Resumen: Programación orientada a objeto con PHP

- **Clase**: Una clase es una plantilla o blueprint para crear objetos. Define propiedades (a menudo llamadas atributos) y métodos (funciones asociadas con un objeto de esa clase).

```php
<?php
/**
 * Clase para representar un libro.
 */
class Libro {
    public $titulo;
    public $autor;

    public function __construct($titulo, $autor) {
        $this->titulo = $titulo;
        $this->autor = $autor;
    }

    public function imprimirDetalles() {
        echo "Título: $this->titulo, Autor: $this->autor\n";
    }
}
?>
```

- **Objeto:** Una instancia de una clase. Cuando se crea un objeto a partir de una clase, se dice que se está "instanciando" la clase.

```php
<?php
$libro1 = new Libro("1984", "George Orwell");
$libro1->imprimirDetalles();  // Salida: Título: 1984, Autor: George Orwell
?>
```

- **Propiedades**: Son variables asociadas a una clase. En el ejemplo anterior, `$titulo` y `$autor` son propiedades de la clase `Libro`.
- **Métodos**: Son funciones asociadas a una clase. En el ejemplo, `imprimirDetalles` es un método de la clase `Libro`.
- **$this**: Una pseudo-variable que se refiere al objeto actual. Es útil para acceder a propiedades y métodos desde dentro de la clase.
- **Encapsulamiento**: Es el acto de agrupar datos (propiedades) y métodos que operan sobre esos datos en una única entidad (clase) y controlar el acceso a esos datos. En PHP, podemos usar modificadores como `public`, `private` y `protected` para controlar el acceso.

```php
<?php
class EjemploEncapsulamiento {
    private $datoPrivado = "Soy privado";

    public function obtenerDato() {
        return $this->datoPrivado;
    }
}
$ejemplo = new EjemploEncapsulamiento();
echo $ejemplo->obtenerDato();  // Salida: Soy privado
// echo $ejemplo->datoPrivado;  // Error: Cannot access private property
?>
```

- Herencia: Permite crear una nueva clase basada en una clase existente, heredando sus propiedades y métodos.

```php
<?php
/**
 * Clase base.
 */
class Vehiculo {
    public $color;

    public function describir() {
        return "Este vehículo es de color $this->color.";
    }
}
/**
 * Clase derivada.
 */
class Auto extends Vehiculo {
    public $marca;

    public function describir() {
        return parent::describir() . " Y es un $this->marca.";
    }
}
$miAuto = new Auto();
$miAuto->color = "rojo";
$miAuto->marca = "Toyota";
echo $miAuto->describir();  // Salida: Este vehículo es de color rojo. Y es un Toyota.
?>
```

- **Polimorfismo**: La capacidad de tratar objetos de diferentes clases de manera uniforme. En PHP, el polimorfismo se logra principalmente a través de la herencia y las interfaces.

```php
<?php
interface Forma {
    public function area();
}
class Circulo implements Forma {
    private $radio;

    public function __construct($radio) {
        $this->radio = $radio;
    }

    public function area() {
        return 3.14 * $this->radio * $this->radio;
    }
}
class Cuadrado implements Forma {
    private $lado;

    public function __construct($lado) {
        $this->lado = $lado;
    }

    public function area() {
        return $this->lado * $this->lado;
    }
}
function imprimirArea(Forma $forma) {
    echo $forma->area() . "\n";
}
imprimirArea(new Circulo(5));  // Salida: 78.5
imprimirArea(new Cuadrado(4)); // Salida: 16
?>
```