Universidad Autónoma de Nuevo León
Facultad de Ciencias Físico Matemáticos

Licenciado en Seguridad en Tecnologías de Información

Diseño Orientado a Objetos

S.O.L.I.D

Tarea

Lic. Miguel Ángel Salazar Santillán

Grupo: 007 Matrícula: 1732645

Diana Elizabeth Díaz Rodríguez

31/ 03/ 2017

SOLID es un acrónimo de los primeros cinco principios de diseño orientado a objetos (OOD) de Robert C. Martin, conocido popularmente como el tío Bob .

SOLID (Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion) es un acrónimomnemónico introducido por Robert C. Martin a comienzos de la década del 2000 que representa cinco principios básicos de la programación orientada a objetos y el diseño. Cuando estos principios se aplican en conjunto es más probable que un desarrollador cree un sistema que sea fácil de mantener y ampliar con el tiempo.3 Los principios SOLID son guías que pueden ser aplicadas en el desarrollo de software para eliminar código sucio provocando que el programador tenga que refactorizar el código fuente hasta que sea legible y extensible. Debe ser utilizado con el desarrollo guiado por pruebas o TDD, y forma parte de la estrategia global del desarrollo ágil de software y desarrollo adaptativo de software.

SOLID es un acrónimo de los primeros cinco principios de diseño orientado a objetos (OOD) de Robert C. Martin, conocido popularmente como el tío Bob .

Estos principios, cuando se combinan juntos, hacen que sea fácil para un programador para desarrollar software que son fáciles de mantener y extender. También hacen que sea fácil para los desarrolladores de código para evitar los olores, refactorizar fácilmente el código, y son también una parte del desarrollo ágil de software o de adaptación.

Cuando se inicia la pregunta de *cómo* , que es un poco como mirar una carrera de maratón y preguntándose como se llega a la línea de meta. Obviamente, para un maratón que llegue a la meta mediante la ejecución de un paso a la vez. El desarrollo de software permite, se mueve un paso a la vez hacia sus objetivos orientados a objetos también. Los pasos están compuestos por principios y objetivos adicionales de implementación, tales como los descritos en el acrónimo SOLID:

- **S** : Responsabilidad solo principio (SRP)
- **O** : abierto-cerrado Principio (OCP)
- **L** : Liskov principio de sustitución (LSP)
- **I** : Interfaz Segregación Principio (ISP)
- **D** : La dependencia Inversion Principio (DIP)

Originalmente compilado por Robert C. Martin en la década de 1990, estos principios proporcionan un camino claro para pasar de código fuertemente acoplado con una mala cohesión y poco encapsulación de los resultados deseados de código de

acoplamiento flexible, que opera muy cohesiva y encapsular las necesidades reales del negocio apropiadamente .

La Responsabilidad Individual principio dice que las clases, módulos, etc., deben tener una y sólo una de las razones para cambiar. Esto ayuda a impulsar la cohesión en un sistema y se puede utilizar como una medida de acoplamiento también.

El principio abierto-cerrado indica cómo un sistema puede ampliarse mediante la modificación del comportamiento de las clases o módulos individuales, sin tener que modificar la clase o módulo en sí. Esto le ayuda a crear sistemas bien encapsulado altamente cohesivos.

El *Liskov* principio de sustitución también ayuda con la encapsulación y la cohesión. Este principio dice que no se debe violar la intención o la semántica de la abstracción que está heredando de o implementar.

La interfaz de Segregación Principio ayuda a hacer que su sistema sea fácil de entender y utilizar. Se dice que no se debe obligar a un cliente a depender de una interfaz (API) que el cliente no necesita. Esto le ayuda a desarrollar bien encapsulado conjunto, de cohesión de las partes.

La dependencia Inversión Principio ayuda a entender cómo enlazar correctamente el sistema en conjunto. Se le informa de que su detalle de implementación dependerá de las abstracciones políticas de más alto nivel, y no al revés. Esto le ayuda a avanzar hacia un sistema que se acopla correctamente, e influye directamente en la encapsulación y la cohesión de ese sistema.

A lo largo del resto de este artículo, voy a caminar a través de un escenario de creación de un sistema de software. Verá cómo los cinco principios sólidos pueden ayudar a lograr la encapsulación fuerte, alta cohesión y bajo acoplamiento. Verá cómo se puede empezar con una de 50 metros “lograr que se haga ahora” guión, y terminar con un maratón de largo plazo de cambios a la funcionalidad del sistema.