The SOLID principles

They are a set of design principles that are intended to guide software developers in writing maintainable, scalable, and flexible code. These principles were introduced by Robert C. Martin (also known as Uncle Bob) and have become fundamental concepts in object-oriented design.

The SOLID acronym stands for:

S - Single Responsibility Principle (SRP):

A class should have only one reason to change.

It states that a class should have only one responsibility or only one job. If a class has more than one responsibility, it becomes tightly coupled and harder to maintain.

O - Open/Closed Principle (OCP):

Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification.

It suggests that classes should be designed in a way that new functionality can be added without altering existing code. This is typically achieved using inheritance, polymorphism, or composition.

L - Liskov Substitution Principle (LSP):

Objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program.It emphasizes that subtypes must be substitutable for their base types without altering the correctness of the program. In other words, derived classes should be able to extend base classes without changing their behavior.

I - Interface Segregation Principle (ISP):

Clients should not be forced to depend on interfaces they do not use.

It states that clients should not be forced to implement interfaces they don't need. Instead of having one large interface, it's better to have smaller, specific interfaces tailored to the needs of clients.

D - 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.

It suggests that the high-level modules and low-level modules should depend on abstractions rather than concrete implementations. This reduces coupling and makes the code more flexible and easier to maintain.

These principles are not strict rules but rather guidelines to help developers create more modular, flexible, and maintainable software systems. Following SOLID principles can lead to cleaner codebases, easier maintenance, and better scalability in the long term.