

SOC

Servei d'Ocupació
de Catalunya



DESENVOLUPAMENT D'APLICACIONS AMB TECNOLOGIES WEB



Posible clasificación

Lenguajes de programación:

- Antiguos (procedimientos, década de los 60 y principios de los 70)
- Modernos (Orientado a Objetos. Más actuales)

Lenguajes de programación:

- **Antiguos** (Fortran, Cobol, Basic...)
 - Código largo en aplicaciones complejas
 - Errores difíciles de encontrar, y muy probable que el programa caiga.
 - Uso por otro programador complicado
 - Difícil de reutilizar (muy difícil de cambiar)
- **Modernos**
 - Abstracción del mundo real.

Ejemplo de código Basic (diferente de Visual Basic)

Listado de un programa en BASIC

```

LOAD TWO SQUARES
LIST

30 HOME : HGR
35 PRINT CHR$(4);"BLOAD SHAPE 1,A$0300"
36 POKE 232,0: POKE 233,03
37 POKE - 16302,0
40 ST$ = "T W O S Q U A R E S "
60 A = 90:B = 5
70 HCOLOR= 7: ROT= 0
80 FOR S = 1 TO 40
90 SCALE= S
100 DRAW 2 AT A,B
101 DRAW 2 AT A + 1,B + 1
105 HPLOT 90,5 TO 10,86
110 NEXT S
120 HCOLOR= 5
130 X = 260:Y = 75
140 ROT= 5
150 FOR S = 60 TO 1 STEP - 1
160 SCALE= S
170 DRAW 2 AT X,Y: DRAW 2 AT X + 1,Y + 1
180 NEXT S
185 PRINT CHR$(4);"BLOAD SHAPE ALPHABET,A$6000"
186 VT = 21:HT = 2
187 SCALE= 2: ROT= 0
*190 GOSUB 5000
195 REM THIS ADDS LINES TO FRAME 1
200 FOR S = 200 TO 175 STEP - 4
```

Beginner's
All-Purpose
Symbolic
Instruction
Code

Programacion en PHP

- Programación lineal: código desarrollado con código php alternado con el HTML.
- Programación estructurada: estructuramos el código con funciones , a las cuales luego llamamos.
- Programación orientada a objetos: se definen clases y objetos.

Programacion orientada a objetos

‘Evolución del paradigma de programación estructurada que permite la modelización de objetos del mundo real, con sus características y propiedades internas.’

Dos tipos de lenguajes orientados a objetos:

- Lenguajes puros: todos los módulos de un programa han de ser clases (Scala).
- Lenguajes híbridos: permiten utilizar clases y objetos que se pueden combinar con estructuras procedimentales clásicas (Java, C++, JavaScript, PHP).

¿En qué consiste?


- Realizar una abstracción del mundo real (objetos)
- Teniendo en cuenta que los objetos tienen unas características y un comportamiento, lo que se llamarán propiedades o atributos y métodos.

1- Ejemplo de objeto; el objeto coche.

Un objeto puede tener un estado, unas propiedades y un comportamiento

Posibles características y qué comportamiento tiene que hace que sea un coche:

- Estado: parado, en marcha, aparcado...
- Características: color, tamaño, marca...
- Comportamiento: arrancar, girar, frenar...



El objeto coche no se refiere a una única entidad sino a un conjunto de entidades con unas características comunes, sería la clase coche.

Si hablamos del coche de Maria, nos estamos refiriendo a un coche en concreto, se ha de entender como un objeto o instancia que identifica un miembro individual y concreto de la clase coche.

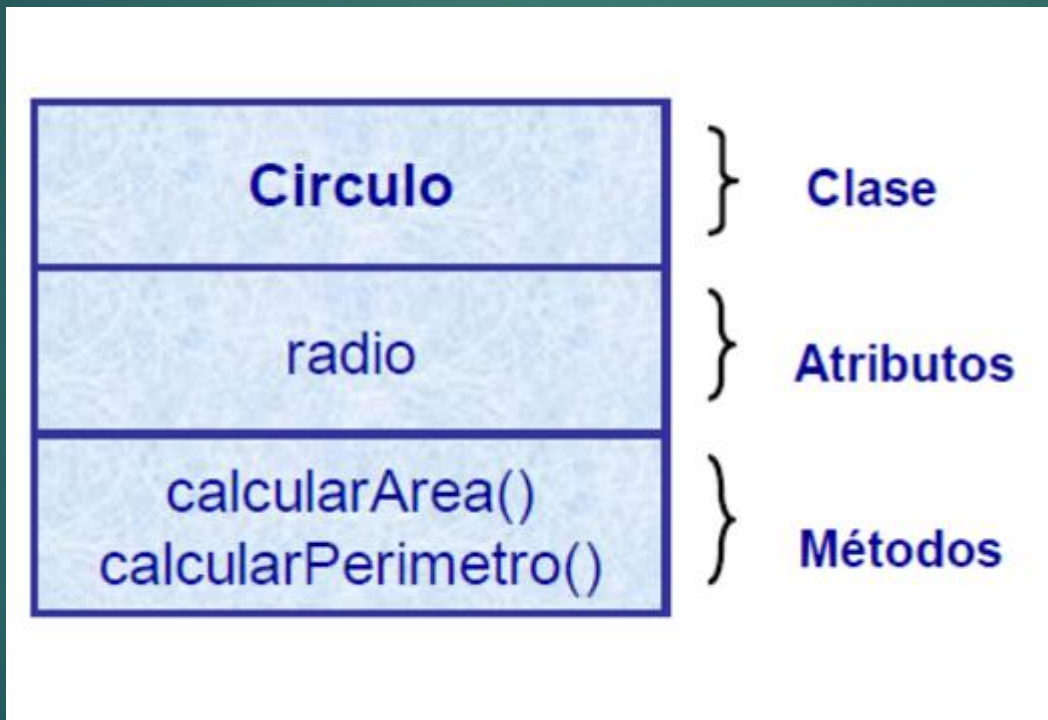
2- Ejemplo de clase Persona.



En la clase persona están representadas las propiedades y comportamientos que caracterizan a una persona (entidad del mundo real), mientras que los objetos representan a individuos concretos.

Pensar en una possible classe...

Por ejemplo la clase círculo:



Ventajas de la POO:

- Un programa se puede dividir en trozos de código, módulos, clases... **Modularización**
- Muy reutilizable. **Herencia.**
- Si existe algún fallo en alguna línea de código, el programa no se caerá. Se tratan los errores (**excepciones**).
- **Encapsulamiento.** Los objetos se pueden comunicar entre sí pero no necesitan saber de los datos de los otros.

Equipo compacto de música:



Equipo modular de música:



Modularización:

Dividir un gran programa en diferentes partes que se unen entre sí para formar un todo.

Ventajas:

- Errores más localizables y más fáciles de resolver

Dos o más clases unidas entre ellas para formar una unidad.

Conceptos

1. Clase
2. Objeto
3. Instancia de clase
4. Modularización
5. Encapsulamiento
6. Herencia
7. *Polimorfismo*

Clase

**Cuenta
bancaria**

Nombre de
la clase

Características
de la clase

Número
Titular
Tipo
Saldo

Objetos de la clase

Objeto 1

Número: 45878
Titular: Sandra Parra
Tipo: Ahorros
Saldo: 500,000

Objeto 2

Número: 75369
Titular: Pedro Torres
Tipo: Corriente
Saldo: 1,800,000

Objeto 3

Número: 85214
Titular: Luz Gómez
Tipo: Ahorros
Saldo: 120,000

Los objetos tienen sus propiedades (atributos) (sus características) y sus métodos(funcionalidades).

El método: es una función asociada a un objeto, es una acción que se ejecuta sobre los datos del objeto (por ejemplo calcular el área). Se definen como las funciones.

Para acceder:

`nombreObjeto.propiedad;`

En php `nombreObjeto->propiedad`

`nombreObjeto.metodo(param1,param2);`

En php `nombreObjeto->metodo(param1,param2`

Creación de clases PHP:



primeraClase.php

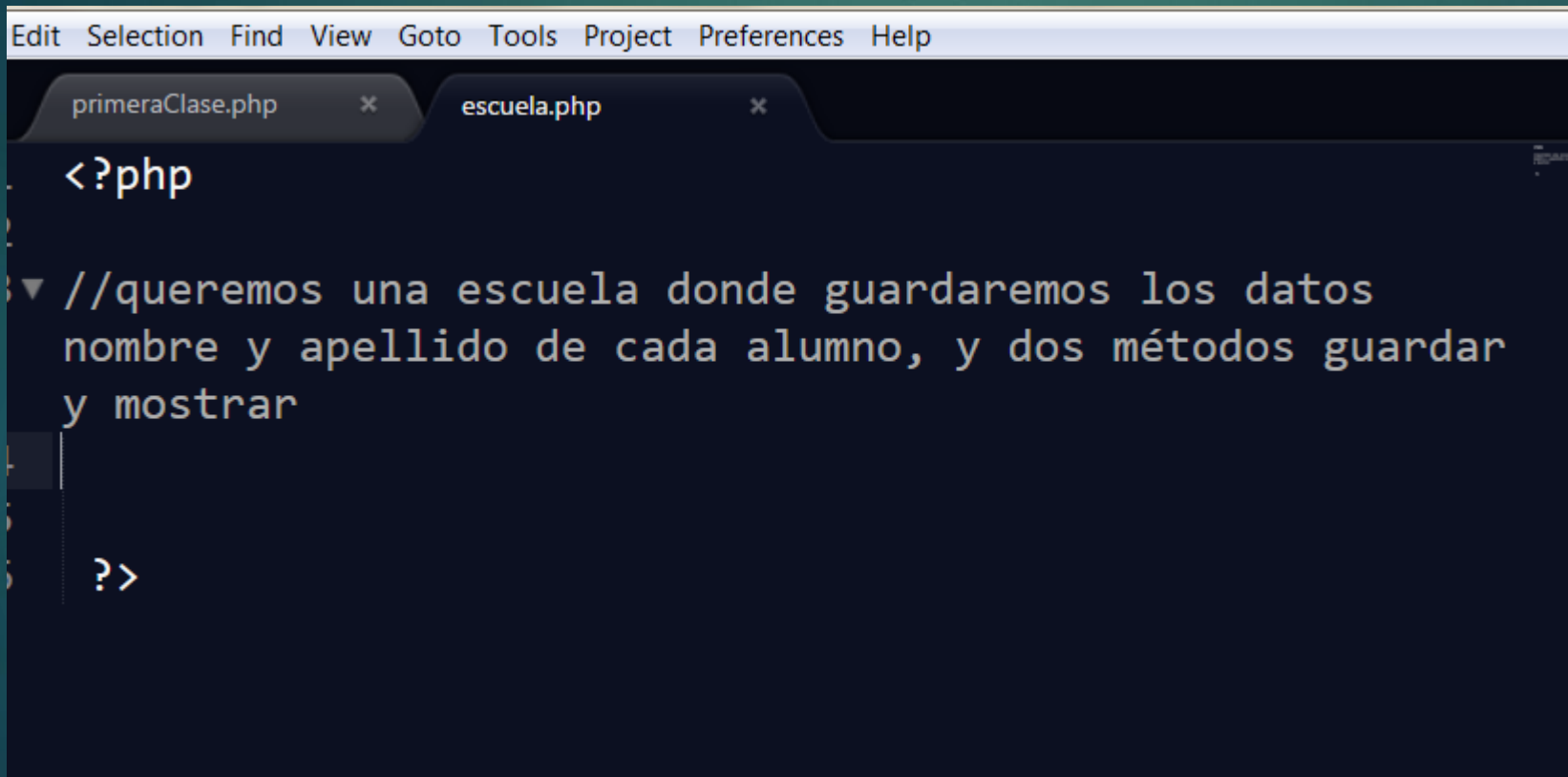
```
<?php

//Crear la clase
class Persona {
    //atributos
    public $nombre="Pedro";
    //métodos
    public function hablar($mensaje){
        echo $mensaje;
    }
}

//instancia de la clase
$persona=new Persona();
echo $persona->nombre; //propiedad o atributo
$persona->hablar("hola"); //método

?>
```

Ejercicio : escuela.php



```

Edit Selection Find View Goto Tools Project Preferences Help
primeraClase.php x escuela.php x
<?php
//queremos una escuela donde guardaremos los datos
nombre y apellido de cada alumno, y dos métodos guardar
y mostrar
?>
```

<?php

```
//queremos una escuela donde guardaremos los datos
nombre y apellido de cada alumno, y dos métodos guardar
y mostrar
```

```
class Alumno {
    public $nombre;
    public $apellido;

    public function guardar($nombre,$apellido){
        $this->nombre=$nombre;
        $this->apellido=$apellido;
    }


    public function mostrar(){
        echo"Su nombre es:". $this->nombre."<br/>";
        echo"Su apellido es:". $this->apellido;
    }
}
```

?>

escuela1.php

crearAlumnoEscuela1.php

```
1 <?php
2
3 include("escuela1.php");
4
5 $alumno1=new Alumno();
6 $alumno1->nombre="Sonia";
7 $alumno1->apellido="Sánchez";
8 $alumno1->mostrar();
9
10 ?>
```


- 
- Crear las clases :
 - La Clase Rectangulo (claseRectangulo.php)
 - la Clase Circulo
 - la clase Triangulo

Con un constructor
sus propiedades y métodos calcularArea y calcularPerímetro

Creación de un objeto (de una clase):

PHP nos permite crear nuestros propios objetos con sus propias propiedades y sus métodos.

Crear un nuevo objeto consiste en declarar una función (un **constructor**).

El **constructor** es el encargado de dar un estado inicial al objeto.

El **constructor** es un método que ha de ser público y no puede devolver nada.


<?php

```
class Rectangulo{
//definición de propiedades
public $lado1;
public $lado2;
//definición del constructor
public function __construct($l1,$l2){
    $this->lado1=$l1;
    $this->lado2=$l2;
}
public function areaRectangulo(){
    return $this->lado1*$this->lado2;
    //cálculo del área de ese objeto que llama
}
public function perimetroRectangulo(){
    //cálculo del perímetro de ese objeto que llama
    return 2*($this->lado1+$this->lado2);
}
}
```

?>

Ejemplo de constructor

claseRectangulo1.php

- 
- Crear las clases con su constructor :
 - La Clase Rectangulo
 - la Clase Círculo
 - la clase Triangulo

```
class Circulo{

    public $radio; //private tb
    public function __construct($r){
        $this->radio=$r;
    }
    public function calcularArea(){
        return pi()*$this->radio*$this->radio;
    }
}

$circulo1=new Circulo(5);
echo "El area es:".$circulo1->calcularArea();
?>
```

claseCirculo.php



La sobrecarga de métodos

Es la creación de varios métodos con el mismo nombre pero que se diferencian en algún aspecto.

PHP sufrimos la carencia de sobrecarga debido a que es un lenguaje poco estricto.

Ejemplo (en java)

/* Métodos sobrecargados */

```
public int calculaSuma(int x, int y, int z){...}
```

```
public int calculaSuma(double x, double y, double z){...}
```

```
public int calculaSuma(int x, int y){...}
```

/* Error de compilación: estos métodos no están sobrecargados */

```
public int calculaSuma(int x, int y, int z){...}
```

```
public double calculaSuma(int x, int y, int z){ ...}
```


Sobrecarga de constructores

Los constructores dan un estado inicial a los objetos.

En general en los POO se puede elegir el estado inicial de los objetos definiendo diferentes constructores. En PHP no.

Tal vez no conoczcamos todos los valores iniciales en el momento de crear la instancia de una clase.

```
Persona p1 = new Persona();  
Persona p2 = new Persona("Alex");  
Persona p3 = new Persona(20);  
Persona p4 = new Persona("Alex", 20);
```

Constructor por defecto, sin parámetros



Sobrecarga de constructores.

En este caso tenemos tres constructores.


Como mínimo uno.
(java)

```
public Persona() {}  
public Persona(String nombre) {  
    this.nombre = nombre;  
}  
public Persona(int edad) {  
    this.edad = edad;  
}  
  
public Persona(String nombre, int edad) {  
    this.nombre = nombre;  
    this.edad = edad;  
}
```

Nos podemos encontrar un constructor que llame a otro constructor.

En el caso en el que unos parámetros ya vengan definidos desde el inicio por defecto (java)

```
public Persona() {}  
public Persona(String nombre) {  
    this(nombre, 18);  
}  
public Persona(int edad) {  
    this.edad = edad;  
}  
  
public Persona(String nombre, int edad) {  
    this.nombre = nombre;  
    this.edad = edad;  
}
```



```
Persona p3 = new Persona("Sara");
```

```
class Rectangulo{
    public $l1;
    public $l2;
    public function __construct($l){
        $this->l1=$l;
        $this->l2=$l;
    }
    public function __construct($lado1,$lado2){
        $this->l1=$lado1;
        $this->l2=$lado2;
    }
    /*public function calcularArea(){
        return pi()*$this->radio*$this->radio;
    }*/
}

$rect1=new Rectangulo(5);
echo "El lado".$rect->$l2;
?>
```

claseRectangulo2.php

Fatal error: Cannot redeclare Rectangulo::__construct() in C:\xampp\htdocs\clases\claseRectangulo.php on line 11

Método destructor:

Se ejecuta también de manera automática, pero al final de la clase. Ha de ser público y no retornar nada.

Ejemplo de constructor(con parámetros) y destructor

Realizar una aplicación tipo lotería.

Dónde le indicaremos un número aleatorio y la cantidad de intentos.

Mostraremos los intentos y al final mostrará el mensaje de si acertó o no.

loteria.php

```
class Loteria{
    //atributos
    public $numero;
    public $intentos;
    public $resultado=false;
    //métodos
    public function __construct($numero,$intentos){
        $this->numero=$numero;
        $this->intentos=$intentos;
    }
    public function sortear(){
        $min=$this->numero / 2;
        $max=$this->numero * 2;

        for($i=0;$i<$this->intentos;$i++)
        {
            $int=rand($min,$max);
            self::intentos($int);// probar con $this->
        }
    }
    public function intentos($int){
        if($int==$this->numero)
        {
            echo "<b>".$int."==" canton . $this->numero."</b><br/>" ;
            $this->resultado=true;
        }
        else { echo $int."diferente de " canton . $this->numero."<br/>"; }
    }
    public function __destruct(){
        if($this->resultado) { echo "Ganador!!:" canton . $this->intentos; }
        else{echo "Perdió...: a los..." canton . $this->intentos ;}
    }
}
```

```
$loteria=new Loteria(10,10);
$loteria->sortear();
```


Exercici:

Queremos una clase donde guardaremos los datos nombre y mail de cada alumno, y un array de cursos en los que se ha apuntado

El constructor inicializa el nombre y el mail

Definimos un método setCurso que añade al array de cursos el curso al que se quiere inscribir. Apuntarlo a tres cursos.

Definimos un método mostrar() que al llamarlo desde el objeto me muestra el nombre, el mail y todos los cursos a los que se ha apuntado.

```

class Persona {
    public $nombre;
    public $mail;
    public $curso=array();
    public function __construct($nombre,$mail) {
        $this->nombre=$nombre;
        $this->mail=$mail;
    }
    public function setCurso($c) {
        $this->curso[]=$c;
    }
    public function mostrar() {
        echo "El nombre:". $this->nombre. "<br/>";
        echo "El mail:". $this->mail. "<br/>";
        echo "Cursos apuntados:<br/>";

        for($i=0;$i<count($this->curso);$i++)
        {
            echo $this->curso[$i]. "<br/>";
        }
    }
}

$persona=new Persona("Luis","fernandez");
$persona->setCurso("php");
$persona->setCurso("js");
$persona->mostrar();

```

?>

El nombre:Luis
El mail:fernandez
Cursos apuntados:
php
js

registroCursos.php



Crear una clase persona que tenga un nombre,apellido y un correo.

Crear una clase estudiante que tenga un nombre,apellido,un correo,
un código y un nombre del la facultad.

Definir un constructor para cada clase, que inicialice todas las propiedades.

Estudiante.php

Persona.php

```
1 <?php
2
3 class Persona{
4
5     public $nombre;
6     public $apellido;
7     public $correo;
8
9     public function __construct($n,$a,$c){
10         $this->nombre=$n;
11         $this->apellido=$a;
12         $this->correo=$c;
13
14     }
15     //método
16     public function verInfo(){
17         echo "nombre:".$this->nombre."<br/>";
18         echo "apellido:".$this->apellido."<br/>";
19         echo "correo:".$this->correo."<br/>";
20     }
21 }
22
23
24 ?>
```

Persona.php

<?php

```
class Estudiante{

    public $nombre;
    public $apellido;
    public $correo;
    public $codigo;
    public $facultad;

    public function __construct($n,$a,$c,$co,$f){
        $this->nombre=$n;
        $this->apellido=$a;
        $this->correo=$c;
        $this->codigo=$co;
        $this->facultad=$f;
    }

    public function verInfo(){
        echo "nombre:". $this->nombre. "<br/>";
        echo "apellido". $this->apellido. "<br/>";
        echo "correo". $this->correo. "<br/>";
        echo "codigo". $this->codigo. "<br/>";
        echo "facultad". $this->facultad. "<br/>";
    }

}
```

Estudiante.php