	UNIVERSIDAD DON BOSCO **FACULTAD DE ESTUDIOS TECNOLÓGICOS** **ESCUELA DE COMPUTACIÓN**
CICLO: 01/2019	*GUIA DE LABORATORIO #6* **Nombre de la Practica:** **Programación Orientada a Objetos en PHP** **Lugar de Ejecución:** **Centro de Cómputo** **Tiempo Estimado:** **2 horas con 30 minutos** **MATERIA:** **Desarrollo de Aplicaciones Web con Software Interpretado en el Servidor** **DOCENTES:** **Ing. Ricardo Ernesto Elías / Ing. Miguel Eduardo Orellana**

I. OBJETIVOS

Con la realización de esta guía de práctica el estudiante estará en capacidad de:

1. Dar una visión general de la metodología de programación orientada a objetos.
2. Estar en capacidad de diseñar clases y crear objetos a partir de estas.
3. Hacer uso de constructores y destructores en el diseño de clases.
4. Hacer uso del control de acceso a las propiedades y métodos de una clase.
5. Sacar provecho de la autocarga de clases con la función *__autoload()*.
6. Realizar clases usando la sobrecarga de propiedades y métodos.
7. Implementar herencia de clases en situaciones que lo requieran.
8. Controlar la herencia de clases con clases abstractas y clases finales.

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*, *private* o *protected*.

En PHP4 se podía utilizar la palabra clave var, sin especificar control de acceso, ya que esta característica sólo está disponible en PHP5.
Los **métodos** o **acciones** de la clase se crean declarando funciones dentro de la definición de la clase.

Control de acceso a los miembros de una clase
En PHP 5, se han incluido especificadores de acceso para los miembros de una clase. Estos, que ya se mencionaron, son:

- 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";
```

Propiedades y métodos estáticos

Una propiedad o método estático pertenece a la clase en la que está definido, no a los objetos creados a partir de dicha clase. De modo que, no puede ser accedido desde un objeto ni redefinido en las clases derivadas. No obstante, es posible llamar a una propiedad o método estático desde fuera del contexto de un objeto.

Para declarar una propiedad o método estático se hace uso de la palabra reservada *static* y debe colocarse después de la declaración de visibilidad o acceso para la propiedad, si es que existe, tal y como se muestra:

```
prívate static $idLibro;
```

Las propiedades estáticas por pertenecer a la clase y no a los objetos deben ser accedidas, desde dentro del contexto de la clase, empleando el nombre de clase reservado *self* y el operador de resolución de ámbito ::. Esta palabra reservada permite hacer referencia a la clase actual en la que se encuentra la declaración. Por ejemplo, para hacer referencia a la propiedad estática $idLibro, definida anteriormente, debe utilizar una instrucción como la siguiente, si está dentro de la clase:

```
self::$idLibro++;
```

Ahora bien, para acceder a la propiedad desde fuera del contexto de la clase, debe hacer uso del nombre de la clase, seguida del operador de resolución de ámbito (::) y, a continuación, el nombre de la propiedad.

En el caso de los métodos estáticos, se procede de igual forma, ya que pertenecen a la clase y no a los objetos. Así que los métodos pueden ser llamados desde fuera del contexto de la clase utilizando la notación:

```
nombreClase::metodo();
```

Recuerde, que al igual que con las propiedades estáticas, no se puede utilizar la variable especial $this dentro de los métodos estáticos, puede emplearse self::.

Constantes de clase

A partir de PHP5 es posible definir constantes dentro de una clase. Estas pueden ser utilizadas para almacenar valores escalares que permanecerán invariables a lo largo de la ejecución del script. Por ejemplo, rutas de direcciones URL, valores constantes en ciertos cálculos (constante PI), nombres de bases de datos, el valor de un impuesto (como el IVA).

Las constantes de clase –como las propiedades estáticas— pertenecen a la clase en la que están definidas, no a los objetos de dicha clase. Por esta razón, no puede accederse a través del objeto a los valores constantes definidos en la clase. El acceso debe realizarse, desde dentro de la clase, a través del nombre de clase reservado self, o bien, desde fuera de la misma a través del nombre de la clase. Por ejemplo:

```
class miClase{
   …
   const IVA = 0.13;
   …
}
```

Si accedemos a la constante desde dentro de un método de la clase, entonces podemos utilizar la siguiente instrucción:

```
$precio = $precioSinIva * self::IVA;
```

Para acceder a la misma constante desde fuera del contexto de la clase, debe utilizar el nombre de la clase y luego el operador de resolución de ámbito:

```
$precio = $precioSinIva * nombreClase::IVA;
```

Constantes mágicas
PHP proporciona dos constantes de clase especiales, denominadas también, constantes mágicas. Estas constantes están disponibles desde dentro de cada clase. Estas constantes son __CLASS__ y __METHOD__.

La constante __CLASS__ almacena el nombre de la clase en el que es empleada. Este nombre también puede ser obtenido a través de la función **`get_class()`**.

Por su parte, __METHOD__ contiene el nombre de clase y método desde el que se accede. Así, si se accede a dicha constante desde el método denominado metodo1 de la clase clase1, la constante mágica contendrá la cadena 'clase1::metodo1'. Adicionalmente, se dispone de la función get_class_methods($objeto), que recibe como argumento un nombre de clase o un objeto y devuelve una matriz indexada conteniendo todos los métodos definidos en la clase correspondiente.

Carga automática de clases
Cuando se trabaja con clases produciendo un script independiente por cada definición de clase, es necesario incluir cada una de estas definiciones de clase en el script principal de la aplicación final. En el caso que el número de clases definidas sea considerable, puede llegar a ser molesto tener que crear una instrucción include o require por cada una de las clases necesarias para el script.
Para brindar un mecanismo para que los programadores puedan cargar las clases en tiempo de ejecución o cargar las clases de forma dinámica o automática, en PHP 5 se ha incorporado un método conocido como método mágico, denominado ***__autolead()***. La llamada a este método se realiza de forma automática, no se puede invocar desde código. Esta llamada se produce cuando se intenta instanciar a una clase, sin que exista en el script una definición para esta. En ese momento, PHP buscará el método ***__autoload()*** para localizar la definición de esa clase en un archivo externo. Si no se encuentra un archivo con la definición de la clase ni con este método, entonces el script terminará con un error fatal.
El objetivo de esta técnica es facilitar al programador la inclusión de clases sin tener que recurrir a la inclusión de un sin número de instrucciones include o require.
La implementación de esta técnica requiere que se programe el método ***__autoload()*** en el script. Una implementación simple sería la siguiente:

```
function __autoload($clase){
   include_once($clase . ".php");
}
```

El método ***__autoload()*** será invocado de forma automática cuando se instancie a una clase para crear un objeto, de esta forma:

```
$objeto = new clase1();
```

Si no se encuentra en el script o en un script externo incluido mediante una instrucción include o require, PHP intentará, como último recurso, encontrar la definición de la clase en el método ***__autoload()***. Si no la encuentra, entonces lanzará un error fatal.

Hay algunas consideraciones a tomar en cuenta para usar el método ***__autoload()***:

1. El método ***__autoload()*** se autoejecuta, lo que significa que no puede ser invocado por el programador mediante código PHP.
2. El método recibe como argumento el nombre de la clase que se ha intentado instanciar sin que haya sido encontrada su definición en el script. PHP es quien envía el valor adecuado de este argumento, siendo este el nombre de la clase.

3. Dentro de la función ***__autoload()*** se intenta incluir un script que debe haber sido nombrado igual que la clase para que sea encontrado.
4. Por último, para que esto funcione, debe haber creado un script PHP, por cada definición de clase.

Sobrecarga de propiedades y métodos

La **sobrecarga de propiedades** en PHP puede ser implementada a través de los métodos especiales, llamados también mágicos, `__set()` y `__get()`. Estos métodos son invocados de forma automática cuando se intenta acceder a una propiedad inexistente de un objeto.
La sintaxis de estos métodos es la siguiente:

```
void __set(string $name, mixed $value);
mixed __get(string $name);
```

El método ***__set()*** se utiliza para asignar un valor, dado por $value, a la propiedad que se ha intentado acceder y que no existe en la definición de la clase.
El método ***__get()*** permite recuperar el valor de una propiedad a la que se ha intentado acceder, sin que exista en la definición de la clase.
Un aspecto importante a considerar es el hecho que primero debe establecerse el valor, antes de intentar accederlo. Esto significa, que primero debe hacerse la asignación del valor en la propiedad y luego, intentar obtener ese valor.
Los métodos ***__set()*** y ***__get()*** son llamados únicamente si la propiedad referenciada no existe en el objeto. Si la propiedad ya fue establecida en una ejecución previa de ***__set()***, ya no se volverán a ejecutar ni ***__get()***, ni ***__set()***.

La **sobrecarga de métodos** se puede implementar con el método mágico ***__call()***. Mediante este método se podrá acceder a métodos no definidos en el objeto.
La sintaxis del método ***__call()*** es la siguiente:

```
mixed __call(string $name, array $arguments);
```

Podemos comprender major el funcionamiento de la sobrecarga de miembros de una clase, mediante un ejemplo simple:

```
class sinPropiedades {
   function __set($propiedad, $valor){
      echo "Asignamos $valor a $propiedad";
      $this->propiedad = $valor;
   }
   function __get($propiedad){
      echo "Acceso a la propiedad $propiedad(clase ", __CLASS__,")\n";
      return $this->propiedad;
   }
   function __call($metodo, $parametros){
      echo "Acceso al método $metodo (clase, __CLASS__, ")\nArgumentos:\n",
var_dump($parametros), "\n";
   }
}  //Fin clase sinPropiedades
```

Para probar nuestra clase sinPropiedades, tenemos el siguiente script:

```
//Creación de un nuevo objeto sinPropiedades
$obj = new sinPropiedades();
//Asignando valores a dos propiedades no definidas
$obj->nombre = "Sergio";
$obj->edad = 25;
```

```
//Hacer un volcado del objeto
echo var_dump($obj), "\n";
//Acceder a las propiedades sobrecargadas y a otra inexistente
echo 'Nombre: ', $obj->nombre, "\n";
echo 'Edad: ', $obj->edad, "\n";
echo 'Apellido: ', @$obj->apellido, "\n";
//Intentar ejecutar un método inexistente
echo $obj->darNombre('Sergio', 'Pérez', 30);
```

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>";
```

Clases y métodos abstractos

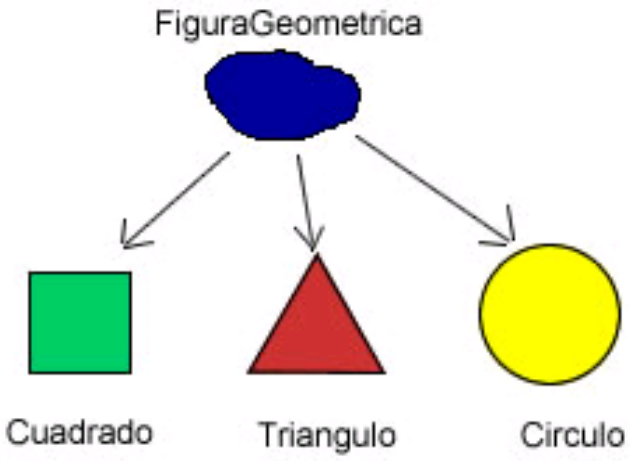
La **abstracción** es un recurso que se utiliza para brindar mayor control sobre el proceso de herencia. La característica principal de una clase abstracta es que no puede instanciarse, lo que quiere decir que no puede utilizarse de forma directa, únicamente se puede acceder a sus miembros a través de una subclase o clase derivada de esta. A esta subclase se le denomina también clase concreta.

Una **clase abstracta**, como cualquier clase padre, además de declarar e implementar propiedades, constantes y métodos, puede definir **métodos también abstractos**, los cuales no pueden ser implementados; es decir, no poseen código. La implementación de los métodos abstractos corresponde a las clases hijas. Además, las clases hijas pueden volver a declarar estos métodos abstractos, lo que significa que la responsabilidad de la implementación del método recaerá en el siguiente nivel jerárquico.

Veamos el siguiente ejemplo:

```
abstract class number {
   //Propiedades
   private $value;
   //Declaración de un método abstracto
   abstract public function value();
   public function reset(){
      $this->value = NULL;
   }
}

class integer extends number {
   //Propiedades de la clase derivada
   private $value;
   //Implementación del método abstracto
   public function value(){
      return (int)$this->value;
   }
}

//Probando la clase
$int = new integer();  //Se ejecutará correctamente
$num = new number();  //Lanzará un error
```

Un buen ejemplo de abstracción son las figuras geométricas. La figura geométrica es una clase abstracta, puesto que representa nada más un concepto. Todos entendemos lo que son las figuras geométricas, pero nadie puede decir que ha visto alguna, lo que pudiera haberse visto es un cuadrado, un triángulo, un rectángulo o un círculo, estas si son figuras geométricas concretas, con dimensiones específicas.



Interfaces

Las interfaces son similares en algunos aspectos a las clases abstractas. Permiten definir protocolos de comportamiento para los objetos en cualquier punto de la jerarquía de clases. Una interfaz permite definir un nombre a un conjunto de definiciones de métodos y a un conjunto de constantes. Una clase que cumple con una determinada interfaz debe implementar todos los métodos incluidos en la misma, adquiriendo por tanto, un cierto comportamiento. La clase podría también, declarar alguno de los métodos definidos por la interfaz, como abstracto, forzando a que dicho método sea implementado por alguna de sus clases hijas. Además, una clase puede implementar más de una interfaz.

Una diferencia entre las clases abstractas y las interfaces es que mientras las primeras proporcionan un medio para expresar conceptos abstractos en programación, las segundas, se han diseñado para asegurar la funcionalidad dentro de una clase.

Para declarar una interfaz en PHP se utiliza la palabra clave **interface** y, a continuación, el identificador de la interfaz. De la siguiente forma:

```
interface printable {
   public function printme();
}
```

Para que una interfaz sea útil, debe ser implementada mediante una o más clases. También, es posible que una sola clase pueda implantar múltiples interfaces diferentes.

```
interface printable {
   public function printme();
}

interface Inumber {
   public function reset();
}

class integer implements printable, Inumber {
   private $value;
   function __construct($value) {
      $this->value = $value;
   }
   //Implementación de la interfaz printable
   public function printme() {
      echo (int)$this->value;
   }
   public function reset() {
```

```
      $this->value = NULL;
   }
   public function value() {
      return (int)$this->value;
   }
}

function resetNumber(Inumber $obj) {
   $obj->reset();
}
function printNumber(printable $obj) {
   $obj->printme();
}
```

Para probar la interfaz:

```
$entero = new integer(12);
printNumber($entero);
resetNumber($entero);
```

Polimorfismo

El **polimorfismo** es uno de los elementos clave o, dicho de otro modo, uno de los pilares de la Programación Orientada a Objetos. Con el **polimorfismo** es posible emplear un mismo método perteneciente a objetos de distinta clase, sin que importe realmente donde está implementado dicho método.

El siguiente ejemplo supone que están definidas las clases derivadas empleado, estudiante y bebé. Todas ellas hijas de la superclase persona. La clase hija empleado posee el método trabaja(), la clase estudiante posee el método estudia() y la clase bebe, posee el método comeyduerme(). La idea es desarrollar una función que muestre la ocupación principal de cada persona. La función tendría la siguiente implementación:

```
function estadoPersona($persona) {
   if($persona instanceof empleado)
      echo $persona->trabaja();
   elseif($persona instanceof estudiante)
      echo $persona->estudia();
   elseif($persona instanceof bebe)
      echo $persona->comeyduerme();
   else
      echo "ERROR: Se desconoce la ocupación de esta persona.";
}
```

El operador instanceof se utiliza para determinar la clase a la que pertenece un objeto o si un objeto es una instancia de una determinada clase. También su utiliza para determinar si un objeto implementa alguna interfaz dada. Puede notar este tipo de implementación es poco escalable. Por ejemplo, si añadimos un nuevo tipo de persona a la aplicación, tendríamos que modificar el código de la función estadoPersona. Sin embargo, existe una mejor solución que consiste en hacer uso del polimorfismo. De modo, que cada una de las clases hijas implementará un método específico, denominado ocupacionPrincipal(). Así, dado un objeto de cualquiera de las tres clases hijas (empleado, estudiante y bebé), se podría realizar la llamada al método correspondiente utilizando la sintaxis:

```
$objeto->ocupacionPrincipal();
```

De este modo la función estadoPersona sería:

```
function estadoPersona($persona) {
   if($persona instanceof persona)
      echo $persona->ocupacionPrincipal();
   else
```

```
      echo "ERROR: Se desconoce la ocupación de esta persona.";
}
```

Manejo de excepciones

Una de las incorporaciones más interesantes de PHP 5 es el manejo de excepciones. Las **excepciones** proporcionan un mecanismo para gestionar errores en el contexto de los objetos. Las **excepciones** también proporcionan ventajas significativas sobre las técnicas tradicionales de gestión de errores. Uno de los principales inconvenientes del manejo tradicional de errores es que la gestión del error está entrelazada con el flujo normal del código del script o programa. Incluir el tratamiento de errores en el punto donde este se produce provoca que el código sea poco flexible y difícil de reutilizar, ya que el mismo error puede precisar de tratamientos distintos en función de las circunstancias en las que se produzca.

La gestión de excepciones utilizando objetos intenta resolver estos problemas, permitiendo delegar el tratamiento de errores al lugar más apropiado, haciendo posible manejar múltiples condiciones de error en un único punto, separando el procesamiento de la excepción de la lógica del programa.

La clase Exception

En la práctica, las excepciones son instancias de clases que contienen información sobre el error que se ha producido durante la ejecución de la secuencia de comandos. PHP 5 proporciona internamente una clase denominada Exception. La definición de la clase Exception es la siguiente:

```
class Exception {
   protected $message;
   private $string;
   protected $code;
   protected $file;
   protected $line;
   private $trace;
   function __construct($message="", $code=0);
   function __toString();
   public function getFile();
   public function getLine();
   public function getMessage();
   public function getCode();
   public function getTrace();
   public function getTraceAsString();
}
```

Las excepciones en PHP contienen dos valores principales que son: una cadena con el mensaje que describe el error que se ha producido y un valor entero que representa el código del error. De forma predeterminada, PHP asignará automáticamente a la excepción la línea y el nombre del archivo donde se ha producido el error, así como una localización de pila que representa la ruta de acceso a la ejecución que ha resultado en error.

Arrojar y capturar excepciones

Como programadores, tenemos la posibilidad de derivar la clase Exception, para definir nuestras propias excepciones. Cuando se lance una excepción, se puede definir un mensaje descriptivo sobre el error y un código de error. Del resto de la clase no debemos preocuparnos como programadores, puesto que PHP maneja los métodos por nosotros.

Uno de los métodos más interesantes de la clase Exception es el método ***__toString()***, el cual pertenece a la categoría de los métodos conocidos como métodos mágicos. Al igual que ***__construct()***, ***__destruct()***, ***__clone()***, ***__autoload()*** y otros que hemos mencionado anteriormente. Este método es susceptible de ser sobrescrito en las clases derivadas de Exception.

Para lanzar una excepción se puede instanciar a la clase Exception o a una clase derivada de esta, incluso aunque no se haya creado un objeto como tal para su uso posterior. Para lanzar la excepción desde la secuencia de comandos se utiliza la sentencia throw junto con la instancia de la excepción a arrojar. Veamos el siguiente ejemplo:

```
class throwExample {
   public function makeError(){
      throw new Exception("Este es un ejemplo de excepción");
   }
}

$inst = new throwExample();
$inst->makeError();
```

El manejo de excepciones en PHP 5 se lleva a cabo utilizando una estructura de control, denominada try/catch. Dentro del bloque try se incluye el código de programa que podría llegar a producir una excepción. Todo bloque try dentro de la secuencia de comando debe tener asociado, al menos un bloque catch. Con el bloque catch se realiza la gestión de las excepciones. La sintaxis es la siguiente:

```
try {
   //código que puede generar una excepción
}
catch(classException1 $e1) {
   //Procesamiento de las excepciones de classException1
}
[catch(classException2 $e2){
   //Procesamiento de las excepciones de classException2
}]
```

El funcionamiento es, en teoría, simple. Si la excepción es una instancia de la clase capturada en el primer bloque catch, será procesada en ese punto. En caso contrario, se buscará otro bloque catch que admita la clase de excepción generada u otra estructura por encima de try/catch en la que capturarla. Si no se encuentra dicho bloque, se generará un error fatal.

El siguiente ejemplo muestra cómo manejar el conocido caso de la división por cero haciendo uso de excepciones.

```
<?php
   try {
      $a=15;
      //$a=0;  //quitar  los  comentarios  luego  a  esta  línea  para  probar  las
excepciones
      if($a==0){
         $error  = "<p style=\"font:bold 10pt Verdana;color:red;\">";
         $error .= "El divisor debe de ser diferente de CERO.</p>";
         throw new Exception($error);
         echo "<p>ESTA PARTE DE CODIGO NO SE EJECUTARÁ";
         echo "TODO ESTO SERÍA IGNORADO.</p>";
      }
      else{
         @$resultado=number_format(10/$a, 4, '.', ',');
         echo  "<p  style=\"font:bold  12pt  Verdana;color:Green\">Respuesta  de  la
division: ";
         echo $resultado . "<br>\n";
         echo 'No se ha producido ninguna excepción.</p>';
      }
   }
   catch (Exception $e){
      echo "<br>ERROR<br><br>";
      echo "Descripcion : ".$e->getMessage(). "<br>";
      echo "Codigo de la Excepcion : ".$e->getCode(). "<br>";
      echo "Archivo del error : ".$e->getFile(). "<br>";
```

```
      echo "Línea de llamado de Exception : ".$e->getLine(). "<br>";
   }
   // Continue execution
   echo "<br><span style=\"font:bold 10pt Arial;color:Blue;\">";
   echo "FIN DEL PROGRAMA</span>";
?>
```

III. MATERIALES Y EQUIPO

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

No.	Requerimiento	Cantidad
1	Guía de práctica #6: Programación Orientada a Objetos con PHP	1
2	Computadora con WampServer y Sublime Text 3 instalado	1
3	Memoria USB	1

IV. PROCEDIMIENTO

Realice ordenadamente cada uno de los siguientes ejercicios. Algunos incluyen más de una script PHP junto con alguna página web en puro HTML.

Ejercicio #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
/*******************************************************
 * Descripción: Ejemplo de definición de una clase auto *
 * Autor: Ricardo Ernesto Elías Guandique               *
 * Archivo: autopoo.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='Rojo',
$image='img/hondacivic.jpg'){
        $this->marca = $marca;
        $this->modelo = $modelo;
        $this->color = $color;
        $this->image = $image;
    }

    //Métodos de la clase
    function mostrar(){
        $tabla  = "<table style=\"width:200;border:ridge 5px rgb(200,50,150)\">\n";
        $tabla .= "<caption>Compra un " . $this->marca . "</caption>";
        $tabla .= "<tr>\n<td style=\"width:35%;\">MARCA</td>\n";
        $tabla .= "<td style=\"width:35%\">" . $this->marca . "</td>\n";
```

```
        $tabla .= "<td rowspan=\"3\" style=\"width:35%\"><img src=\"" . $this->image .
"\"></td></tr>";
        $tabla .= "<tr>\n<td>MODELO</td>\n";
        $tabla .= "<td>" . $this->modelo . "</td>\n</tr>\n";
        $tabla .= "<tr>\n<td>COLOR</td>\n";
        $tabla .= "<td>\n" . $this->color . "</td>\n</tr>\n";
        $tabla .= "</table>\n";
        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">
    <title>Venta de autos</title>
    <link rel="stylesheet" href="css/autospoo.css" />
    <!--[if lt IE 9]>
        <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>
<body>
<header>
    <h1>Autos disponibles</h1>
</header>
<section>
<article>
<?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", "Marron", "img/renaultclio.jpg");
  $movil[2] = new auto("BMW", "Serie6", "Azul", "img/bmwserie6.jpg");
  //Esta llamada mostrará los valores por defecto en los argumentos
  //del método constructor.
  $movil[3] = new auto();

  //Mostrando la tabla con los autos disponibles
  for($i=0; $i<count($movil); $i++){
      $movil[$i]->mostrar();
  }
?>
</article>
</section>
</body>
</html>
```

El resultado en el navegador de su preferencia sería el siguiente:

Ejercicio #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 $operacion;
         private $balance = 0.0;

      //Métodos de la clase
      function openAccount($owner, $operacion, $amount){
         self::$numberAccount++;
         $this->idcuenta = self::$numberAccount;
         $this->owner = $owner;
         $this->operacion = $operacion;
         $this->balance = $amount;
         $comprobante  = "\n<table class=\"responstable\">\n";
         $comprobante .= "<tr>\n<td>Número de cuenta: </td>\n";
         $comprobante .= "<td>" . self::$numberAccount . "</td>\n</tr>\n";
         $comprobante .= "<tr>\n<td>Propietario: </td>\n";
         $comprobante .= "<td>" . $this->owner . "</td>\n</tr>\n";
         $comprobante .= "<tr>\n<td>Operación: </td>\n";
         $comprobante .= "<td>" . mb_convert_case($this->operacion, MB_CASE_TITLE,
"UTF-8") . "</td>\n</tr>\n";
         $comprobante .= "<tr>\n<td>Saldo inicial: </td>\n";
         $comprobante  .=  "<td>$  "  .  number_format($this->balance,2,'.',',')  .
"</td>\n</tr>\n";
         $comprobante .= "</table>";
         echo $comprobante;
      }

      function makeDeposit($amount, $operacion, $saldo=250){
         //Se añade al saldo actual la cantidad ($amount)
```

```
         //recibida como argumento del método
         $this->operacion = $operacion;
         //Actualizar el saldo antes de realizar cualquier operación
         $this->balance = $saldo;
         $this->balance += $amount;
         $comprobante  = "\n<table class=\"responstable\">\n";
         $comprobante .= "\t<tr>\n\t\t<td>Saldo inicial: </td>\n";
         $comprobante  .=  "\t\t<td>\$  "  .  number_format($saldo,2,".",",")  .
"</td>\n\t</tr>\n";
         $comprobante .= "<tr>\n<td>Operación: </td>\n";
         $comprobante .= "<td>" . mb_convert_case($this->operacion, MB_CASE_TITLE,
"UTF-8") . "</td>\n</tr>\n";
         $comprobante .= "<tr>\n<td>Cantidad depositada: </td>\n";
         $comprobante  .=  "<td>\$  "  .  number_format($amount,2,'.',',')  .
"</td>\n</tr>\n";
         $comprobante .= "<tr>\n<td>Nuevo saldo: </td>\n";
         $comprobante  .=  "<td>\$  "  .  number_format($this->balance,2,'.',',')  .
"</td>\n</tr>\n";
         $comprobante .= "</table>\n";
         echo $comprobante;
      }

      function makeWithdrawal($amount, $operacion, $saldo=250){
         //Se resta del saldo actual de la cuenta
         //la cantidad ($amount) recibida como argumento
         $this->operacion = $operacion;
         //Actualizar el saldo antes de realizar cualquier operación
         $this->balance = $saldo;
         $saldoinicial = $this->balance;
         $this->balance -= $amount;
         if($this->balance > 0) {
             $comprobante  = "\n<table class=\"responstable\">\n";
             $comprobante .= "\t<tr>\n\t\t<td>Saldo inicial: </td>\n";
             $comprobante .= "\t\t<td>\$ " . number_format($saldoinicial,2,".",",")
. "</td>\n\t</tr>\n";
             $comprobante .= "<tr>\n<td>Operación: </td>\n";
             $comprobante   .=   "<td>"   .   mb_convert_case($this->operacion,
MB_CASE_TITLE, "UTF-8") . "</td>\n</tr>\n";
             $comprobante .= "\t\t<td>Cantidad retirada: </td>\n";
             $comprobante  .=  "\t\t<td>\$  "  .  number_format($amount,2,'.',',')  .
"</td>\t\n</tr>\n";
             $comprobante .= "\t<tr>\n<td>Nuevo saldo: </td>\n";
             $comprobante .= "<td>\$ " . number_format($this->balance,2,'.',',') .
"</td>\n</tr>\n";
             $comprobante .= "</table>\n";
         }
         else {
             $comprobante  = "\n<table class=\"responstable\">\n";
             $comprobante .= "\t<tr>\n\t\t<td>Aviso: </td>\n";
             $comprobante .= "\t\t<td>Su cuenta presenta fondos
insuficientes.</td>\n\t</tr>\n";
             $comprobante .= "</table>\n";
         }
         echo $comprobante;
      }

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

Archivo 2: bankform.php

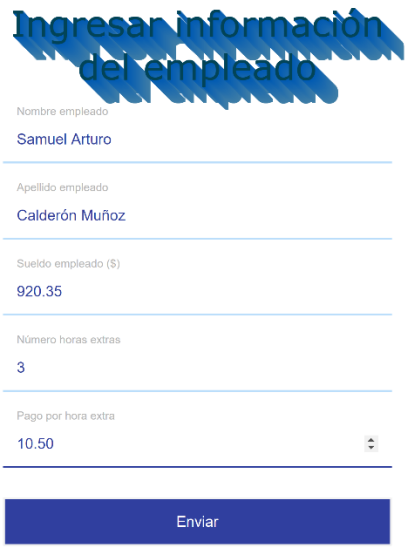
```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1, maximum-
scale=1">
    <title>Cuentas de ahorro</title>
    <link rel="stylesheet" href="css/formoid-flat-green.css" />
    <link rel="stylesheet" href="css/styles.css" />
    <link rel="stylesheet" href="css/table.css" />
    <!--[if lt IE 9]>
      <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
    <script src="js/jquery.min.js"></script>
</head>
<body>
<section>
<article>
<form name="operaciones" method="POST" action="<?php echo $_SERVER['PHP_SELF'] ?>"
class="formoid-flat-green">
    <div class="title">
        <h2>Operaciones bancarias</h2>
    </div>
    <div class="element-input">
        <label class="title">
            Nombre:<span class="required">*</span>
        </label>
        <input type="text" name="nombre" maxlength="40" required="required"
class="large" />
    </div>
    <div class="element-input">
        <label class="title">
            Cantidad:<span class="required">*</span>
        </label>
        <input type="text" name="cantidad" maxlength="10" required="required"
class="large" />
    </div>
    <div class="element-radio">
        <label class="title">Operación:</label>
        <div class="column column1">
            <label>
                <input type="radio" name="operacion" value="apertura"
checked="checked">
                <span>Apertura</span>
            </label>
        </div>
        <!--<span class="clearfix"></span>-->
        <div class="column column1">
            <label>
                <input type="radio" name="operacion" value="depósito">
                <span>Depósito</span>
            </label>
        </div>
        <!--<span class="clearfix"></span>-->
        <div class="column column1">
            <label>
                <input type="radio" name="operacion" value="retiro">
                <span>Retiro</span>
            </label>
        </div>
        <!--<span class="clearfix"></span>-->
    </div>
```

```
    <div class="submit">
        <input type="button" name="restablecer" value="Cancelar" />
        <input type="submit" name="enviar" value="Enviar">
    </div>
</form>
<?php
  //Usando autocarga de clases para iniciar el programa
  function __autoload($classname){
      include_once("class/" . $classname . ".class.php");
  }

  //Verificar que se ha enviado el formulario mediante el método POST
  if($_SERVER["REQUEST_METHOD"] == "POST" && isset($_POST['enviar'])){
      $msg = "";
      $titular = isset($_POST['nombre']) ? $_POST['nombre'] : "";
      if($titular == ""){
          $msg  = "<h3>El nombre de la cuenta no puede estar vacío</h3><br />";
      }
      $cantidad = isset($_POST['cantidad']) && is_numeric($_POST['cantidad']) ?
$_POST['cantidad'] : 0;
      if($cantidad == 0 || $cantidad < 0){
         $msg .= "<h3>La cantidad no puede ser negativa, ni cero.</h3><br />";
      }
      if($msg != ""){
          echo $msg;
          echo "<a href=\"{$_SERVER['PHP_SELF']}\">Volver al formulario</a><br />";
          exit(0);
      }
      $operacion = isset($_POST['operacion']) ? $_POST['operacion'] : null;
      $nuevacuenta = new bankAccount();
      switch($operacion){
          case "apertura":
              $nuevacuenta->openAccount($titular, $operacion, $cantidad);
              break;
          case "depósito":
              $nuevacuenta->makeDeposit($cantidad, $operacion);
              break;
          case "retiro":
              $nuevacuenta->makeWithdrawal($cantidad, $operacion, 500.00);
              break;
          default:
              echo "<p>No se ha seleccionado la operación a realizar</p>";
              exit();
      }
   }
?>
</article>
</section>
</body>
<script src="js/formoid-flat-green.js"></script>
</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:

Número de cuenta:	1
Propietario:	Juan Carlos Miranda
Operación:	Apertura
Saldo inicial:	$ 320.00

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. Esta aplicación se realizará con Programación Orientada a Objetos.
El formulario para ingreso debe ser como el siguiente junto con la boleta generada:



Id empleado:	1
Nombre empleado:	Samuel Arturo Calderón Muñoz
Salario nominal:	$ 920.35
Salario por horas extras:	$ 31.50
Descuentos	
Descuento seguro social:	$ 28.56
Descuento AFP:	$ 59.49
Descuento renta:	$ 83.23
Total descuentos:	$ 171.28
Sueldo líquido a pagar:	$780.57

El descuento de Seguro Social (ISSS) está establecido en 3%, el de AFP en 6.25% y, por último, el de la renta se rige en la siguiente tabla de niveles de ingresos:

Tramos	Desde	Hasta	Porcentaje
Tramo 1	$ 0.01	$ 472.00	0%
Tramo 2	$ 472.01	$ 895.24	10%
Tramo 3	$ 895.25	$ 2,038.10	20%
Tramo 4	$ 2,038.11	En adelante	30%

A diferencia de los otros dos descuentos, el de la Renta se aplica sobre la diferencia del salario nominal, más bonificaciones, horas extras y aguinaldos menos los descuentos realizados de ISSS y AFP. Una vez obtenida esa diferencia se ubica la cantidad obtenida en la tabla y se le aplica el respectivo descuento.

Archivo 1: empleado.class.php

```
<?php
  //Definición de la clase empleado
  class empleado     {
     //Estableciendo las propiedades de la clase
     private static $idEmpleado = 0;
     private $id;
     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 aquí porque pertenecen a la clase.
     //Se utilizarán los siguientes valores:
     //1. Descuento ISSS del 3%
     //2. Descuento Renta en base a tabla de ingresos
     //3. Descuento AFP de 6.25%
     const descISSS = 0.03;
     // const descRENTA = 0.10;
     const descAFP = 0.0625;

     //Constructor de la clase
     function __construct(){
        $this->id = 0;
        $this->nombre = "";
        $this->apellido = "";
        $this->sueldoLiquido = 0.0;
        $this->pagoxhoraextra = 0.0;
     }

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

     }
```

```
    //Métodos de la clase empleado
    function obtenerSalarioNeto($nombre, $apellido, $salario, $horasextras,
$pagoxhoraextra=0.0){
        $this->id = ++self::$idEmpleado;
        $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;
            $salariocondescuento = $this->sueldoNominal + $this->pagoxhoraextra -
($this->isss + $this->afp);
        }
        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;
        }
        $this->sueldoNominal = $salario;
        $this->sueldoLiquido = $this->sueldoNominal + $this->pagoxhoraextra - ($this-
>isss + $this->afp + $this->renta);
        $this->imprimirBoletaPago();
        }

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

```
        $tabla .= "<td> $" . number_format($this->sueldoLiquido, 2, '.', ',') .
"</td>\n</tr>\n";
        $tabla .= "</table>\n";
        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="css/material.css" />
    <link rel="stylesheet" href="css/links.css" />
    <link rel="stylesheet" href="css/table.css" />
</head>
<body>
<div class="contenedor-formulario">
  <div class="wrap">
<?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{
?>
    <h1>Ingresar información del empleado</h1>
    <form action="<?php echo $_SERVER['PHP_SELF'] ?>" method="POST"
class="formulario" name="formulario_registro">
      <div>
        <div class="input-group">
          <input type="text" id="nombre" name="nombre" maxlength="30">
          <label class="label" for="nombre">Nombre empleado</label>
        </div>
        <div class="input-group">
          <input type="text" id="apellido" name="apellido" maxlength="30">
          <label class="label" for="apellido">Apellido empleado</label>
        </div>
        <div class="input-group">
          <input type="number" id="sueldo" name="sueldo" maxlength="9" required
min="0.00" step=".01">
          <label class="label" for="sueldo">Sueldo empleado ($)</label>
        </div>
```

```
        <div class="input-group">
          <input type="number" id="horasextras" name="horasextras" maxlength="2">
          <label class="label" for="horasextras">Número horas extras</label>
        </div>
        <!--<div class="input-group radio">
          <input type="radio" name="sexo" id="hombre" value="Hombre">
          <label for="hombre">Hombre</label>
          <input type="radio" name="sexo" id="mujer" value="Mujer">
          <label for="mujer">Mujer</label>
        </div>-->
        <!--<div class="input-group checkbox">
          <input type="checkbox" name="terminos" id="terminos" value="true">
          <label for="terminos">Acepto los Terminos y Condiciones</label>
        </div>-->
        <div class="input-group">
          <input type="number" id="pagohoraextra" name="pagohoraextra"
maxlength="6" min="0.00" step=".01">
          <label class="label" for="pagohoraextra">Pago por hora extra</label>
        </div>
        <input type="submit" id="btn-submit" name="enviar" value="Enviar">
      </div>
    </form>
<?php
}
?>
  </div>
</div>
</body>
<script src="js/material.js"></script>
</html>
```

Ejercicio #4: En el presente ejercicio, se realiza un inicio de sesión utilizando el paradigma de Programación Orientada a Objetos en el cual se hace uso de dos clases para controlar la información de los usuarios y otra que controla el login de los mismos. Se ejemplifica el uso de objetos independientes de dos clases diferentes y la forma en que estos pueden compartir información de un objeto a otro. Observe que los script de clases deben almacenarse dentro de una carpeta denominada libs.

Archivo 1: usuario.class.php

```
<?php
class Usuario {
       /*****************************
        * Declaramos las propiedades *
        *****************************/
       private $_id;
       private $_nombre;
       private $_pass;
       private $_email;
       private $_usuarios;

       //contructor con dos parametros opcionales
       function __construct($nombre = '', $pass = ''){
              $this->_nombre = $nombre;
              $this->_pass = $pass;
       }

       /*********************************************************
        * Métodos para establecer/obtener información del usuario *
        *********************************************************/
    public function setId($id){
        $this->_id = $id;
```

```
    }

    public function getId(){
        return $this->_id;
    }

    public function setNombre($nombre){
        $this->_nombre = $nombre;
    }

       public function getNombre(){
        return $this->_nombre;
    }

    public function setPass($pass){
        $this->_pass = $pass;
    }

       public function getPass(){
        return $this->_pass;
    }

    public function setEmail($email){
        $this->_email = $email;
    }

    public function getEmail(){
        return $this->_email;
    }

       public function obtener_datos(){
              $arreglo["username"] = $this->_nombre;
              $arreglo["pass"] = $this->_pass;
              return $arreglo;
       }

       //cifrar password a SHA254
       private function cifrarPass($pass){
              $this->_pass = hash('sha256', $pass);
              return $this->_pass;
       }
}
?>
```

Archivo 2: login.class.php

```
<?php
class Login  {
       private $_id;
       private $_nombre;
       private $_pass;
       private $_status = false;

       //constructor con dos parametros opcionales
       function __construct($nombre = '', $pass = ''){
              $this->_nombre = $nombre;
              $this->_pass = $pass;
       }

       /*********************************************************
        * Clases para establecer/Obtener información del usuario *
        *********************************************************/
    public function setId($id){
        $this->_id = $id;
```

```
    }

    public function getId(){
        return $this->_id;
    }

    public function setNombre($nombre){
        $this->_nombre = $nombre;
    }

    public function getNombre(){
        return $this->_nombre;
    }

       public function setPass($pass){
         $this->_pass = $pass;
    }

    public function getPass(){
        return $this->_pass;
    }

       //verifica si el password coincide y retorna falso o verdadero
       public function verificarUsuario($fila_datos){
              $username = array('docente','estudiante','maria','ricardo');
              $password = array('udbFET','fetUDB','maria','P@$$w0rD');

              if (in_array($fila_datos["pass"], $password) &&
in_array($fila_datos["username"], $username)){
                     $this->_status = true;
              } else {
                     $this->_status = false;
              }
              return $this->_status;
       }

       //verifica el estado de la session
       public function getStatus(){
              return $this->_status;
       }

       //cierra la session y elimina las variables
       public function cerrarSession(){
              $this->_nombre = '';
              $this->_pass = '';
              $this->_status = false;
       }
}
?>
```

Archivo 3: login.php

```
<?php
//Utilizando autocarga de clases
function __autoload($classname){
    require "class/" . $classname . ".class.php";
}

//Condicional - si existen datos por medio $_POST se hace
if ($_POST) {
       $nombre = isset($_POST["nombre"]) ? $_POST["nombre"] : '';
       $pass = isset($_POST["pass"]) ? $_POST["pass"] : '';

       //Instaciamos clases
```

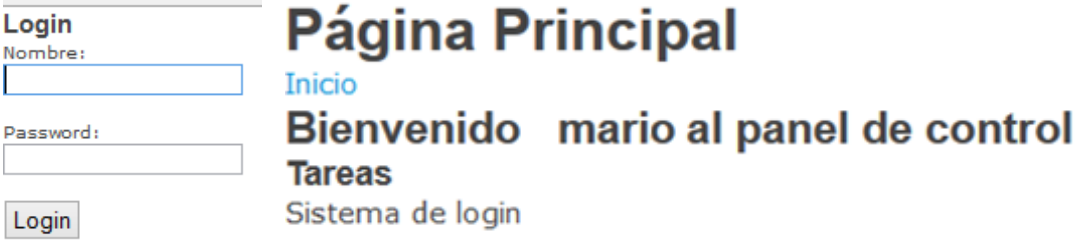
```
	$Usuario = new Usuario($nombre, $pass);
	$Login = new Login( $nombre, $pass );

	$filaUsuario = $Usuario->obtener_datos();
	$login = $Login->verificarUsuario($filaUsuario);

	if($login) {
		//direccionar
		header("Location: index.php?usuario=" . $Usuario->getNombre());
	} else {
		//direccionar
		header("Location: login.php?error=1");
	}
}
?>
<!DOCTYPE html>
<html lang="es">
<head>
	<meta charset="utf-8" />
	<title>Login</title>
	<link rel="stylesheet" href="css/style.css">
	<!--[if lt IE 9]>
	 <script src="
https://cdnjs.cloudflare.com/ajax/libs/html5shiv/3.7.3/html5shiv.js"></script>
    <![endif]-->
</head>
<body>
	<section id="login">
		<header>
			<h2>Login</h2>
		</header>
		<form method="post" action="login.php">
			<?php
				if (isset($_GET["error"]) and $_GET["error"] == 1){
			?>
			<h3>Error al intentar iniciar sesión</h3>
			<?php
			   }
			?>
			<dl>
				<dt><label for="nombre">Nombre:</label></dt>
				<dd><input type="text" name="nombre" value=""></dd><br>
				<dt><label for="pass">Password:</label></dt>
				<dd><input type="password" name="pass" value=""></dd><br>
			</dl>
			<input type="submit" value="Login">
		</form>
	</section>
</body>
</html>
```

Archivo 4: index.php

```
<?php
	$nombre = isset($_GET['usuario']) ? $_GET['usuario'] : null;
	if(!isset($nombre)){
		header("Location: login.php");
	}
?>
<!DOCTYPE html>
<html>
<head>
	<meta charset="utf-8" />
	<title>Index de login</title>
```

```
        <link rel="stylesheet" href="css/style.css">
    <!--[if lt IE 9]>
       <script src="http://html5shiv.googlecode.com/svn/trunk/html5.js"></script>
    <![endif]-->
</head>
<body>
        <header>
                <h1>Página Principal</h1>
        </header>
        <nav>
                <ul>
                        <li><a href="index.php">Inicio</a></li>
                </ul>
        </nav>
        <section id="intro">
                <header>
                        <h2>Bienvenido <?php echo $nombre; ?> al panel de control</h2>
                </header>
                <section>
                        <header>
                                <h3>Tareas</h3>
                        </header>
                </section>
        </section>
        <aside>
        </aside>
        <footer>
          <p>Sistema de login</p>
        </footer>
</body>
</html>
```

Debe iniciar el script login.php en cualquier navegador, ingresar los datos que aparecen en las matrices $username y $password de la clase login.class.php y posteriormente al dar clic en el botón Login si ingresó bien los datos debería ver la página de inicio del sitio.



Ejercicio #5: En el presente ejercicio, se realiza una aplicación donde las páginas web del sitio se crean utilizando el paradigma de la Programación Orientada a Objetos (POO) considerando una clase llamada página (page), en donde se identifican como propiedades los elementos característicos de una página web como el contenido en una propiedad $content, el título de la página en una propiedad $title, las palabras clase, típicamente utilizadas en una etiqueta meta, los enlaces del menú principal de una propiedad $buttons, que para este caso contendrá una matriz o arreglo con todos los enlaces que incluyen el título del enlace que será el texto visible para los usuarios que naveguen en la página y el enlace o URL hacia donde apunta la página. Luego, se definen dentro de esta misma clase los métodos que permitirán generar la página web solicitada, como display(), displayTitle(), displayStyles(), displayScripts(), etc.

Archivo 1: page.class.php

```
<?php
/*************************************************
```

```
*  Descripción: Clase para crear páginas web      *
*  Creador: Ricardo Elías                         *
*  Fuente: Libro Desarrollo web con PHP y MySQL   *
*  Fecha: 10-marzo-2013                           *
************************************************/
class page {
       //Atributos de la clase
       public $content;
       public $title = "Centro de Estudios de Postgrados - Universidad Don Bosco
&copy;";
       public $keywords = "Universidad Don Bosco, UDB, Educaci&oacute;n con estilo
salesiano";
       public $buttons = array(
                              "Inicio" => "home.php",
                              "Carreras" => "carreras.php",
                              "Institucional" => "institucional.php",
                              "Contacto" => "contacto.php"
                           );
       //Operaciones de la clase
       public function __set($name, $value){
    $this->name = $value;
       }

       public function display(){
              echo "<!DOCTYPE html>\n";
              echo "<html lang=\"es\">\n<head>\n";
    echo "\t<meta charset=\"utf-8\" />\n";
              $this->displayTitle();
              $this->displayKeywords();
              $this->displayStyles("css/home.css");
              $this->displayScripts("js/modernizr.custom.lis.js");
              echo "</head>\n<body>\n";
    $this->displayHeader();
    $this->displayMenu($this->buttons);
    echo $this->content;
    $this->displayFooter();
    echo "</body>\n</html>";
       }

       public function displayTitle(){
              echo "\t<title>" . $this->title . "</title>\n";
       }

       public function displayKeywords(){
              echo  "\t<meta  name=\"keywords\"  content=\""  .  $this->keywords  .  "\"
/>\n";
       }

       public function displayStyles($estilos){
              //Patrón de expresión regular para verificar
    //si la extensión del archivo es .css
    $patron = "%\.{1}(css)$%i";
    $styles = "";
    if(is_array($estilos)):
      foreach($estilos as $cssfile):
        $styles .= "\t<link rel=\"stylesheet\" href=\"" . $cssfile . "\" />\n";
      endforeach;
    else:
      $styles .= "\t<link rel=\"stylesheet\" href=\"" . $estilos . "\" />\n";
    endif;
    echo $styles;
       }
```

```
  public function displayScripts($scripts){
    //Patrón de expresión regular para verificar
    //que la extensión del archivo es .js
    $patron = "%\.{1}(js)$%i";
    if(is_array($scripts)):
      foreach($scripts as $scriptfile):
        echo   "\t<script   type=\"text/javascript\"   src=\""   .   $scriptfile   .
"\"></script>\n";
      endforeach;
    else:
        if(!empty($scripts)):
          if(preg_match($patron, $scripts)):
            echo   "\t<script   type=\"text/javascript\"   src=\""   .   $scripts   .
"\"></script>\n";
          endif;
        endif;
    endif;
    $scripts   =   "\t<script   type=\"text/javascript\"   src=\""   .   $scripts   .
"\"></script>\n";
  }

      public function displayHeader(){
             $header = <<<HEADER
    <!-- page header -->
    <section>
        <article>
        <table width="100%" cellpadding="12" cellspacing="0" border="0">
            <tr bgcolor="black">
                <td align="left">
                    <img src="img/logo.gif" alt="Logo URL" height="70" width="70" />
                </td>
                <td>
                    <h1>Universidad Don Bosco</h1>
                </td>
                <td align="right">
                    <img src="img/logo.gif" alt="Logo URL" height="70" width="70" />
                </td>
            </tr>
        </table>
HEADER;
        echo $header;
       }

  public function displayMenu($buttons){
    $menu  = "<ul id=\"mainmenu\">\n\t";
    //Calcular tamaño
    $width = 100/count($buttons);
    while(list($name, $url) = each($buttons)){
      $menu .= "<li>\n\t\t";
      $menu     .=     $this->displayButton($width,     $name,     $url,     !$this-
>isURLCurrentPage($url)) . "\n\t\t";
      $menu .= "</li>\n";
    }
    $menu .= "</ul>\n";
    echo $menu;
      }

  function isURLCurrentPage($url){
    if(strpos($_SERVER['PHP_SELF'], $url) == false):
      return false;
    else:
      return true;
    endif;
```

```
  }

  public function displayButton($width, $name, $url, $active=true){
    $button = "";
    if($active):
      $button .= "<a href=\"" . $url . "\">\n\t\t";
      $button .= "<img src=\"img/url-icon.png\" alt=\"" . $name . "\" />\n\t";
      $button .= "</a>\n\t";
      $button .= "<a href=\"" . $url . "\">\n\t\t";
      $button .= "<span class=\"menu\">" . $name . "</span>\n\t";
      $button .= "</a>\n";
    else:
      $button .= "<img src=\"img/url-icon.png\" alt=\"" . $name . "\" />\n\t";
      $button .= "<span class=\"menu\">" . $name . "</span>\n";
    endif;
    return $button;
  }

  public function displayFooter(){
    $footer = <<<FOOT
        <!-- Pie de la página -->
        <table id="footer">
          <tr>
            <td>
              <p class="foot">
                <a   href="http://www.udb.edu.sv"   target="_blank">Universidad   Don
Bosco</a>
              </p>
              <p class="foot">Centro de Estudios de Postgrados.</p>
            </td>
          </tr>
        </table>
      </article>
    </section>
FOOT;
    echo $footer;
  }
}
?>
```

Archivo 2: home.php

```
<?php
  function __autoload($class_name){
      require("class/" . $class_name . ".class.php");
  }

  //Creando el objeto página
  $homepage = new page();

  $homepage->content = <<<PAGE
        <!-- page content -->
        <div id="topcontent">
           <div id="textbox">
              <div id="title">
                 <h2>BIENVENIDOS</h2>
              </div>
              <div id="paragraph">
                 <p>
                    La Universidad Don Bosco en sus 27 años de experiencia educativa,
                    ha mantenido una expansión constante en su oferta académica,
                    lo cual puede comprobarse en su trayectoria desde su creación en
                    1984.<br />
```

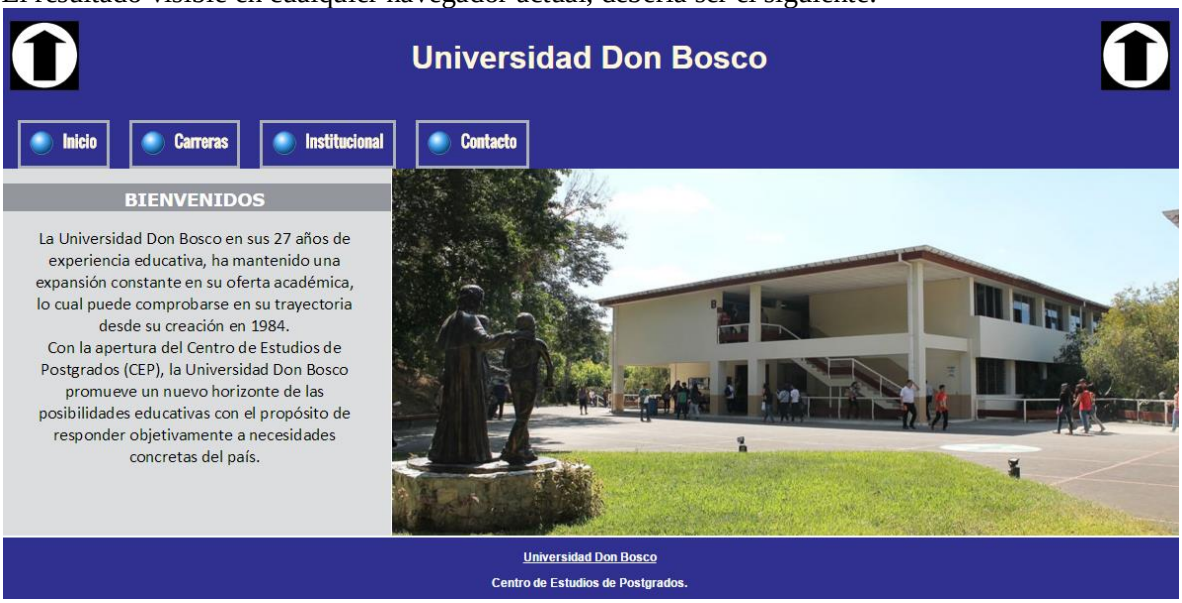
```
                    Con la apertura del Centro de Estudios de Postgrados (CEP), la
Universidad
                    Don Bosco promueve un nuevo horizonte de las posibilidades
educativas
                    con el propósito de responder objetivamente a necesidades
concretas
                    del país.
                  </p>
               </div>
            </div>
            <div id="picture">
               <img  src="img/plaza-de-las-banderas.jpg"  alt="Imagen de portada"
width="800" height="370" longdesc="Imagen de portada" />
            </div>
         </div>
PAGE;
  echo $homepage->display();
?>
```

Archivo 3: contacto.php

```
<?php
<?php
   function __autoload($class_name){
      require("class/" . $class_name . ".class.php");
   }

   //Creando el objeto página
   $contactpage = new page();
   $contactpage->content = <<<PAGE
      <!-- contact content -->
        <div id="topcontent">
           <div id="textbox">
              <div id="title">
                 <h2>CONTACTO</h2>
              </div>
              <div id="paragraph">
                 <h4>Números de atención:</h4>
                 <p>
                    Universidad Don Bosco.<br />
                    Tel. (503)2251-8200.
                 </p>
                 <p>
                    Administración Académica.<br />
                    Tel. (503)2251-8200 ext. 1710.
                 </p>
                 <p>
                    Administración Financiera.<br />
                    Tel. (503)2251-8200 ext. 1700.
                 </p>
                 <p>
                    Nuevo Ingreso.<br />
                    Tel. (503)2527-2314.
                 </p>
              </div>
           </div>
           <div id="picture">
              <img src="img/entradapostgrados.jpg" alt="Imagen de portada" width="800"
height="370" longdesc="Imagen de portada" />
           </div>
         </div>
PAGE;
   echo $contactpage->display();
```

```
?>
```

El resultado visible en cualquier navegador actual, debería ser el siguiente:



Ejercicio #6: 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";
         }
      }
   }
?>
```

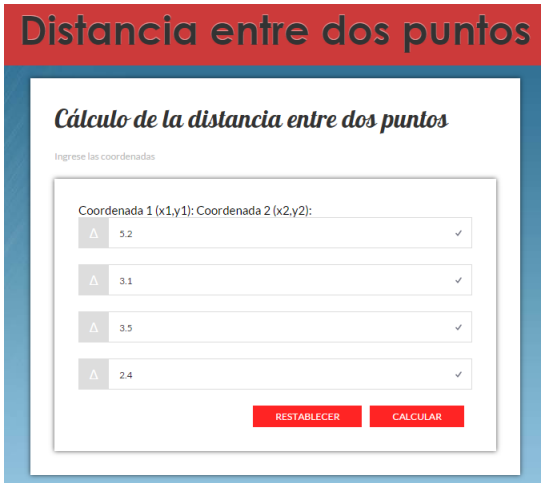
Archivo 2: distanciadospuntos.php

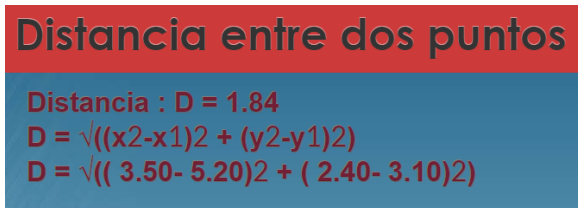
```
<!DOCTYPE html>
<html lang="es">
<head>
```

```
    <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:





Ejercicio #7: Ejemplo que muestra cómo crear imágenes miniatura de muestra a partir de otra imagen de tamaño mayor en formato PNG. Para la implementación se ha utilizado el concepto de herencia de clases creando una superclase llamada archivo y a partir de ella una subclase llamada archivoPNG que explota las características de una imagen en dicho formato. Se utilizan varias funciones para manejo de imágenes de PHP.

Archivo 1: archivoPNG.class.php

```
<?php
  class archivo{
    //Definiendo las propiedades del objeto
    protected $nombrearchivo;
    private $octetos;
    private $fechamodificacion;
    const PATH = 'img/';

    //Creando los métodos
    function __construct($f){
        if(is_file(self::PATH . $f)){
            $this->nombrearchivo = self::PATH . $f;
            $this->octetos = (filesize($this->nombrearchivo));
            $this->fechamodificacion = (filemtime($this->nombrearchivo));
        }
        else{
            die("**** ERROR: No se encuentra el archivo '$f'  ****");
        }
    }

    function octetos(){
        return $this->octetos;
    }

    function fechamodificacion(){
        return $this->fechamodificacion;
    }

    function renombrar($nombrenuevo){
        if(rename($this->nombrearchivo, $nombrenuevo)){
            $this->nombrearchivo = $nombrenuevo;
            return 1;
        }
        else {
            echo "**** ERROR: No se puede renombrar el archivo";
        }
    }
}

class archivoPNG extends archivo{
    //Propiedades de la clase
    private $alto;
    private $ancho;
    private $bitsporcolor;
    //Métodos de la clase
```

```
    function __construct($f){
        parent::__construct($f);
        $props = getimagesize(parent::PATH . $f);
        $ind_tipo_img = 2;
        if($props[$ind_tipo_img] != 3){   //Es un archivo PNG
            die("%%%% ERROR: '$f' no tiene formato gráfico PNG %%%%");
        }
        else{
            $ind_alto_img = 0;
            $ind_ancho_img = 1;
            $this->alto = $props[$ind_alto_img];
            $this->ancho = $props[$ind_ancho_img];
            $this->bitsporcolor = $props['bits'];
        }
    }

    function creamuestra($archivomuestra, $anchomuestra, $altomuestra){
        $imgmuestra = ImageCreate($anchomuestra, $altomuestra);
        $imgoriginal = ImageCreateFromPNG($this->nombrearchivo);
        ImageCopyResized($imgmuestra,   $imgoriginal,   0,0,0,0,   $anchomuestra,
$altomuestra, ImageSX($imgoriginal), ImageSY($imgoriginal));
        ImagePNG($imgmuestra, parent::PATH . $archivomuestra);
    }

    function   mostrarimagenesPNG($imagenoriginal,   $imagenmuestra,   $objorig,
$objmuestra){
        echo "<table id=\"emblemas\">\n
                    <tr>\n<th>\nImagen\n</th>\n
                       <th>\nTamaño\n</th>\n
                       <th>\nDimensiones</th>\n</tr>\n
                       <tr                                  class=\"even\">\n<td>\n<img
src='img/$imagenoriginal'>\n</td>\n
                       <td>\n" . $objorig->octetos() . " Kb</td>\n
                       <td>\n" . $objorig->ancho() . "x" . $objorig->alto() . "
px</td>\n</tr>\n
                       <tr                                   class=\"odd\">\n<td>\n<img
src='img/$imagenmuestra'>\n</td>\n
                       <td>\n" . $objmuestra->octetos() . " Kb</td>\n
                       <td>\n" . $objmuestra->ancho() . "x" . $objmuestra->alto() .
" px</td>\n</tr>\n
                       </table>";
    }

    function bitsporcolor(){
        return $this->bitsporcolor;
    }

    public function alto(){
        return $this->alto;
    }

    public function ancho(){
        return $this->ancho;
    }
  }
?>
```

Archivo 2: miniaturas.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>Creación de imágenes de muestra</title>
```

```
    <link rel="stylesheet" href="css/miniaturas.css" />
</head>
<body>
<header>
    <h1 class="engraved">Imágenes miniatura con PHP</h1>
</header>
<section>
<article>
<?php
   function __autoload($classname){
      require_once "class/" . $classname . ".class.php";
   }

   $objpng = new archivoPNG("escudo-el-salvador.png");
   //Probando la creación de la imagen en miniatura
   $archivoimagen = "escudo-el-salvador.png";
   $archivoimagenmuestra = "thumbs-escudo-es.png";
   $objpng = new archivoPNG($archivoimagen);
   $objpng->creamuestra($archivoimagenmuestra,64,64);
   $objpngmuestra = new archivoPNG($archivoimagenmuestra);
   $objpng->mostrarimagenesPNG($archivoimagen, $archivoimagenmuestra, $objpng,
$objpngmuestra);
?>
</article>
</section>
</body>
</html>
```

Puedes verificar en el navegador de tu preferencia ejecutando el script miniaturas.php el resultado.

Ejercicio #8: Este ejemplo muestra cómo implementar una clase para crear campos de formulario. La clase campoformulario se crea como una clase abstracta con uno de sus métodos abstractos, el método que crea un campo de formulario específico. Todas las clases derivadas de esta tendrán que implementar este método abstracto para poder crear el tipo de campo de formulario adecuado.

Archivo 1: campotexto.class.php

```
<?php
  //Clase abstracta para algún tipo de campo de formulario
  abstract class campoformulario {
        //Propiedades de la clase abstracta
        protected $idcampo;
        protected $etiqueta;
        protected $capaayuda;

        //Constructor de la clase
```

```
        function __construct($id, $etiq, $ayuda){
              $this->idcampo = $id;
              $this->etiqueta = $etiq;
              $this->capaayuda = $ayuda;
       }

       //Método abstracto que será implementado
       //por alguna de las clases hijas o derivadas
       abstract function pinta_campo();

       protected function poner_eventos_js(){
              $cmd_js               =       'document.getElementById("c_"       +
this.name).style.visibility';
              $cmd2_js = 'document.getElementById("c_" + this.name).style.display';
              return " onfocus='$cmd_js=\"visible\"; $cmd2_js=\"inline-block\";'
onblur='$cmd_js=\"hidden\"; $cmd2_js=\"none\"'";
       }

       protected function poner_capa_ayuda(){
              //El identificador de la capa es
              //'c_' + nombre de la capa
              $s  = "background: Lavender; ";
              $s .= "border: 1px solid #4D274F; ";
              $s .= "color: #7B0F86; ";
              $s .= "font: Bold 0.85em \"Open Sans\",Arial,Helvetica,sans-serif; ";
              $s .= "padding: 4px 6px; ";
              $s .= "position:relative; ";
              $s .= "text-shadow: 0 -1px 0 #BFADC0,\n 0 -2px 0 #B099B2,\n 0 -3px 0
#9C889F,\n 0 0 2px #816883,\n 0 -2px 3px #715973,\n 3px -3px 15px #000; ";
              $s .= "display: none; ";
              $s .= "visibility:hidden;";
              return "<span id='c_$this->idcampo' style='$s'>
                      $this->capaayuda</span>\n";
       }
  }

  //Clase para un campo de formulario de tipo text
  class campotexto extends campoformulario {
        //Definiendo las propiedades
        private $placeholder;
        private $maxcar;
        //Creando un nuevo constructor
        function __construct($id, $etiq, $ayuda, $placehold, $maxcar){
              parent::__construct($id, $etiq, $ayuda);
              $this->placeholder = $placehold;
              $this->maxcar = $maxcar;
        }
        //Implementando el método abstracto pinta_campo
        function pinta_campo(){
              $stag  = "";
              $stag .= "\t<div class=\"row\">\n";
              $stag .= "\t<section class=\"col col-6\">\n";
              $stag .= "\t\t<label for=\"$this->idcampo\" class=\"input\">\n";
           $stag .= "\t\t<i class=\"icon-prepend icon-user\"></i>\n";
              $stag  .= "\t\t<input  type=\"text\"  name=\""  .  $this->idcampo  .  "\"
id=\"".   $this->idcampo   .   "\"   placeholder=\""   .   $this->placeholder   .   "\"
maxlength=\"" . $this->maxcar . "\"" . $this->poner_eventos_js() . " />\n";
              $stag .= $this->poner_capa_ayuda();
              $stag .= "</label>";
              $stag .= "\t</section>\n";
              $stag .= "\t</div>\n";
              echo $stag;
        }
```

```
  }

  //Clase para un campo de formulario tipo textarea
  class campotextarea extends campoformulario {
        //Propiedades
     private $placeholder;
        private $lineas;
        private $cols;
        //Definiendo un constructor para esta clase
        function __construct($id, $etiq, $lineas, $cols, $ayuda, $placehold){
              parent::__construct($id, $etiq, $ayuda);
              $this->placeholder = $placehold;
              $this->lineas = $lineas;
              $this->cols = $cols;
        }

        function pinta_campo() {
           $tag  = "\t<section>\n";
           $tag  .=  "\t\t<label  for=\"$this->idcampo\"  class=\"textarea\">$this-
>etiqueta\n";
              $tag .= "\t\t<textarea name=\"" . $this->idcampo . "\" id=\"" . $this-
>idcampo  . "\" rows=\"" . $this->lineas  . "\" cols=\"" . $this->cols  . "\" " .
$this->poner_eventos_js() . ">";
              $tag .= $this->placeholder;
              $tag .= "</textarea>\n";
              $tag .= $this->poner_capa_ayuda();
              $tag .= "</label>";
              $tag .= "\t</section>\n";
              echo $tag;
        }
  }

  class campocheckbox extends campoformulario {
     //Definiendo las propiedades
     private $options = array();
     private $listed;
     //Creando el constructor de la clase campocheckbox
     function __construct($id, $name, $etiq, $ayuda, $options, $enlistados=false){
        parent::__construct($id, $name, $etiq, $ayuda);
        $this->options = $options;
        $this->etiqueta = $etiq;
        $this->enlistados = $enlistados;
     }
     //Implementando el método abstracto crearcampo
     function pinta_campo(){
        $pos = 0;  //Indica la posición en el arreglo de opciones del checkbox
        //echo    "<label    for=\"$this->idcampo"    .    $pos    .    "\">$this-
>etiqueta</label><br />\n";
        $stag  = "";
        $stag .= "\t<section>\n";
        $stag .= "\t\t<label>" . $this->etiqueta . "</label>\n";
        //Recorriendo el array con las opciones del checkbox
        foreach($this->options['opciones'] as $key => $value){
           $stag .= "\t\t<label class=\"checkbox\">\n";
           $stag  .=  "\t\t<input  type=\"checkbox\"  value=\"$value\"  name=\"$this-
>idcampo" . $pos . "\" id=\"$this->idcampo" . $pos . "\" ";
           $stag .= $this->options['estados'][$pos] == true ? "checked=\"checked\"
" : "";
           $stag .= " /><i class=\"fa-check\"></i>$key</label>\n";
           $stag .= $this->enlistados == true ? "<br />\n" : "";
           $pos++;
        }
        $stag .= "\t</section>\n";
```

```
        echo $stag;
        echo $this->enlistados == true ? "<br />\n" : "";
    }
  }

  class camposelect extends campoformulario{
     //Propiedades de la clase
        private $size;
        private $multiple;
        private $options = array();
        //Método constructor
        function __construct($id, $etiq, $ayuda, $size, $multiple, $options){
        parent::__construct($id, $etiq, $ayuda);
        $this->size = $size;
        $this->multiple = $multiple;
        $this->options = $options;
        }
     function pinta_campo(){
       $mult = ($this->multiple != "") ? " " . $this->multiple : "";
        $seltag  = "<div class=\"row\">\n";
        $seltag .= "\t<label for=\"" . $this->idcampo . "\" class=\"label col col-
4\">$this->etiqueta</label><br />\n";
        $seltag .= "\t<section class=\"col col-5\">";
        $seltag .= "\t\t<label class=\"select\">";
             $seltag .= "\t\t<select name=\"" . $this->idcampo . "\" id=\"" . $this-
>idcampo . "\" size=\"" . $this->size . "\"" . $this->poner_eventos_js() . ">\n";
        foreach($this->options as $key => $value){
           $seltag .= "\t\t\t<option value=\"$key\">$value</option>\n";
        }
        $seltag .= "\t\t</select>\n";
        $seltag .= $this->poner_capa_ayuda();
        $seltag .= "\t\t<i></i>\n";
        $seltag .= "\t\t</label>\n";
             $seltag .= "</div>\n";
        echo $seltag;
     }
  }

  class camposubmit extends campoformulario{
     //Propiedades adicionales
        private $value;
        //Constructor de la clase camposubmit
        function __construct($id, $etiq, $value, $ayuda){
        parent::__construct($id, $etiq, $ayuda);
        $this->value = $value;
     }

     function pinta_campo(){
        $subtag  = "<input type=\"submit\" name=\"" . $this->idcampo . "\" id=\""
. $this->idcampo . "\" value=\"" . $this->value . "\" class=\"button\"";
             $subtag .= $this->poner_eventos_js() . " />\n";
             $subtag .= $this->poner_capa_ayuda();
             $subtag .= "<br />\n";
        echo $subtag;
     }
  }
?>
```

Archivo 2: abstractforms.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
```

```
    <title>Formulario dinámico</title>
    <!-- <link rel="stylesheet" href="css/fonts.css" /> -->
    <link rel="stylesheet" href="css/demo.css" />
    <!-- Estilos tomados desde de los ejemplos de http://voky.com.ua/ -->
    <link rel="stylesheet" href="css/sky-forms.css" />
    <link rel="stylesheet" href="css/sky-forms-purple.css" />
    <!--[if lt IE 9]>
      <link rel="stylesheet" href="css/sky-forms-ie8.css">
    <![endif]-->

    <!--[if lt IE 10]>
      <script
src="http://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
      <script src="js/jquery.placeholder.min.js"></script>
    <![endif]-->
    <!--[if lt IE 9]>
      <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
      <script src="js/sky-forms-ie8.js"></script>
    <![endif]-->
</head>
<body class="bg-purple">
<div class="body">
    <h1>Generador de formulario</h1>
<?php
  function __autoload($classname){
    include_once "class/" . $classname . ".class.php";
  }

  echo "<form name=\"form\" action=\"mostrarform.php\" method=\"POST\" class=\"sky-
form\"
onsubmit=\"return true\">\n\t";
  echo "\t<header>Formulario dinámico</header>\n";
  echo "\t<fieldset>\n";
  //En el array $campos se crean todos los campos que van a conformar el formulario
  //utilizando todas las clases que se han creado para cada uno de los controles
  //de formulario considerados para este ejemplo
  $campos = array(
                new campotexto       ("nombre", "Nombre: ", "El nombre completo",
"Nombre completo",
40),
                new campotexto          ("apellido", "Apellido: ", "El apellido
completo", "Apellido
completo", 30),
                new campocheckbox   ("deportes", "Deportes", "Deportes:", "Deportes
favoritos",
                                     array(
                                         "opciones" => array(
                                             "Fútbol"      => "Fútbol",
                                             "Basketball"  => "Basketball",
                                             "Volleyball"  => "Volleyball",
                                             "Beisball"    => "Beisball",
                                             "Tenis"       => "Tenis"
                                         ),
                                         "estados"  => array(
                                             true,
                                             false,
                                             false,
                                             true,
                                             false
                                         )
                                     ),
                                    false
                                    ),
```

```
                new campotextarea    ("observaciones", "Observaciones: ", 6, 50,
"Háganos sus
comentarios", "Envíenos sus comentarios"),
                new camposelect     ("nacionalidad", "Nacionalidad: ", "Seleccione
su acionalidad",
1, '',
                                     array(
                                        "El Salvador" => "El Salvador",
                                        "Guatemala" => "Guatemala",
                                        "Honduras" => "Honduras",
                                        "Costa Rica" => "Costa Rica",
                                        "Nicaragua" => "Nicaragua",
                                        "Panamá" => "Panamá"
                                     )
                                    ),
                new camposubmit    ("enviar", "", "Enviar", "Enviar el formulario")
            );
  foreach($campos as $campo){
      $campo->pinta_campo();
  }
  echo "</fieldset>\n";
  echo "</form>\n";
?>
</div>
</body>
</html>
```

Archivo 3: mostrarform.php

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="utf-8" />
    <title>Datos del formulario</title>
    <link rel="stylesheet" href="http://fonts.googleapis.com/css?family=Bitter" />
    <link rel="stylesheet" href="css/liststyle.css" />
</head>
<body class="bg-cyan">
<section>
<div class="form-style">
<h1>Datos del formulario</h1>
<div class="numberlist">
<?php
   if(isset($_POST['enviar'])){
     $lista = "<ol>\n";
      foreach($_POST as $name => $value){
         if(gettype($value) == "array" && $name != "enviar"){
            foreach($value as $dato){
               $lista .= "\t<li><a href=\"javascript:void(0);\">$name: " . $dato .
"</a></li>\n";
            }
         }
         if($name != "enviar"){
            $lista .= "\t<li><a href=\"javascript:void(0);\">$name: " . $value .
"</a></li>\n";
         }
      }
      $lista .= "</ol>\n";
      echo $lista;
   }
?>
</div>
</div>
</article>
```

```
</section>
</body>
</html>
```

Al visualizar en el navegador el script abstractforms.php podremos observar el formulario creado dinámicamente haciendo uso de la clase campotexto.class.php, que en realidad sirve para definir todos campos de formulario que se muestran en dicho formulario.

Ejercicio #9: El siguiente ejemplo ilustra cómo pueden utilizarse las interfaces para que dos clases no relacionadas (Television y TennisBall) implementen los métodos definidos en la interfaz, cada una por su cuenta.

Archivo 1: sellable.class.php

```
<?php
   interface sellable {
      public function addStock($numItems);
      public function sellItem();
      public function getStockLevel();
   }
?>
```

Archivo 2: television.class.php

```
<?php
   class television implements sellable {
      //Propiedades de la clase television
      private $_screenSize;
      private $_stockLevel;
      //Métodos de la clase television
      public function getScreenSize() {
         return $this->_screenSize;
      }

      public function setScreenSize($screenSize) {
         $this->_screenSize = $screenSize;
      }

      public function addStock($numItems) {
         $this->_stockLevel += $numItems;
      }
```

```
      public function sellItem() {
         if($this->_stockLevel > 0){
            $this->_stockLevel--;
            return true;
         }
         else {
            return false;
         }
      }

      public function getStockLevel() {
         return $this->_stockLevel;
      }
   }
?>
```

Archivo 3: tennisBall.class.php

```
<?php
   class tennisBall implements sellable {
      //Propiedades de la clase tennisBall
      private $_color;
      private $_ballsLeft;

      //Métodos de la clase TennisBall
      public function getColor() {
         return $this->_color;
      }

      public function setColor($color) {
         $this->_color = $color;
      }

      public function addStock($numItems) {
         $this->_ballsLeft += $numItems;
      }

      public function sellItem() {
         if($this->_ballsLeft > 0) {
            $this->_ballsLeft--;
            return true;
         }
         else {
            return false;
         }
      }

      public function getStockLevel() {
         return $this->_ballsLeft;
      }
   }
?>
```

Archivo 4: storeManager.class.php

```
<?php
   class storeManager {
      //Propiedades de la clase StoreManager
      private $_productList = array();
      //Métodos de la clase StoreManager
      public function addProduct(Sellable $product){
         $this->_productList[] = $product;
      }
```

```
      public function stockUp() {
         foreach ($this->_productList as $product){
            $product->addStock(100);
         }
      }
   }
?>
```

Archivo 5: sell.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>Crear y usar interfaces</title>
    <link rel="stylesheet" href="css/common.css" />
</head>
<body>
<header>
    <h1>Almacén El Machetazo"</h1>
</header>
<section>
   <article>
<?php
   function __autoload($clase){
      require_once("class/" . $clase . ".class.php");
   }
   $tv = new television;
   $tv->setScreenSize(42);
   $ball = new tennisBall;
   $ball->setColor("amarilla");
   $manager = new storeManager();
   $manager->addProduct($tv);
   $manager->addProduct($ball);
   $manager->stockUp();
   echo "<p>Existen " . $tv->getStockLevel() . " televisores de " . $tv-
>getScreenSize();
   echo "-pulgadas y " . $ball->getStockLevel() . " pelotas de tenis " . $ball-
>getColor();
   echo " en existencia.</p>";
   echo "<p>Vendiendo un televisor...</p>";
   $tv->sellItem();
   echo "<p>Vendiendo dos pelotas de tenis...</p>";
   $ball->sellItem();
   $ball->sellItem();
   echo "<p>Ahora existen ". $tv->getStockLevel() . " televisores de " . $tv-
>getScreenSize();
   echo "-pulgadas y " . $ball->getStockLevel() . " pelotas de tenis " . $ball-
>getColor();
   echo " en existencia.</p>";
?>
   </article>
</section>
</body>
</html>
```

En el navegador de su preferencia, al ejecutar el script sell.php podremos ver lo siguiente:

Ejercicio #10: Ejemplo completo que ilustra todos los conceptos de la programación orientada a objetos.

Archivo 1: poocompleto.class.php

```
<?php
   //Definiendo una interfaz
   interface leer{
      function tieneLibro($tituloLibro);
      function leeLibro();
   }
   //Definiendo la clase abstracta persona
   abstract class persona {
   //Propiedades de la clase abstracta persona.
      //Esta clase definirá el método abstracto ocupacionPrincipal()
      private static $numPersonas = 0;  //número total de personas
      protected $nombre = '';    //Nombre de la persona
      protected $fechaNac = '';  //Fecha de nacimiento de la persona
      //Constructor de la clase abstracta persona
      function __construct($nombrePersona = 'sin nombre'){
         self::$numPersonas++;
         $this->nombre = $nombrePersona;
      }
      function __clone(){
         self::$numPersonas++;
      }
      //Destructor de la clase abstracta persona
      function __destruct(){
         self::$numPersonas--;
      }
      function __toString(){
         $cadena  = 'nombre(' . $this->nombre;
         $cadena .= ') Poblaci&oacute;n(' . self::$numPersonas . ')';
         return $cadena;
      }
      //Métodos de la clase abstracta persona
      final public static function poblacion(){
         return self::$numPersonas;
      }
      final public function asignaNombre($nombreAsignado){
         $this->nombre = $nombreAsignado;
      }
      public function dameNombre(){
         return $this->nombre;
      }
      //Se fuerza la implementación del método en las subclases
      abstract public function ocupacionPrincipal();
   }  //Final clase persona

   //Definiendo la clase derivada empleado.
   //Esta clase implementará la interfaz leer
   class empleado extends persona implements leer {
      //Propiedades de la clase empleado
      private static $idEmpleados = 0;  //Número de empleados
```

```
      protected $id;          //Id del empleado
      private $libro = NULL;  //Título del libro que está leyendo
      //Constructor de la clase empleado
      function __construct($nombreEmpleado){
         parent::__construct($nombreEmpleado); //Invocando al constructor de clase
padre
         self::$idEmpleados++;
         $this->id = self::$idEmpleados;
      }
      //Destructor de la clase empleado
      function __destruct(){
         parent::__destruct();
              self::$idEmpleados--;
      }
      function __clone(){
         parent::__clone();
         self::$idEmpleados++;
         $this->id = self::$idEmpleados;
      }
      function __toString(){
         $cadena  = __CLASS__ . ": id($this->id) nombre($this->nombre)";
         $cadena .= ' poblaci&oacute;n(' . parent::poblacion() . ')';
         if(!empty($this->libro)){
            $cadena .= $this->leerLibro();
         }
         return $cadena;
      }
      public function ocupacionPrincipal(){
         return 'trabajar';
      }
      public function tieneLibro($tituloLibro){
         $this->libro = $tituloLibro;
      }
      public function leeLibro(){
         if(empty($this->libro))
            return 'No est&aacute; leyendo ning&uacute;n libro actualmente.';
         else
            return "Est&aacute; leyendo \"$this->libro\"";
      }
   }  //Final clase empleado
   //Definiendo la clase estudiante.
   //Esta clase implementará también la interface leer
   class estudiante extends persona implements leer{
      //Métodos de la clase abstracta estudiante
        protected $estudios = '';  //Estudios realizados
        private $libro = NULL;     //Título de libro que está leyendo
        //Cosntructor de la clase abstracta estudiante
        function __construct($nombreEstudiante, $estudios = ''){
        parent::__construct($nombreEstudiante);
        $this->estudios = $estudios;
        }
        function __destruct(){
        parent::__destruct();
        }
      function __clone(){
         parent::__clone();
      }
      public function __toString(){
         $cadena  = __CLASS__ . ": nombre($this->nombre) ";
              $cadena .= "estudios($this->estudios) poblaci&oacute;n(";
              $cadena .= parent::poblacion() . ")<br />";
              if(!empty($this->libro)){
            $cadena .= $this->leeLibro();
```

```
                }
                return $cadena;
        }
        //Métodos de la clase abstracta estudiante
        public function ocupacionPrincipal(){
           return "estudiar($this->estudios)";
        }
        public function tieneLibro($tituloLibro){
           $this->libro = $tituloLibro;
        }
        public function leeLibro(){
           if(empty($this->libro))
                   return 'No est&aacute; leyendo ning&uacute;n libro actualmente.';
           else
              return " est&aacute; leyendo \"$this->libro\"";
        }
     }  //Fin clase estudiante
     //Definiendo la clase bebe.
     class bebe extends persona {
        //Constructor de la clase bebe
           function __construct($nombreBebe){
           parent::__construct($nombreBebe);
        }
        //Destructor de la clase bebe
        function __destruct(){
           parent::__destruct();
        }
        function __clone(){
           parent::__clone();
        }
        //Métodos de la clase bebe
        public function ocupacionPrincipal(){
           return 'Comer y dormir';
        }
     }  //Final clase bebe
?>
```

Archivo 2: poocompleto.php

```
<?php
   function __autoload($classname){
      if($classname == "empleado" || $classname == "estudiante" || $classname ==
"bebe"){
         require_once "class/poocompleto.class.php";
      }
      else{
         require_once "class/" . $classname . ".class.php";
      }
   }

   //Creando al primer empleado
   echo "&ndash; Primera persona y empleado: <br />\n";
   $employee1 = new empleado('Carlos L&oacute;pez');
   echo $employee1 . "<br /><br />\n";
   //Creando al empleado 2 utilizando clonación
   echo "&ndash; Segundo empleado creado con clonaci&oacute;n: <br />\n";
   $employee2 = clone $employee1;
   $employee2->asignaNombre('Mar&iacute;a Calder&oacute;n');
   echo $employee2 . "<br /><br />\n";
   //Se crea el empleado 3 y luego se anula
   echo "&ndash; Se crea al empleado 3 y despu&eacute;s se anula su referencia: <br
/>\n";
   $employee3 = new empleado('Sonia Cu&eacute;llar');
   echo $employee3 . "<br /><br />\n";
```

```
   $employee3 = NULL;
   //Se crea un par de personas más
   echo "&ndash; Creaci&oacute;n de 2 personas m&aacute;s que no son empleados: <br
/>\n";
   $persona1 = new estudiante('Medardo Enrique Somoza', 'Ingenier&iacute;a en
Computaci&oacute;n');
   $persona1->tieneLibro('La Biblia de PHP 5');
   echo $persona1 . "<br /><br />\n";
   $persona2 = new bebe('Beb&eacute; 1');
   echo $persona2 . "<br />\n";
   echo "Principal ocupaci&oacute;n: ";
   echo $persona2->ocupacionPrincipal() . "<br /><br />\n";
   echo "Poblaci&oacute;n actual: " . persona::poblacion() . "<br /><br />\n";
   //Se crea un nuevo empleado reutilizando el identificador $id
   echo "&ndash; Creando al empleado 4 y mostrando informaci&oacute;n: <br />\n";
   $employee4 = new empleado('Pedro Cruz');
   echo $employee4 . "<br /><br />\n";
   echo "Poblaci&oacute;n actual: " . persona::poblacion() . "<br /><br />\n";
?>
```

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 deducirlo 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.
3. Implemente en el Ejercicio 5 del procedimiento las opciones de menú Carreras e Institucional que deberán llamar a los scripts carreras.php e institucional.php, respectivamente. Utilice la lógica orientada a objetos que se ha utilizado en los scripts home.php y contacto.php. Modifique el diseño de las páginas Carreras e Institucional, de modo que muestre otra distibución de elementos de página. Puede utilizar párrafos, imágenes, tablas y cualquier otro elemento HTML5. Sea creativo.
4. Modifique el ejemplo de interfaces (Ejercicio 9 del procedimiento) creando una nueva clase a la que llamará movie con tres propiedades, una para el título de la película ($title), otra para el género de la película ($genre) y una más para el número de existencias de esa película que deberá actualizarse cada vez que se vende una nueva película y cada vez que se adquieren nuevos ejemplares de la misma. Cree métodos similares a las de las clases television y tennisBall para establecer valores a las propiedades título y género y otro para controlar las existencias, así como se hace en las las mencionadas clases television y tennisBall. Anexe a los resultados de la ejecución del script sell.php los datos de la película que está en venta.

VI. DISCUSIÓN DE RESULTADOS

1. Investigue sobre las constantes y métodos mágicos de PHP relacionados con la utilización de clases y objetos. Además, averigüe sobre los métodos de PHP que realizan una tarea similar

a la realizada por las constantes mágicas, así como los métodos o funciones de PHP para determinar si una clase está definida y para determinar si un método existe en una clase.

VII. BIBLIOGRAFÍA

- Cabezas Granado, Luis Miguel. PHP 6 Manual Imprescindible. 1ra. Edición. Editorial Anaya Multimedia. Madrid, España. 2010.
- Doyle, Matt. Fundamentos de PHP Práctico. Editorial Anaya Multimedia. 1ª. Edición. Madrid, España. 2010.
- Gutierrez, Abraham / Bravo, Ginés. PHP 5 a través de ejemplos. 1ra Edición. Editorial Alfaomega. Junio 2005. México.
- Welling, Luke / Thomson, Laura. Desarrollo web con PHP y MySQL. Traducción de la 3ra Edición en inglés. Editorial Anaya Multimedia. 2005. Madrid, España.
- Gil Rubio / Francisco Javier, Villaverde / Santiago Alonso. Creación de sitios web con PHP5. 1a. edición en español. Editorial McGraw Hill. Madrid, España. 2006.
- John Coggeshall. LA BIBLIA DE PHP 5. 1ra. Edición. Editorial Anaya Multimedia. Madrid, España 2005.
- Ellie Quigley / Marko Gargenta. PHP y MySQL Práctico para Diseñadores y Programadores web. Primera edición. Editorial Anaya Multimedia. Madrid, España 2007.