SOLID Object Oriented Programming Principles

Aplicaciones Móviles

Mario Alejandro Gil Lázaro

15 de octubre de 2016

S ingle responsibility principle
O pen-closed principle
L iskov substitution principle
I nterface segregation principle
D ependency inversion principle

Índice general

1 Single Responsibility Principle

> Una clase debe tener una y sólo una razón para cambiar, es decir, una clase debe tener sólamente un trabajo.

Si una clase se encarga de varios trabajos a la vez, modificar su comportamiento se vuelve una tarea compleja. Esto reduce la flexibilidad y la mantenibilidad del código. Se trata de promover la modularidad del código al distribuir los trabajos entre distintas clases especializadas.

2 Open-Closed Principle

> Los objetos y entidades deben estar abiertos para su extensión, pero cerrados a la modificación.

3 Liskov Substitution Principle

> Sea $\phi(x)$ una propiedad comprobable acerca de los objetos x de tipo T. Entonces $\phi(y)$ debe ser verdad para los objetos y del tipo S donde S es un subtipo de T.

Esto dicta que cada subclase debe ser sustituible por su clase base o, de otro modo, que puede ser usada como su clase padre.

4 Interface Segregation Principle

> Un cliente nunca debe ser obligado a utilizar una interfaz que no usa, o los clientes no deben ser forzados a depender de métodos que no usan.

Las interfaces deben planearse de modo que nuestras clases no tengan que implementar métodos que no serán usados. Esto se logra implementando interfaces más generales, por ejemplo.

5 Dependency inversion principle

> Las entidades deben depender de abstracciones, no te implementaciones. El módulo de alto nivel no debe depender del módulo de bajo nivel, ambos deben depender de abstracciones.

Este principio trata de utilizar las propiedades que propone el Principio de Sustitución de Liskov para programar dependiendo de una interfaz que abstraerá casos de implementación específicos, haciendo nuestro código más robusto.