Ajuntament de Barcelona

Desenvolupament web amb PHP
Conceptes clau de POO (I)

IT Academy

Desembre de 2020

Barcelona Activa

Unió Europea
Fons Europeu de Desenvolupament Regional

POO: Operador “this”

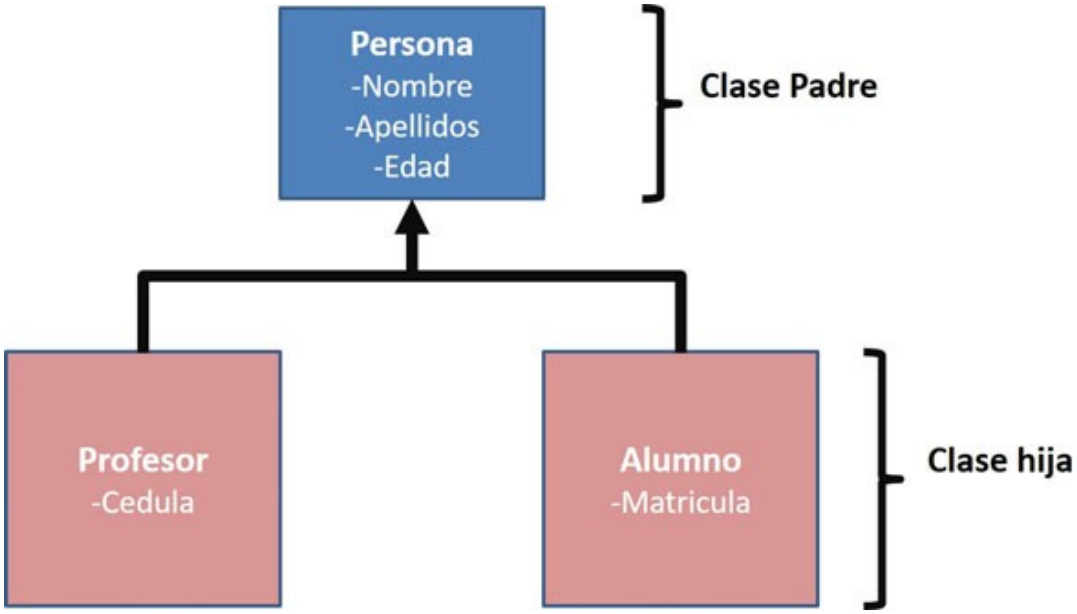
L’operador “this” es una variable especial que **només té sentit dins del context d’un objecte instanciat**. Representa al mateix objecte, i serveix, per tant, per referirse als mètodes i atributs d’ell mateix.

```php
<?php
//Clase Padre
class Operacion {
  //Atributos de clase
  protected $valor1;
  protected $valor2;
  protected $resultado;
  //Metodos de clase
  public function cargar1($v)
  {
    $this->valor1=$v;
  }
  public function cargar2($v)
  {
    $this->valor2=$v;
  }
  public function imprimirResultado()
  {
    echo $this->resultado.'<br>';
  }
}
```

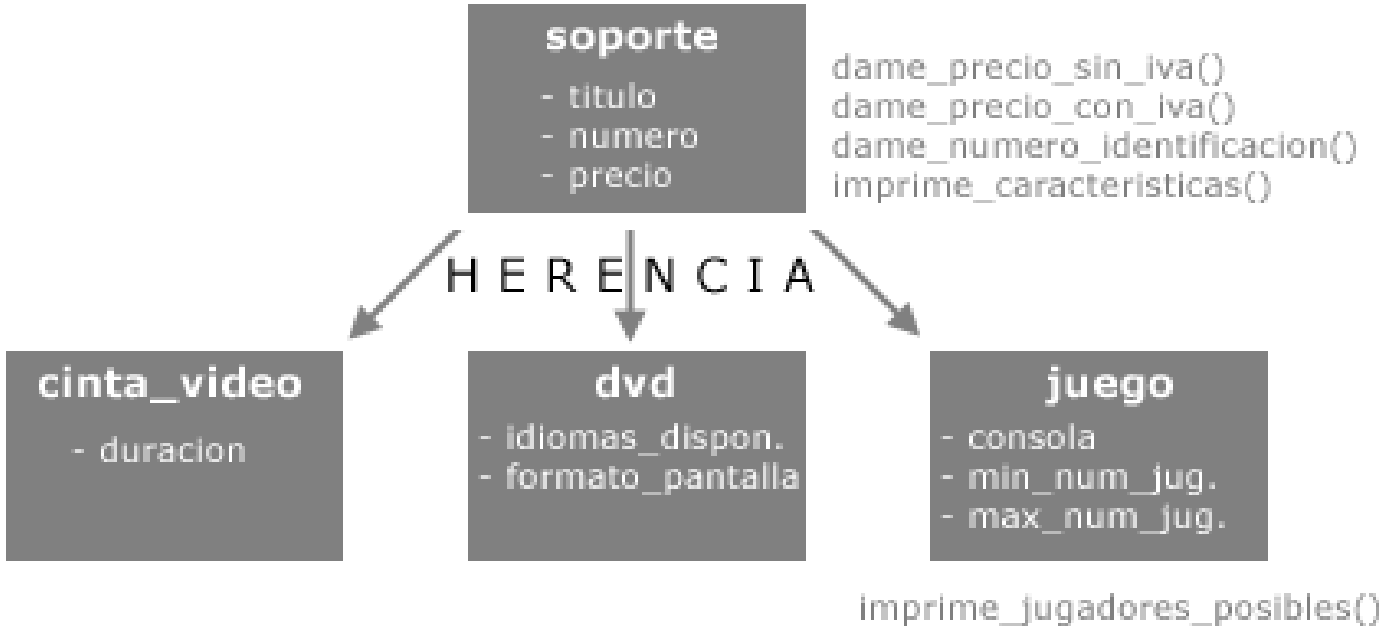
Aquí, per exemple, **$this→valor1** refereix a l’atribut de l’objecte, i li assigna el valor de la variable **$v.** Si la variable que entra pel paràmetre tingués el mateix nom($valor1) no hi hauria conflicte sempre i quan ens referíssim a l’atribut de l’objecte amb $this.

POO: Herència

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



POO: Herència



POO: Herència

```php
<?php
//Clase Padre
class Operacion {
  //Atributos de clase
  protected $valor1;
  protected $valor2;
  protected $resultado;
  //Metodos de clase
  public function cargar1($v)
  {
    $this->valor1=$v;
  }
  public function cargar2($v)
  {
    $this->valor2=$v;
  }
  public function imprimirResultado()
  {
    echo $this->resultado.'<br>';
  }
}
```

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

```php
//Clase hija
class Suma extends Operacion{
    //Metodos de clase
    public function operar()
      {
        $this->resultado=$this->valor1+$this->valor2;
      }
}
//Clase hija
class Resta extends Operacion{
    //Metodos de clase
    public function operar()
      {
        $this->resultado=$this->valor1-$this->valor2;
      }
}
```

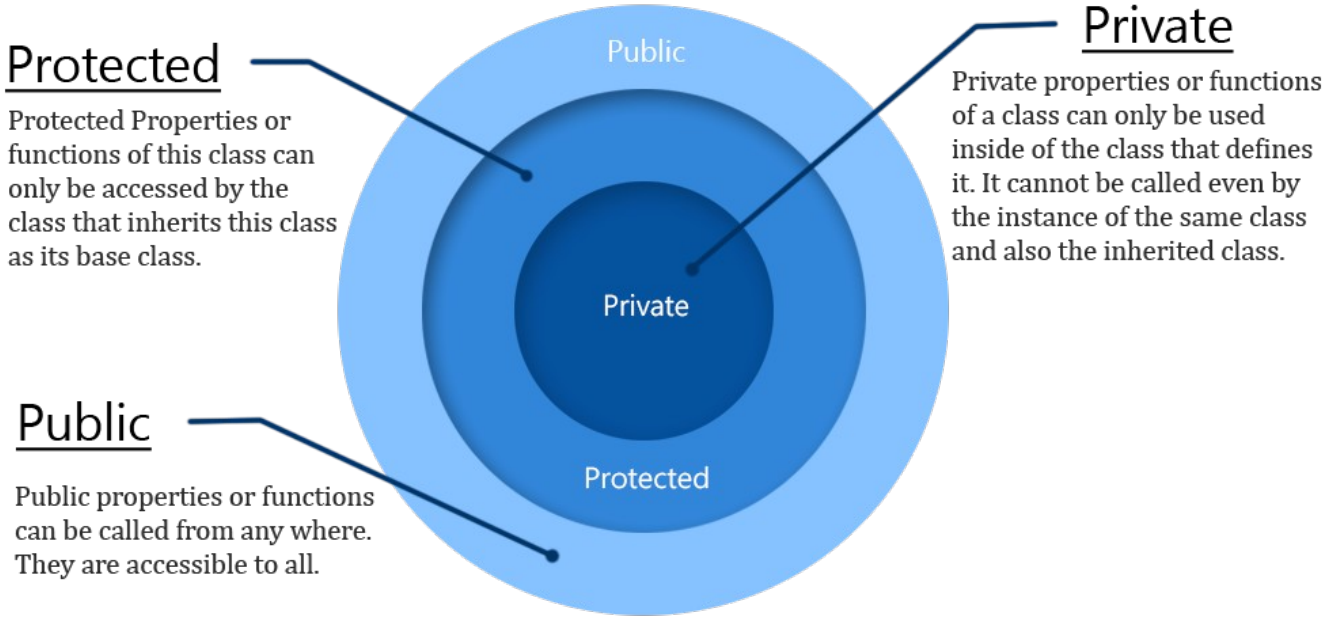
La principal “gràcia” d’aquest mecanisme es que s’hereten els mètodes i atributs depenent de la seva **visibilitat(**terme del qual parlem més endavant). En aquest cas, les classes Suma i Resta hereten $valor1,$valor2,$valor3 i els mètodes cargar1,cargar2 e imprimirResultado, afegint-ho a les seves pròpies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

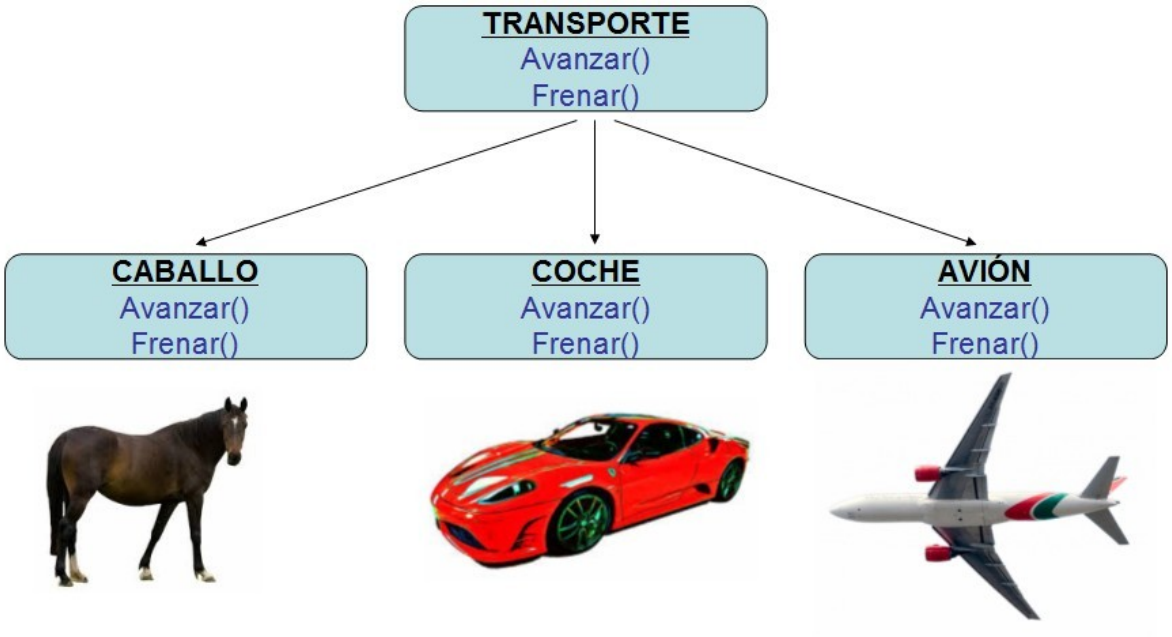
POO: Polimorfisme

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:



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

```php
<?php
//Clase Padre
class Operacion {
  //Atributos de clase
  protected $valor1;
  protected $valor2;
  protected $resultado;
  //Metodos de clase
  public function cargar1($v)
  {
    $this->valor1=$v;
  }
  public function cargar2($v)
  {
    $this->valor2=$v;
  }
  public function imprimirResultado()
  {
    echo $this->resultado.'<br>';
  }
}
```

```php
//Clase hija
class Suma extends Operacion{
  //Metodos de clase
  public function operar()
  {
    $this->resultado=$this->valor1+$this->valor2;
  }
  public function imprimirResultado()
  {
    echo "La suma de $this->valor1 y $this->valor2 es:";
    parent::imprimirResultado();
  }
}
```

```php
//Clase hija
class Resta extends Operacion{
  //Metodos de clase
  public function operar()
  {
    $this->resultado=$this->valor1-$this->valor2;
  }
  public function imprimirResultado()
  {
    echo "La diferencia de $this->valor1 y $this->valor2 es:";
    parent::imprimirResultado();
  }
}
```

POO: Polimorfisme i sobreescriptura 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 **sobreescriptura de mètodes** que ens permet el **polimorfisme**, el mètode **imprimirResultado** es comportarà de manera diferent **depenent del tipus d'objecte**(suma o resta en aquest cas) que l'executi

POO: Polimorfisme i sobreescriptura de mètodes(constructor)

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

```php
//Clase Padre
class Operacion {
  //Atributos de clase
  protected $valor1;
  protected $valor2;
  protected $resultado;
  //Constructor de clase
  public function __construct($v1,$v2)
  {
    $this->valor1=$v1;
    $this->valor2=$v2;
  }
  //Metodos de clase
  public function imprimirResultado()
  {
    echo $this->resultado.'<br>';
  }
}
```

```php
//Clase hija
class Suma extends Operacion{
  //Atributos de clase
  protected $titulo;
  //Constructor de clase
  public function __construct($v1,$v2,$tit)
  {
    Operacion::__construct($v1,$v2);
    $this->titulo=$tit;
  }
  //Metodos de clase
  public function operar()
  {
    echo $this->titulo;
    echo $this->valor1.'+'.$this->valor2.' es ';
    $this->resultado=$this->valor1+$this->valor2;
  }
}
```

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.cat/barcelonactiva