Solid Principles

⇒ S.O.L.I.D

S → Single Responsibility principle
O → Open - closed - principle
L → Liskov substitution principle
I → Interface segregation principle
D → Dependancy Inversion principle

⇒ Benefits of SOLID Principle

- Loose - coupling
- Code maintainability
- Dependency management

⇒ Single Responsibility Principle

A class should have only one & Only one reason to change, meaning that a class should have only one job

⇒ Open for Extension, closed for Modification

You should be able to extend a classes behaviour, without modifying it.

~~⇒ Liskov Solution~~

⇒ Liskov Substitution

"Derived classes must be suitable for their base classes."

* If S is a subtype of T, then objects of type T may be replaced (or substituted) with objects of type S.

T is parent & S is child.

⇒ Interface Segregation

Make fine grained interface that are client specific

⇒ Dependency Inversion

High-level modules should not depend on low-level modules. Both should depend on abstraction.

Abstraction should not depend on details.
Details should depend on abstractions