**Patrones de diseño aplicables a JAVA**

**Concepto**

Un patrón de diseño es una abstracción de una solución en un nivel alto. Los patrones solucionan problemas que existen en muchos niveles de abstracción. Hay patrones que abarcan las distintas etapas del desarrollo; desde el análisis hasta el diseño y desde la arquitectura hasta la implementación.

Muchos diseñadores y arquitectos de software han definido el término de patrón de diseño de varias formas que corresponden al ámbito a la cual se aplican los patrones. Luego, se dividió los patrones en diferentes categorías de acuerdo a su uso.

Se han clasificado los patrones en 3 grandes categorías basadas en su PROPÓSITO: **creacionales, estructurales y de comportamiento.**

* Creacionales: Patrones creacionales tratan con las formas de crear instancias de objetos. El objetivo de estos patrones es de abstraer el proceso de instanciación y ocultar los detalles de cómo los objetos son creados o inicializados.
* Estructurales: Los patrones estructurales describen como las clases y objetos pueden ser combinados para formar grandes estructuras y proporcionar nuevas funcionalidades. Estos objetos adicionados pueden ser incluso objetos simples u objetos compuestos.
* Comportamiento: Los patrones de comportamiento nos ayudan a definir la comunicación e iteración entre los objetos de un sistema. El propósito de este patrón es reducir el acoplamiento entre los objetos.

Tambien se dividen en 2 ámbitos: Clases y objetos. Es así que, tenemos 6 tipos de patrones:

|  |  |
| --- | --- |
| **Creacionales** | * Creacional de la Clase Los patrones creacionales de Clases usan la herencia como un mecanismo para lograr la instanciación de la Clase. Por ejemplo el método Factoría. * Creacional del objeto Los patrones creacionales de objetos son más escalables y dinámicos comparados de los patrones creacionales de Clases. Por ejemplo la Factoría abstracta y el patrón Singleton. |
| **Estructurales** | * Estructural de la Clase Los patrones estructurales de Clases usan la herencia para proporcionar interfaces más útiles combinando la funcionalidad de múltiples Clases. Por ejemplo el patrón Adaptador (Clase). * Estructural de Objetos Los patrones estructurales de objetos crean objetos complejos agregando objetos individuales para construir grandes estructuras. La composición de l patrón estructural del objeto puede ser cambiado en tiempo de ejecución, el cual nos da flexibilidad adicional sobre los patrones estructurales de Clases. Por ejemplo el Adaptador (Objeto), Facade, Bridge, Composite. |
| **Comportamiento** | * Comportamiento de Clase Los patrones de comportamiento de Clases usan la herencia para distribuir el comportamiento entre Clases. Por ejemplo Interpreter. * Comportamiento de Objeto Los patrones de comportamiento de objetos nos permite analizar los patrones de comunicación entre objetos interconectados, como objetos incluidos en un objeto complejo. Ejemplo Iterator, Observer, Visitor. |