SOLID principles are a set of five design principles that help software developers create more maintainable, flexible, and scalable software. The SOLID principles are:

1. **Single Responsibility Principle (SRP):** A class should have only one reason to change. In other words, a class should have only one responsibility or job.
2. **Open/Closed Principle (OCP):** Software entities (classes, modules, functions, etc.) should be open for extension but closed for modification. In other words, the behavior of a module can be extended without modifying its source code.

This encourages the use of interfaces and abstract classes to allow for easy extension without altering existing code.

1. **Liskov Substitution Principle (LSP):** Subtypes must be substitutable for their base types without altering the correctness of the program. In other words, objects of a superclass should be replaceable with objects of a subclass without affecting the correctness of the program.
2. **Interface Segregation Principle (ISP):** A client should never be forced to implement interfaces it does not use. This principle encourages creating small, specific interfaces rather than large, general-purpose ones.
3. **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 principle promotes the use of interfaces and abstract classes to decouple high-level and low-level modules.