Principios y Patrones

Principios SOLID

Patrones

Antipatrones

Cosas extras

- Override

  En Java, el término "override" se refiere a la capacidad de una clase hija (subclase) de proporcionar una implementación diferente para un método que ya está definido en su clase padre (superclase).

  Cuando una subclase hereda un método de su superclase, puede decidir si desea utilizar la implementación existente del método o proporcionar su propia implementación personalizada. Si la subclase decide proporcionar una implementación diferente para el método, utiliza la anotación `@Override` antes de la declaración del método para indicar que está anulando (override) el método de la superclase.

  El uso de la palabra clave "override" es importante para garantizar el principio de polimorfismo y permitir que los objetos de una subclase sean tratados como objetos de la superclase en tiempo de ejecución, pero con la implementación específica de la subclase para el método anulado.

- Virtual

  En lenguajes que admiten la palabra clave "virtual", cuando un método se declara como virtual en una clase base, se permite que las subclases anulen ese método con su propia implementación. Esto significa que, en tiempo de ejecución, se invocará la versión del método específica de la subclase si está presente, en lugar de la versión de la superclase.

  La declaración de un método como virtual en la clase base se realiza utilizando la palabra clave "virtual" en el momento de su definición. A continuación, en las subclases, se utiliza la palabra clave "override" para indicar que se está anulando el método virtual de la superclase.

- Diferencia clase abstracta y interfaz.

En resumen, una clase abstracta proporciona una base común para clases derivadas y puede tener métodos abstractos y concretos, mientras que una interfaz define una serie de métodos que una clase puede implementar, sin proporcionar implementaciones concretas.

▼ Constructor

El uso de la palabra clave "override" es importante para garantizar el principio de polimorfismo y permitir que los objetos de una subclase sean tratados como objetos de la superclase en tiempo de ejecución, pero con la implementación específica de la subclase para el método anulado.

▼ Sobrecarga

La sobrecarga de métodos es una característica de la programación orientada a objetos que permite definir varios métodos con el mismo nombre pero con diferentes parámetros en una clase. Estos métodos pueden tener el mismo nombre, pero deben tener una lista de parámetros distinta, ya sea en términos de cantidad, tipo o ambos.

▼ Encapsulamiento

El encapsulamiento se refiere a la capacidad de ocultar los detalles internos de un objeto y proporcionar una interfaz pública para interactuar con él. Los objetos encapsulados mantienen su estado interno y los detalles de implementación ocultos, lo que facilita el mantenimiento y la modificación del código sin afectar otras partes del sistema.

El encapsulamiento protege los datos y evita que se modifiquen de manera inadecuada, mejorando la seguridad y mantenibilidad del código.

▼ Herencia

La herencia permite crear nuevas clases basadas en clases existentes, heredando sus propiedades y comportamientos. La clase derivada (hija) puede extender o modificar la funcionalidad de la clase base (padre). La herencia facilita la reutilización de código, la organización jerárquica de las clases y la creación de relaciones entre ellas.

▼ Abstracción

La abstracción implica identificar las características y el comportamiento esencial de un objeto en el dominio del problema y representarlos en el código como una clase. Al crear una abstracción, se enfoca en los aspectos relevantes del objeto y se ignoran los detalles irrelevantes. Esto permite modelar objetos del mundo real de manera efectiva en el software.

▼ Polimorfismo

El polimorfismo permite que los objetos de diferentes clases respondan de manera diferente a los mismos mensajes o métodos. Se puede lograr mediante el uso de la herencia y la implementación de métodos con el mismo nombre en diferentes clases. El polimorfismo ayuda a escribir código más genérico y flexible, lo que permite tratar a los objetos de diferentes clases de manera uniforme.

El uso de la palabra clave "override" es importante para garantizar el principio de polimorfismo y permitir que los objetos de una subclase sean tratados como objetos de la superclase en tiempo de ejecución, pero con la implementación específica de la subclase para el método anulado.

▼ Principio de composición sobre herencia

Este principio sugiere que la composición de objetos (combinar y utilizar objetos existentes) es preferible a la herencia de clases para lograr la reutilización de código. En lugar de heredar todas las características de una clase base, se pueden utilizar objetos como componentes o colaboradores dentro de una clase para obtener mayor flexibilidad y modularidad.