The SOLID Principles are five principles of Object-Oriented class design. They are a set of rules and best practices to follow while designing a class structure.

Let's look at each principle one by one. Following the SOLID acronym, they are:

* The **S**ingle Responsibility Principle
* The **O**pen-Closed Principle
* The **L**iskov Substitution Principle
* The **I**nterface Segregation Principle
* The **D**ependency Inversion Principle

The Single-Responsibility Principle (SRP):

The single responsibility principle states that every module or class should have responsibility over a single part of the functionality provided by the software, and that responsibility should be entirely encapsulated by the class, module or function. All its services should be narrowly aligned with that responsibility.

The **O**pen-Closed Principle (OCP):

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

The **L**iskov Substitution Principle (LSP):

Subtypes must be substitutable for their base types.

Here make sure that inheritance is about behaviour, not about data, the contract of base types is adhered to the required concept.

The **I**nterface Segregation Principle (ISP):

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

The Dependency Inversion Principle (DIP):

The Dependency Inversion Principle (DIP) tells us that the most flexible systems are those in which source code dependencies refer only to abstractions, not to concretions.