EJEMPLO: CLASE-CONSTRUCTOR-GETTER-SETTER

Versiones PHP > 8.0

```php
<?php
// Definición NUEVA de la clase Persona
class Persona {
    // Constructor con propiedades promovidas
    public function __construct(
        private string $nombre,
        private int $edad
    ) {}
    // Método getter para el nombre
    public function getNombre(): string {
        return $this->nombre;
    }
    // Método setter para el nombre
    public function setNombre(string $nombre): void {
        $this->nombre = $nombre;
    }
    // Método getter para la edad
    public function getEdad(): int {
        return $this->edad;
    }
    // Método setter para la edad
    public function setEdad(int $edad): void {
        if ($edad > 0) {
            $this->edad = $edad;
        } else {
            echo "La edad debe ser un número
positivo.<br>";
        }
    }
    // Método para mostrar la información de la persona
    public function mostrarInformacion(): void {
        echo "Nombre: " . $this->getNombre() . "<br>";
        echo "Edad: " . $this->getEdad() . "<br>";
    }
}
// Crear un objeto de la clase Persona
$persona = new Persona("Juan", 30);

// Mostrar la información inicial de la persona
echo "<h2>Información Inicial:</h2>";
$persona->mostrarInformacion();
// Modificar los valores de las propiedades usando los
métodos setter
$persona->setNombre("María");
$persona->setEdad(25);
// Mostrar la información actualizada de la persona
echo "<h2>Información Actualizada:</h2>";
$persona->mostrarInformacion();
?>
```

Versiones PHP anteriores

```php
<?php
// Definición de la clase Persona
class Persona {
    // Propiedades privadas
    private $nombre;
    private $edad;
    // Constructor
    public function __construct($nombre, $edad) {
        $this->nombre = $nombre;
        $this->edad = $edad;
    }
    // Método getter para el nombre
    public function getNombre() {
        return $this->nombre;
    }
    // Método setter para el nombre
    public function setNombre($nombre) {
        $this->nombre = $nombre;
    }
    // Método getter para la edad
    public function getEdad() {
        return $this->edad;
    }
    // Método setter para la edad
    public function setEdad($edad) {
        if ($edad > 0) {
            $this->edad = $edad;
        } else {
            echo "La edad debe ser un número positivo.<br>";
        }
    }
    // Método para mostrar la información de la persona
    public function mostrarInformacion() {
        echo "Nombre: " . $this->getNombre() . "<br>";
        echo "Edad: " . $this->getEdad() . "<br>";
    }
}
// Crear un objeto de la clase Persona
$persona = new Persona("Juan", 30);
// Mostrar la información inicial de la persona
echo "<h2>Información Inicial:</h2>";
$persona->mostrarInformacion();
// Modificar los valores de las propiedades usando los métodos setter
$persona->setNombre("María");
$persona->setEdad(25);
// Mostrar la información actualizada de la persona
echo "<h2>Información Actualizada:</h2>";
$persona->mostrarInformacion();
?>
```

En el constructor de `Persona`, las propiedades `nombre` y `edad` se definen y se inicializan directamente.

Se especifican los tipos de datos para las propiedades (`string` para `nombre` y `int` para `edad`) y también para los métodos (`string` para `getNombre`, `int` para `getEdad`, y `void` para los setters y `mostrarInformacion`).

Los getters y setters funcionan de la misma manera que en el ejemplo anterior, permitiendo el acceso y la modificación de las propiedades privadas.

EJEMPLO: CLASE-SUBCLASE

Index.php

```
<?php
require_once 'Telefono.php';
require_once 'Tablet.php';
// Crear objetos de teléfonos y tablets
$telefono = new Telefono("iPhone 13", 999, "iPhone 13 Pro");
$tablet = new Tablet("iPad Air", 599, "10.9 pulgadas");
// Mostrar la descripción de los productos
echo $telefono->descripcion() . "<br>";
echo $tablet->descripcion() . "<br>";
```

/* En index.php , incluimos los archivos de las clases Telefono y Tablet . Luego, creamos objetos de teléfonos. */

Producto.php

```
<?php
class Producto {
protected $nombre;
protected $precio;
public function __construct($nombre, $precio) {
$this->nombre = $nombre;
$this->precio = $precio;
}
public function getNombre() {
return $this->nombre;
}
public function getPrecio() {
return $this->precio;
}
}
public function descripcion() {
return "Producto: {$this->nombre}, Precio: {$this->precio}";
}
```

Tablet.php

```
<?php
require_once 'Producto.php';
class Tablet extends Producto {
private $tamano;
public function __construct($nombre, $precio, $tamano) {
parent::__construct($nombre, $precio);
$this->tamano = $tamano;
}
}
public function descripcion() {
return "Tablet: {$this->nombre}, Tamaño: {$this->tamano}, Precio: {$this->precio}";
}
```

Telefono.php

```
<?php
require_once 'Producto.php';
class Telefono extends Producto {
private $modelo;
public function __construct($nombre, $precio, $modelo) {
parent::__construct($nombre, $precio);
$this->modelo = $modelo;
}
}
public function descripcion() {
return "Teléfono: {$this->nombre}, Modelo: {$this->modelo}, Precio: {$this->precio}";
}
```

/***La clase Producto es la superclase que contiene propiedades comunes como el nombre y el precio de un producto.** También tiene métodos para obtener el nombre y el precio, así como un método descripcion() que proporciona una descripción general.*/

/* **La clase Tablet es otra clase derivada de la superclase Producto .** Al igual que la clase Telefono , tiene propiedades adicionales y su propio constructor. También reemplaza el método descripcion() de la superclase para proporcionar una descripción específica de una tablet. */

/* **La clase Telefono es una clase derivada de la superclase Producto** . Además de las propiedades heredadas, tiene una propiedad adicional llamada modelo. Su constructor llama al constructor de la superclase y luego establece el valor del modelo. La clase Telefono también reemplaza el método descripcion() de la superclase para proporcionar una descripción específica de un teléfono */