

INGENIERÍA MECATRÓNICA



DI\_CERO

DIEGO CERVANTES RODRÍGUEZ

JAVA 8

APACHE NETBEANS IDE 12.5

Programación Orientada  
a Objetos (POO)

# Contenido

<b>Programación Orientada a Objetos (POO)</b> .....	2
Objetos .....	2
Clase .....	2
Método Main .....	3
Constructor .....	4
Operador this .....	4
Instancia .....	5
Llamada a Atributos de un Objeto .....	6
Llamada a Métodos de un Objeto .....	6
Funciones Include Require .....	6
Referencias: .....	6



# Programación Orientada a Objetos (POO)



La programación orientada a objetos o POO es una metodología que sirve para abstraer el concepto de programación y de esa manera poder crear de manera más fácil programas y/o algoritmos. Esta es la idea que se quería alcanzar, aunque más que nada esto nos sirve para entender la lógica con la que se manejan los programas predefinidos del lenguaje Java. Este concepto está tan arraigado en el lenguaje Java que todos los programas a fuerza deben estar compuestos de mínimo una clase.

## Objetos

La programación orientada a objetos consiste en **trasladar el comportamiento de los objetos de la vida real al código de programación** para hacer más fácil el proceso.

Los objetos de la vida real tienen **atributos** (que describen sus características) y **métodos** (que describen lo que el objeto puede hacer), estos aspectos deben ir descritos en cada objeto de programación que creemos.

- Objeto:
  - Tiene propiedades (atributos):
    - Color
    - Peso
    - Alto
    - Largo
  - Tiene un comportamiento (¿Qué es capaz de hacer?):
    - Arrancar
    - Frenar
    - Girar
    - Acelerar



## Clase

Una clase es un **modelo donde se redactan las características comunes de objetos similares**. Los programas en Java deben estar compuestos de por lo menos una clase.

Por ejemplo, siguiendo el ejemplo del objeto coche, todos los coches tendrán 4 ruedas, un motor, una rueda de repuesto, un ancho, un largo y un peso.



Por convención siempre la primera letra de las clases se debe poner en mayúscula.

```
modificador_de_acceso class NombreDeLaClase {  
    //Atributos o propiedades de los objetos pertenecientes a la clase.  
    var $atributo_1;  
  
    var $atributo_2;...  
  
    var $atributo_n;  
  
    //Métodos o funcionalidades de los objetos pertenecientes a la clase.  
    function metodo_1() {  
        Acción 1 que puede ejecutar nuestro objeto  
    }  
  
    function metodo_2() {  
        Acción 2 que puede ejecutar nuestro objeto  
    }...  
  
    function metodo_n() {  
        Acción n que puede ejecutar nuestro objeto  
    }  
}
```

## Método Main

Además, es importante decir que siempre en Java una clase será la principal y esto lo sabemos porque esa clase mandará a llamar a las demás por medio de un método llamado método main:

```
public class ClasePropia {  
    //Esta es la clase principal  
    public static void main(String args[]){  
        /*Lo sé porque es la que contiene el método main, dentro de este método  
        se manda a llamar a las demás clases*/  
    }  
}
```

Esto no se cumple cuando estemos construyendo un programa tipo Applet.



## Constructor

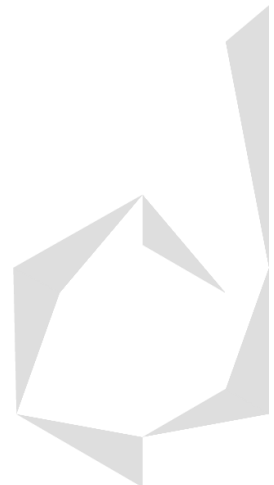
Los objetos en programación deben de tener un **estado inicial en sus atributos y en sus métodos**, esto se declara dentro de la clase con un método especial llamado **constructor** que indica el estado inicial de los objetos o instancias de mi clase y **siempre se debe declarar con el mismo nombre de la clase**, aunque esto se va a cambiar para nuevas versiones de PHP y el constructor se debe declarar con la instrucción `__construct()`.

## Operador this

El operador `$this` sirve para referirnos a los atributos de la clase para poder asignárselos a las futuras instancias que creamos, se usa en conjunto con el símbolo `->` que se refiere a los objetos de mi clase, esto se usa para darle un estado inicial a los atributos de nuestros objetos, además algo importante a mencionar es que el atributo se tiene que poner sin el signo de pesos porque ya lo tiene incluido el operador `$this`.

Si no usamos el operador `$this` dentro de los métodos de nuestra clase, el método no podrá acceder a los atributos de nuestra clase.

```
class NombreDeLaClase {  
    //Atributos o propiedades de las instancias de esta clase.  
    var $atributo_1;  
    var $atributo_2; ...  
    var $atributo_n;  
  
    //Métodos o funcionalidades de las instancias de esta clase.  
    function método_1() {  
        Acción 1 que puede ejecutar nuestro objeto  
    }  
    function método_2() {  
        Acción 2 que puede ejecutar nuestro objeto  
    } ...  
    function método_n() {  
        Acción n que puede ejecutar nuestro objeto  
    }  
}
```



```
/*MÉTODO CONSTRUCTOR que indica el estado inicial de los objetos que pertenezcan a esta clase*/
```

```
function __construct () {  
  
    $this->atributo_1 = valor;  
  
    $this->atributo_2 = valor;  
  
    $this->atributo_n = valor;  
  
}  
}
```

## Instancia

Es un **ejemplar perteneciente a una clase**, osea **cada objeto** en particular **que ocupe el modelo descrito por la clase**.

Por ejemplo, cada objeto Renault, Ford, Mercedes, BMW, etc. es una instancia de la clase coche y donde cada objeto tiene características comunes, pero también tienen elementos diferenciadores entre ellos



```
$nombreObjeto_o_Instancia = new NombreDeLaClase ();
```

Aquí algo importante a mencionar es que el operador **new** lo que está haciendo es mandar a llamar al constructor de la clase.

## Llamada a Atributos de un Objeto

Para llamar al atributo o campo de un objeto se debe usar el nombre de la instancia u objeto y el símbolo de flecha -> seguido del nombre del atributo.

**\$nombreObjeto\_o\_Instancia -> atributo\_n;**

## Llamada a Métodos de un Objeto

Para llamar al método o comportamiento de un objeto se debe usar el nombre de la instancia u objeto y el símbolo de flecha -> seguido del nombre del método con sus paréntesis.

**\$nombreObjeto\_o\_Instancia -> método\_n();**

## Funciones Include Require

Estas funciones sirven para poder incluir el código de una clase externa PHP dentro de la etiqueta php de un documento.

**include ("NombreArchivoExterno.php");**

**require ("NombreArchivoExterno.php");**

la diferencia entre **include** y **require** es que:

- Si con **include** hacemos referencia a un archivo que no existe, al llegar a esa línea de código donde estoy tratando de importar el archivo me saldrá un error, pero el resto del programa lo ejecutará.
- Si usamos **require** no se ejecutará nada después de ese error.

## Referencias:

Pildoras Informáticas, "Curso Java", 2023 [Online], Available:  
<https://www.youtube.com/playlist?list=PLU8oAIHdN5BktAXdEVCLUYzvDyqRQJ2lk>

