
12
OREO
Principios
SOLID
11
10

¿Que son?

Un Conjunto de Principios aplicables a la Programación Orientada a Objetos que, usados correctamente, ayudan a escribir un Software de Calidad, con Código que será más fácil de **Leer**, **Testear** y **Mantener**.

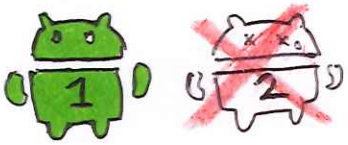
La Idea

Robert Martin (AKA **Uncle Bob**) escribió de esto en los 2000 y se convirtió en uno de los acrónimos más famosos de la programación.

Los Principios

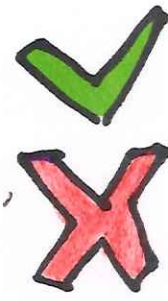
(S) Principio de Responsabilidad Única

{ Básicamente dice que una clase debe encargarse solo de una funcionalidad si ya es necesario que se realicen más acciones entonces deben de crearse otras clases y llamarlas para que así el código sea más entendible y escalable. }



(O) Principio de ser Abierto y Cerrado

[Es decir que el código que ya ha sido escrito no debe modificarse, porque ya pasó las pruebas unitarias, por lo que lo único que debe de hacer es lograr expandirse y que sirva de guía para heredar a otra clase para agregar más funcionalidades, como se dice día tras día



" SI FUNCIONA NO TOQUES!! "]

(L) Principio de Sustitución de Liskov

Dice que si se tiene una clase base y de esta se extiende una clase hija, esta debe de usarla sin alterar nada de la clase base, si no se utiliza nada de la clase entonces no debería de usarse.

```
Class ONE

Class TWO EXTENDS ONE
```

(I) Principio de Integración de Diseño

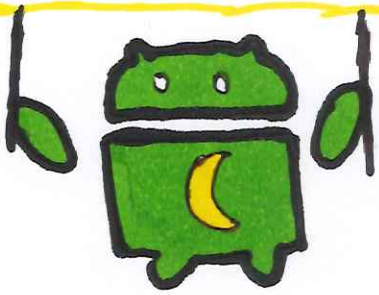
Se debe de dividir una interfaz en interfaces más pequeños para no tener metodos llamadas sin que en verdad se estén utilizando, esto aplica mas que todo con los clases padres que tienen mas metodos por lo que es mejor dividirlo tal como dice el Principio.

IPTERFAZ
INTERFAZ
INTERFAZ

(D) Principio de Inversión de Dependencias

Establece que el modulo de alto nivel no debe depender del modulo de bajo nivel, sino que deben depender de abstracciones, normalmente se realiza esto pasando objetos como parametros del constructor de una clase, llamamos esta acción

Inyección de Dependencias



Beneficios de los Principios

- Software más flexible
- Entender mejor los arquitecturas

- Simplifican la creación de tests

Notas Importantes

1. Unos principios **NO** pueden existir sin los otros: Estan conectados entre si es decir que a veces para cumplir uno se debe cumplir el otro.

2. Al cumplir un principio puede que se incumpla otro: Esto es lo mas dificil de aceptar y entender, muchas veces es imposible cumplir todos los principios a la vez, ya que al aplicar un principio se puede dar la espalda a otro, el consejo es, NO obsesionarse con eso al final lo importante es poder comprender el potencial de cada principio.

The Conclusión End

El uso de todos estos principios permite al desarrollador crear un software de calidad facilmente, escalable y entendible por todos, tolerante a tests unitarios y facil de mantener debido a la facilidad de sustitución de cada uno de los elementos que componen el sistema.