CONCEPTO DE VARIABLE Y MÉTODO STATIC

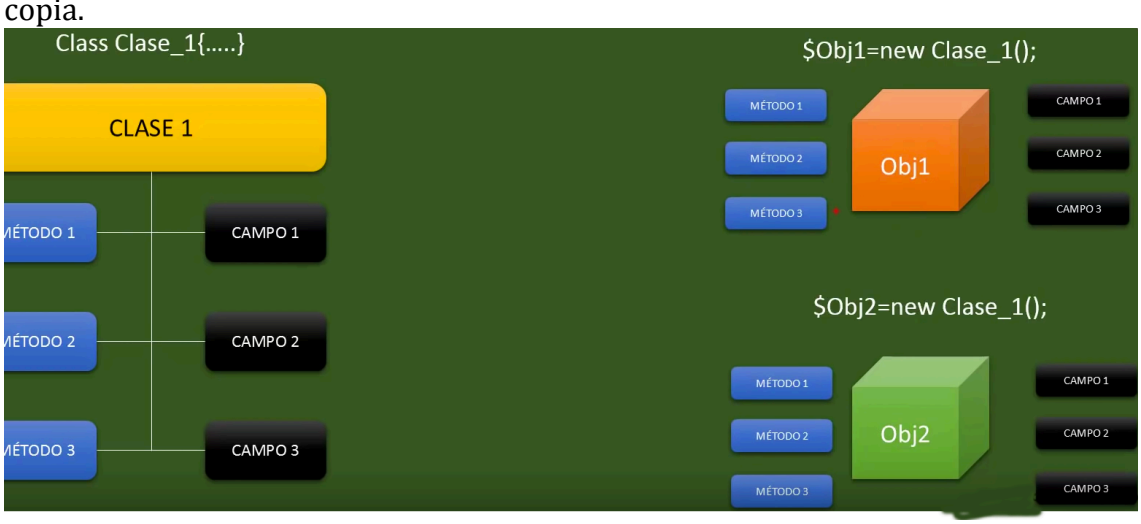
1º empezamos por ver :

- el método o función no estático
- variable o campo no estática que pertenezcan a una clase.

EJEMPLO:

1. Hemos definido en el documento php, clase 1
2. Class Classe_1{….} En ella definimos unos métodos y unas propiedades , y unas variables normales
3. Creamos instancias que pertenecen a esta clase 1 $OBJ1= NEW Clase_1(),$OBJ2= NEW Clase_1(),……Así crearíamos todas las instancias pertenecientes a la clase 1

Cuando creamos una instancia las propiedades y métodos de esa clase crea una copia.



Por ejemplo los métodos número 2 son independientes en las dos instancias, por lo que podrán comportarse de forma diferente por cada uno de ellos.

Cuando la variable es normal (es decir no es estática), se podrán comportar cada uno de los métodos numero 2, de forma diferentes.

Ejemplo
tenemos un archivo php llamado concesionario.

```php
<?php
    class Compra_vehiculo{
        private $precio_base;

        function Compra_vehiculo($gama){
            if($gama=="urbano"){
                    $this->precio_base=10000;
            }else if($gama=="compacto"){

                    $this->precio_base=20000;
            }
            else if($gama=="berlina"){
                    $this->precio_base=30000;
            }

        }// fin constructor
```

el fin, naturalmente es vender coches a clientes
Creamos un class Compre_vehiculo, con un constructor . Esta class dependiendo de la gama del automóvil, de compra, va a crear un precio (urbano, compacto o berlina) . Se podrá utilizar uno u otro
Además del constructor tenemos un conjunto de funciones que van agregando extras
Cada extra, se suma al precio base.

```php
        }// fin constructor


        function climatizador(){

                $this->precio_base+=2000;

        }// fin climatizador

        function navegador_gps(){
            $this->precio_base+=2500;
        }//fin navegador gps


        function tapiceria_cuero($color){
```

```
}//fin navegador gps


function tapiceria_cuero($color){
    if($color=="blanco"){
        $this->precio_base+=3000;
    }
    else if($color=="beige"){
        $this->precio_base+=3500;
    }
    else{
        $this->precio_base+=5000;
    }
}// fin tapicería
```

```
            if($color== blanco ){
                $this->precio_base+=3000;
            }
            else if($color=="beige"){
                $this->precio_base+=3500;
            }
            else{
                $this->precio_base+=5000;
            }
        }// fin tapicería

        function precio_final(){
            return $this->precio_base;
        }// fin precio final

    }// fin clase

?>
```

Función precio final: Calcula el precio final del vehículo en función de los extras

Foto

Además tenemos el archivo de código fuente con el inlude del archivo
Imaginemos que entra un cliente que quiere comprar un vehiculo y le damos la función precio total

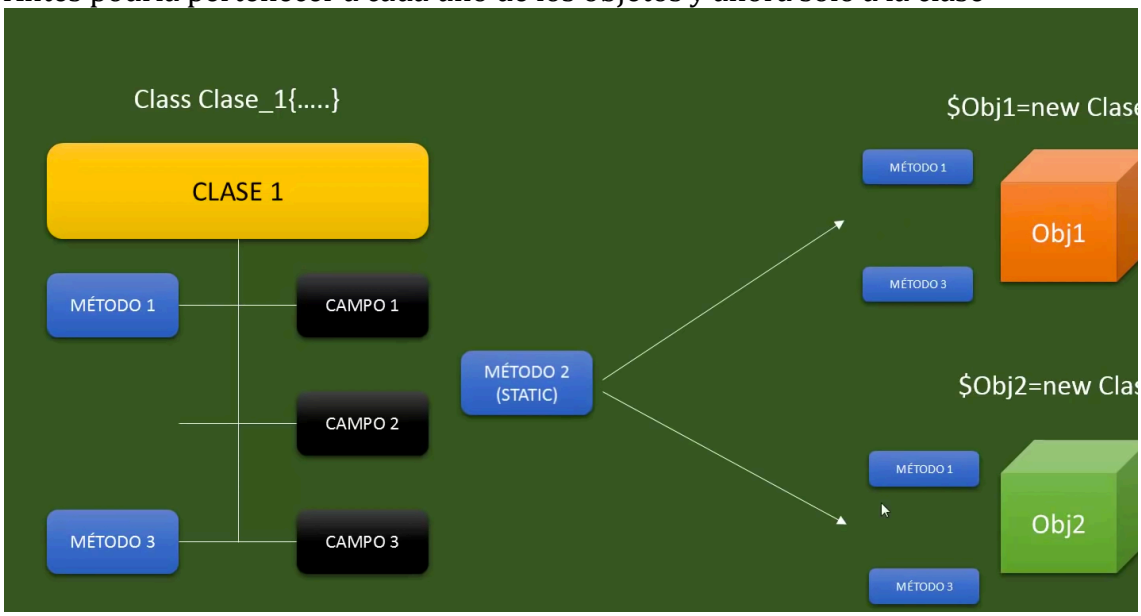
```php
<?php

    include("Concesionario.php");


    $compra_Antonio=new Compra_vehiculo("compacto");

    $compra_Antonio->climatizador();

    $compra_Antonio->tapiceria_cuero("blanco");

    echo $compra_Antonio->precio_final();

    $compra_Ana=new Compra_vehiculo("compacto");

    $compra_Ana->climatizador();

    $compra_Ana->tapiceria_cuero("rojo");

    echo $compra_Ana->precio_final();
```

Con esto me tiene que devolver 20000€.
Si antes de precio final quiere un extra climatizador, que incremente el precio base en 2000, devolverá 220000
Si además incremente la tapicería en blanco, mas
Luego entra un segundo cliente

Cada una de esas instancias va a tener su propia copia de los métodos.

Comprobemos como tapicería de cuero se comporta de forma diferente en Antonio que en Ana. Solo cambia el color
Si hacemos una visualización saldrán diferentes
A pesar de haber elegido los mismos métodos de precio es diferente. Por qué?. Por que se comporta de forma diferente para el primero y el 2ª

Si un método lo definimos como estático, en ese caso al ser estático las instancias ya no tendrían la copia de ese método independiente
Pasaría a ser un método compartido por todos los elementos de una instancia.
Antes podría pertenecer a cada uno de los objetos y ahora solo a la clase



¿Esto que implica?
Pues imagina te que queremos hacer un descuento y ese descuento es 450€.
Ese descuento no lo hacemos nosotros como concesionario, sino el gobierno para fomentar la compra.
Creamos entonces un campo o variable estático .

```php
<?php

    class Compra_vehiculo{

        private $precio_base;

        static $ayuda=4500;


        function Compra_vehiculo($gama){

            if($gama=="urbano"){

                    $this->precio_base=10000;

            }else if($gama=="compacto"){
```

Ahora en precio final debemos cambiarlo, porque debemos restar el descuento.
Creamos la variable $valor final, para hacer referencia a un campo o variable estática, utilizamos self::

Self::$ayuda

```
function precio_final(){

    $valor_final=$this->precio_base-self::$ayuda;

    return $valor_final;

}// fin precio final
```

Cuando creamos un método como estático, quiere decir que pertenece solo a la clase donde se ha creado
Ninguna de las instancias de dicha clase o clases heredadas, podrá tener copia del método 2.
Ahora en lugar de ser métodos independientes, van a ser compartidos. Es el mismo. Antes podría pertenecer a cada uno de los objetos ahora solo puede pertenecer a la clase

EJEMPLO DESCUENTO
Creamos un campo o variable astática y cambiamos el precio final . Ojo el campo estático ya no pertenece a ningún objeto sino que pertenece a la clase.
Por lo que para hacer referencia al campo estático $ayuda, ya no podemos con el $this (este operador hacer referencia al objeto que estamos crando en cada momento (compra Antonios/Compra Ana).
como , al ser estático solo pertenece a la clase, y no pertenece a ningún objeto, no podemos usar el this ahora, sino el operador SELF

Para hacer referencia a un campo o variable estática utilizamos self:: (habría que quitar el this del return.
Tiene que devolver 20.500 /25,500

PROBLEMA
Al no ser un campo privado (encapsulado), podemos s modificarlo desde fuera de la clase, dando como consecuencia que se pueda manipular el precio de la ayuda .

Por otra parte si queremos hacer referencia a un campo estático desde fura de la clases con
Compra_vehiculo::$ayuda

```php
include("Concesionario.php");

Compra_vehiculo::$ayuda=10000;


$compra_Antonio=new Compra_vehiculo("compacto");

$compra_Antonio->climatizador();

$compra_Antonio->tapiceria_cuero("blanco");

echo $compra_Antonio->precio_final() . "<br>";

$compra_Ana=new Compra_vehiculo("compacto");

$compra_Ana->climatizador();

$compra_Ana->tapiceria_cuero("rojo");

echo $compra_Ana->precio_final();
```

Al no ser privado puedo cambiar el valor. (automáticamente se resta a todos las cantidad de forma predeterminada.
Solución: Encapsular la variable estática ayuda, haciéndola además de estática privada (Nos asegura que solo sea accesible desde la compra vehículo. NO podrá manipularse desde fuera de la clase

```
class Compra_vehiculo{
    private $precio_base;
    private static $ayuda=0;

    function Compra_vehiculo($gama){
        if($gama=="urbano"){
                $this->precio_base=10000;
        }else if($gama=="compacto"){

                $this->precio_base=20000;
        }
        else if($gama=="berlina"){
                $this->precio_base=30000;
```

¿Por qué no aplicar el descuento con una variable normal?
Hay que tener en cuenta de que es un descuento obligatorio
Si no se hace estática tendríamos que hacer e l descuento de forma manual
Con static nos evitamos escribir 20 veces en caso de que fueran 20 compradores , la misma línea de código

Como decíamos para que no se manipule la ayuda hacemos la variable estática privada . De esta manera será accesible desde la propia clase utilizando el operador SELF pero no será accesible desde fuera de la clase

Si pusiéramos ahora la línea de código de antes para cambiar el valor, , nos daría error

Ingeniemos nos ahora que la ayuda del gobierno queremos aplicara a veces si y a veces no . Ej. Los tres primeros meses del año, el resto no.
creamos un método pera que cuando queramos aplicar la ayuda llamamos al método y cuando no no lo llamemos

1. Declaramos $ayuda y le damos el valor cero
2. Creamos método (en cualquier sitio º), un método estático (esto implica que pertenecerá a la clase y a no a ninguna instancia.
3. Lo creamos con la palabra static