



El empleo  
es de todos

Mintrabajo

# PROGRAMACIÓN ORIENTADA A OBJETOS

## Usando PHP 7.4



@SENAcomunica

[www.sena.edu.co](http://www.sena.edu.co)




# PARADIGMA ORIENTADO A OBJETOS

- La P.O.O es un **paradigma de programación** (o técnica de programación) que utiliza objetos e interacciones en el diseño de un sistema.
- El paradigma de programación OBJECT ORIENTED se utiliza en todos los lenguajes de programación modernos (Java, .Net, c#, Python, JS, PHP).
- PARADIGMA : El término **paradigma** se origina en la palabra griega [parádeigma] que en griego antiguo **significa** "modelo" o "ejemplo".



# VENTAJAS DE LA P.O.O.

- Facilidad de mantenimiento, modificación y escalado del código (clean code). 
- Los componentes se pueden reutilizar.
- Se puede obtener una estructura modular clara , la cual NO revelará el mecanismo detrás del diseño.
- Se acopla bien a la utilización de bases de datos, debido a la correspondencia entre las estructuras.
- El código es mas fácil de entender y documentar .
- Mas información : <https://desarrolloweb.com/articulos/beneficios-poo-resumen.html>.

# PILARES DE LA P.O.O.

# PILARES DE LA P.O.O.



ABSTRACCIÓN

HERENCIA

ENCAPSULAMIENTO

POLIMORFISMO

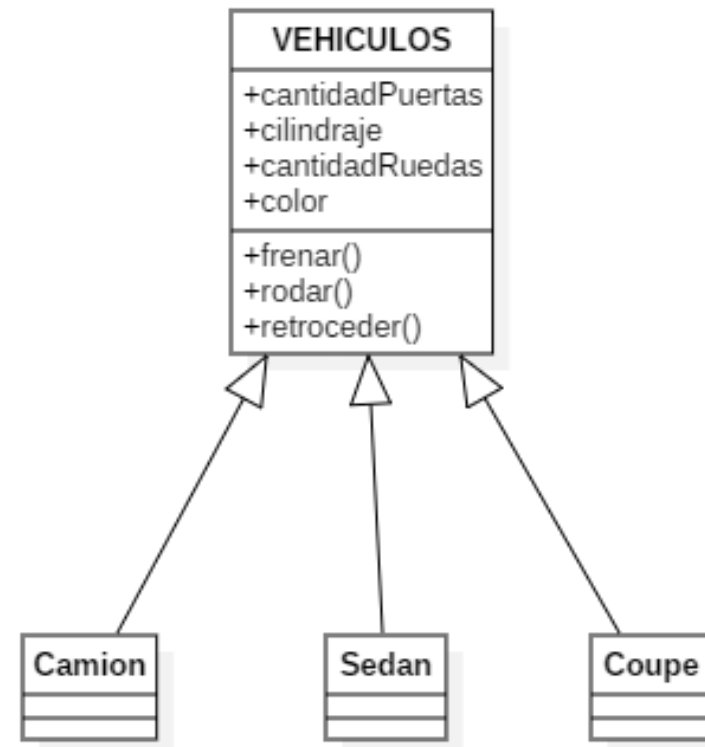


# ABSTRACCIÓN

- Es el pilar fundamental de la POO, que permite identificar y abstraer las características y comportamientos de un objeto real y con los cuales se construirá la clase base (ó plantilla).
- Pensar en términos de objetos es muy parecido a cómo lo haríamos en la vida real. Por ejemplo vamos a pensar en un Vehículo para tratar de modelarlo en un esquema de P.O.O.

# DIAGRAMAS DE CLASES - UML

- Para representar las clases y objetos de un sistema utilizando UML se modelan en el DIAGRAMA DE CLASES.



# PROCESO DE ABSTRACCIÓN.



Abstracción

| VEHICULOS  |
|--|
| +cantidadPuertas<br>+cilindraje<br>+cantidadRuedas<br>+color |
| +frenar()<br>+rodar()<br>+retroceder()                       |

La clase es  
como: una  
plantilla, plano,  
molde.

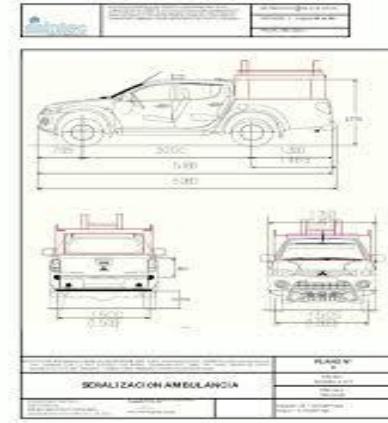




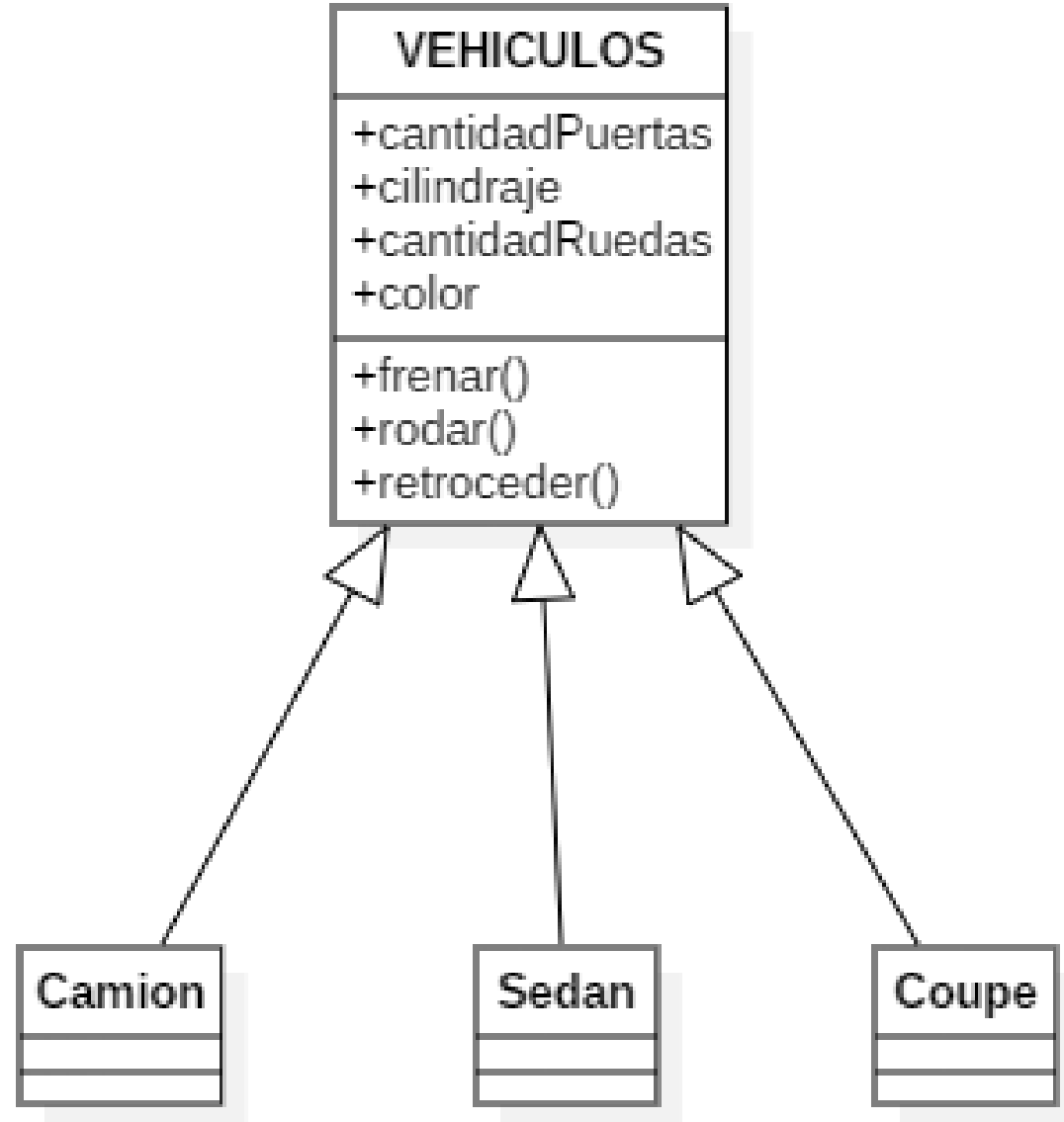
# HERENCIA

- Es el pilar más fuerte que asegura la **reutilización de código**, ya que a partir de esta característica es posible reutilizar (heredar) las características y comportamientos de una clase superior llamada clase padre o clase base, a sus clases hijas, denominadas clases derivadas.

# HERENCIA



# HERENCIA





# ENCAPSULAMIENTO

- Es la característica de la POO que permite el ocultamiento de la complejidad del código, pertenece a la parte privada de la clase y que no puede ser vista desde ningún otro programa.
- El encapsulamiento tiene que ver también con la visibilidad de los atributos y métodos.

# ENCAPSULAMIENTO

- Sabe usted como funciona internamente el T.V. ?

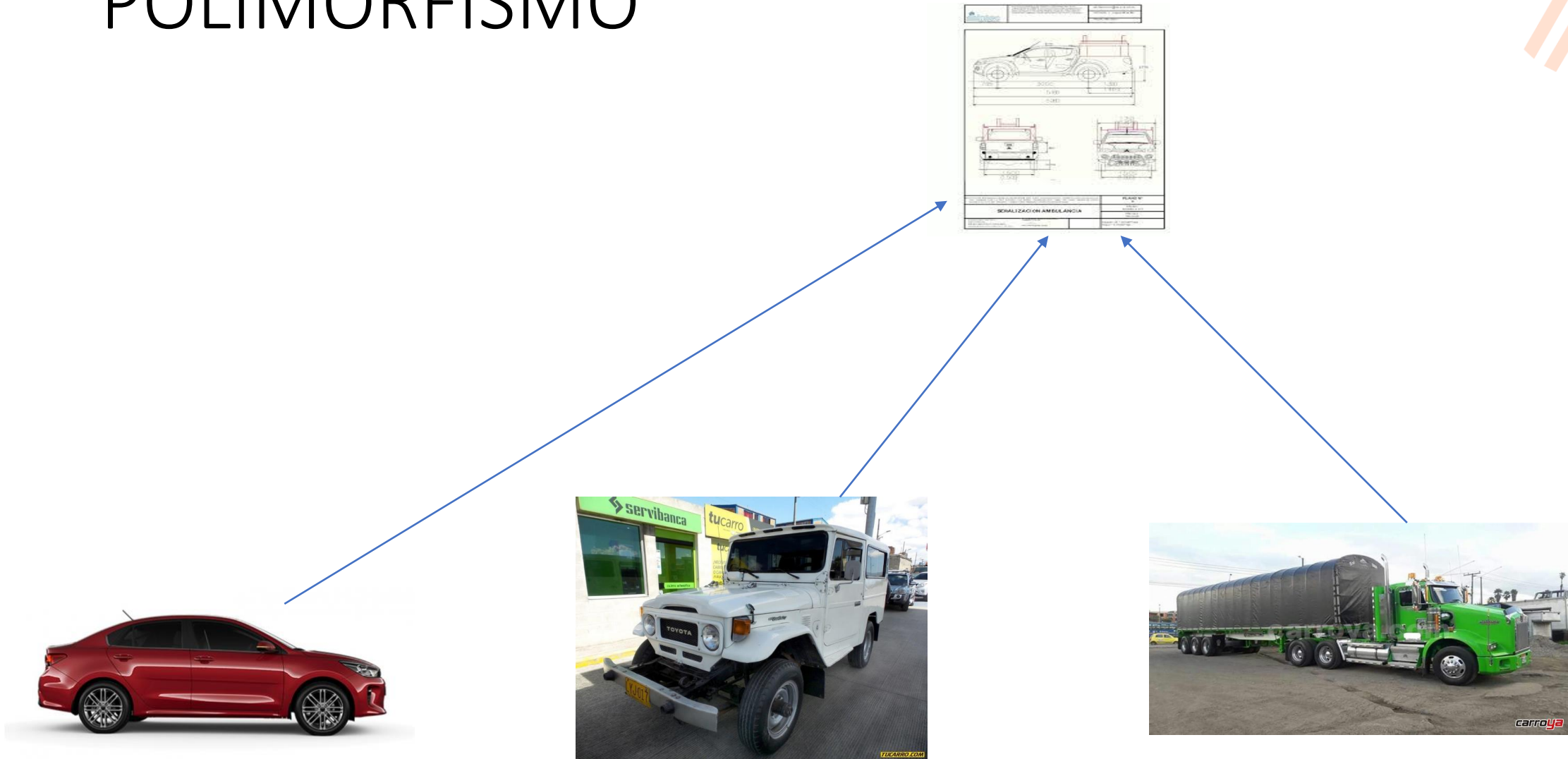




# POLIMORFISMO

- A través de esta característica es posible definir varios métodos o comportamientos de un objeto bajo un mismo nombre, de forma tal que es posible modificar los parámetros del método, o reescribir su funcionamiento, o incrementar más funcionalidades a un método.
- Los objetos “pueden tomar diferentes formas”.

# POLIMORFISMO





# APLICACIÓN PRÁCTICA EN P.H.P

Versión 7.4





# CLASES

- Son “planos”.
- Se definen las Propiedades(Properties) y métodos (Methods)
- Puedo crear (*INSTANCIAR*) Múltiples objetos basados en una clase.





# CREAR UNA CLASE

```
class Vehiculo
{
    public $cantidadPuertas;
    public $cilindraje;
    private $cantidadRuedas;
    protected $placa;

    /* Método mágico usado para darle un valor a los atributos del objeto al crearlo. */

    public function __construct($cantidadPuertas, $cilindraje, $cantidadRuedas, $placa)
    {
        $this->cantidadPuertas =$cantidadPuertas;
        $this->cilindraje =$cilindraje;
        $this->cantidadRuedas =$cantidadRuedas;
        $this->placa =$placa;
    }

    public function mostrarInformacion()
    {
        $info = 'Hola soy un objeto creado a partir de la clase vehiculo y tengo:<br>';
        $info.= "Cilindraje : ".$this->cilindraje.'<br>';
        $info.= "Ruedas : ".$this->cantidadRuedas.'<br>';
        $info.= "Placa Numero : ".$this->placa.'<br>';
        return $info;
    }
};
```



# INSTANCIAR UNA CLASE (CREAR UN OBJETO)

- Una vez que las clases han sido declaradas, será necesario crear los objetos y utilizarlos, para instanciar una clase, solo es necesario utilizar la palabra clave ***new***. El objeto será creado, asignando esta instancia a una variable (**la cual, adoptará la forma de objeto**).
- Lógicamente, la clase debe haber sido declarada antes de ser instanciada



# INSTANCIAR LA CLASE

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Orientado a Objetos con PHP</title>
</head>

<body>
  <pre>
    <?php
      require_once('Vehiculo.php');
      $camion = new Vehiculo(2, 2500, 18, 'XYZ85');
      echo $camion ->mostrarInformacion();
    ?>
  </pre>
</body>

</html>
```



# VISIBILIDAD DE LOS ATRIBUTOS Y MÉTODOS.

- PUBLIC: pueden ser accedidas desde cualquier parte de la aplicación, sin restricción.
- PRIVATE: solo pueden ser accedidas por la clase que las definió.
- PROTECTED: pueden ser accedidas por la propia clase que la definió, así como por las clases que la heredan, pero no, desde otras partes de la aplicación.
- PUBLIC STATIC: puede ser accedida sin necesidad de instanciar un objeto. y su valor es estático
- CONST: Constantes, funcionan igual que PUBLIC STATIC

# ACCESANDO A PROPIEDADES Y MÉTODOS DENTRO DE LA CLASE



Se accede a una propiedad **no estática** dentro de la clase, utilizando la pseudo-variable ***\$this*** siendo esta pseudo-variable una referencia al objeto mismo:

```
return $this->nombre;
```

Cuando la variable **es estática**, se accede a ella mediante el operador de resolución de ámbito, doble dos-puntos **::** anteponiendo la palabra clave ***self*** o ***parent*** según si trata de una variable de la misma clase o de otra de la cual se ha heredado, respectivamente:

```
print self::$variable_estatica_de_esta_clase;  
print parent::$variable_estatica_de_clase_madre;
```

# ACCESANDO A PROPIEDADES Y MÉTODOS FUERA DE LA CLASE



## ACCESO A VARIABLES DESDE EL EXTERIOR DE LA CLASE

Se accede a una propiedad **no estática** con la siguiente sintáxis: `$objeto->variable`

Nótese además, que este acceso dependerá de la visibilidad de la variable. Por lo tanto, solo variables públicas pueden ser accedidas desde cualquier ámbito fuera de la clase o clases heredadas.



# HERENCIA DE CLASES

## HERENCIA DE CLASES

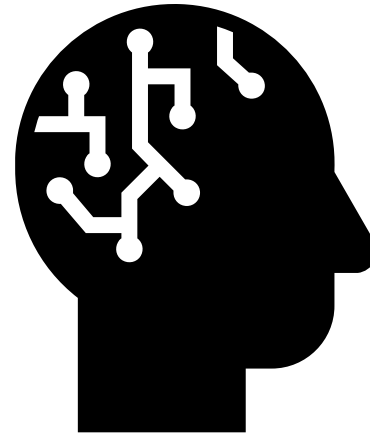
Los **objetos pueden heredar propiedades y métodos de otros objetos**. Para ello, PHP permite la “extensión” (herencia) de clases, cuya característica representa la relación existente entre diferentes objetos. Para definir una clase como extensión de una clase “madre” se utiliza la palabra clave ***extends***.

```
class NombreDeMiClaseMadre {
```



# A TENER EN CUENTA

- En la programación OO se interactúa con los OBJETOS.
- Las clases son solo plantillas. !





# Utiliza siempre un Estándar de Codificación.

- Ejemplo 1:  
<http://www.juntadeandalucia.es/servicios/madeja/contenido/subsistemas/desarrollo/especificaciones-codificacion-y-construccion>
- Ejemplo 2: <http://zfdes.com/manual/es/coding-standard.html>
- RECOMENDACIÓN: Puede empezar usando el documento guía del instructor.



# GRACIAS

Línea de atención al ciudadano: 018000 910270  
Línea de atención al empresario: 018000 910682



@SENAcomunica

[www.sena.edu.co](http://www.sena.edu.co)