

ÉRICA PALOMINO

TEMA 9

# PHP ORIENTADO A OBJETOS

**[erica.palomino@escuelaartegranada.com](mailto:erica.palomino@escuelaartegranada.com)**

ESCUELAARTEGRANADA

# INTRODUCCIÓN

La sintaxis para crear una clase en PHP es bastante sencilla: usando la palabra reservada **class**.

Dentro de una clase podremos declarar **propiedades** y **métodos** muy parecido a como lo hacíais en JAVA.

Para acceder a una propiedad o método de la clase desde cualquier punto de ella será ESCRITAMENTE necesario utilizar **\$this**.

Para acceder a cualquier elemento (método o propiedad) de un objeto, en PHP se utiliza el operador **->** (no el **.** como en JAVA)

# INTRODUCCIÓN

Para indicar desde dónde y cómo se puede acceder a los atributos y a los métodos utilizaremos las palabras reservadas:

- **public:** podrán ser accedidos desde cualquier parte, incluso desde fuera de la clase
- **protected:** sólo podrán ser accedidos desde el interior de la propia clase o de sus hijos.
- **private:** sólo podrán ser accedidos desde el interior de la clase donde se han definido.
- **static:** podrán ser accedidos sin necesidad de instanciar la clase.



# **MÉTODOS MÁGICOS**

# MÉTODOS MÁGICOS

Un método mágico es un método que ya existe por defecto en cualquier clase que creemos. Pero estos métodos estarán vacíos, deberemos sobrecargarlos siempre.

Un método mágico suele ser llamado de forma automática, sin que nosotros lo llamemos de forma explícita.

El ejemplo más sencillo es el método **`_construct()`**.

Este método nos permite crear un objeto nuevo de una clase, pero nunca lo llamamos de forma explícita si no que se llama automáticamente cuando hacemos un **`new`**.

# MÉTODOS MÁGICOS

No podremos crear ningún método con el mismo nombre de un método mágico.

Habrà que **sobrecargarlos** para que hagan lo que queramos.

Siembre van precedidos de 2 guiones bajos (..)

**\_\_construct()**: método constructor

# MÉTODOS MÁGICOS

**`__destruct()`**: Permite destruir el objeto y liberar su memoria.

**`__get()`**: Se utiliza para consultar el contenido de las propiedades privadas de la clase.

**`__set()`**: Se utiliza para introducir contenido en las propiedades privadas de la clase

**`__isset()`**: Se utiliza para ver si tiene contenido una propiedad definida como privada

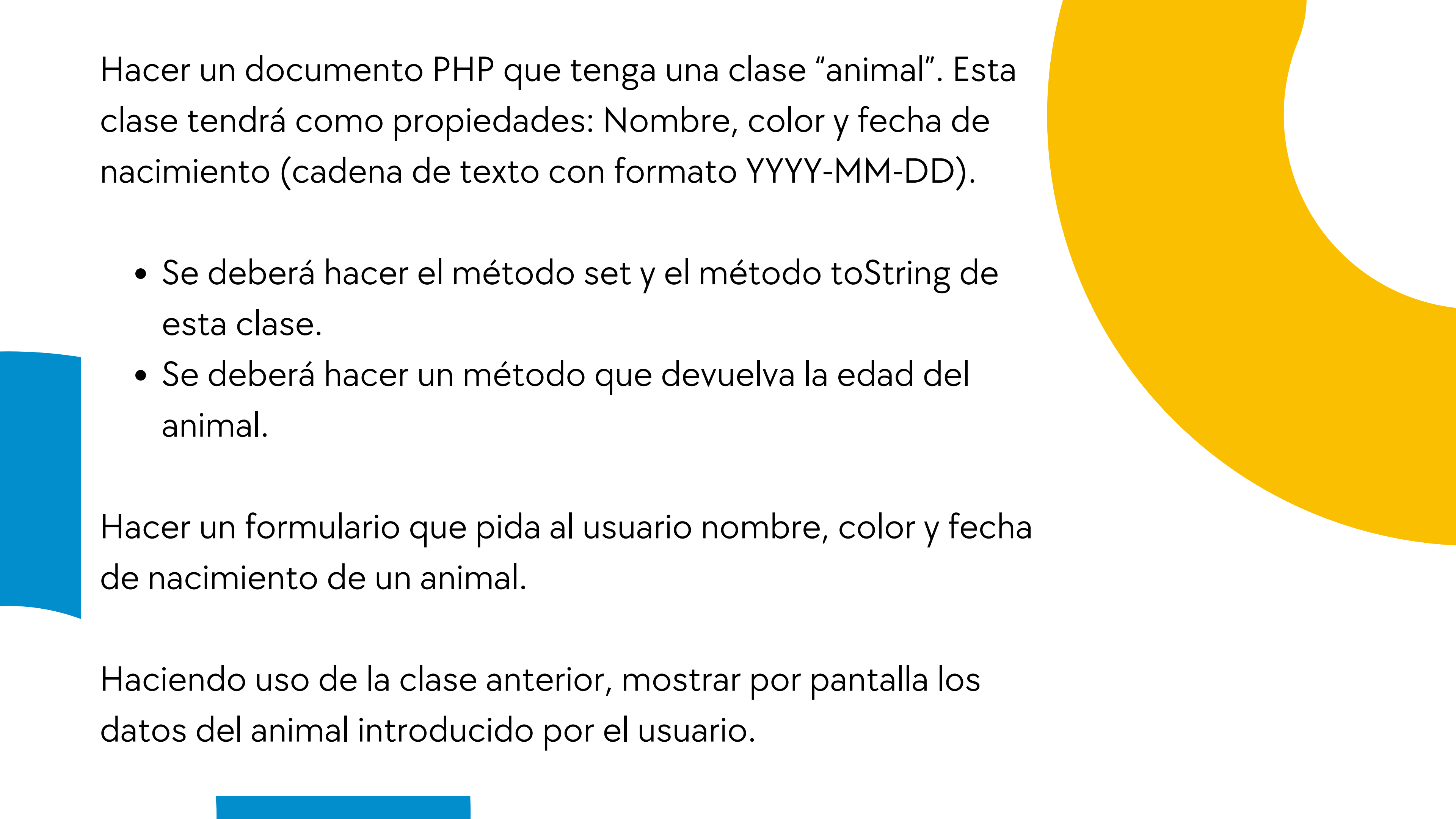
# MÉTODOS MÁGICOS

**`_unset()`**: Se utiliza para vaciar una propiedad privada.

**`_toString()`**: método que será llamado cuando intentemos mostrar por pantalla un objeto (completo) de esta clase.



# EJERCICIOS



Hacer un documento PHP que tenga una clase “animal”. Esta clase tendrá como propiedades: Nombre, color y fecha de nacimiento (cadena de texto con formato YYYY-MM-DD).

- Se deberá hacer el método set y el método toString de esta clase.
- Se deberá hacer un método que devuelva la edad del animal.

Hacer un formulario que pida al usuario nombre, color y fecha de nacimiento de un animal.

Haciendo uso de la clase anterior, mostrar por pantalla los datos del animal introducido por el usuario.

Crear una clase «Vehículo». Esta clase tendrá 3 propiedades: nombre, tipo y peso.

Ambas propiedades serán privadas.

- En la propiedad tipo se guardará
  - 'C' para camión
  - 'M' para moto
  - 'T' para turismo
- En la propiedad peso se guardarán las toneladas que pesa el vehículo.

Además, la clase debe tener los métodos:

- Constructor: para crear un objeto metiendo todos los valores
- Get: para obtener el valor de las propiedades
- Set: para cargar el valor de las propiedades.
- toString: para mostrar una frase con todos los datos del vehículo

Crear ahora un documento en PHP que, por medio de formularios, pida al usuario tipo y peso de dos vehículos diferentes. El formulario debe tener unos campos tipo radio para el tipo de vehículo y un campo de texto para el peso.

El documento deberá comprobar si los dos vehículos son del mismo tipo o no.

- Si son del mismo tipo, mostrará por pantalla el vehículo que pese más.
- Si son del mismo tipo y pesan igual, dará un mensaje indicándolo.
- Si son de distinto tipo deberá indicar al usuario que no se pueden comparar.

Crea una clase llamada **Imagen**. Esa clase será la encargada mostrar una imagen en código HTML.

La clase Imagen contendrá tres atributos de ámbito privado:

- **src** (atributo source de la etiqueta img): nombre o ruta donde se encuentra la imagen.
- **border** (atributo border de la etiqueta img): ancho que rodea la imagen
- **ruta\_images**: ruta donde se encuentran las imágenes.

Dos métodos:

- El constructor al cual le pasamos como parámetro el src (nombre del fichero o ruta) y el borde que será un parámetro optativo que por defecto estará inicializado a 0. El atributo src hay que concatenarle el atributo ruta\_images y el src que nos pasan como parámetro. Deberá comprobarse que ruta\_images, existe.
- El método mágico **\_\_toString** que devuelve el objeto Imagen como cadena devolviendo el código HTML para construir la imagen: (<img src='<ruta>' border='<borde>' />)





# **CONSTANTES MÁGICAS**

# MÉTODOS MÁGICOS

De la misma forma, PHP tiene definidas una serie de **constantes** que nos permitirán encontrar información fácilmente.

Siempre irán precedidas de `_` y seguidas de `_` (dos guiones bajos)

Constante	Descripción
<code>__LINE__</code>	Número de línea actual del fichero
<code>__FILE__</code>	Ruta completa del fichero
<code>__DIR__</code>	Directorio en el que se encuentra el fichero
<code>__FUNCTION__</code>	Nombre de la función que se está ejecutando
<code>__CLASS__</code>	Nombre de la clase
<code>__METHOD__</code>	Nombre del método de la clase



# **HERENCIA DE CLASES**



# MÉTODOS MÁGICOS

Para que una clase que creamos pueda heredar las características de otra, utilizaremos la palabra reservada **extends**.

PHP permite que una subclase sobrecargue métodos de la clase superior.

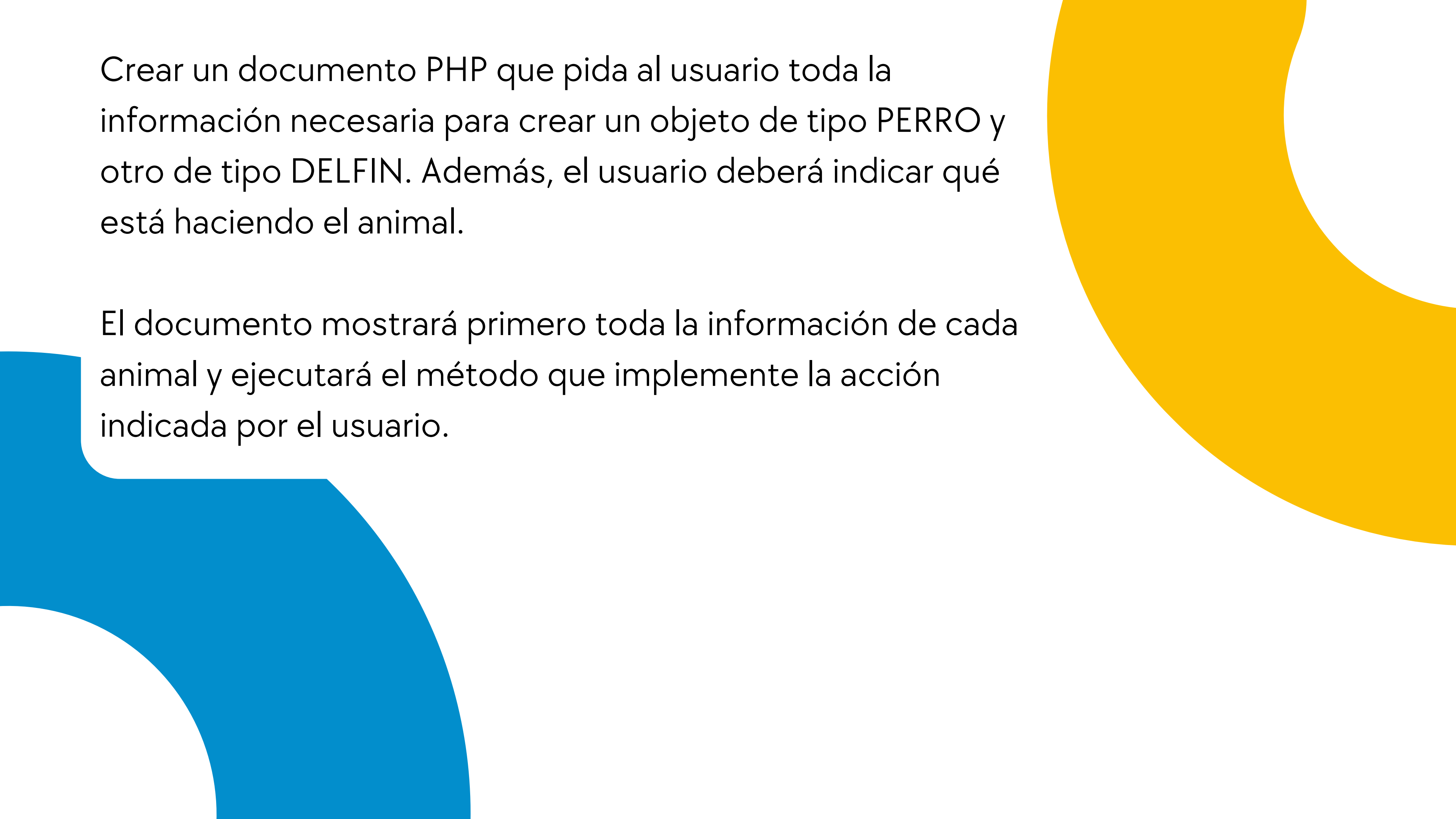
Si queremos preservar lo que hacía el método en la clase superior utilizaremos la palabra reservada **parent** junto con el operador de resolución de ámbito (::)

# EJERCICIOS

Partiendo de la clase "animal" que creamos antes, vamos a crear varias clases que hereden de esta:

- Clase "perro":
  - La clase "perro" tendrá dos propiedades "raza" y "sexo".
  - Además, tendrá los siguientes métodos:
    - "ladrar": mostrará el mensaje "<nombre> dice GUAU"
    - "dormir": mostrará el mensaje "<nombre> se ha dormido"
- Clase "delfin"
  - La clase "delfin" tendrá una propiedad "longitud"
  - Además, tendrá los siguientes métodos:
    - "saltar": mostrará el mensaje "<nombre> está saltando por los aires"
    - "comer": mostrará el mensaje "<nombre> tiene hambre"

Ambas clases deberán tener también sus métodos `_set` y `_get` así como `_toString`



Crear un documento PHP que pida al usuario toda la información necesaria para crear un objeto de tipo PERRO y otro de tipo DELFIN. Además, el usuario deberá indicar qué está haciendo el animal.

El documento mostrará primero toda la información de cada animal y ejecutará el método que implemente la acción indicada por el usuario.

Crea dos clases, **Articulo** y **ArticuloRebajado** que será una subclase de **Articulo**.

La clase **Articulo** se guardará en un fichero llamado "class.articulo.php" y contendrá dos atributos protegidos **nombre** y **precio**.

El constructor recibirá como parámetros obligatorios **pNombre** y **pPrecio** y le asignará dichos valores a sus correspondientes atributos.

Implementar el método **\_toString()** que devuelva la siguiente cadena:

```
'Nombre:' . $this->nombre . '<br />' . 'Precio: ' .  
$this->precio . '&euro;<br />' ;
```

Un método **getPrecio** que devuelva el valor actual del precio y **setPrecio(\$pPrecio)** que compruebe si la variable **\$pPrecio** es un valor numérico y si es así, se lo asigne al atributo **precio** de la clase **Articulo**.

Crea un nuevo fichero llamado "herencia.php" que requiera la definición (usa la función que comprueba que no se haya requerido la definición de la clase anteriormente y si es así provoca un error fatal) "Articulo" y dentro de dicho fichero crea la clase **ArticuloRebajado** que será declarada como **final** (no se podrá heredar de ella) y será hija de la clase **Articulo**. Esta clase contendrá:

- Un atributo privado llamado \$rebaja
- El constructor por defecto que recibe como parámetros obligatorios \$pNombre, \$pPrecio y \$pRebaja. Dentro de dicho constructor habrá que llamar al constructor del padre para darle valor a nombre y precio. Después le daremos valor al atributo rebaja.
- Un método privado llamado calculaDescuento() que nos devuelve el precio por la rebaja dividido por 100.

- Un método público llamado `precioRebajado()` que nos devuelve la diferencia entre el precio y el descuento
- El método `_toString()` que nos devuelve:
  - Lo que nos devuelve el método `_toString()` de la clase padre `Articulo` (habrá que realizar una llamada explícita al método `_toString()` del padre)
  - Junto con la cadena 'La rebaja es: ' . `$this->rebaja . '%'`;
  - y por último 'El descuento es ' . `self::calculaDescuento(). '€'`

Después de definir la clase `**ArticuloRebajado**` crea una instancia de la misma con los parámetros: `nombre='Bicicleta'`, `precio=352.10` y `rebaja=20`.

Imprime con un `echo` el objeto y en la siguiente línea esta cadena "El precio del artículo rebajado es" concatenado con la función que nos devuelve el valor del precio rebajado junto el signo del euro y un salto de línea.