Principios S.O.L.I.D.

Single Responsibility (Unica Responsabilidad)

Mantener la cohesión, cada clase, método debería tener una sola razón de existir. Se debe reunir las cosas que cambian por las mismas razones, y separar aquellas que cambian por razones diferentes. En el momento en el que una clase adquiere más responsabilidad pasa a estar acoplada.

Open – Close (Abierto – Cerrado)

En otras palabras: las clases que usas deberían estar abiertas para poder extenderse y cerradas para modificarse.

Liskov Sustitution (Sustitución)

Significa que los objetos deben poder ser reemplazados por instancias de sus subtipos sin alterar el correcto funcionamiento del sistema o lo que es lo mismo: si en un programa utilizamos cierta clase, deberíamos poder usar cualquiera de sus subclases sin interferir en la funcionalidad del programa.

Interface Segregation (Segregación de la interfaz)

Hacer interfaces que sean específicas para un tipo de cliente. Es preferible contar con muchas interfaces que definan pocos métodos que tener una interface forzada a implementar muchos métodos a los que no dará uso.

Dependency Inversion (inversión de dependencia)

Los módulos de alto nivel no deberían depender de módulos de bajo nivel, ambos deberían depender de abstracciones. Las abstracciones no deberían depender de los detalles, si no al revés. El objetivo de este es alcanzar un bajo acoplamiento.