POO/OOP

Programación Orientado a Objetos

Object-Oriented Programming

Vocabulario

- Clase, objeto
- Ejemplar de clase, instancia de clase, ejemplarizar una clase, instanciar una clase
- Modularización
- Encapsulamiento / encapsulación
- Herencia
- Poliformismo

- Constantes de clase
- Keyword: static

Modularización

- El concepto de **Modularización** como su nombre indica consiste en dividir un programa en módulos que puedan compilarse por separado, sin embargo mantendrán conexiones con otros módulos.
- Principios
  - **Capacidad de descomponer un problema complejo**
    Divide y vencerás
  - **Capacidad de componer a través de módulos**
    Posibilidad de componer el programa desde los problemas más pequeños
  - **Comprensión de sistema en partes**
    El tener separada cada parte separada nos ayuda a un mejor entendimiento del código y del sistema

Modularización

«Programe pensando en quien mantendrá el código»

Constantes de clase

Como vimos con anterioridad las constantes son espacios de memoria donde almacenamos datos pero que no se pueden modificar. Veamos las particularidades de las variables de clase:

- Las **constantes de clase** se declaran e inicializan al mismo tiempo
- Una **constante de clase** por defecto es **pública** y al contrario que las **variables de clase** no necesitan utilizar la palabra reservada **public**
- Como las **constantes** no se pueden modificar afectan al conjunto de los **objetos instanciados**, por tanto a la **clase** misma. Por este motivo para acceder a ellos no se utiliza la nomenclatura del punto, en su lugar "::", y tampoco $this, en su lugar: self o el propio nombre de la clase
- No se pueden cargar en los métodos, ni tan siquiera en el **método constructor**

```
class Empleado{

     //declaración de atributos
     private string $nombre;
     private float $sueldo;
     private DateTime $altaContrato;
      const PI = 3.1416;//No es necesario poner la palabra reservada public

     //constantes
     private const SUELDO_BASE =1000; //declaración e inicialización
```

Campos static

- Los atributos, propiedades o campos afectan a los objetos
- Un **campo static** afecta a la clase no al objeto
- **No hace falta ser instanciada**
- Se accede a ellos a través del **nombre de la clase** o con la palabra reservada **self**

```php
class Empleado{
     //declaración de atributos
     private string $nombre;
     private float $sueldo;
     private DateTime $altaContrato;
     private int $orden; //un número que identifica en el orden que creamos a los empleados
```

```php
$empleado1 = new Empleado('Xurxo',1800,2018,4,6);
$empleado1->orden=1;
```

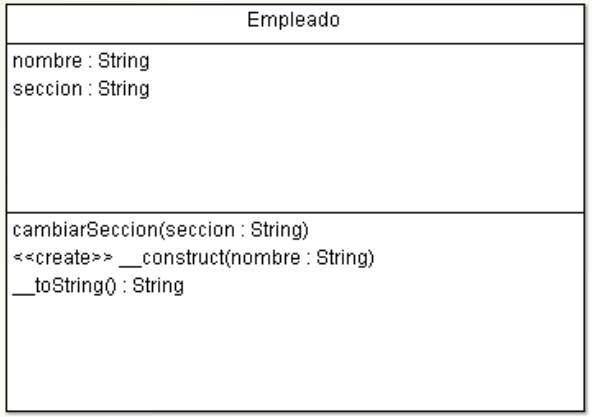
```php
$empleado2 = new Empleado('Carolina',2200,2020,3,7);
$empleado2->orden=2;
```

```php
private string $nombre;
private float $sueldo;
public int $orden
```

El orden de entrada tendría que ser automático

```php
private string $nombre;
private float $sueldo;
public int $orden;
```

Ejercicio



Manipula el ejercicio anterior para que te diga siempre cuál es el número total de empleados utilizando la palabra reservada static

Ejercicio 2

Realizaremos una clase llamada **Persona** que tendrán los atributos (**privados**):
nombre, edad, DNI, sexo (H hombre, M mujer), peso y altura.
Iniciaremos las variables en el método constructor.
El Sexo será mujer por defecto.
Se implantaran un **método constructor** el **nombre**, **peso**, **altura** como parámetros.
Los demás campos serán manipulados a través de getters y setters

Métodos adicionales

toString(): devuelve toda la información del objeto.

comprobarSexo(string sexo): comprueba que el sexo introducido es correcto. Si no es correcto, será H. No será visible al exterior.

esMayorDeEdad(): indica si es mayor de edad, devuelve un booleano.

Ejercicio 2

`calcularIMC()`: Calculará si la persona esta en su peso ideal(peso en kg/(altura^2 en m)). Si esta fórmula devuelve un valor menor que 20, la función devuelve un -1 e indica bajo peso. Si devuelve un número entre 20 y 25 (incluidos) significa que está en su peso ideal y la función devolverá un 0. Si devuelve un valor mayor que 25 significa que tiene sobrepeso, la función devolverá un 1.
Te recomiendo que uses constantes para devolver estos valores.

`$peso/($altura*$altura)` → `$peso/pow($altura,2)`