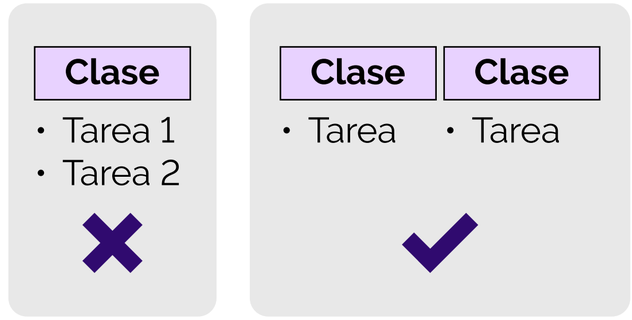
Principios

S

Principio de Responsabilidad Única (Single Responsibility Principle): Una clase debe tener solo una razón para cambiar, lo que significa que debe tener solo una tarea o responsabilidad.

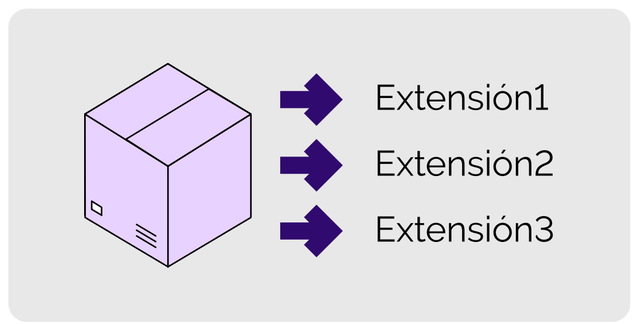
Beneficio: Simplifica el entendimiento y la prueba de la clase.



O

Principio de Abierto/Cerrado (Open/Closed Principle): Las entidades de software como clases, módulos y funciones deben estar abiertas para extensión, pero cerradas para modificaciones.

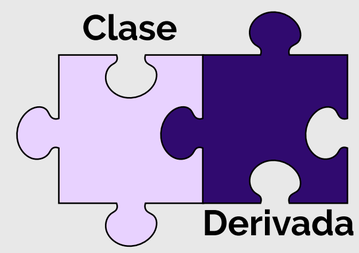
Beneficio: Fomenta la extensibilidad del código sin alterar el código existente, reduciendo el riesgo de errores.



L

Principio de Sustitución de Liskov (Liskov Substitution Principle): Las clases derivadas deben ser completamente sustituibles por sus clases base.

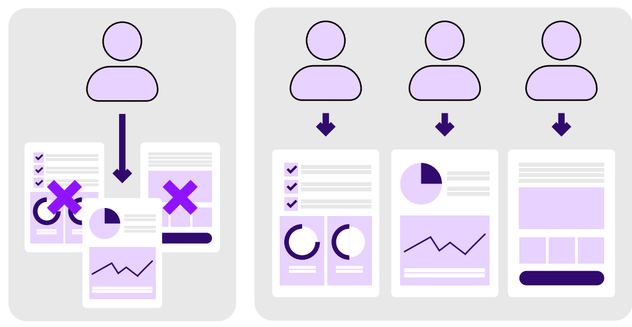
Beneficio: Mejora la compatibilidad entre las subclases y sus bases, lo que a su vez aumenta la reutilización de código.

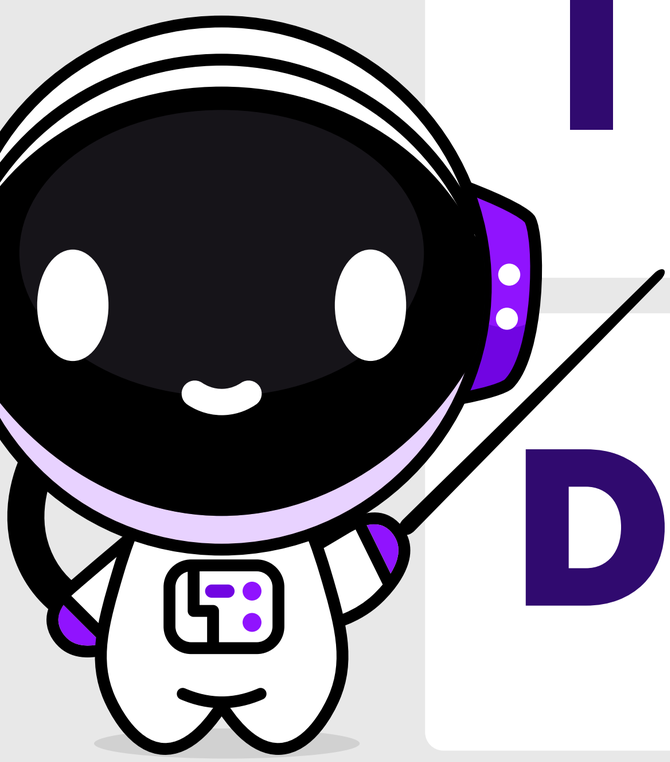


I

Principio de Segregación de Interfaz (Interface Segregation Principle): Los clientes no deben ser forzados a depender de interfaces que no utilizan.

Beneficio: Evita la dependencia de métodos innecesarios, reduciendo el acoplamiento.

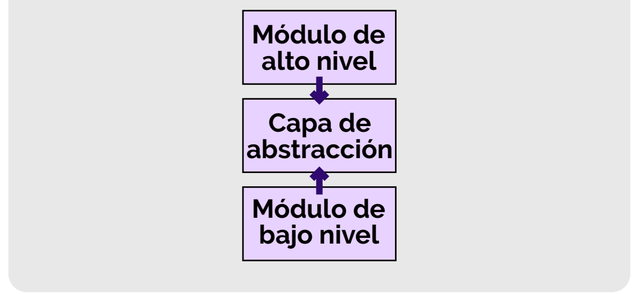




D

Principio de Inversión de Dependencias (Dependency Inversion Principle): Los módulos de alto nivel no deberían depender de módulos de bajo nivel. Ambos deberían depender de abstracciones.

Beneficio: Disminuye la dependencia entre módulos concretos, permitiendo una mayor flexibilidad y reutilización.



FUENTE: Diseño Ágil con Patrones de Diseño y Principios SOLID - *Robert C. Martin*

Curso disponible

Recursos

@Fernando_Her85

@DevTalles

www.devtalles.com