

# PRINCIPIO DE SEGREGACIÓN DE INTERFACES (ISP) EN SOLID

Creando Interfaces Eficientes y Específicas

# EQUIPO 3

## LETRA 'I' DE 'S-O-L-I-D'



Issam



Matias



Jeni



Fran



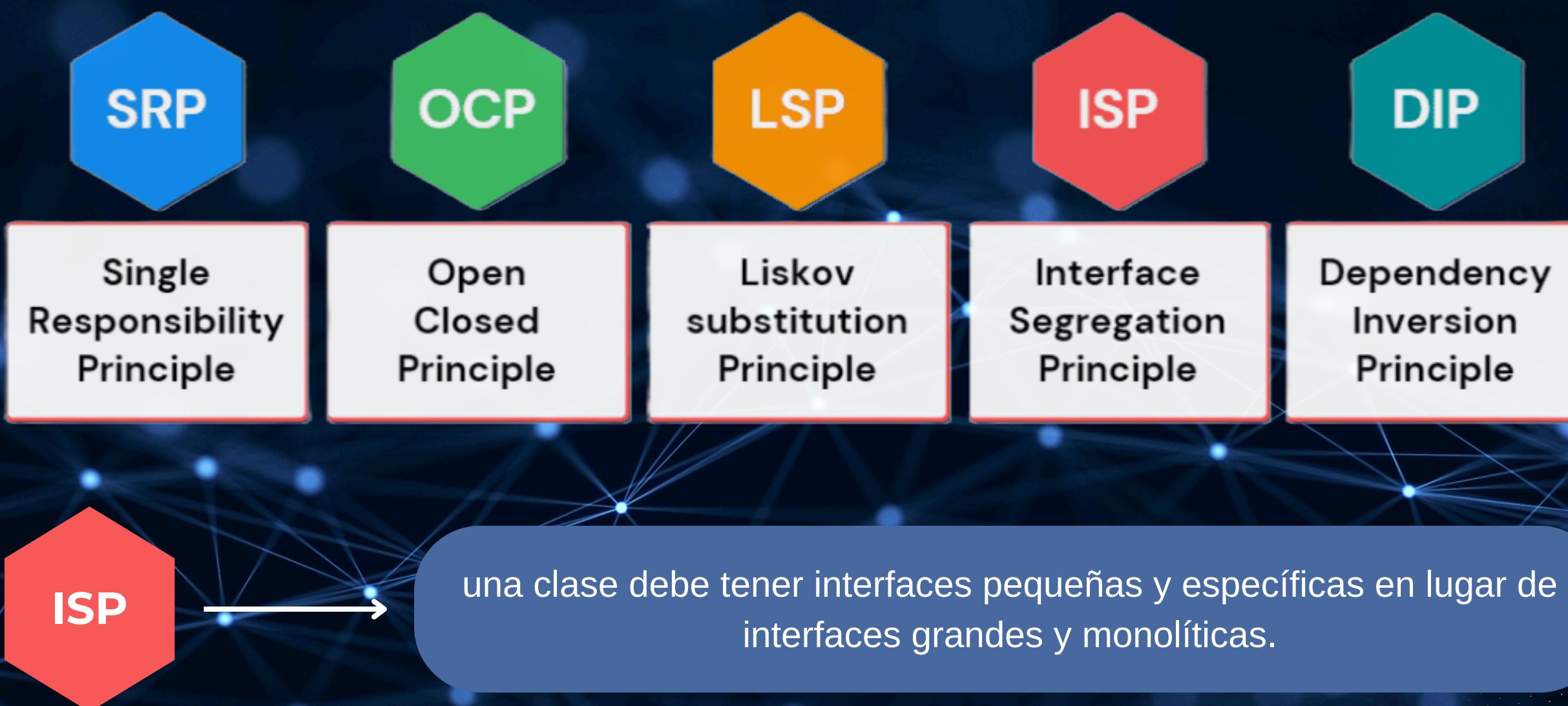
Rene



Jonathan

# INTRODUCCION PRINCIPIOS DE SOLID

SOLID es un **conjunto de principios para diseño de software** **mantenible y escalable.**



# ¿QUE ES ISP?

## ¿Qué es el Interface Segregation Principle de los principios SOLID?

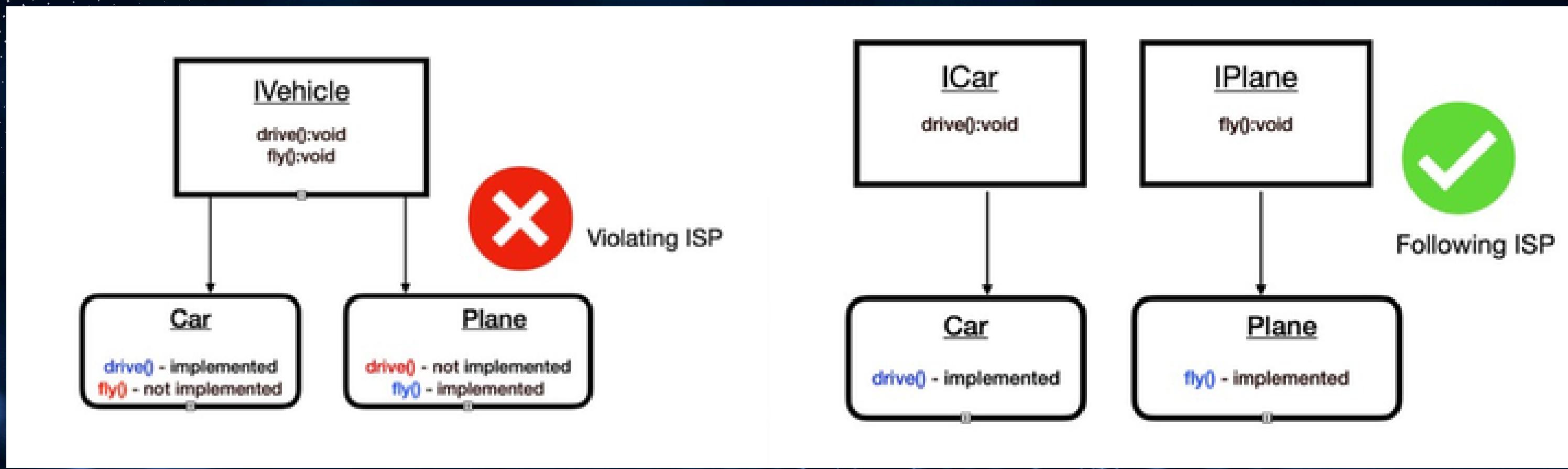
Este principio nos dice que una clase nunca debe extender de interfaces con métodos que no usa.

busca que las interfaces sean lo más pequeñas y específicas posible, cada clase con su método específico.

## ¿Cómo detectar que estamos violando el principio de segregación de interfaces?

- Estaremos violando este principio si tenemos clases que implementan métodos de interfaces que no se usan.
- Cuando definimos interfaces con parámetros que no se van a utilizar en todas las clases.
- Cuando tenemos interfaces muy grandes, ya que esto nos indica que estamos definiendo métodos no genéricos.

# EJEMPLO DE ISP



## Problemas de no respetar el Principio de Segregacion de Interfaces:

- Fat interfaces (interfaces gordas)
- Fuerzas a los objetos que se conforman al protocolo a implementar métodos que no tienen sentido para ellos

# APLICACIÓN DE ISP

Sistema que procesa pagos para diferentes métodos (tarjeta de crédito, PayPal, transferencia bancaria).

**Sin ISP**

```
1 <?php
2
3 interface PaymentProcessor {
4     public function processCreditCard($cardNumber);
5     public function processPayPal($email);
6     public function processBankTransfer($accountNumber);
7 }
```

**Con ISP**

```
1 <?php
2
3 interface CreditCardProcessor {
4     public function processCreditCard($cardNumber);
5 }
6
7 interface PayPalProcessor {
8     public function processPayPal($email);
9 }
10
11 interface BankTransferProcessor {
12     public function processBankTransfer($accountNumber);
13 }
```

# ¿CUÁLES SON LOS BENEFICIOS DEL ISP?

01

Reducción del  
Acoplamiento

02

Mejora de la  
Cohesión

03

Facilidad de  
Implementación

04

Facilidad de  
Pruebas

07

Promoción de Buenas  
Prácticas de Diseño

06

Mejor Comprensión  
del Código

05

Flexibilidad y  
Escalabilidad

S  
O  
L  
I  
D

# ISP EN RELACIÓN CON OTROS PRINCIPIOS SOLID

## ISP y SRP

- **Explicación:** Ambos se enfocan en la especialización.
- **SRP:** Cada clase tiene una sola responsabilidad.
- **ISP:** Las interfaces contienen solo lo que la clase necesita.

## ISP y OCP

- **Explicación:** ISP facilita la expansión sin modificar el código existente.
- **OCP:** Las clases son abiertas a la extensión y cerradas a la modificación.

## ISP y LSP

**Explicación:** ISP asegura que las subclases puedan reemplazar a las clases base sin romper el comportamiento esperado.

## ISP y DIP

**Explicación:** ISP ayuda a crear abstracciones adecuadas, permitiendo una mejor gestión de dependencias.

# CONCLUSIÓN

01

## Interfaces



**Definición Básica:** Principio de segregación de interfaces que promueve interfaces pequeñas, específicas y enfocadas.

02

## Cosas a evitar

- Mezclar conceptos
- Funcionalidades diversas
- Falta de cohesión
- Diseño poco claro
- Múltiples responsabilidades
- Métodos no relacionados
- Complejidad innecesaria
- Alto acoplamiento

03

## Soluciones

Interfaces pequeñas Propósito específico Fácil implementación  
Máxima flexibilidad

04

## Conclusión

ISP busca simplicidad, claridad y enfoque en el diseño de interfaces.

# ¡GRACIAS POR SU ATENCIÓN!

## CONTACTOS:



Issam



Matias



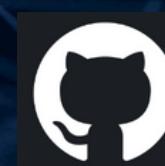
Fran



René



Jeni



Jonathan