SOLID

Principio de Responsabilidad Única: Cada clase debe tener una única razón para cambiar. En otras palabras, una clase debe tener una única responsabilidad o función en el sistema. Esto facilita el mantenimiento y evita efectos en cascada cuando se realizan cambios.

Principio de Abierto/Cerrado: Las entidades del software (clases, módulos, etc.) deben estar abiertas para extensión pero cerradas para modificación. Esto significa que se deben poder agregar nuevas funcionalidades sin modificar el código existente, fomentando la reutilización y la flexibilidad.

Principio de Sustitución de Liskov : Las instancias de una clase base deben poder ser reemplazadas por instancias de sus clases derivadas sin afectar la integridad del programa. En resumen, las clases derivadas no deben romper el comportamiento esperado de la clase base.

Principio de Segregación de Interfaces : Los clientes no deben estar obligados a depender de interfaces que no utilizan. Esto se traduce en crear interfaces específicas para las necesidades de cada cliente en lugar de interfaces monolíticas que contienen métodos irrelevantes para algunas clases.

Principio de Inversión de Dependencia: Los módulos de alto nivel no deben depender de los módulos de bajo nivel; ambos deben depender de abstracciones. Además, las abstracciones no deben depender de los detalles; los detalles deben depender de las abstracciones. Esto promueve la flexibilidad y la facilidad de cambio al invertir la dirección de las dependencias.