PATRON DE DISEÑO

DOO-TAREA 6

Victor Hugo Martinez Garcia

1657393

Grupo: 007

Patrón

Cada patrón describe un problema que ocurre una y otra vez en nuestro entorno también descríbela idea central de la solución al problema, para que se pueda utilizar un millón de veces sin tener que hacer dos veces lo mismo.

Patrón de diseño

Es una descripción de clases y objetos que se comunican entre sí para brindar una solución a problemas de software en contextos similares.

Los patrones nos dan soluciones concretas a problemas que encontramos normalmente a desarrollar una solución en cualquier lenguaje de programación pues como ya sabemos son diseños básicos de clases e interfaces. Para facilitar la programación, pero a veces no es posible aplicarlas o la solución no se adapta a lo que necesitamos y para esto se usa un diferente patrón, para especificaciones personales.

¿Cuál es el objetivo de un patrón de diseño?

Tener guardados catálogos que puedan ser reutilizables en diseños de software pues ayudara a evitar hacer soluciones a problemas ya conocidos, así mismo el programador se ahorra tiempo.

Los patrones de diseño se documentan y utilizan plantillas formales que siguen una norma.

Pero todo documento sobre esto debe llevar el nombre (Describe el problema de diseño), el Problema (Cuando se aplica el patrón) y La solución (Que describe los elementos que componen el diseño).

Un ejemplo de esta plantilla sería el GOF y algunos de sus campos son:

A. Nombre del patrón: Ayuda a recordar la esencia del patrón.
B. Clasificación: Los patrones originalmente definidos por GOF se clasifican en tres categorías "Creacional", "Estructural", "Comportamiento".
C. Propósito / Problema: ¿Qué problema aborda el patrón?
D. Colaboraciones: ¿Cómo trabajan en equipo los participantes para llevar a cabo sus responsabilidades?

E. Consecuencias: ¿Cuáles son los beneficios y resultados de la aplicación del patrón?
F. Patrones relacionados: Comparación y discusión de patrones relacionados. Escenarios donde puede utilizarse en conjunción con otros patrones.

Los patrones se dividen en 3 categorías que son:

Patrones creacionales: utilizados para instanciar objetos, y así separar la implementación del cliente de la de los objetos que se utilizan. Con ellos intentamos separar la lógica de creación de objetos y encapsularla.

Ejemplos de estos patrones son: Factory Method,Abstract Factory o Singleton

Patrones de comportamiento: Permiten definir la comunicación entre los objetos del sistema y el flujo de la información entre los mismos.

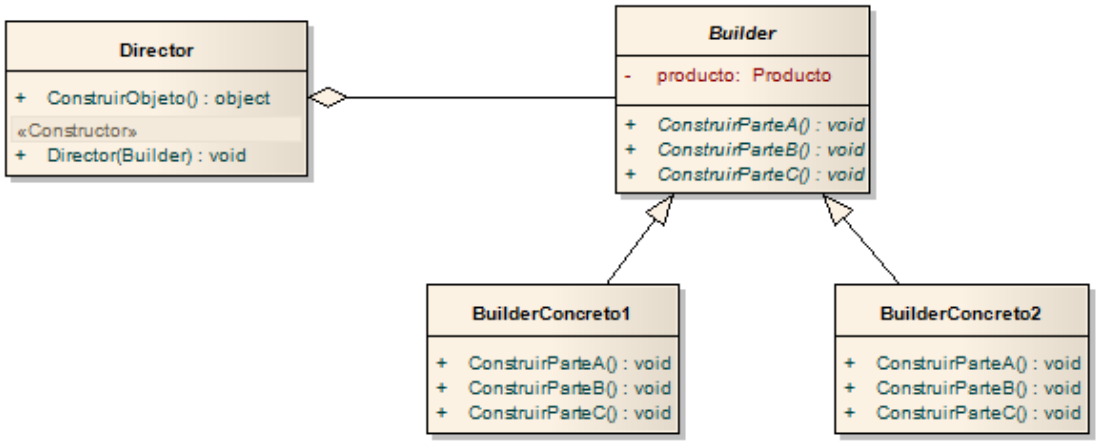
Ejemplos de este patrón son: Strategy,Interpreter y Template Method.

Patrones Estructurales: Permiten crear grupos de objetos para ayudarnos a realizar tareas complejas.

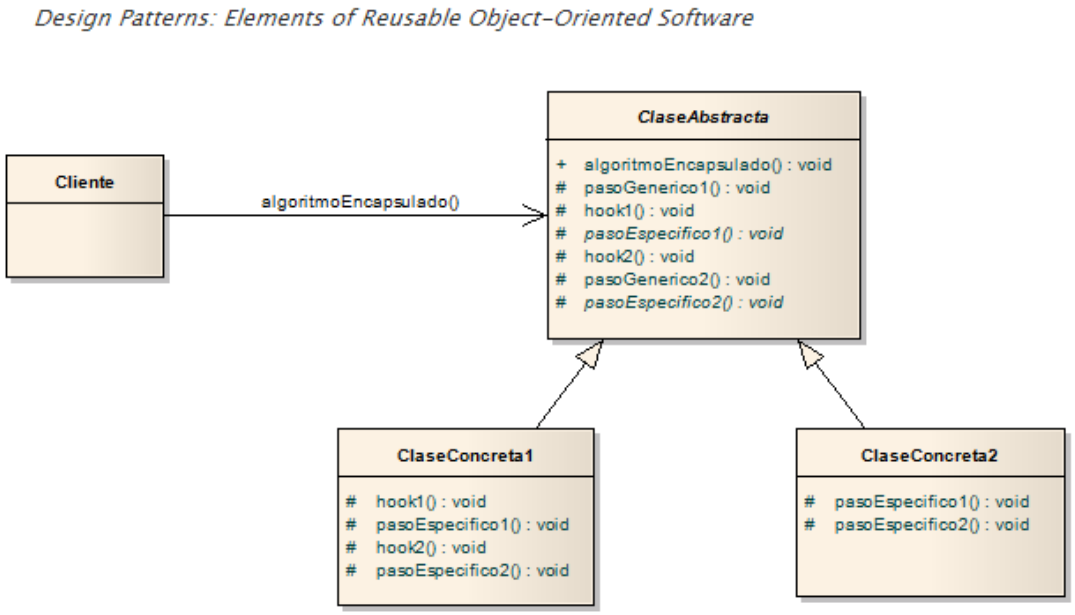
Ejemplos: Adapter,Composite,Proxy y Bridge

Patrón Builder

Design Patterns: Elements of Reusable Object-Oriented Software



Su objetivo es instanciar objetos complejos que generalmente están compuestos por varios elementos y que admiten diversas configuraciones.

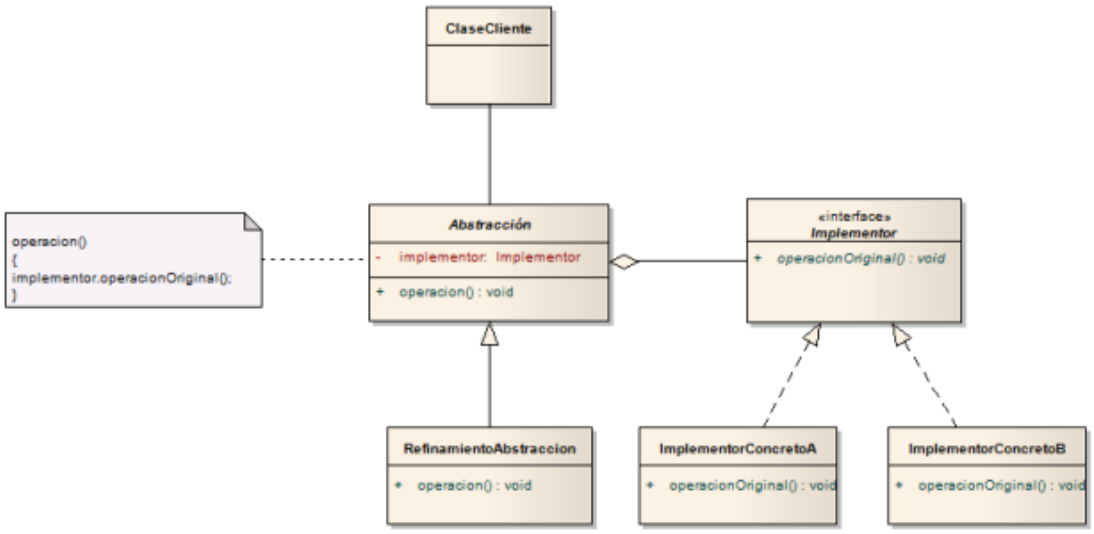


Template method

Establece una forma de encapsular algoritmos.

Este patrón se basa en un principio muy sencillo: si un algoritmo puede aplicarse a varios supuestos en los que únicamente cambie un pequeño número de operaciones, la idea será utilizar una clase para modelarlo a través de sus operaciones.

Design Patterns: Elements of Reusable Object-Oriented Software



Bridge

Cualquier cambio que se realice sobre abstracción afectara a todas las clases que la implementan, Bridge propone añadir un nuevo nivel de abstracción entre ambos elementos que permitan que puedan desarrollarse cada uno por su lado.

BIBLIOGRAFIA

Daniel Garcia. (s.f.). Obtenido de https://danielggarcia.wordpress.com/2014/03/17/patrones-estructurales-iv-patron-bridge/

Microsoft. (s.f.). *MSDN Microsoft*. Obtenido de https://msdn.microsoft.com/es-es/library/bb972240.aspx