



Ajuntament  
de Barcelona

# Desenvolupament web amb PHP

## Conceptes clau de POO (I)

**IT Academy**

Desembre de 2020



# POO: Operador “this”

L'operador “this” es una variable especial que **només té sentit dins del context d'un objecte instanciat**. Representa al mateix objecte, i serveix, per tant, per referir-se als mètodes i atributs d'ell mateix.

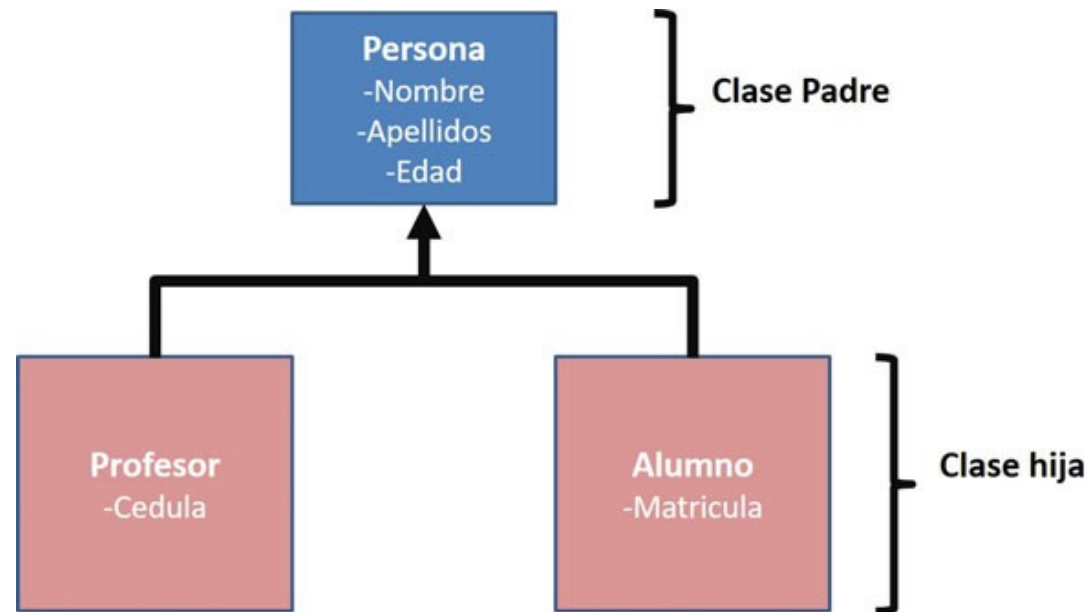
```
8  <?php
9  //Clase Padre
10 class Operacion {
11     //Atributos de clase
12     protected $valor1;
13     protected $valor2;
14     protected $resultado;
15     //Metodos de clase
16     public function cargar1($v)
17     {
18         $this->valor1=$v;
19     }
20     public function cargar2($v)
21     {
22         $this->valor2=$v;
23     }
24     public function imprimirResultado()
25     {
26         echo $this->resultado.'<br>';
27     }
28 }
```

Aquí, per exemple, **\$this** → **valor1** refereix a l'atribut de l'objecte, i li assigna el valor de la variable **\$v**. Si la variable que entra pel paràmetre tingués el mateix nom (\$valor1) no hi hauria conflicte sempre i quan ens referíssim a l'atribut de l'objecte amb \$this.



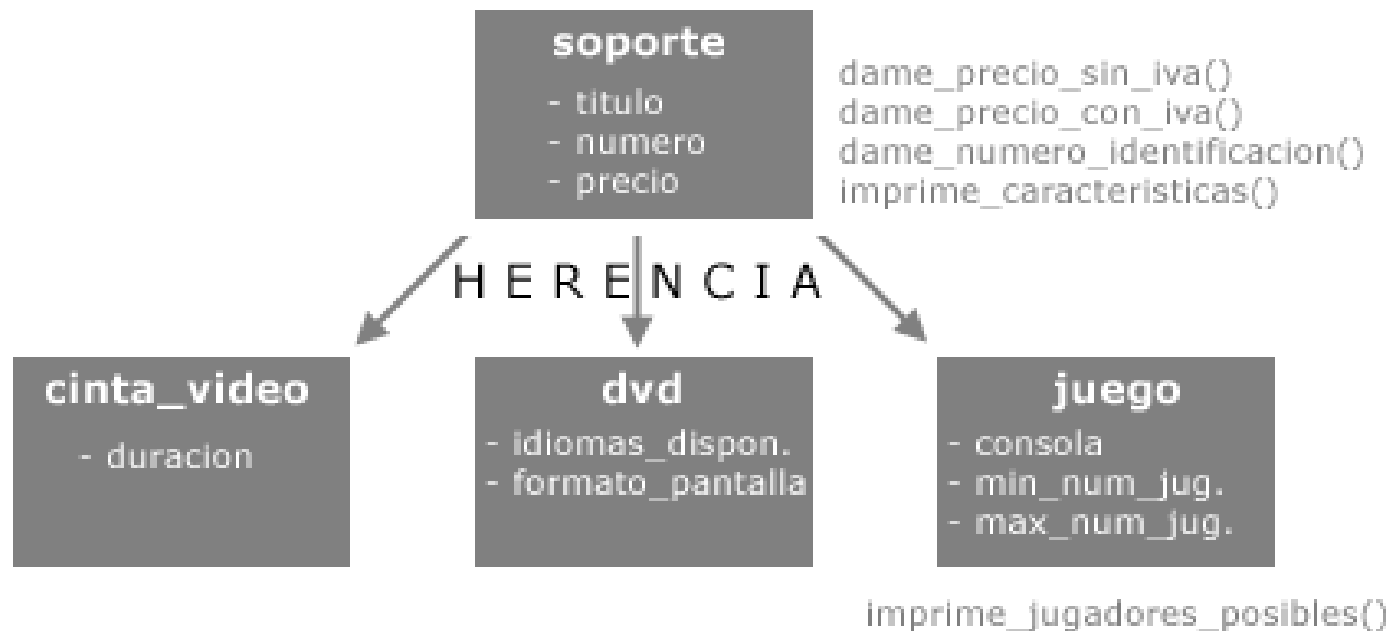
# POO: Herència

L'**herència** és un mecanisme de POO mitjançant el qual podem crear **jerarquies** de classes.





# POO: Herència





# POO: Herència

```
8 <?php
9 //Clase Padre
10 class Operacion {
11     //Atributos de clase
12     protected $valor1;
13     protected $valor2;
14     protected $resultado;
15     //Metodos de clase
16     public function cargar1($v)
17     {
18         $this->valor1=$v;
19     }
20     public function cargar2($v)
21     {
22         $this->valor2=$v;
23     }
24     public function imprimirResultado()
25     {
26         echo $this->resultado.'<br>';
27     }
28 }
```

La keyword **extends** ens serveix per a indicar de quina classe heretarem

```
29 //Clase hija
30 class Suma extends Operacion{
31     //Metodos de clase
32     public function operar()
33     {
34         $this->resultado=$this->valor1+$this->valor2;
35     }
36 }
37 //Clase hija
38 class Resta extends Operacion{
39     //Metodos de clase
40     public function operar()
41     {
42         $this->resultado=$this->valor1-$this->valor2;
43     }
44 }
```

La principal “gràcia” d’aquest mecanisme es que s’hereten els mètodes i atributs depenent de la seva **visibilitat**(terme del qual parlem més endavant). En aquest cas, les classes Suma i Resta hereten \$valor1,\$valor2,\$valor3 i els mètodes cargar1,cargar2 e imprimirResultado, afegint-ho a les seves pròpies característiques.



# POO: Herència

```
45 //Creacion e inicializacion de objetos
46 $suma=new Suma();
47 $suma->cargar1(10);
48 $suma->cargar2(10);
49 $suma->operar();
50 echo 'El resultado de la suma de 10+10 es: ';
51 $suma->imprimirResultado();
52 $resta=new Resta();
53 $resta->cargar1(10);
54 $resta->cargar2(5);
55 $resta->operar();
56 echo 'El resultado de la diferencia de 10-5 es: ';
57 $resta->imprimirResultado();
58
59 ?>
```



# POO: Polimorfisme

La **visibilitat** és un mecanisme de POO que ens permet **limitar l'accés de mètodes o atributs d'una classe**.

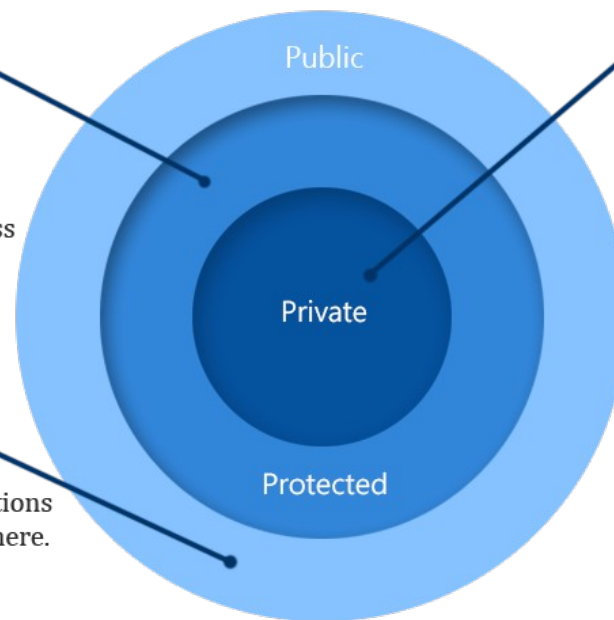
Les *keywords* **protected**, **private** o **public** davant d'un atribut o mètode d'una classe signifiquen, doncs:

## Protected

Protected Properties or functions of this class can only be accessed by the class that inherits this class as its base class.

## Public

Public properties or functions can be called from any where. They are accessible to all.



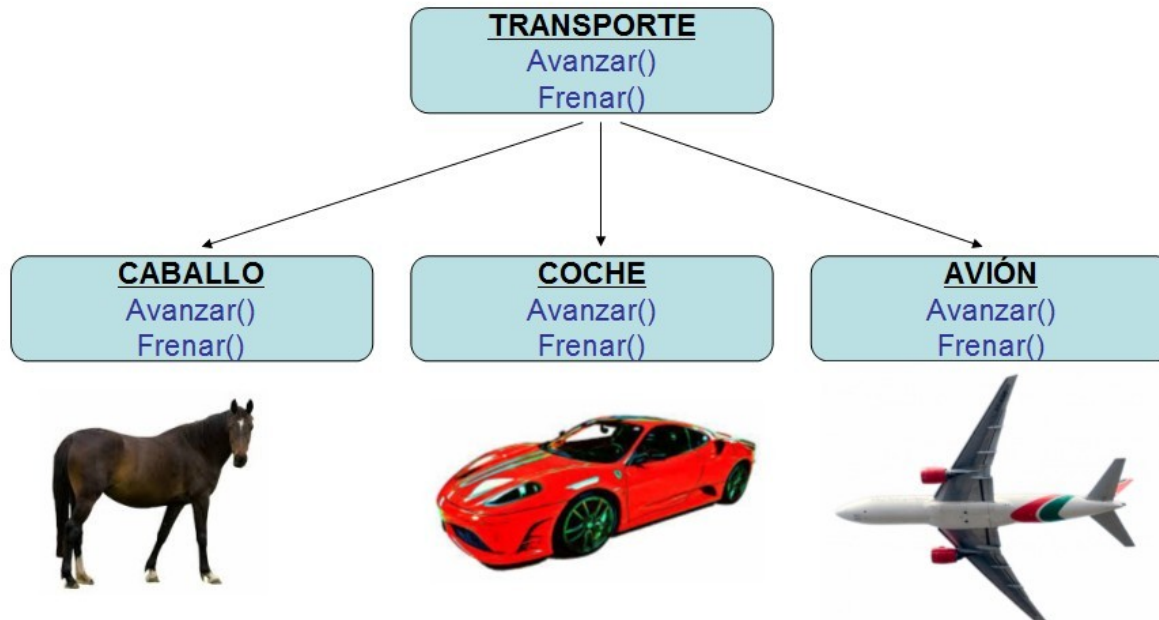
## Private

Private properties or functions of a class can only be used inside of the class that defines it. It cannot be called even by the instance of the same class and also the inherited class.

Copyright Dreamsoft 2013 [www.dreamsoftworks.blogspot.com](http://www.dreamsoftworks.blogspot.com)

# POO: Polimorfisme

El **polimorfisme** és un mecanisme de POO mitjançant el qual podem donar varies formes a un mateix objecte, permetent, entre d'altres coses, respondre de manera diferent a crides a funcions del mateix nom. El polimorfisme té sentit gràcies a l'**herència** en POO.





# POO: Polimorfisme i sobreescriptura de mètodes

Recuperem l'exemple anterior on la classe "pare" Operacion té com a "filles" les classes Suma y resta

```
117 <?php
118 //Clase Padre
119 class Operacion {
120     //Atributos de clase
121     protected $valor1;
122     protected $valor2;
123     protected $resultado;
124     //Metodos de clase
125     public function cargar1($v)
126     {
127         $this->valor1=$v;
128     }
129     public function cargar2($v)
130     {
131         $this->valor2=$v;
132     }
133     public function imprimirResultado()
134     {
135         echo $this->resultado.'<br>';
136     }
137 }
```

```
138 //Clase hija
139 class Suma extends Operacion{
140     //Metodos de clase
141     public function operar()
142     {
143         $this->resultado=$this->valor1+$this->valor2;
144     }
145     public function imprimirResultado()
146     {
147         echo "La suma de $this->valor1 y $this->valor2 es:";
148         parent::imprimirResultado();
149     }
150 }
```

```
151 //Clase hija
152 class Resta extends Operacion{
153     //Metodos de clase
154     public function operar()
155     {
156         $this->resultado=$this->valor1-$this->valor2;
157     }
158     public function imprimirResultado()
159     {
160         echo "La diferencia de $this->valor1 y $this->valor2 es:";
161         parent::imprimirResultado();
162     }
163 }
```



# POO: Polimorfisme i sobreescritura de mètodes

```
164 //Creacion e inicializacion de objetos
165 $suma=new Suma();
166 $suma->cargar1(10);
167 $suma->cargar2(10);
168 $suma->operar();
169 $suma->imprimirResultado();
170 $resta=new Resta();
171 $resta->cargar1(10);
172 $resta->cargar2(5);
173 $resta->operar();
174 $resta->imprimirResultado();
175
176 ?>
```

“La suma de \$this → valor1 y \$this → valor2 es: ”

“La diferencia de \$this → valor1 y \$this → valor2 es: ”

Gràcies a la **sobreescritura de mètodes** que ens permet el **polimorfisme**, el mètode **imprimirResultado** es comportarà de manera diferent **dependent del tipus d'objecte**(suma o resta en aquest cas) que l'executi



# POO: Polimorfisme i sobreescritura de mètodes(constructor)

També es poden sobre escriure mètodes per defecte de les classes com en **constructor**

```
187 //Clase Padre
188 class Operacion {
189     //Atributos de clase
190     protected $valor1;
191     protected $valor2;
192     protected $resultado;
193     //Constructor de clase
194     public function __construct($v1,$v2)
195     {
196         $this->valor1=$v1;
197         $this->valor2=$v2;
198     }
199     //Metodos de clase
200     public function imprimirResultado()
201     {
202         echo $this->resultado.'<br>';
203     }
204 }
```

```
205 //Clase hija
206 class Suma extends Operacion{
207     //Atributos de clase
208     protected $titulo;
209     //Constructor de clase
210     public function __construct($v1,$v2,$tit)
211     {
212         Operacion::__construct($v1,$v2);
213         $this->titulo=$tit;
214     }
215     //Metodos de clase
216     public function operar()
217     {
218         echo $this->titulo;
219         echo $this->valor1.'+'. $this->valor2.' es ';
220         $this->resultado=$this->valor1+$this->valor2;
221     }
222 }
```



# POO: Polimorfisme i sobreescriptura de mètodes(constructor)

---

```
223 //Creacion e inicializacion de objetos
224 $suma=new Suma(10,10,'Suma de valores:');
225 $suma->operar();
226 $suma->imprimirResultado();
227
228 ?>
229 </body>
230 </html>
```



Barcelona  
**Activa**



[barcelona.cat/barcelonactiva](http://barcelona.cat/barcelonactiva)