Los principios SOLID son un conjunto de cinco principios de diseño de software que se consideran fundamentales para desarrollar sistemas de software flexibles, mantenibles y de alta calidad. Estos principios fueron introducidos por el ingeniero de software Robert C. Martin, también conocido como "Uncle Bob". Cada uno de los principios se centra en un aspecto específico del diseño de software y juntos forman una guía para escribir código limpio y modular.

A continuación, se describen brevemente cada uno de los principios SOLID:

1. Principio de Responsabilidad Única (Single Responsibility Principle - SRP): Este principio establece que una clase debe tener una sola razón para cambiar. En otras palabras, una clase debe tener una única responsabilidad o función en el sistema. Esto promueve la cohesión y evita que una clase tenga demasiadas responsabilidades, lo que dificultaría el mantenimiento y la reutilización del código.

2. Principio de Abierto/Cerrado (Open/Closed Principle - OCP): Según este principio, las entidades de software (clases, módulos, funciones, etc.) deben ser abiertas para su extensión pero cerradas para su modificación. En lugar de modificar el código existente, se debe permitir la extensión del comportamiento a través de la adición de nuevas clases o funciones. Esto fomenta el diseño modular y evita cambios no deseados en el código existente.

3. Principio de Sustitución de Liskov (Liskov Substitution Principle - LSP): El principio establece que los objetos de una clase derivada deben poder sustituirse por objetos de su clase base sin alterar el correcto funcionamiento del programa. Esto significa que las clases derivadas deben ser sustituibles por las clases base sin introducir errores o comportamientos inesperados. Cumplir este principio garantiza una correcta interoperabilidad entre objetos y favorece la reutilización de código.

4. Principio de Segregación de Interfaces (Interface Segregation Principle - ISP): Este principio indica que los clientes no deben depender de interfaces que no utilizan. En lugar de tener interfaces monolíticas y sobrecargadas con muchos métodos, se deben crear interfaces específicas para cada cliente. Esto evita la dependencia innecesaria y garantiza que los clientes solo utilicen los métodos que necesitan.

5. Principio de Inversión de Dependencias (Dependency Inversion Principle - DIP): El principio establece que los módulos de alto nivel no deben depender de módulos de bajo nivel, sino que ambos deben depender de abstracciones. Además, las abstracciones no deben depender de los detalles, sino que los detalles deben depender de las abstracciones. Este principio promueve el desacoplamiento entre módulos y facilita la flexibilidad y el intercambio de componentes.

Estos principios SOLID se consideran fundamentales para lograr un diseño de software flexible, mantenible y escalable, y son ampliamente utilizados en el desarrollo de software orientado a objetos.