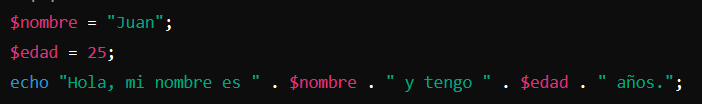
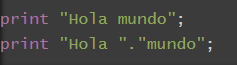
**IMPRIMIR**









**VARIABLES**

****

**TIPOS DE DATOS**

* **gettype($variable):** obtiene el tipo de variable
* **settype($variable,”float”)**: convierte una variable al tipo indicado por parámetro
* **is\_array( )**, **is\_bool( )**, **is\_float( )**, **is\_integer( ):** comprobar si es de un tipo concreto
* **isset($variable)**: indica si una variable está definida y no es null.
* **unset($variable):** destruye una variable

**FECHAS**

* **date\_default\_timezone\_set:** establece la zona horaria

****

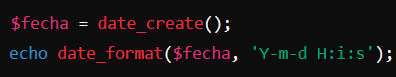
* **date\_create():** Crea un objeto DateTime a partir de una cadena o por defecto la fecha actual

****

* **date( ):** obtener una cadena de texto a partir de una fecha y hora con el formato que se elija.

****

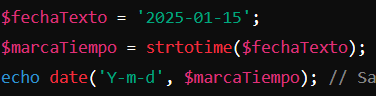
* **date\_format():Formatea un objeto DateTime en una cadena de texto**

****

* **checkdate():**Validar una fecha

****

* **strtotime():**Convierte una cadena textual de fecha/hora en una marca de tiempo

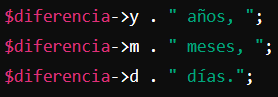
**ç**

* **getdate():**Devuelve un array con información sobre la fecha actual o una marca de tiempo específica

****

* **date\_diff():**Calcula la diferencia entre dos fechas, devuelve un objeto DateInterval





* **date\_modify():**Modifica una fecha existente

** **

****

* **idate():** Devuelve un solo componente de la fecha como un entero

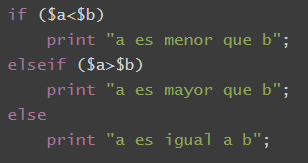
****

**TERNARIO**

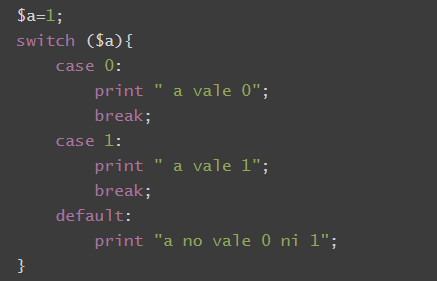
****

**CONDICIONALES**

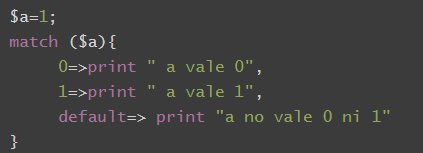
* **if**

****

* **switch**

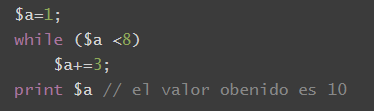
****

* **match**

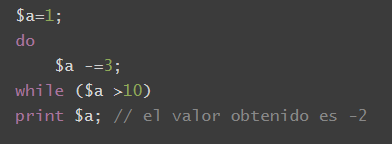
****

**BUCLES**

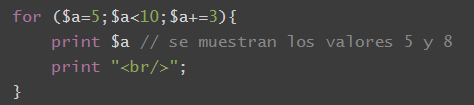
* **while**

****

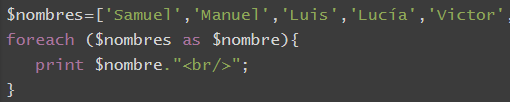
* **do while**

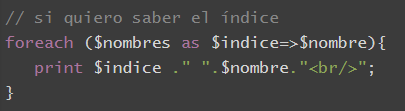
****

* **for**

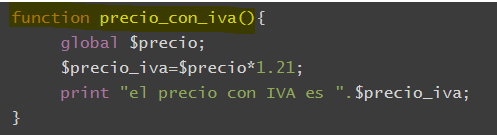
****

* **foreach**

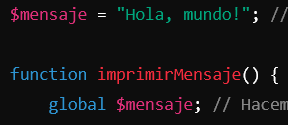
****

****

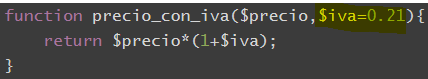
**FUNCIONES**

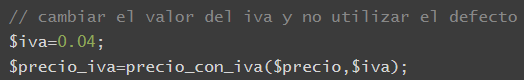
****

* **global:** permite utilizar una variable definida fuera de la funcion

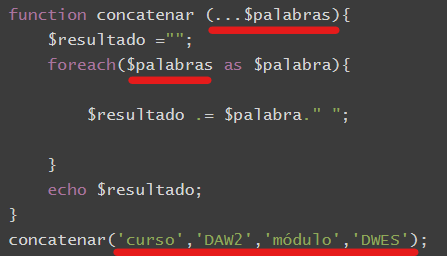
****

* **valores por defecto**

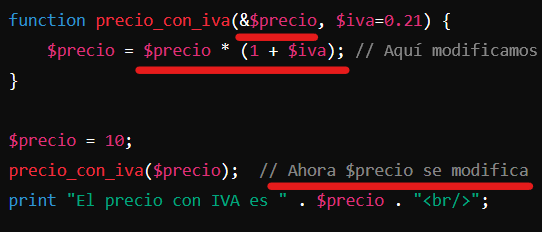
****

****

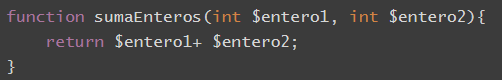
* **tres puntos:** permite pasar un número indeterminado de parámetros, lo trata como un array

****

* **parámetro por referencia (&):** permite que la función modifique la variable original que se pasa por parámetro

****

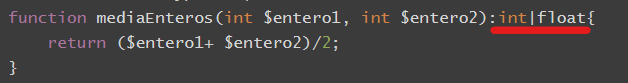
* **parámetros con tipos**

****

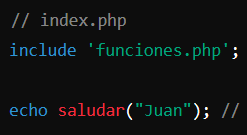
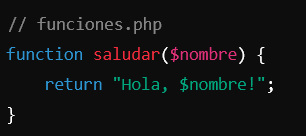
* + declarar que el tipo es restrictivo dando error si no hay correspondencia

****

* **tipo de dato que la función va a devolver**

****

* **Funciones en archivos externos:** permite que en un archivo, puedas utilizar funciones definidas en un archivo externo

****

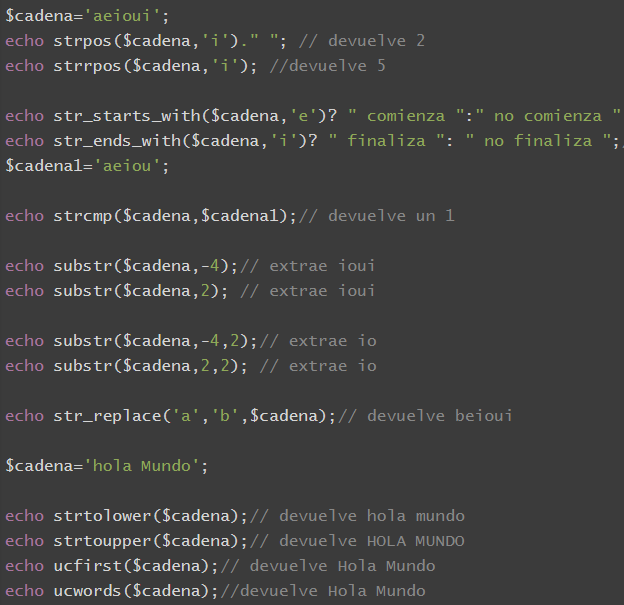
* + **Include:** evalúa el contenido del fichero que se indica y lo incluye como parte del fichero actual, si el fichero no se encuentra, PHP emitirá una advertencia
  + **include\_once:** evita que el mismo fichero sea incluido más de una vez.
  + **require:** si el fichero no se encuentra, require generará un error fatal (fatal error) y detendrá la ejecució
  + **require\_once:** Si el fichero ya ha sido incluido, no se volverá a incluir, y si no se puede encontrar, el script se detendrá con un error fatal.

**CADENAS DE CARACTERES**

* **strpos( exp1, exp2)**, busca la primera coincidencia de izquierda a derecha y devuelve la posición
  + exp1 - cadena donde buscar
  + exp1 - cadena donde buscar
  + **strrpos( exp1,exp2)**, sentido de búsqueda de derecha a izquierda
* **str\_starts\_with(exp1,exp2)**, si comienza con la cadena que queremos buscar
* **str\_ends\_with(exp1,exp2)**, si termina con la cadena a buscar
* **strcmp(exp1, exp2)**, compara dos cadenas:
  + si son iguales devuelve 0, si exp1 > exp2 devuelve un 1, si exp1 < exp2 devuelve un -1
* **substr(exp1,exp2, exp3)**,obtiene una subcadena
  + exp1 - cadena donde extraer subcadena
  + exp2 - posición desde donde comienza a extraer la subcadena (el 0 es la primera posición).

Si ponemos un número negativo comienza desde el final de la cadena

* + exp3 - indica el número de caracteres a extraer
* **str\_replace(exp1,exp2,exp3)**, reemplazar una cadena
  + exp1 - cadena que se quiere reemplazar en la exp3
  + exp2 - cadena por la que se quiere reemplazar
  + exp3 - cadena donde se quiere operar
* **strtolower(exp1)**, convertir toda la cadena exp1 a minúsculas
* **strtoupper(exp1)**, convertir toda la cadena exp1 a mayúsculas
* **ucfirst(exp1)**, convertir la primera letra de la cadena exp1 a mayúsculas
* **ucwords(exp1)**, convertir la primera letra de cada palabra de la cadena exp1 a mayúsculas

****

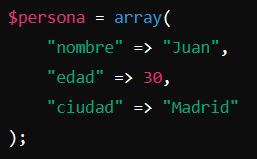
**ARRAYS**

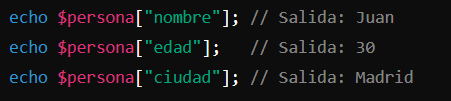
* **declaración**

****

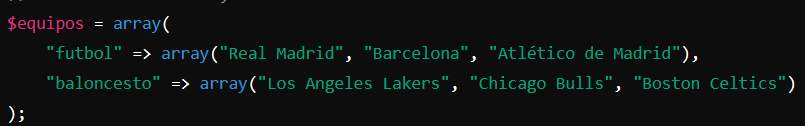
****

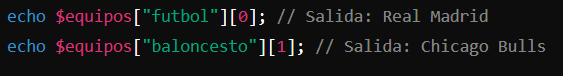
* + **asociativo**

****

****

* + **multidimensional**

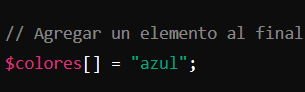
****

****

* **mostrar**



* **agregar elementos**
  + **al final**

****

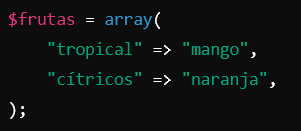
* + **posición especifica**

****

* + - $colores: array al que añadir
    - 1: posición en la que se añadirá
    - 0: cantidad de elementos que se deben eliminar del array original en la posición (moverá una posición los elementos siguientes)

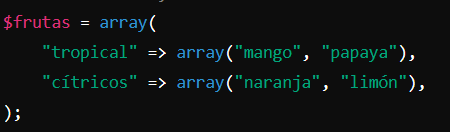
Si sería 1, el elemento original se eliminaría (remplazo), si fuera 2, se remplazaría en las 2 posiciones

* + - “amarillo”: elemento a añadir
  + Agregar a array asociativo





* + Agregar a array multidimensional



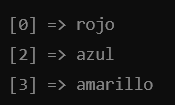




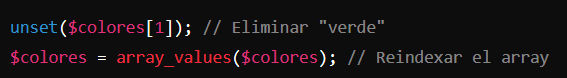
* **Eliminar elementos**
  + **unset():** conociendo el índice

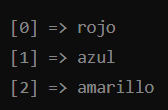
****

****

****

* + - **reindexarlo**

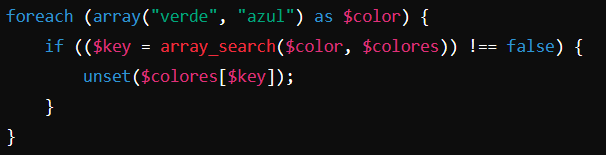
****

****

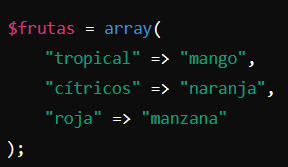
* + **array\_diff():** Si no conoces el índice, solo el valor

****

* + **eliminar varios elementos**

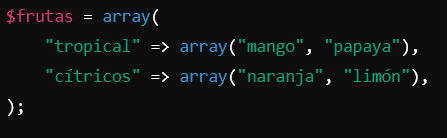
****

* + **eliminar de array asociativo**

****

****

* + **eliminar de array multidimensional**

****

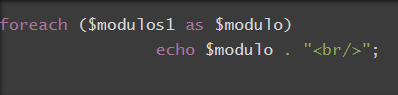
Eliminar naranja

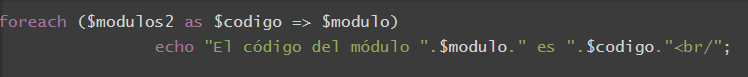
****

Eliminar todos los citricos

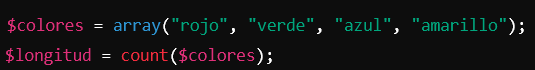
****

* **Recorrer un array**

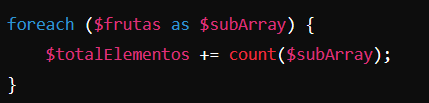
****

****

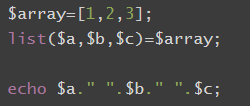
* **Longitud del array**
  + **count():** numero de elementos del array

** =>** Saldría 4

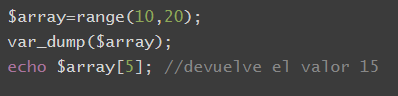
Para array multidimensional o asociativo



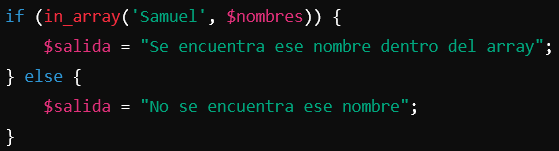
* **Mas funciones**
  + **list(variable1,…,variablen)=nombrearray** : almacenar los valores del array en variables
    - list(variable1,…,variablen)=nombrearray
    - nombrearray , es el array que queremos extraer a las variables

****

* + **range(valor1,valor2):** rellenar un array con un intervalo de valores comprendido entre valor1 y valor 2

****

* + **in\_array( cadena a buscar, nombrearray):** saber si un elemento se encuentra dentro de un array, devuelve un boolean

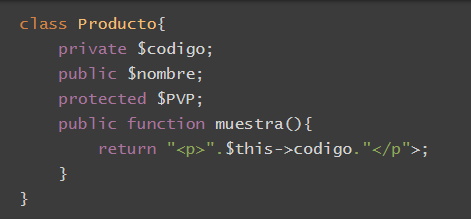
****

**VARIABLES ESPECIALES DE PHP**

* **$\_SERVER**: contiene información sobre el entorno del servidor web y de ejecución.
* **$\_GET, $\_POST y $\_COOKIE**: contienen las variables que se han pasado al guión actual utilizando respectivamente los métodos GET (parámetros en la URL), HTTP POST y Cookies HTTP.
* **$\_REQUEST**: junta en uno solo el contenido de los tres arrays anteriores.
* **$\_ENV**: contiene las variables que se puedan haber pasado a PHP desde el entorno en que se ejecuta.
* **$\_FILES**: contiene los ficheros que se puedan haber subido al servidor utilizando el método POST.
* **$\_SESSION**: contiene las variables de sesión disponibles para el guión actual.

**OBJETOS**

* **Declaración**

****

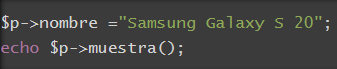
* **Instanciar**

****

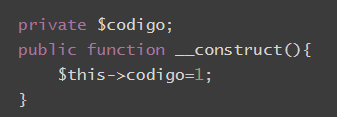
****

* **Acceder a sus atributos o métodos**

****

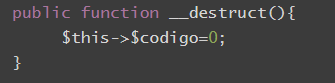
****

* **Constructores**

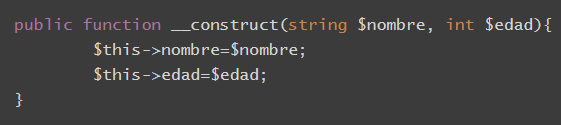
****

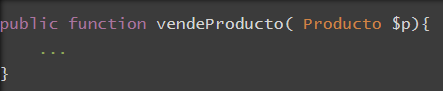
**\*** Solo puede haber un constructor en cada clase

* **Destructor:** definir acciones que se ejecutarán cuando se elimine el objeto

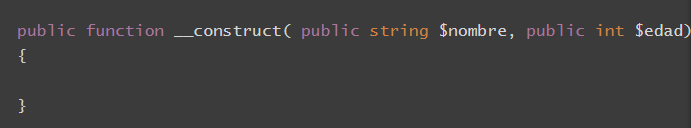


* **Constructor con tipo de dato**

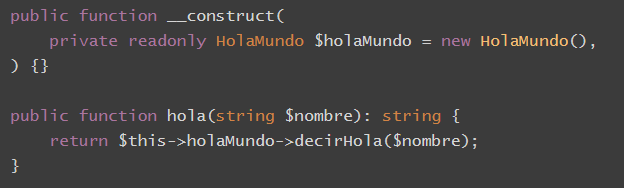
****

****

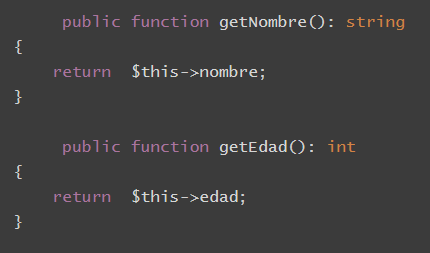
* **Constructor que define los parámetros**

****

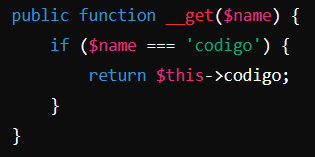
* **Crear objeto en el constructor**

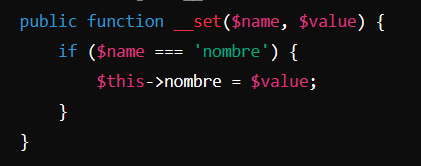
****

* **Metodos get y set**

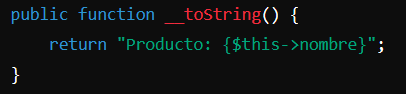
****

* **Metodos mágicos ( \_\_ )**
  + **Get y set**

****

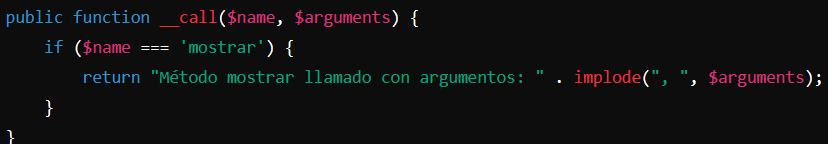
****

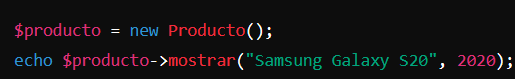
* + **toString:** Definir la representación en cadena del objeto

****

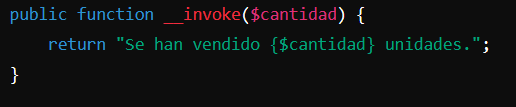
****

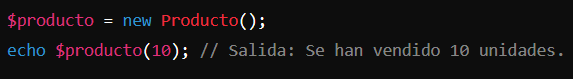
* + **Call:** Llamada a métodos no accesibles

****

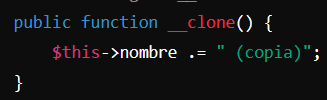
****

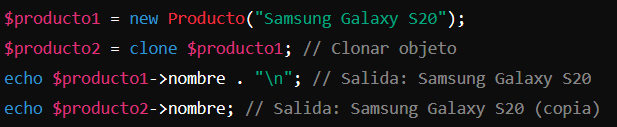
* + **Invoke: I**nvocar el objeto como una función.

****

****

* + **Clone:** Clonación del objeto

****

****

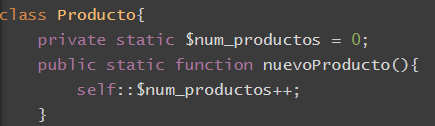
* **Operador this:** desde un objeto se invoca un **método de la clase**

****

* **Constantes:** variable que no se puede cambiar

****

* **Metodos estaticos: no** pueden ser llamados desde un objeto de la clase utilizando el operador “ -> “

****

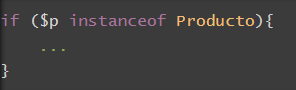
* + Si el método o atributo es público , deberá accederse utilizando el nombre de la clase y el operador de resolución de ámbito **::**

****

* + Si es privado se podrá acceder a él desde los métodos de la propia clase, utilizando la palabra **self**

****

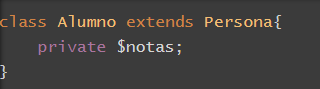
* **Utilización de objetos**
  + **Instance of:** comprobar si es o no una instancia de una clase

****

|  |  |  |
| --- | --- | --- |
| Función | Significado | Ejemplo |
| **get\_class** | Devuelve el nombre de la clase del objeto |  |
| **class\_exists** | Devuelve true si la clase está definida o false en caso contrario |  |
| **get\_declared\_classes** | Devuelve un array con los nombres de las clases definidas |  |
| **class\_alias** | Crea un alias para una clase |  |
| **get\_class\_methods** | Devuelve un array con los nombres de los métodos de una clase que son accesibles desde donde se hace la llamada |  |
| **method\_exists** | Devuelve true si existe el método en el objeto o la clase que se indica, o false en caso contrario, independientemente de si es accesible o no |  |
| **get\_class\_vars** | Devuelve un array con los nombre de los atributos de una clase que son accesibles desde dónde se hace la llamada |  |
| **get\_object\_vars** | Devuelve un array con los nombres de los atributos de un objeto que son accesibles desde dónde se hace la llamada |  |
| **property\_exits** | Devuelve true si existe el atributo en el objeto o la clase que se indica, o false en caso contrario, independientemente de si es accesible o no |  |

**HERENCIA**

* **Declaracion**

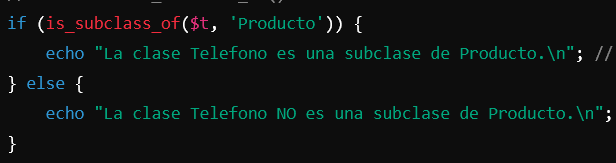
****

* **Funciones**
  + **get\_parent\_class:** Devuelve el nombre de la clase padre del objeto o la clase que se indica

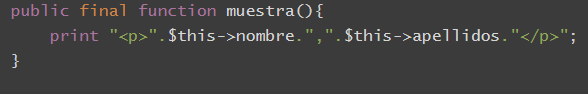
****

****

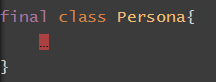
* + **is\_subclass\_of:** Devuelve true si el objero o la clase del primer parámetro, tiene como clase base a la que se indica en el segundo parámetro, o false en caso contrario

****

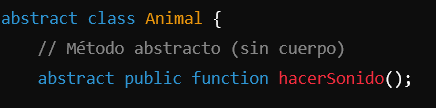
* **Modificadores de acceso en herencia**
  + **public:** atributos o metodos que deberían ser accesibles desde cualquier lugar, incluyendo instancias de la clase y subclases.
  + **protected:** para atributos o metodos que van a ser utilizados por las clases hijas
  + **private:** para atributos o metodos específicos de una clase o clase hija
  + **Final**
    - **En métodos:** no podrán ser sobrescritos en las subclases

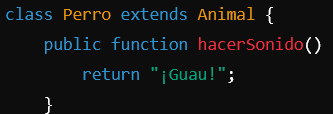
****

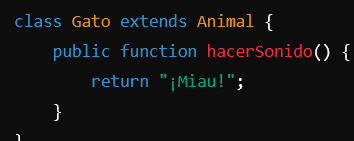
* + - **En clases:** No podrá ser heredada por otras clases

****

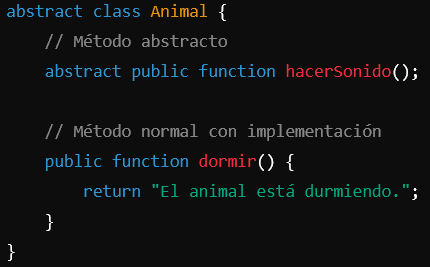
* + **Abstract**
    - **En métodos:** no tiene implementación en la clase donde se declara, debe ser implementado en las subclases

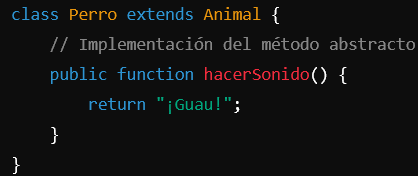
****

****

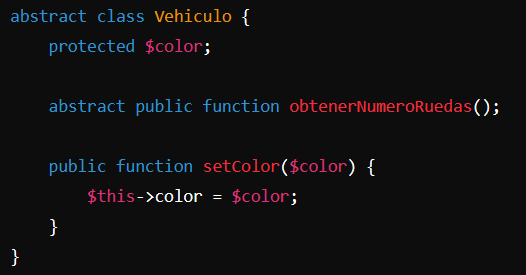
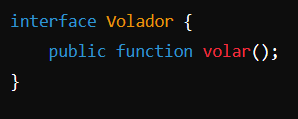
****

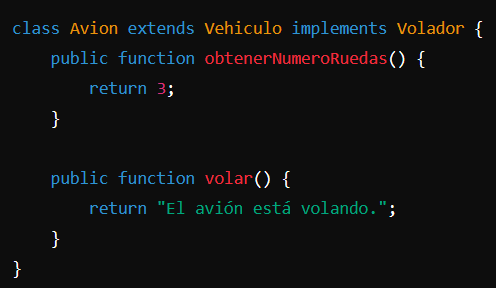
* + - **En clases:** no pueden ser instanciadas directamente, puede contener metodos normales y abstractos

****

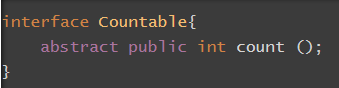
****

**INTERFACES**



* **Interfaces definidas** 
  + **Countable**

****

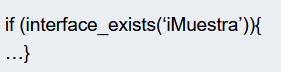
* + **Iterator**
  + **Serializable**

**….**

* **Funciones**
  + **get\_declared\_interfaces:** Devuelve un array con los nombres de los interfaces declarados

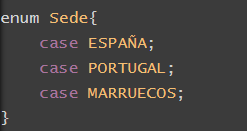
****

* + **interface\_exists:** Devuelve true si existe el interface que se indica o false en caso contrario

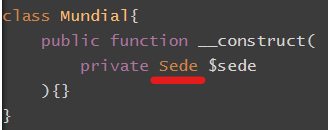
****

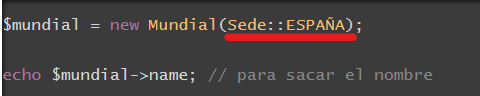
**ENUMS**

* **Declaracion**

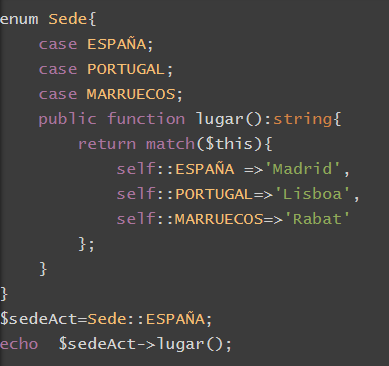
****

* **Uso**

****

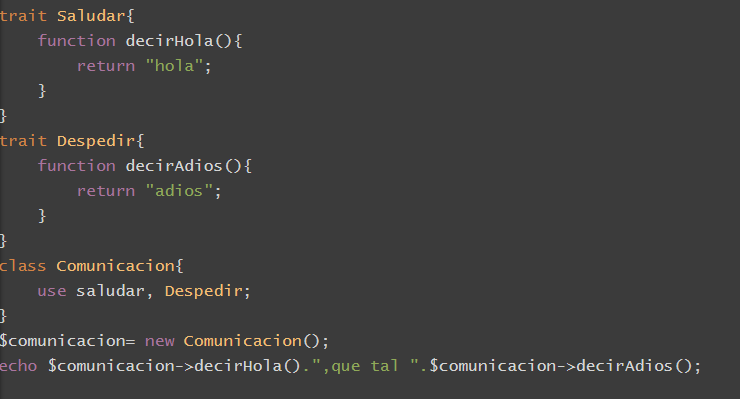
****

* **Metodos dentro de un enum**

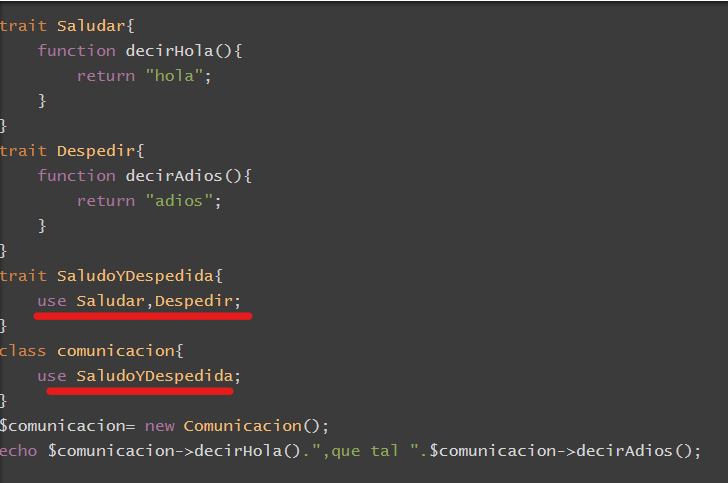
****

**TRAITS**

* Permiten reutilizar métodos en varias clases independientes y de distintas jerarquías
* No se puede instanciar, simplemente facilita comportamientos a las clases sin necesidad de usar la herencia.
* No puede implementar interfaces ni extender clases normales ni abstractas

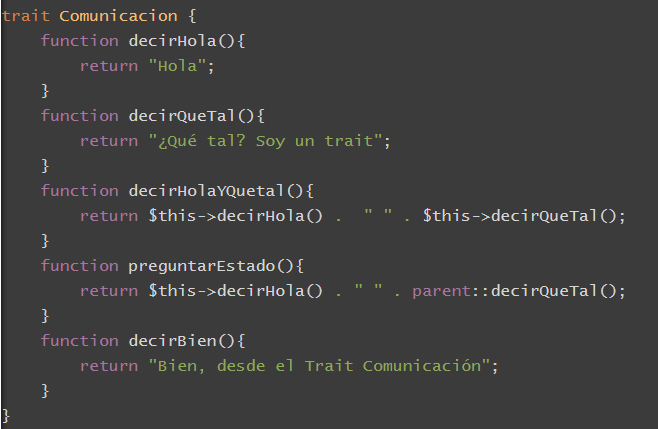
****

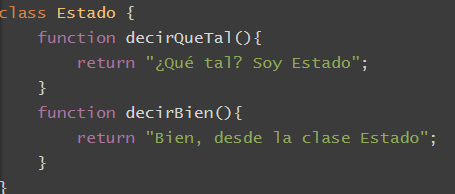
* **Traits dentro de otros traits**

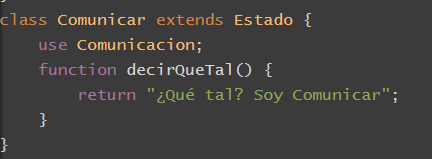
****

* **Orden de precedencia entre traits y clases**

1. Métodos de un trait sobreescriben métodos heredados de una clase padre
2. Métodos definidos en la clase actual sobreescriben a los métodos de un trait



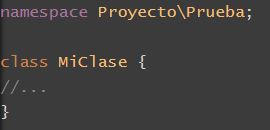
****

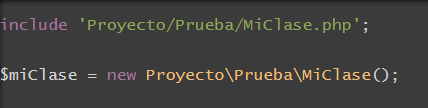
****

* **Reflection**
  + **ReflectionClass::getTraits()**. Devuelve un array con todos los traits disponibles en una clase
  + **ReflectionClass::getTraitNames()**. Devuelve un array con los nombres de los traits en una clase.
  + **ReflectionClass::isTrait()**. Comprueba si algo es un trait o no.
  + **ReflectionClass::getTraitAliases()** Devuelve un trait con los alias como keys y sus nombres originales como values.

**NAMESPACES**

* **Uso**

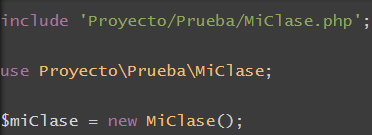
****

****

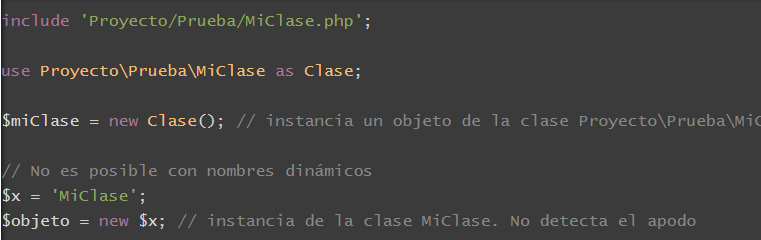
* **Constante**

****

* **Importar namespace**

****

* **Alias en los namespace**

****

**FORMULARIOS**

* **HTML**