SOLID Principles

Friday, May 10, 2024 1:38 PM

Single Responsibility Principle: A class should have only one reason to change. This means that a class should have a single responsibility and should not be responsible for multiple unrelated tasks.

Benefits

The Less code we put, the more hard is to mess up.

Open/Closed Principle: Software entities should be open for extension but closed for modification. This means that classes should be designed in a way that allows them to be extended without requiring modifications to the original code. (Abstract Class Shape (Calculate Area, with Rectangle and Circle as child class).

How

This can be done with the Polymorphic way, where an interface can be used and you create your own concrete implementation (Hierarchical Inheritance)

Benefits

TDD, we can test the class once and disable the execution of the tests once it's already tested.

Liskov Substitution Principle: Subtypes should be substitutable for their base types. This means that any object of a superclass should be able to be replaced by an object of its subclass without affecting the correctness of the program. (Method overriding, dynamic polymorphism).

OR

Definition 2

A subclass should behave in such a way that it will not cause problems when used instead of the superclass.

Interface Segregation Principle: Clients should not be forced to depend on interfaces they do not use. This means that interfaces should be designed to be small, focused, and specific to the needs of the clients that use them.

Dependency Inversion Principle: High-level modules should not depend on low-level modules. Both should depend on abstractions. This means that the dependencies between modules should be inverted so that high-level modules depend on abstractions, rather than on low-level implementation details.

References

Link from Microsoft