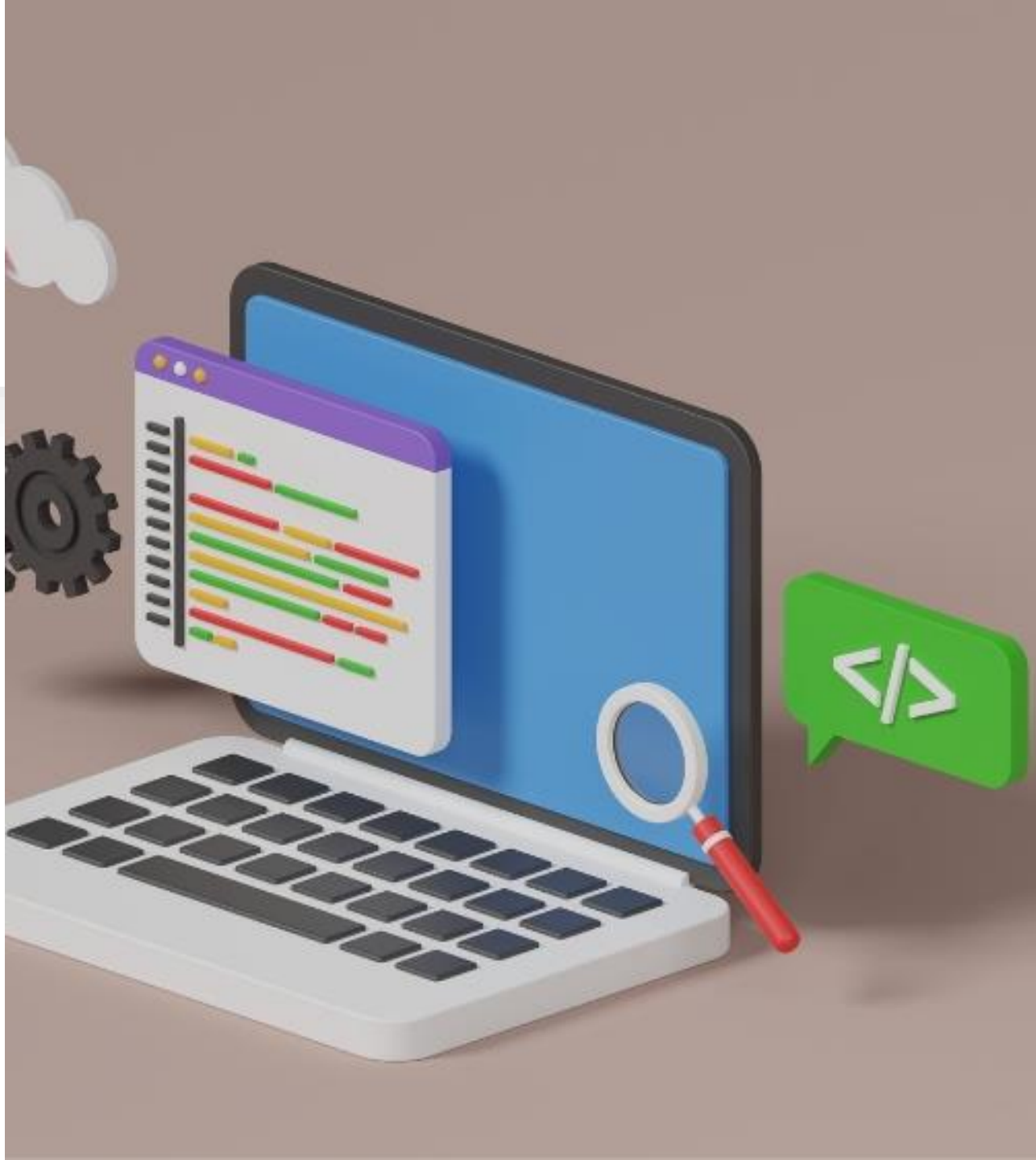
Programación Orientada a Objetos

Ing. Oriel A. Cedeño

POO

- POO significa Programación Orientada a Objetos.
- Es una forma de organizar y estructurar tu código creando "objetos" que representan cosas del mundo real.



Objetos

- Un objeto es algo con propiedades y acciones.

- Ejemplo: Un teléfono tiene propiedades como color y marca, y acciones como llamar y enviar mensajes.

Clases

- Una clase es como un molde para crear objetos.
- Define las propiedades y acciones que los objetos tendrán.
- Ejemplo: La clase "Teléfono" define que todos los teléfonos tienen un color, una marca y pueden llamar o enviar mensajes.

Cómo Crear una Clase en PHP

```php
class Telefono {
    public $color;
    public $marca;

    public function llamar() {
        echo "El teléfono está llamando";
    }

    public function enviarMensaje() {
        echo "Mensaje enviado";
    }
}
```

Propiedades

- Las propiedades son características de un objeto.
- Ejemplo: En la clase "Teléfono", $color y $marca son propiedades.

Métodos

- Los métodos son acciones que un objeto puede realizar.
- Ejemplo: En la clase "Telefono", llamar() y enviarMensaje() son métodos.

Cómo Crear un Objeto en PHP

```php
$miTelefono = new Telefono();
$miTelefono->color = "Negro";
$miTelefono->marca = "Samsung";
$miTelefono->llamar();
```

¿Qué es la Herencia?

- La herencia permite crear una nueva clase basada en una clase existente.
- Ejemplo: Una clase "Smartphone" puede heredar de la clase "Telefono".

```php
class Smartphone extends Telefono {
    public function navegarInternet() {
        echo "Navegando en internet";
    }
}

$miSmartphone = new Smartphone();
$miSmartphone->color = "Blanco";
$miSmartphone->marca = "iPhone";
$miSmartphone->navegarInternet();
```

¿Qué es el Polimorfismo?

- El polimorfismo permite que diferentes clases usen el mismo método de diferentes maneras.
- Ejemplo: Tanto la clase "Telefono" como la clase "Smartphone" pueden tener un método llamar(), pero cada uno funciona de manera diferente.

```php
class Telefono {
    public function llamar() {
        echo "Llamando desde un teléfono normal";
    }
}

class Smartphone extends Telefono {
    public function llamar() {
        echo "Llamando desde un smartphone";
    }
}

$miTelefono = new Telefono();
$miTelefono->llamar();

$miSmartphone = new Smartphone();
$miSmartphone->llamar();
```

Constructores y Destructores

- Constructor: Una función especial que se ejecuta cuando se crea un objeto.
- Destructor: Una función especial que se ejecuta cuando el objeto se destruye.

```php
class Telefono {
    public $color;
    public $marca;

    public function __construct($color, $marca) {
        $this->color = $color;
        $this->marca = $marca;
        echo "Teléfono creado: $this->marca de color $this->color";
    }

    public function __destruct() {
        echo "El teléfono $this->marca de color $this->color ha sido destruido";
    }
}

$miTelefono = new Telefono("Azul", "Nokia");
```

¿Qué es el Encapsulamiento?

- El encapsulamiento protege los datos de un objeto ocultando sus propiedades y permitiendo el acceso a través de métodos.
- Ejemplo: Usar métodos getter y setter para acceder a las propiedades privadas.

```php
class Telefono {
    private $color;
    private $marca;

    public function setColor($color) {
        $this->color = $color;
    }

    public function getColor() {
        return $this->color;
    }

    public function setMarca($marca) {
        $this->marca = $marca;
    }

    public function getMarca() {
        return $this->marca;
    }
}

$miTelefono = new Telefono();
$miTelefono->setColor("Verde");
$miTelefono->setMarca("LG");

echo "Color: " . $miTelefono->getColor();
echo "Marca: " . $miTelefono->getMarca();
```

¿Qué es la Abstracción?

- La abstracción simplifica la complejidad al mostrar solo los detalles esenciales y ocultar los innecesarios.
- Ejemplo: Una clase "Dispositivo" que tiene métodos como encender() y apagar(), sin mostrar cómo exactamente cada dispositivo realiza estas acciones.

```php
abstract class Dispositivo {
    abstract protected function encender();
    abstract protected function apagar();
}

class Telefono extends Dispositivo {
    public function encender() {
        echo "El teléfono está encendido";
    }

    public function apagar() {
        echo "El teléfono está apagado";
    }
}

$miTelefono = new Telefono();
$miTelefono->encender();
```