Principios de SOLID

Utilizando correctamente el paradigma de la POO

Autor: Luz Elena Galindo Gamboa

¿Por qué usar SOLID?

S.O.L.I.D. es un acrónimo de los 5 principios de diseño de POO. Estos principios permiten a los desarrolladores de software:

- Diseñar software fácil de mantener y extender
- Evitar el código sucio
- Facilitar la refactorización
- Facilitar el desarrollo ágil

Seguir los principios de SOLID nos permitirán evitar las malas prácticas y podremos desarrollar utilizando el paradigma de la POO con el verdadero propósito por el que fue creado.

Autor: Luz Elena Galindo Gamboa

Principios de S.O.L.I.D.

S - Single Responsability (Responsabilidad Única)

O - Open/Closed (Abierto/Cerrado)

L - Liskov Substitution (Sustitución de Liskov)

I - Interface Segregation (Segregación de Interfaces)

D - Dependency Inversion (Inversión de Dependencias)

¿Cómo detectar que se está violando el principio?

¿Cómo corregirlo?

Autor: Luz Elena Galindo Gamboa

Single Responsability - Responsabilidad Única

Una clase debe tener una y solo una única causa por la cual puede ser modificada, por lo que la clase debería tener una única responsabilidad.

Clases con muchos métodos para diferentes tareas, métodos que dependen de otros métodos en la misma clase.

Separar en diferentes clases las diferentes responsabilidades de un proceso.

Autor: Luz Elena Galindo Gamboa

Open/Closed - Abierto/Cerrado

Las entidades deben estar abiertas a extensiones, pero cerradas a las modificaciones. Para nuevos requisitos debemos extender el comportamiento añadiendo código sin modificar el existente.

Mucha lógica condicional para realizar determinada acción, abuso del uso de if/else y switch.

Utilizar clases derivadas o interfaces para extender (sobreescribir) funcionalidad, haciendo uso del polimorfismo para sustituir clases.

Autor: Luz Elena Galindo Gamboa

Liskov Substitution - Sustitución de Liskov

Los objetos de tipos diferentes que derivan de una misma clase base deben ser intercambiables donde el tipo base esté implementado. Nos indica la mejor forma de crear jerarquías de herencia entre clases.

Implementación de la clase base a través de clases derivadas con métodos particulares que no comparten las demás clases derivadas.

Identificar los miembros que comparten las clases derivadas, agregarlos a la clase base y extender su funcionalidad en cada clase derivada.

Autor: Luz Elena Galindo Gamboa

Interface Segregation - Segregación de Interfaces

Los clientes no deben ser forzados a implementar una interface que no usan ni depender de métodos que no usan. Debemos segregar las operaciones en pequeñas interfaces.

Clases que al implementar una interfaz dejan métodos vacíos. Clases con métodos que no son de la interfaz y pueden ser usadas en otras clases.

Reorganizar las interfaces en otras más pequeñas, recordando que una clase puede implementar varias interfaces.

Autor: Luz Elena Galindo Gamboa

Dependency Inversion - Inversión de dependencias

> Los módulos de alto nivel no deben depender de los módulos de bajo nivel. Ambos deben depender de las abstracciones.
> Las abstracciones no deben depender de los detalles. Los detalles deben depender de las abstracciones.

Uso de instancias a clases dentro de otras clases. Se complican los tests de forma aislada por las dependencias.

Eliminar las instancias de las clases, inyectar las dependencias en el constructor de la clase.

Autor: Luz Elena Galindo Gamboa

Fuentes

Autor: Luz Elena Galindo Gamboa

Fuentes

https://scotch.io/bar-talk/s-o-l-i-d-the-first-five-principles-of-object-oriented-design

https://itnext.io/solid-principles-explanation-and-examples-715b975dcad4

https://desarrolloweb.com/manuales/programacion-orientada-objetos-dotnet.html

https://desarrolloweb.com/articulos/principio-reponsabilidad-unica-I-dotnet.html

https://desarrolloweb.com/articulos/principio-reponsabilidad-unica-II-dotnet.html

https://desarrolloweb.com/articulos/principio-open-closed-I-dotnet.html

https://desarrolloweb.com/articulos/principio-open-closed-II-dotnet.html

https://desarrolloweb.com/articulos/principio-sustitucion-liskov-dotnet.html

https://desarrolloweb.com/articulos/principio-de-segregacion-interfaces-dotnet.html

https://devexperto.com/principio-de-inversion-de-dependencias/

Autor: Luz Elena Galindo Gamboa