PATRONES DEL SOFTWARE

INDICE

INDICE 1

¿Qué son? 2

Ventajas 2

Clases de patrones 2

- Creacional: 3
- Estructural: 3
- Comportamiento: 3

Tabla resúmen 4

Bibliografía 4

¿Qué son?

Los patrones de software es la solución predefinida a problemas que se plantean en la programación de software.

Se lleva programando muchos años y los problemas han ido surgiendo suelen repetirse en el tiempo, cuando algo se repite en el tiempo podemos decir que se crea un patrón.

Los desarrolladores han sido capaces de detectar esos patrones, esos problemas, y estandarizar unas soluciones que se consideran óptimas, para que ni siquiera tengan que plantearse cómo hacerlo.

No solo es un acercamiento teórico a los problemas, si no que se facilita la reutilización de código.

Ventajas

- La ventaja más evidente es que facilita enormemente la resolución de problemas, haciendo que, llegados al punto de detección de un patrón, sea rápido.

- Se promueve la reutilización de código para solucionar lo que ocurre.

- Es un código ya validado, y cuya estandarización facilita también la lectura a terceros, pues está sujeto a los convencionalismos propios del lenguaje de programación que se utilice.

Clases de patrones

La clasificación de patrones se puede ver desde dos perspectivas:

1. Según el **propósito** para lo que sirven, es decir, el tipo de problema que solucionan.
2. Según el **ámbito**, pueden afectar a *Clases* u *Objetos*.

A principios de los noventa, se definió la llamada Gang of four (GoF), creada por cuatro autores en el libro “Design Patterns: Elements of Reusable Object-Oriented Software”.

Estos son un conjunto de 23 patrones que, divididos bajo el prisma de su propósito, se pueden dividir en tres categorías, que son:

Creacional:

Este primer patrón son los patrones que se refieren a la creación tanto de clases como de objetos.

Los patrones creacionales son 4, y de ellos los más usados son:

- factory method: interface que se usa para crear objetos en una superclase.
- builder: crea etapa por etapa objetos, a partir de otros objetos más simples.
- singleton: asegura una única instancia de una clase.

Estructural:

Si en el anterior patrón creamos la clase/objeto, en estos patrones creamos sus estructuras, y la relación entre objetos o clases.

Los patrones estructurales son 7, y el más usado es:

- adapter: convierte la interface de una clase en otra interface.

Comportamiento:

Este tercer y último tipo de patrón se encarga de la comunicación entre clases u objetos.

Son los más numerosos con 9 patrones, pero de entre ellos los más usados son:

- observer: define una dependencia entre objetos del tipo “uno-a-muchos”.
- strategy: define un tipo de algoritmo encapsulado e intercambiables entre si.
- state: permite que cuando hayan cambios internos en un objeto, este alteren su comportamiento.

Tabla resúmen

Purpose / Scope		Purpose		
		Creational (5)	Structural (7)	Behavioral (11)
Scope	Class	Factory Method	Adapter	Interpreter Template Method
	Object	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Façade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

Bibliografía

https://www.gofpatterns.com/behavioral/index.php

https://www.startertutorials.com/patterns/organizing-catalog.html

https://profile.es/blog/patrones-de-diseno-de-software/

https://dev.to/gelopfalcon/los-7-patrones-de-diseno-de-software-mas-importantes-28l2

https://www.digitalocean.com/community/tutorials/gangs-of-four-gof-design-patterns

https://medium.com/codechai/what-is-gang-of-four-gof-5f55a6942913