INGENIERÍA MECATRÓNICA

DI_CERO

DIEGO CERVANTES RODRÍGUEZ

JAVA 8

APACHE NETBEANS IDE 12.5

Atributos y Métodos Estáticos

Contenido

Atributos o Métodos Estáticos:

Clase

Recordemos que las clases representan un **modelo donde se redactan las características comunes de objetos similares,** los programas grandes y complejos los dividimos en módulos o partes representados por estas clases.

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

```
class NombreDeLaClase{
    //Atributos o propiedades de las instancias de esta clase.
    modificadorDeAcceso $atributo_1;
    modificadorDeAcceso $atributo_2;...
    modificadorDeAcceso $atributo_n;

    //Métodos o funcionalidades de las instancias de esta clase.
    modificadorDeAcceso function metodo_1(){
        Acción 1 que puede ejecutar nuestro objeto
    }
    modificadorDeAcceso function metodo_2(){
        Acción 2 que puede ejecutar nuestro objeto
    }...
    modificadorDeAcceso function metodo_n(){
        Acción n que puede ejecutar nuestro objeto
    }
}
```

Constructor

Los objetos en programación deben de tener un **estado inicial en sus atributos** (que representan las características de los objetos) **y en sus métodos** (que representan las funcionalidades que pueden hacer los objetos), esto se declara dentro de la clase con un método especial llamado **constructor** que indica el estado inicial de los objetos o instancias, se debe declarar con la instrucción **_ _construct()** en donde debía ir el nombre de la función.

this

El concepto de métodos y atributos estáticos está íntimamente relacionado con **el operador $this ya que éste sirve para referirnos a los elementos de la misma clase en la que nos encontramos para que se pueda crear una copia de los atributos para cada instancia de la clase**, se usa dentro del constructor para poder darle un estado inicial a los atributos y métodos y se usa en conjunto con el símbolo **->**, además algo importante a mencionar es que el atributo se tiene que poner sin el signo de pesos porque ya lo tiene incluido el operador **$this**.

Si no usamos el operador **$this->** en los métodos de nuestra clase, el método no podrá acceder a los atributos de nuestra clase crear una copia de éstos y asignársela a cada objeto.

```
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**.

```
$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.

Modificadores de Acceso

Para poder encapsular algunas cosas sí y otras no y que algunos atributos y métodos puedan ser accedidos y otros no, es que se usan los modificadores de acceso, generalmente se usan sólo 3:

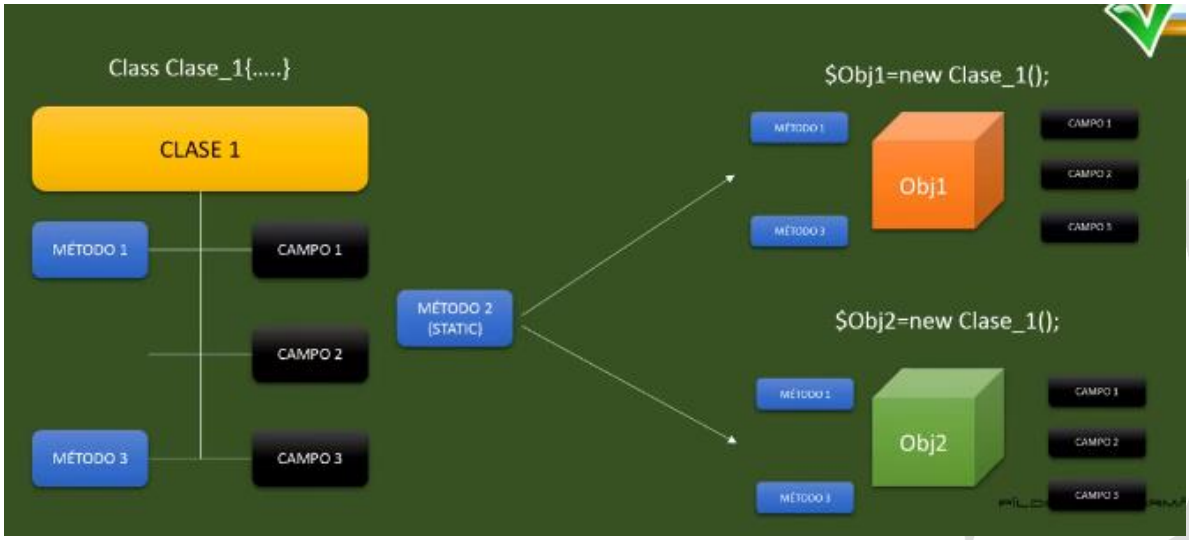
- **public:** Hace accesible a esa propiedad desde cualquier parte del programa (**desde cualquier clase**).
- **private:** Hace accesible a esa propiedad **solamente desde la propia clase**.
- **protected:** Hace accesible a esa propiedad **desde la propia clase o desde las clases que hereden de ella**.

Cuando no existe modificador en un método o atributo, el modificador por defecto es el **public**.

Atributos o Métodos Estáticos

Siempre que creemos una clase le debemos asignar sus atributos o campos y sus métodos o comportamientos, el concepto de atributos o métodos estáticos aplica cuando nosotros creamos instancias u objetos que pertenecen a nuestra clase, ya que cada que cree una instancia, el operador new llamará al constructor de la clase para darle un estado inicial a los atributos o métodos del objeto, pero sí creo otro objeto de la misma clase **si los atributos o métodos de la clase NO son estáticos, se creará una copia de cada atributo y método para cada instancia de la clase**, eso quiere decir que cada uno de los objetos que creemos podrá tener atributos con valores diferentes o pueden tener métodos que podrían comportarse de manera diferente para cada objeto ya que son copias independientes entre sí, esto puede llegar a ser un problema dependiendo de la función del método o atributo de mis clases.

Si dentro de la clase definimos uno de los métodos o atributos como estático lo que pasará es que de ese método o atributo no se crearán copias para cada uno de los objetos creados por la clase, cuando pasa esto se dice que el método o atributo ahora pertenece a la clase y por eso no se crean copias para cada instancia.



Esto implica que con el objeto no podremos acceder al atributo estático o al método estático, para poder hacerlo necesitamos usar un operador nuevo llamado **self** en conjunto con el símbolo **::** ya que **éstos hacen referencia a la clase** en vez de usar el operador **this** en conjunto con el símbolo **->** que **hacen referencia al objeto**.

self

E**l operador self sirve para referirnos a los elementos de clase en la que nos encontramos pero sin crear una copia para cada instancia de la clase**, se usa para llamar atributos estáticos donde no se le asignará una copia a cada objeto ya que el elemento pertenece a la clase y se usa en conjunto con el símbolo **::**, otra diferencia con el operador $this es que self no se usa en conjunto con el signo de pesos, el signo de pesos se debe poner alado del atributo y cuando se declara un método estático simplemente no se pone.

Si no usamos el operador **self::** en los métodos de nuestra clase, el método no podrá acceder a los atributos de nuestra clase **sin crear una copia de éstos para asignársela a cada objeto**.

Para acceder al atributo estático de la clase lo que podría hacer si es que el atributo no está encapsulado debemos poner:

NombreDeLaClase :: $atributo_estático;

NombreDeLaClase :: método_estático();

Y ese atributo o método estático lo que hará es afectar a todos los objetos declarados posteriores.

Referencias:

Pildoras Informáticas, “Curso Java”, 2023 [Online], Available: https://www.youtube.com/playlist?list=PLU8oAlHdN5BktAXdEVCLUYzvDyqRQJ2lk