CREATE A SOLID APP

Single-Responsibility Principle (SCP)

A class should have one, and only one, reason to change.

An object should have only one responsibility (a reason to change). It is supposed to simplify modifications. The cohesion is stronger, dependency coupling is looser and the code is less complex.

Open-Closed Principle (OCP)

Entities should be open for extension, but closed for modification.

Entity can be extended by adding what is needed, but it can never be modified. This significantly reduces the risk of breaking existing functionality and provides a looser coupling.

Liskov Substitution Principle (LSP)

Subtypes must be substitutable for their base types.

The principle says that it is possible to use base type and get a correct result as the outcome. It can be said that the LSP confirmes abstractions are correct.

Interface Segregation Principle (ISP)

Classes that implement interfaces, should not be forced to implement methods they do not use.

Big interfaces should be splitted into smaller ones so there are no methods that are not used implemented. Classes know only about methods related to them providing decoupling and easier modifications.

Dependency Inversion Principle (DIP)

High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.

This reduces dependencies in the code modules. Code is much more easier to maintain if abstractions and details are isolated from each other