Universidad de La Laguna

Principios SOLID

DAA

Darío Domínguez González……..……..……alu0101408095@ull.edu.es

Los principios SOLID son un conjunto de cinco directrices fundamentales en el diseño de software orientado a objetos, introducidos por Robert C. Martin. Estos principios buscan crear sistemas más comprensibles, flexibles y mantenibles. A continuación, se detallan cada uno de ellos:

1. **Principio de Responsabilidad Única (Single Responsibility Principle):** Establece que una clase debe tener una única responsabilidad o motivo para cambiar. Es decir, cada clase debe encargarse de una sola función o tarea dentro del sistema. Esto facilita la comprensión y el mantenimiento del código, ya que los cambios en una funcionalidad específica afectan únicamente a la clase correspondiente.
2. **Principio Abierto-Cerrado (Open-Closed Principle):** Indica que las entidades de software (clases, módulos, funciones, etc.) deben estar abiertas para su extensión pero cerradas para su modificación. Esto significa que se debe poder añadir nueva funcionalidad sin alterar el código existente, promoviendo la reutilización y reduciendo el riesgo de introducir errores en el sistema.
3. **Principio de Sustitución de Liskov (Liskov Substitution Principle):** Propone que las clases derivadas deben ser sustituibles por sus clases base sin alterar el correcto funcionamiento del programa. En otras palabras, los objetos de una subclase deben poder reemplazar a los de la superclase sin que el comportamiento esperado se vea afectado. Esto asegura la coherencia en la jerarquía de herencia y promueve un diseño más robusto.
4. **Principio de Segregación de Interfaces (Interface Segregation Principle):** Establece que una clase no debe verse obligada a implementar interfaces que no utiliza. Es preferible crear interfaces específicas y pequeñas en lugar de interfaces generales y grandes. Esto reduce la dependencia de las clases respecto a métodos que no necesitan y facilita la implementación de cambios.
5. **Principio de Inversión de Dependencias (Dependency Inversion Principle):** Sugiere que los módulos de alto nivel no deben depender de módulos de bajo nivel, sino que ambos deben depender de abstracciones. Además, las abstracciones no deben depender de los detalles, sino que los detalles deben depender de las abstracciones. Este principio promueve el desacoplamiento y facilita la modificación y prueba del código.

Aplicar estos principios en el desarrollo de software contribuye a la creación de sistemas más robustos, fáciles de mantener y escalables, permitiendo adaptarse a futuros cambios y reduciendo la complejidad del código.