SOLID

Acronimo de 5 principios de diseño que hacen el diseño orientado a objetos mas entendible, flexible, mantenible.

Single-responsability principle → Principio de responsabilidad única
↳ Quiere decir que una clase no debería haber mas de una razón para modificar la clase - Toda clase debería tener una unica responsabilidad.

Open-Closed Principle => Las entidades deben ser abiertas para extención, cerradas para modificación

Liskov sostitution principle => Las funciones que utilizan punteros o referencias a clases base deben poder utilizar objetos de clases derivadas sin saberlo.

Interface segregation principle → No se debe obligar a los clientes a depender de interfaces que no usan.

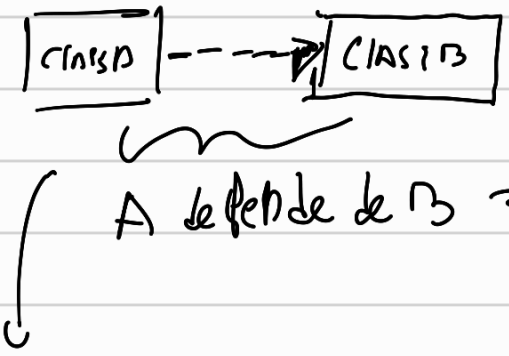
DIP → Principio de inversión de dependencias

Malos diseños => por Interdependencias indeseadas entre clases

- Un cambio afecta muchas partes del programa.
- Fragilidad de partes del programa al dejar de funcionar ante un cambio.
- Inmovilidad al no poder desencadenar clases del programa actual.

=> Clases de alto nivel no deben depender de clases de bajo nivel, ambas deben depender de abstracciones.

→ Los detalles también deben depender de ellas.



A depende de B => A usa responsabilidades de B.
↳ más alto nivel.

- Da igual que A que si introduzco cambios en B, probablemente tenga que cambiar A
- La reutilización de la clase A se ve condicionada en que también se necesite B.