendo el archivo de clase
  function __autoload($classname) {
      include_once("class/" . $classname . ".class.php");
  }
  //Creando los objetos para cada tipo de auto. Notar que se están
  //asignando a elementos de una matriz que tendrá por nombre
$movil
  $movil[0] = new auto("Peugeot", "307", "Gris",
"img/peugeot.jpg");
  $movil[1] = new auto("Renault", "Clio", "Rojo",
"img/renaultclio.jpg");
  $movil[2] = new auto("BMW", "X3", "Negro", "img/bmwserie6.jpg");
  $movil[3] = new auto("Toyota", "Avalon", "Blanco",
"img/toyota.jpg");
  //Esta llamada mostrará los valores por defecto en los
argumentos
  //del método constructor.
  $movil[4] = new auto();
  //Mostrando la tabla con los autos disponibles
  for($i=0; $i<count($movil); $i++){
    $movil[$i]->mostrar();
    }
?>
</div>
</div>
</body>
</html>
```

El resultado en el navegador seria:



Ejemplo #2: Ejemplo de manejo de cuentas de ahorro con clases y objetos en el que se manejan los tres tipos de operaciones básicas con cuentas de ahorro, como son la apertura de la cuenta, depósitos y retiros de efectivo a la cuenta. En todas las operaciones se piden siempre el nombre del titular de la cuenta y la cantidad de dinero con el que se va a aperturar la cuenta, o la cantidad que se depositará o retirará de dicha cuenta.

Archivo 1: bankaccount.class.php

```
<?php
  class bankAccount {
      //Propiedades de la clase
      private static $numberAccount = 0;
      protected $idcuenta;
      private $owner;
      private $balance = 0.0;

      //Métodos de la clase
      function openAccount($owner, $amount){
         self::$numberAccount++;
         $this->idcuenta = self::$numberAccount;
         $this->owner = $owner;
         $this->balance = $amount;

         $comprobante  = "<table class='table table-hover'>";
         $comprobante .=" <thead> <td colspan='2' scope='col'>DATOS
DE TRANSACCIÓN<td></thead>";
         $comprobante .= "<tr scope='row'>\n\t\t<td>Número de
cuenta: </td>\n";
         $comprobante .= "<td>\$ " . self::$numberAccount .
```

```
"</td></tr>";
         $comprobante .= "<td>Propietario: </td>\n";
         $comprobante .= "<td>\$ " . $this->owner .  "</td></tr>";
         $comprobante .= "<tr>\n<td class='success'>Saldo inicial:
</td>";
         $comprobante .= "<td>\$ " .
number_format($this->balance,2,'.',',') . "</td></tr>";
         $comprobante .= "</table>\n";
         echo $comprobante;
      }

      function makeDeposit($amount, $saldo=250){
         //Se añade al saldo actual la cantidad ($amount)
         //recibida como argumento del método
         $this->balance = $saldo;
         $this->balance += $amount;
         $comprobante  = "<table class='table table-hover'>";
         $comprobante .=" <thead> <td colspan='2' scope='col'>DATOS
DE TRANSACCIÓN<td></thead>";
         $comprobante .= "<tr scope='row'>\n\t\t<td>Saldo inicial:
</td>\n";
         $comprobante .= "<td>\$ " .
number_format($saldo,2,".",",") . "</td></tr>";
         $comprobante .= "<td>Cantidad depositada: </td>\n";
         $comprobante .= "<td>\$ " .
number_format($amount,2,'.',',') . "</td></tr>";
         $comprobante .= "<tr>\n<td class='success'>Nuevo saldo:
</td>";
         $comprobante .= "<td>\$ " .
number_format($this->balance,2,'.',',') . "</td></tr>";
         $comprobante .= "</table>\n";
         echo $comprobante;
      }

      function makeWithdrawal($amount, $saldo=250){
         //Se resta del saldo actual de la cuenta
         //la cantidad ($amount) recibida como argumento
         $this->balance = $saldo;
         $saldoinicial = $this->balance;
         $this->balance -= $amount;
         if($this->balance > 0) {
             $comprobante  = "<table class='table table-hover'>";
             $comprobante .=" <thead> <td colspan='2'
scope='col'>DATOS DE TRANSACCIÓN<td></thead>";
             $comprobante .= "<tr scope='row'>\n\t\t<td>Saldo
inicial: </td>\n";
             $comprobante .= "<td>\$ " .
number_format($saldoinicial,2,".",",") . "</td></tr>";
             $comprobante .= "<td>Cantidad retirada: </td>\n";
             $comprobante .= "<td>\$ " .
```

```
number_format($amount,2,'.',',') . "</td></tr>";
            $comprobante .= "<tr>\n<td class='success'>Nuevo
saldo: </td>";
            $comprobante .= "<td>\$ " .
number_format($this->balance,2,'.',',') . "</td></tr>";
            $comprobante .= "</table>\n";
        }
        else {
            $comprobante  = "<table class='table table-hover'>";
            $comprobante .= "<tr><td>Aviso: </td>";
            $comprobante .= "<td>Su cuenta presenta insuficiendia
de fondos.</td></tr>";
            $comprobante .= "</table>";
        }
        echo $comprobante;
    }

    function getBalance(){
        return $this->balance;
    }
  }
?>
```

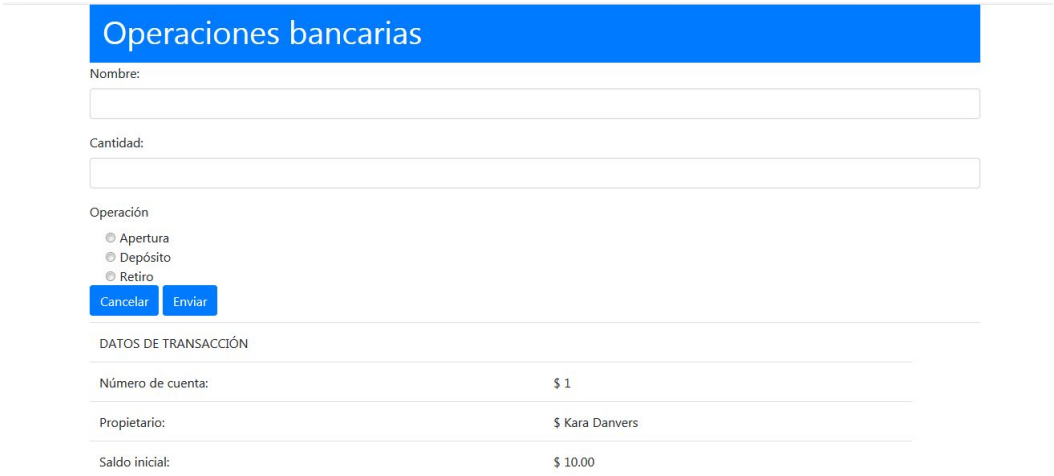
Archivo 1: bankform.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1">
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/boots
trap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeWPG
FN9MuhOf23Q9Ifjh" crossorigin="anonymous">
    <title>Cuentas de ahorro</title>
    <!--[if lt IE 9]>
      <script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>
<body>
<section class="container">
<nav class="navbar navbar-dark bg-primary text-white">
<h1>Operaciones bancarias</h1>
</nav>
    <article>
```

```
<form name="operaciones" id="operaciones" method="POST"
action="<?php echo $_SERVER['PHP_SELF'] ?>">


      <div class="form-group">
      <label for="nombre">Nombre:</label>
            <div class="campo">
               <input class="form-control" type="text"
name="nombre" size="25" maxlength="40" />
            </div>
      </div>
      <div class="form-group">
      <label for="cantidad">Cantidad:</label>
            <div class="campo">
               <input class="form-control" type="text"
name="cantidad" size="8" maxlength="10" />
            </div>
      </div>
      <label for="operacion">Operación</label>
            <div class="opciones">
            <div class="form-check">
               <input type="radio" name="operacion"
value="apertura" />
               <label class="form-check-label" >Apertura</label>
               </div>
               <div class="form-check">
               <input type="radio" name="operacion"
value="deposito" />
               <label class="form-check-label" >Depósito</label>
               </div>
               <div class="form-check">
               <input type="radio" name="operacion" value="retiro"
/>
               <label class="form-check-label" >Retiro</label>
              </div>
            </div>
      </div>
               <input type="reset" class="btn btn-primary mb-2"
name="restablecer" value="Cancelar" />
               <input type="submit" class="btn btn-primary mb-2"
name="enviar" value="Enviar" />

</form>
<?php
  function __autoload($classname){
      include_once("class/" . $classname . ".class.php");
  }
  if(isset($_POST['enviar'])){
      $msg = "";
      $titular = isset($_POST['nombre']) ? $_POST['nombre'] : "";
```

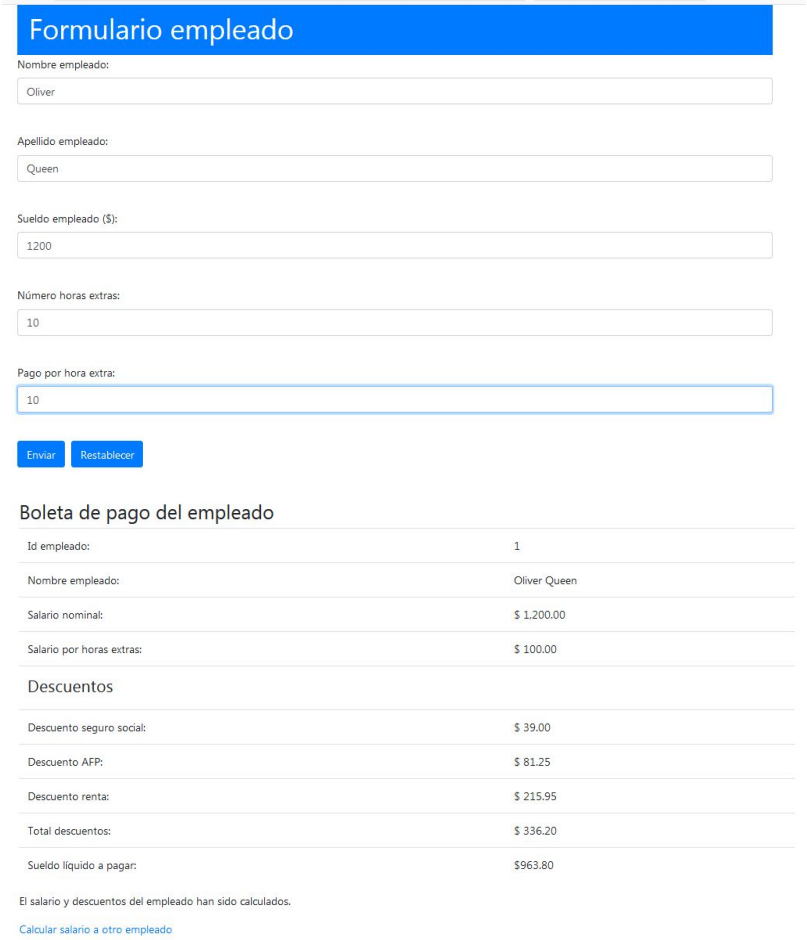
```
      if($titular == ""){
          $msg  = "<h2>El nombre de la cuenta no puede estar
vacío</h2><br />";
      }
      $cantidad = isset($_POST['cantidad']) &&
is_numeric($_POST['cantidad']) ? $_POST['cantidad'] : 0;
      if($cantidad == 0 || $cantidad < 0){
         $msg .= "<h2>La cantidad no puede ser negativa, ni
cero.</h2><br />";
      }
      if($msg != ""){
          echo $msg;
          echo "<a href=\"{$_SERVER['PHP_SELF']}\">Volver al
formulario</a><br />";
          exit(0);
      }
      $operacion = isset($_POST['operacion']) ?
$_POST['operacion'] : "apertura";
      $nuevacuenta = new bankAccount();
      switch($operacion){
          case "apertura":
              $nuevacuenta->openAccount($titular, $cantidad);
              break;
          case "deposito":
              $nuevacuenta->makeDeposit($cantidad);
              break;
          case "retiro":
              $nuevacuenta->makeWithdrawal($cantidad);
              break;
      }
   }
?>
   </article>
</section>
</body>
</html>
```

En el navegador de nuestra preferencia podremos visualizar el formulario para ingresar los datos solicitados y al enviarlos la página que nos muestra la operación realizada:



Ejercicio #3: Una empresa desea crear un sistema que le permita obtener una boleta del pago que realiza a cada uno de sus empleados. En la boleta debe reflejarse el salario nominal, el detalle del descuento por seguro social (ISSS), administradora de pensiones (AFP) y renta, teniendo en cuenta las consideraciones de ley en su aplicación. Se deben totalizar esos descuentos y mostrar el total de descuentos realizados. Además, la empresa paga las horas extras a $10.00, por lo que a parte del salario nominal el empleado recibe remuneración adicional por las horas extras. Los descuentos deben aplicarse tanto al salario como a las entradas por horas extras. Por último, debe mostrar también el sueldo líquido a pagar al empleado. Realice esta aplicación con Programación Orientada a Objetos.

El formulario para ingreso debe ser como el siguiente junto con la boleta generada:



Archivo 1: empleado.class.php

```php
<?php
  //Definición de la clase empleado
  class empleado     {
     //Estableciendo las propiedades de la clase
     private static $idEmpleado = 0;
     private $nombre;
     private $apellido;
     private $isss;
     private $renta;
     private $afp;
     private $sueldoNominal;
     private $sueldoLiquido;
     private $pagoxhoraextra;
     //Declaración de constantes para los descuentos del empleado
     //Se inicializan porque pertenecen a la clase
     const descISSS = 0.03;
     const descRENTA = 0.075;
     const descAFP = 0.0625;
```

```
    //Constructor de la clase
    function __construct(){
       self::$idEmpleado++;
       $this->nombre = "";
       $this->apellido = "";
       $this->sueldoLiquido = 0.0;
       $this->pagoxhoraextra = 0.0;
    }

    //Destructor de la clase
    function __destruct(){
       echo "<p class=\"msg\">El salario y descuentos del
empleado han sido calculados.</p>";
       $backlink  = "<a class=\"a-btn\"
href=\"sueldoneto.php\">";
       $backlink .= "<span class=\"a-btn-text\">Calcular salario
</span>";
       $backlink .= "<span class=\"a-btn-slide-text\">a otro
empleado</span>";
       $backlink .= "<span class=\"a-btn-slide-icon\"></span>";
       $backlink .= "</a>";
       echo $backlink;

    }

    //Métodos de la clase empleado
    function obtenerSalarioNeto($nombre, $apellido, $salario,
$horasextras, $pagoxhoraextra=0.0){
       $this->nombre = $nombre;
       $this->apellido = $apellido;
       $this->pagoxhoraextra = $horasextras * $pagoxhoraextra;
       $this->sueldoNominal = $salario;
       if($this->pagoxhoraextra > 0) {
           $this->isss = ($salario + $this->pagoxhoraextra) *
self::descISSS;
           $this->afp = ($salario + $this->pagoxhoraextra) *
self::descAFP;
       }
       else {
           $this->isss = $salario * self::descISSS;
           $this->afp = $salario * self::descAFP;
       }
       $salariocondescuento = $this->sueldoNominal - ($this->isss
+ $this->afp);
       //De acuerdo a criterios del Ministerio de Hacienda
       //el descuento de la renta varía según el ingreso
percibido
       if($salariocondescuento>2038.10){
           $this->renta = $salariocondescuento * 0.3;
```

```
        }
        elseif($salariocondescuento>895.24 &&
$salariocondescuento<=2038.10){
            $this->renta = $salariocondescuento * 0.2;
        }
        elseif($salariocondescuento>472.00 &&
$salariocondescuento<=895.24){
            $this->renta = $salariocondescuento * 0.1;
        }
        elseif($salariocondescuento>0 &&
$salariocondescuento<=472.00){
            $this->renta = 0.0;
        }
        /* else {
            //Significa que el salario obtenido es negativo
        } */
        $this->sueldoNominal = $salario;
        $this->sueldoLiquido = $this->sueldoNominal +
$this->pagoxhoraextra - ($this->isss + $this->afp + $this->renta);
        $this->imprimirBoletaPago();
     }

     function imprimirBoletaPago(){
        $tabla  = "<table class='table '><tr>";
        $tabla .= "<td>Id empleado: </td>";
        $tabla .= "<td>" . self::$idEmpleado . "</td></tr>";
        $tabla .= "<tr><td>Nombre empleado: </td>\n";
        $tabla .= "<td>" . $this->nombre . " " . $this->apellido .
"</td></tr>";
        $tabla .= "<tr><td>Salario nominal: </td>";
        $tabla .= "<td>$ " . number_format($this->sueldoNominal,
2, '.', ',') . "</td></tr>";
        $tabla .= "<tr><td>Salario por horas extras: </td>";
        $tabla .= "<td>$ " . number_format($this->pagoxhoraextra,
2, '.', ',') . "</td></tr>";
        $tabla .= "<tr class='success'><td
colspan=\"2\"><h4>Descuentos</h4></td></tr>";
        $tabla .= "<tr ><td >Descuento seguro social: </td>";
        $tabla .= "<td>$ " . number_format($this->isss, 2, '.',
',') . "</td></tr>";
        $tabla .= "<tr><td>Descuento AFP: </td>";
        $tabla .= "<td>$ " . number_format($this->afp, 2, '.',
',') . "</td></tr>";
        $tabla .= "<tr><td>Descuento renta: </td>";
        $tabla .= "<td>$ " . number_format($this->renta, 2, '.',
',') . "</td></tr>";
        $tabla .= "<tr><td>Total descuentos: </td>";
        $tabla .= "<td>$ " . number_format($this->isss +
$this->afp + $this->renta, 2, '.', ',') . "</td></tr>";
        $tabla .= "<tr><td>Sueldo líquido a pagar: </td>";
```

```
        $tabla .= "<td> $" . number_format($this->sueldoLiquido,
2, '.', ',') . "</td></tr>";
        $tabla .= "</table>";
        echo $tabla;
    }
  }
?>
```

Archivo 2: sueldoneto.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width,
initial-scale=1, maximum-scale=1">
    <title>Datos del empleado</title>
    <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.4.1/css/boo
tstrap.min.css"
integrity="sha384-Vkoo8x4CGsO3+Hhxv8T/Q5PaXtkKtu6ug5TOeNV6gBiFeW
PGFN9MuhOf23Q9Ifjh" crossorigin="anonymous">

    <!--[if lt IE 9]>
        <script
src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></scrip
t>
    <![endif]-->
</head>
<body class='container'>
<?php
function __autoload($class_name) {
    include_once("class/" . $class_name . ".class.php");
}

if(isset($_POST['enviar'])){
    if(isset($_POST['enviar'])){
        echo "<h3>Boleta de pago del empleado</h3>";
        $name = (isset($_POST['nombre'])) ? $_POST['nombre'] :
"";
        $apellido = (isset($_POST['apellido'])) ?
$_POST['apellido'] : "";
        $sueldo = (isset($_POST['sueldo'])) ?
doubleval($_POST['sueldo']) : 0.0;
        $numHorasExtras = (isset($_POST['horasextras'])) ?
intval($_POST['horasextras']) : 0;
        $pagohoraextra = (isset($_POST['pagohoraextra'])) ?
floatval($_POST['pagohoraextra']) : 0.0;
```

```
        //Creando instancias de la clase empleado
        $empleado1 = new empleado();
        $empleado1->obtenerSalarioNeto($name, $apellido,
$sueldo, $numHorasExtras, $pagohoraextra);
    }
}
else{
?>

<section class="container">


<nav class="navbar navbar-dark bg-primary text-white">
<h1>Formulario empleado</h1>
</nav>
   <article>
<form  action="<?php echo $_SERVER['PHP_SELF'] ?>"
method="POST">
   <fieldset>

   <div class="form-group">
    <label for="nombre">Nombre empleado:</label>
    <input type="text" name="nombre" id="nombre" size="25"
maxlength="30" class="inputField form-control" /><br />
</div>
    <div class="form-group">
    <label for="apellido">Apellido empleado:</label>
    <input type="text" name="apellido" id="apellido" size="25"
maxlength="30" class="inputField form-control" /><br />
</div>
    <div class="form-group">
    <label for="sueldo">Sueldo empleado ($):</label>
    <input type="text" name="sueldo" id="sueldo" size="8"
maxlength="8" class="inputField form-control" /><br />
</div>
    <div class="form-group">
    <label for="horasextras">Número horas extras:</label>
    <input type="text" name="horasextras" id="horasextras"
size="4" maxlength="2" class="inputField form-control" /><br />
 </div>
    <div class="form-group">
    <label for="pogohoraextra">Pago por hora extra:</label>
    <input type="text" name="pagohoraextra" id="pagohoraextra"
size="4" maxlength="6" class="inputField form-control" /><br />
</div>

   <input type="submit" name="enviar" class="btn btn-primary
mb-2" value="Enviar" class="inputButton" /> 
   <input type="reset" name="limpiar" class="btn btn-primary
mb-2" value="Restablecer" class="inputButton" />
```

```
   </fieldset>
</form>
<?php
}
?>
   </article>
</section>
</body>
</html>
```

Ejercicio #4: El siguiente ejemplo nos muestra cómo se puede implementar el cálculo de la distancia entre dos puntos, dadas dos coordenadas ingresadas por el usuario. Se ha utilizado una clase que no posee propiedades y que se definirán dinámicamente, haciendo uso de la sobrecarga implementada con los métodos mágicos __set() y __get().

Archivo 1: coordenadas.class.php

```
<?php
   class coordenadas {
      private $coords = array('x' => 0, 'y' => 0);
      //Métodos especiales __get() y __set()
     function __get($property) {
         if(array_key_exists($property, $this->coords)) {
            return $this->coords[$property];
         }
       else {
            print "Error: Sólo se aceptan coordenadas x y y.<br
/>\n";
         }
      }
      function __set($property, $value) {
         if(array_key_exists($property, $this->coords)) {
            $this->coords[$property] = $value;
         }
       else {
            print "Error: No se puede escribir otra coordenada más
que x y y.<br />\n";
         }
      }
   }
?>
```

Archivo 2: distanciadospuntos.php

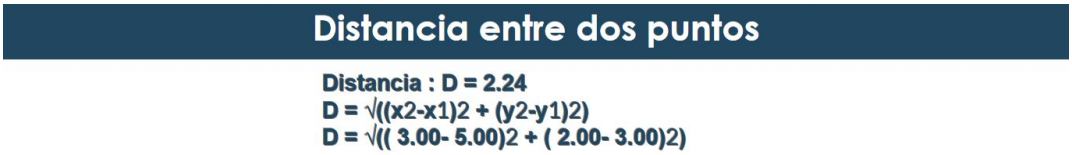
```
<!DOCTYPE html>
<html lang="es">
```

```
<head>
    <meta charset="utf-8" />
    <title>La distancia entre dos puntos</title>
    <link rel="stylesheet" href="css/slick.css" />
</head>
<body>
<header id="demo">
    <h1 class="demo1">Distancia entre dos puntos</h1>
</header>
<section id="slick">
<?php
   if(isset($_POST['submit'])){
      //Capturando los datos de formulario
      $x1 = is_numeric($_POST['coordx1']) ? $_POST['coordx1'] :
"Error";
      $x2 = is_numeric($_POST['coordx2']) ? $_POST['coordx2'] :
"Error";
      $y1 = is_numeric($_POST['coordy1']) ? $_POST['coordy1'] :
"Error";
      $y2 = is_numeric($_POST['coordy2']) ? $_POST['coordy2'] :
"Error";
      if($x1 == "Error" || $x2 == "Error" || $y1 == "Error" || $y2
== "Error"){
         die("<h3 style=\"color:red;\">Los valores de x1, x2, y1 y
y4 deben ser numéricos</h3>");
      }

     //Utilizando autocarga de clases para invocar la clase
      function __autoload($class){
         require_once "class/" . $class . ".class.php";
      }

     //Creando las coordenadas
      $coord1 = new coordenadas();
      $coord2 = new coordenadas();
      //Definiendo las coordenadas del primer punto
      $coord1->x = $x1;
      $coord1->y = $y1;
      //Definiendo las coordenadas del segundo punto
      $coord2->x = $x2;
      $coord2->y = $y2;
      //Obteniendo la distancia entre dos puntos
      $difx = pow($coord2->x - $coord1->x, 2);
      $dify = pow($coord2->y - $coord1->y, 2);
      $dist = sqrt($difx + $dify);
      printf("<p class=\"resp\">Distancia : D = " .
number_format($dist, 2, '.', ',') . "</p>\n");
      printf("<p class=\"resp\">D =
√((x<sub>2</sub>-x<sub>1</sub>)<sup>2</sup> +
(y<sub>2</sub>-y<sub>1</sub>)<sup>2</sup>)</p>\n");
```

```
      printf("<p class=\"resp\">D = √((%5.2lf-%5.2lf)<sup>2</sup>
+ (%5.2lf-%5.2lf)<sup>2</sup>)</p>\n", $coord2->x, $coord1->x,
$coord2->y, $coord1->y);
   }
   else{
?>
<div class="contact-form">
   <!-- Título -->
   <div class="title">Cálculo de la distancia entre dos
puntos</div>
   <!-- Texto indicativo -->
   <p class="intro">Ingrese las coordenadas</p>
   <!-- Área de formulario -->
   <div class="contact-form">
   <!-- Formulario -->
   <div class="w-100">
      <!-- Campos de formulario -->
      <form name="frmrectangulo" id="frmrectangulo" action="<?php
echo $_SERVER['PHP_SELF'] ?>" method="POST">
      <!-- <form name="frmrectangulo" id="frmrectangulo"
action="javascript:void(0);"> -->
         <!-- Coordenada 1 (x1,y1) -->
         <label>Coordenada 1 (x1,y1): </label>
         <div class="field">
            <input type="number" name="coordx1" id="coordx1"
min="0" max="1000" step=".1" placeholder="(x1)" required />
            <span class="entypo-base icon"></span>
            <span class="slick-tip left">Ingrese la coordenada
x1:</span>
         </div>
         <div class="field">
            <input type="number" name="coordy1" id="coordy2"
min="0" max="1000" step=".1" placeholder="(y1)" required />
            <span class="entypo-base icon"></span>
            <span class="slick-tip left">Ingrese la coordenada
y1:</span>
         </div>
         <!-- Coordenada 2 (x2,y2) -->
         <label>Coordenada 2 (x2,y2): </label>
         <div class="field">
            <input type="number" name="coordx2" id="coordx2"
min="0" max="1000" step=".1" placeholder="(x2)" required />
            <span class="entypo-base icon"></span>
            <span class="slick-tip left">Ingrese la coordenada
x2:</span>
         </div>
         <div class="field">
            <input type="number" name="coordy2" id="coordy2"
min="0" max="1000" step=".1" placeholder="(y2)" required />
            <span class="entypo-base icon"></span>
```

```
            <span class="slick-tip left">Ingrese la coordenada
y2:</span>
         </div>
         <!-- Botones para hacer los cálculos -->
         <input type="submit" value="Calcular" class="send"
name="submit" id="perimetro" />
         <input type="reset" value="Restablecer" class="send"
name="reset" id="area" />
      </form>
   </div>
</div>
<?php
   }
?>
</section>
</body>
</html>
```

Al visualizarlo en el navegador de su preferencia puede ingresar los datos de las coordenadas y luego verificar que el cálculo de la distancia entre los dos puntos es correcta:



V. DISCUSIÓN DE RESULTADOS

1. Modifique el script de autos (ejercicio 1) para que en el script autospoo.php en lugar de mostrar todos los autos que existen, se incluya un formulario con un campo SELECT-OPTION que permita al usuario seleccionar la información del auto que desea, de modo que en la misma página se pueda visualizar únicamente la información de la marca de auto que se seleccione.
2. Modifique la clase del ejemplo de salario (Ejercicio 3), de modo que, si el empleado posee algún préstamo que deba ser descontado mediante orden de descuento, se pueda introducir este valor y deducir del salario del empleado. Este valor descontado debe aparecer en la boleta de pago, si es que el empleado posee descuentos por concepto, de lo contrario, no debe aparecer el concepto de descuento por préstamo.

VI. BIBLIOGRAFIA

- Gutiérrez, Abraham / Bravo, Ginés. PHP 5 a través de ejemplos. Editorial Alfaomega RAMA. 1ra edición. México. Junio 2005.
- Gil Rubio, Francisco Javier/Villaverde, Santiago Alonso/Tejedor Cerbel, Jorge A. Creación de sitios web con PHP 5. Editorial McGraw-Hill. 1ra edición. Madrid, España, 2006.
- John Coggeshall. La Biblia de PHP 5. 1ra Edición. Editorial Anaya Multimedia. Madrid España.
- http://www.php.net/manual/en