PRINCIPIOS SOLID

OPEN/CLOSED PRINCIPLE (OCP)

PRESENTED BY GEMA YÉBENES

¿Quésignifica Open/Closed?

El Principio Abierto/Cerrado (OCP) es uno de los cinco principios SOLID de la programación orientada a objetos. Su objetivo es crear sistemas de software que sean robustos y fáciles de mantener a lo largo del tiempo.

✅ Abierto a la extensión → podemos añadir nuevas funcionalidades.

🚫 Cerrado a la modificación → no necesitamos cambiar código existente.

Esto reduce errores y facilita el mantenimiento.

Este principio es crucial porque previene la introducción de errores en funcionalidades ya probadas y facilita enormemente la evolución y adaptación del software a nuevos requisitos sin efectos secundarios indeseados.

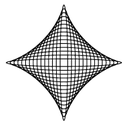
¿Por qué es importante OCP?

⚠ Sin OCP:
Cada vez que añadimos una funcionalidad debemos modificar código.
Riesgo de romper lo que ya funciona.

✅ Con OCP:
El código existente permanece estable.
Solo añadimos nuevas clases o componentes.
El sistema crece de manera ordenada.

Ejemplo SIN OCP

```
function calcularArea(figura) {
  if (figura.tipo === "cuadrado") {
    return figura.lado * figura.lado;
  } else if (figura.tipo === "círculo") {
    return Math.PI * figura.radio * figura.radio;
  }
}
```



⚠ **Cada nueva figura requiere modificar esta función.**

Ejemplo CON OCP

```
class Figura {
calcularArea() {
throw new Error("Este método debe ser
implementado por la subclase");
  }
}
class Cuadrado extends Figura {
  constructor(lado) {
    super();
    this.lado = lado;
  }

  calcularArea() {
    return this.lado * this.lado;
  }
}
```

✅ **Nuevas figuras → creo otra clase. No modifico lo ya escrito.**

¿Cómo se aplica en programación?

1

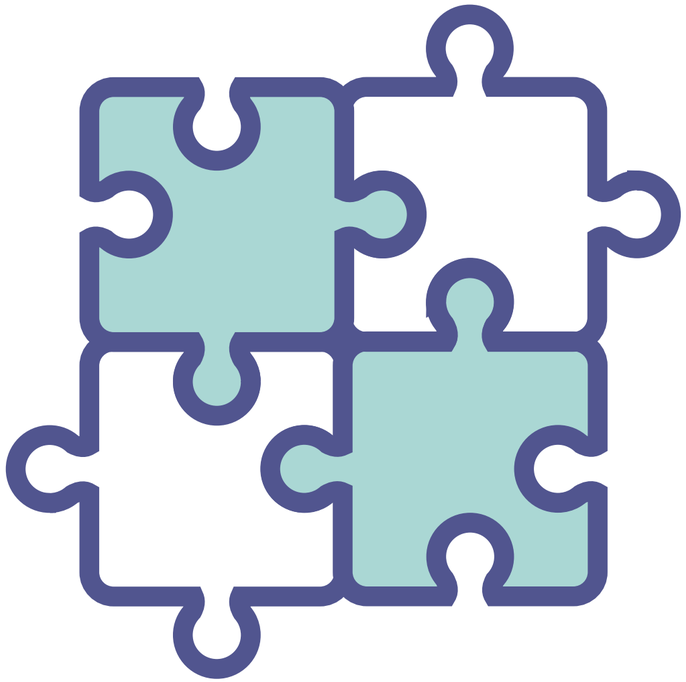
Definir Abstracciones

En lugar de programar algo concreto, creamos un “contrato” común (interface o clase abstracta) que dice lo que todas las clases deben hacer. Este contrato no explica el “cómo”, solo el “qué”.

2

Implementar Comportamientos Base

Las clases reales cumplen ese contrato y escriben su propia lógica. Así, cada clase concreta decide cómo funciona su versión del método.



3

Extender, No Modificar

Si necesitamos una nueva funcionalidad, no tocamos el código ya hecho. Creamos una nueva clase que siga el mismo contrato. De esta forma, lo viejo se mantiene estable y lo nuevo se añade sin romper nada.

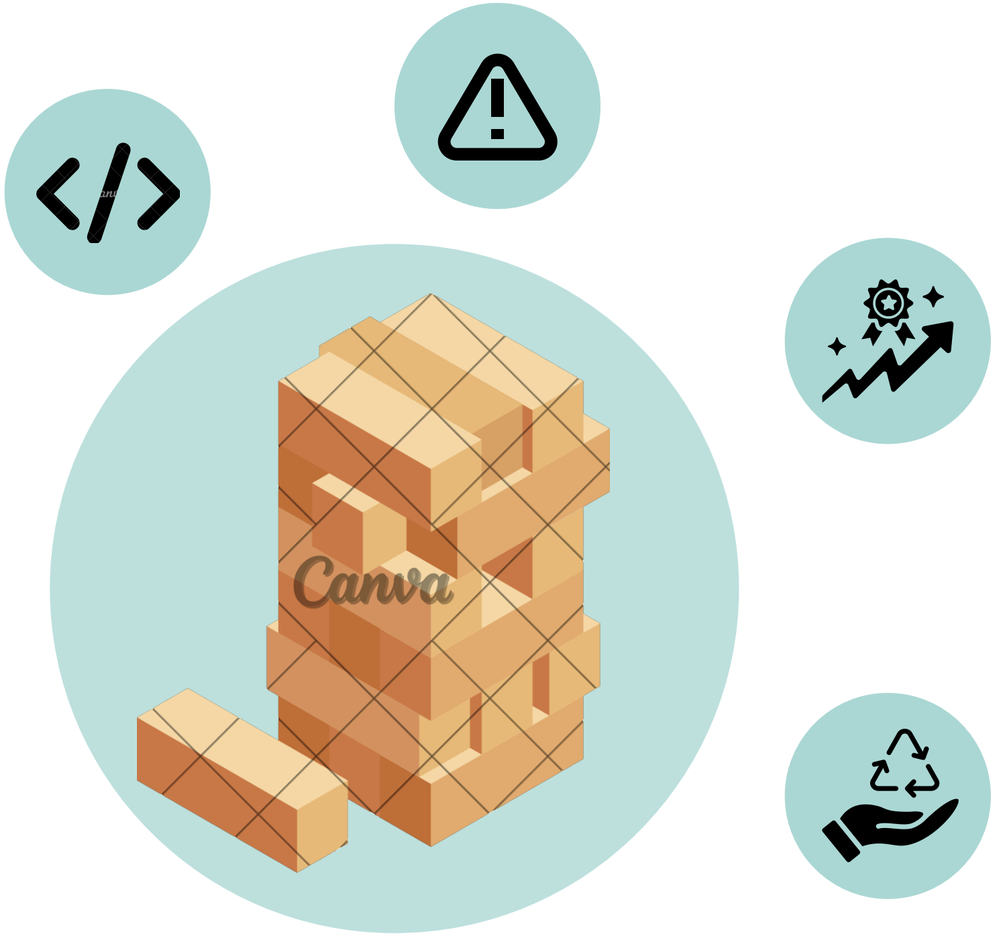
4

Inyección de Dependencias

El código principal trabaja contra el contrato (la abstracción) y no contra una clase específica. Cuando el programa corre, le “inyectamos” la implementación que queremos usar (la original o una nueva). Así podemos cambiar o ampliar el comportamiento sin modificar el código principal.

Beneficios de aplicar OCP

Código flexible y fácil de mantener.
Menos errores al extender funcionalidades.
Posibilidad de crecimiento sin modificar lo existente.
Reutilización de componentes y clases.



Thank you very much!

PRESENTED BY GEMA YÉBENES