**SOLID** is the composition of the five principles of object-oriented design and programming, which were first introduced by Robert C. Martin.

The SOLID principles provide a set of guidelines for designing and building robust, maintainable, and scalable software systems. They help to improve the quality of code and reduce the risk of bugs and errors.

 These principles are:

1. **Single Responsibility Principle (SRP):** The Single Responsibility Principle (SRP) states that a class should have only one reason to change. This means that each class should have a single, well-defined responsibility and that this responsibility should be encapsulated by the class.
2. **Open/Closed Principle (OCP):** The Open/Closed Principle (OCP) states that a class should be open for extension but closed for modification. This means that a class should be designed in such a way that its behaviour can be extended without changing its source code.
3. **Liskov Substitution Principle (LSP):**The Liskov Substitution Principle (LSP) states that objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program. This means that subclasses should be able to extend the behaviour of their superclasses while preserving the contract established by the superclass.
4. **Interface Segregation Principle (ISP):** The Interface Segregation Principle (ISP) states that clients should not be forced to depend on interfaces they do not use. This means that an interface should be designed to be small and focused, with only the methods that are necessary for the client to use.
5. **Dependency Inversion Principle (DIP):**The Dependency Inversion Principle (DIP) states that high-level modules should not depend on low-level modules. Both should depend on abstractions. On the other way, a class should depend on abstractions, not on concrete implementations. This helps to reduce coupling between different parts of the system and makes it easier to change one part of the system without affecting other parts.

By following the SOLID principles, developers can build software systems that are robust, maintainable, and scalable, with a reduced risk of bugs and errors. Additionally, by following these principles, developers can create a modular design that makes it easier to change one part of the system without affecting other parts of the system and to write code that is easy to test and maintain.