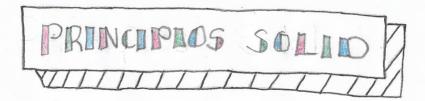
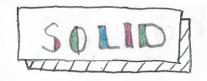
CAPITULO





Es uno de los acrónimos más famosos en el mundo de la Programación. Introducido por Robert C. Martin a principios del 2000, se compone de 5 Principios de la programación orientada a objetos.

PRINCIPIO DE LA RESPONSABILIDAD UNICA (S)

El principio nos dice que una clase debe contener una ónica funcionalidad, caso contrario habrá que separar las clases en múltiples clases. De esta forma conseguimos que la clase sea más enten dible y fácil de mantener.

PRINCIPIO DE SER ABIERTO Y CERRADO (0)

Este principio nos indica que nuestro código debe estar abierto a extensiones y cerrado a modificaciones, es decir, que on código ya escrito y finalizado no se toque, ya que prodeía afectar a su funcionamiento. Sin embargo, dado que nuestro sistema evolucionará con el tiempo, debe estar abiado a cambios, es dear, que podamos extender las clases por medio de clases abstractas, que nos permitan crear nuevos objetos que here den de estas clases, sin alterar su funcionamiento.

Ejemplo:

Public abstract class figura {...}

+ Herencia

Public class Trinangulo extends figura { ... }

PRINCIPIO DE SUSTITUCION DE LISKOV (L)

El principio nos dice que toda clase que extienda la foncionalidad de una clase base, debe hacerlo sin alterar su funcionamiento. De esta forma, cual guier subclase que herede de la clase padre podrá ser sustituida por Otra subclase, sin asectar el comportamiento de la clase padre. Si una clase hija no implementa una propredad o método de la clase padre. la hija no quede ser sub clase de la clase base, puesto que estaría Violando el principio de sustitución de Liskov.

5jem Plo

Public abstract class Figura &...?

Public class Triongulo extend figura { ... } -D Subclase de

Public abstract class figura {...}

Public class circulo {..} - No puede ser subclase

PRINCIPIO DE SEGREGACION DE INTERFAZ (1)

El principio nos dice que no debemos tener. métodos no implementa dos en nuestras clases extendidas, muchas veces se da el caso que tenemos métodos que nunca implementa mos Para evitar esto, nos recomienda dividir la interfaz en interfaces más pequeñas, así todos los métodos que implemente mos tendrán su propósito.

PRINCIPIO DE INVERSION DE DEPENDENCIA (D)

Este Principio establece que, en el dominio de nuestro sistema, debemos Utilizar interfaces o abstracciónes, elementos que cambien con poca frecuencia, de tal forma que sean las concreciones de menor las que dependan de elementos y no a la inversa.