Patrones de Diseño

- Diseño de Software
  - Ciclo 2023-II
  - Hugo R. Cordero

Principios y Patrones de Diseño

- Software Design Patterns and Principles (quick overview)

  https://www.youtube.com/watch?v=WV2Ed1QTst8

  techlead

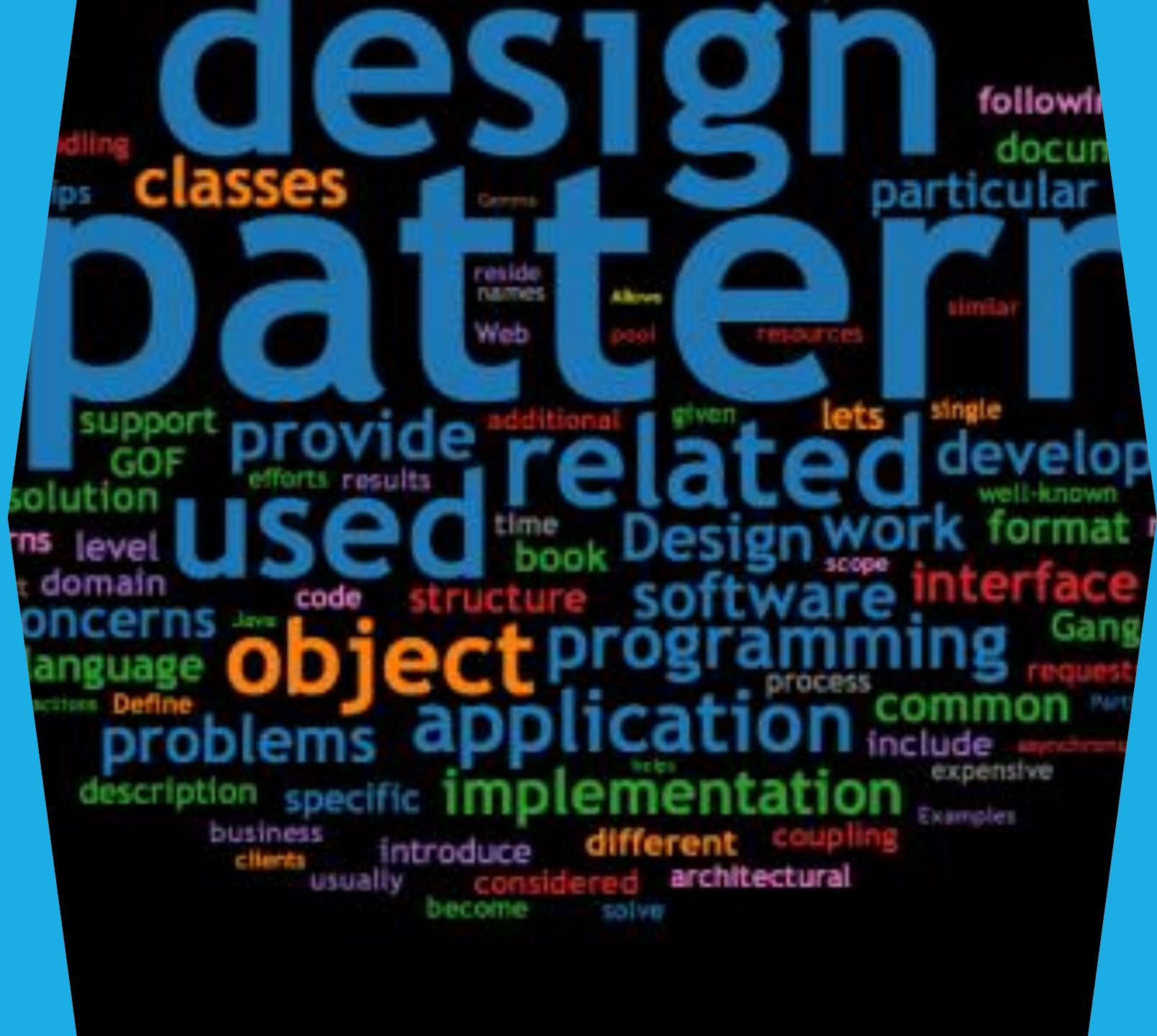
- Visualice los primeros 5 minutos del video y responda:
  - ¿Qué pasa en los proyectos muy grandes?
  - ¿A qué se refiere con las clases de Dios?
  - ¿Qué debemos considerar al separar el programa en componentes?

Patrones

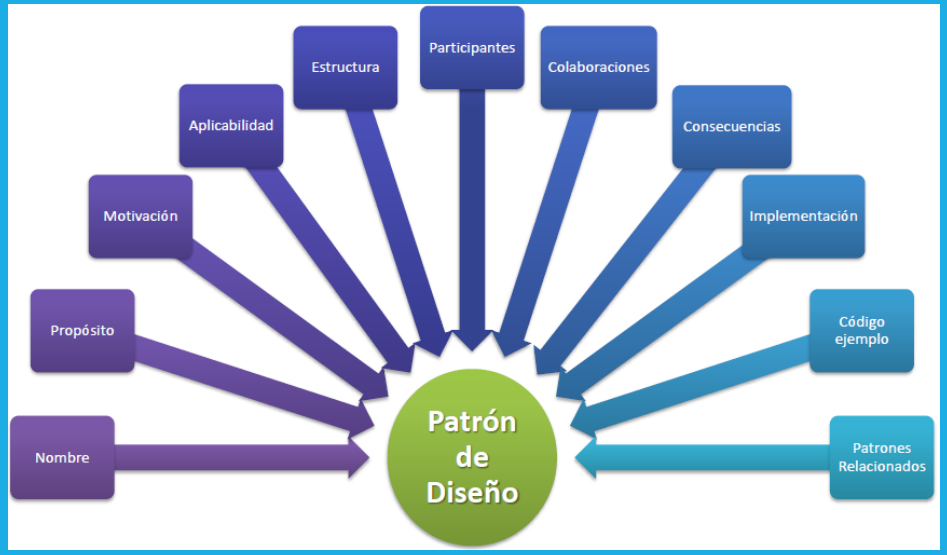
- En la obra “The Timeless Way of Building” (1979), cada patrón es una regla de tres partes, entre un contexto, un problema y una solución.
- Es utilizado en el desarrollo de software orientado a objetos y se aplica al diseño. Luego se utiliza en las etapas del desarrollo de software.
- Tipos de Patrones:
  - De Arquitectura: subsistemas y responsabilidades, con tácticas y estrategias arquitectónicas para organizar las relaciones entre ellos.
  - De Diseño: expresa esquemas de estructuras de diseño (o sus relaciones) con las que se puede construir software en detalle.

Patrones de Diseño

- Describen un problema recurrente y una solución en un contexto dado (Alexander et al. 1977).
- Tienen un **contexto** y se refiere al conjunto de situaciones en las que el patrón es aplicable.
- Son soluciones elegantes y **reutilizables** a problemas de desarrollo de software.
- Están **probados**, pero puede no ser bueno si se aplica en otro contexto diferente.
- Puede **adaptarse** de un problema general a un problema concreto.



Elementos en un Patrón de Diseño



Nombre

Describe un problema de diseño, su solución, y consecuencias en una o dos palabras

El Problema

Describe cuando aplicar el patrón. Algunas veces el problema incluirá una lista de condiciones que deben cumplirse

La Solución

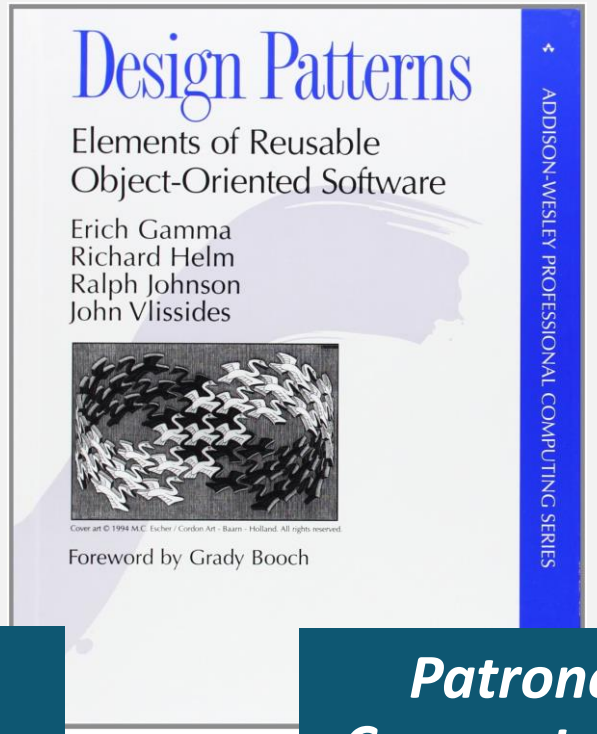
Describe los elementos que forman el diseño, sus relaciones, responsabilidades y colaboraciones

Consecuencias

Son los resultados, importantes para la evaluación de alternativos y para comprender los costes y beneficios

Clasificación de los Patrones según GoF

"GoF" es la banda de los cuatro: Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides



Patrones de Creación

- ***(Creational Patterns)***, son aquellos que crean objetos por nosotros, en vez de tener que instanciar los objetos directamente.
- Los patrones de diseño de creación se enfocan en la mejor manera de crear objetos.

Patrones de Estructura

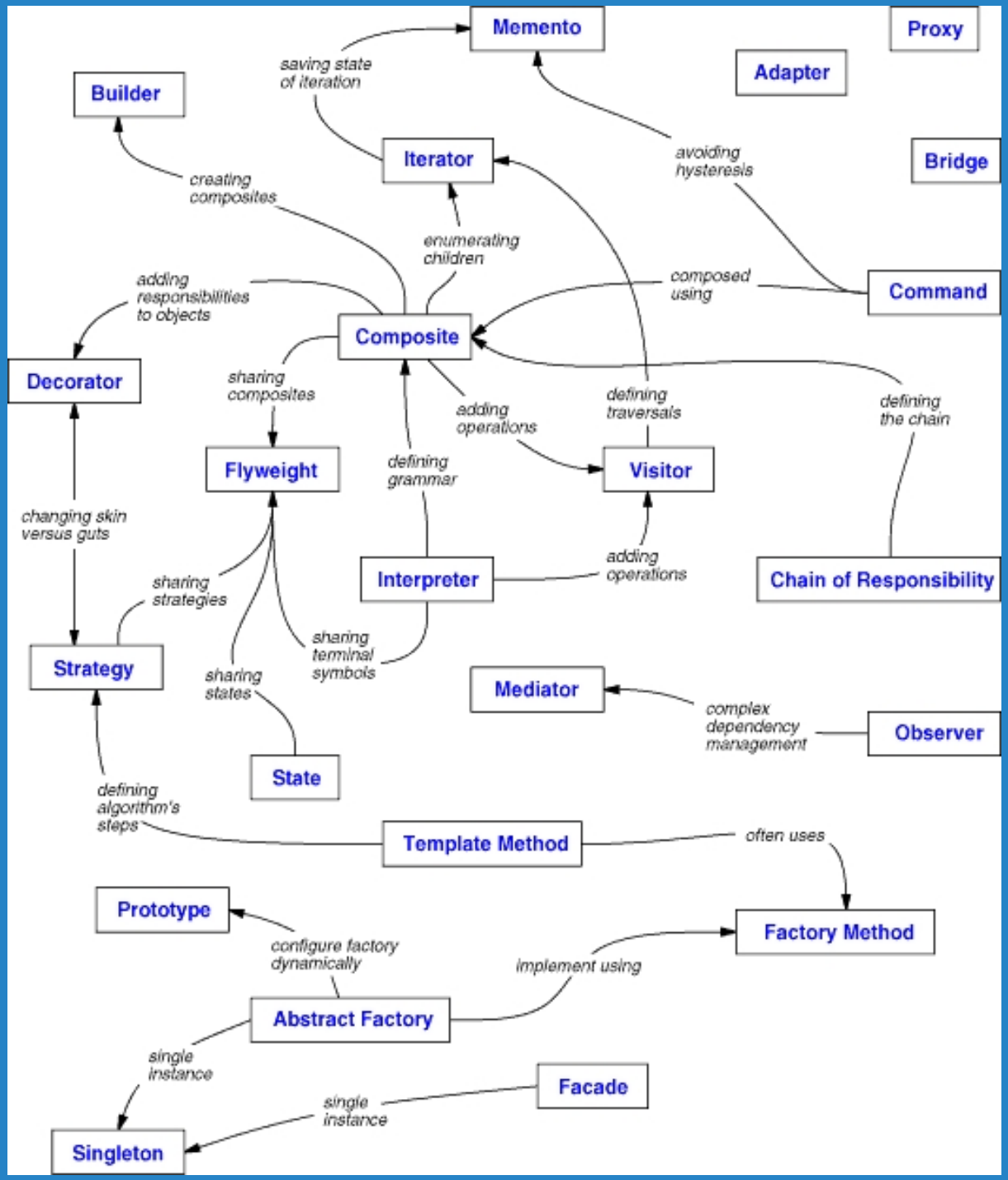
- ***(Structural Patterns)***, le ayudan a componer grupos de objetos en estructuras más grandes y complejas, como interfaces de usuario o datos de contabilidad.
- Los patrones de diseño de estructura se concentran en la composición de objetos

Patrones de Comportamiento

- ***(Behavioral Patterns)***, ayudan a definir la comunicación entre objetos en su sistema y cómo se controla el flujo de información y control en programas complejos.
- Los patrones de diseño de comportamiento se enfocan en la comunicación entre objetos.

Clasificación de los Patrones de Diseño según GOF

- Este libro se publicó por primera vez en 1994 y es uno de los libros más populares para aprender patrones de diseño.
- Los patrones de diseño de Gangs of Four sientan las bases de los patrones de diseño básicos en la programación.
- Se documentaron unos 23 patrones inicialmente.



Patrones de Diseño de Creación

Aquellos que se desarrollan en el proceso de instanciar objetos con la finalidad de facilitar su creación y configuración.

Singleton

Restringe la creación de objetos pertenecientes a una clase o el valor de un tipo a un único objeto

Prototype

Crea objetos personalizados sin conocer la clase exacta de donde son creados y sin conocer los detalles de cómo crearlos.

Factory

Crea varios subtipos de una determinada interface y que todos los objetos concretos fabricados hagan una tarea similar, pero con detalles de implementación diferentes.

Abstract Factory

Crea conjuntos o familias de objetos (denominados productos) que dependen mutuamente y todo esto sin especificar cual es el objeto concreto.

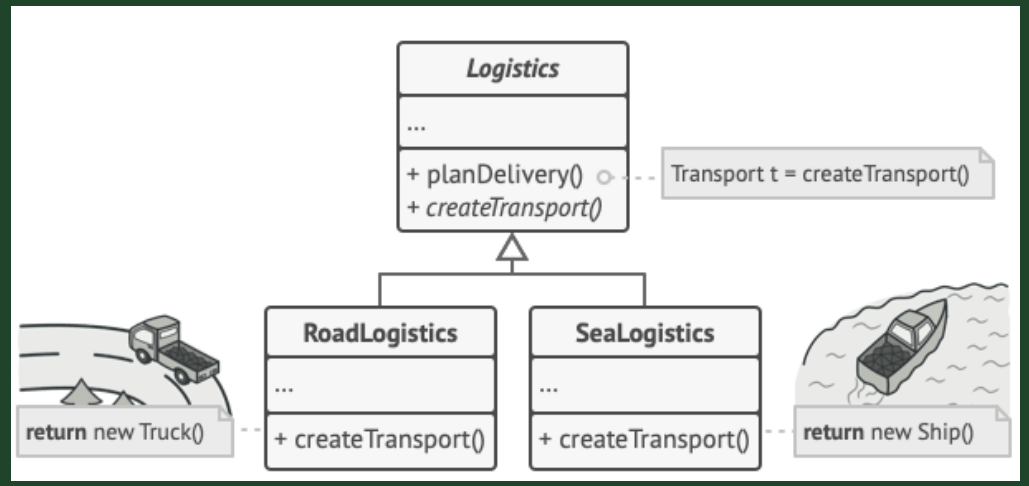
Builder

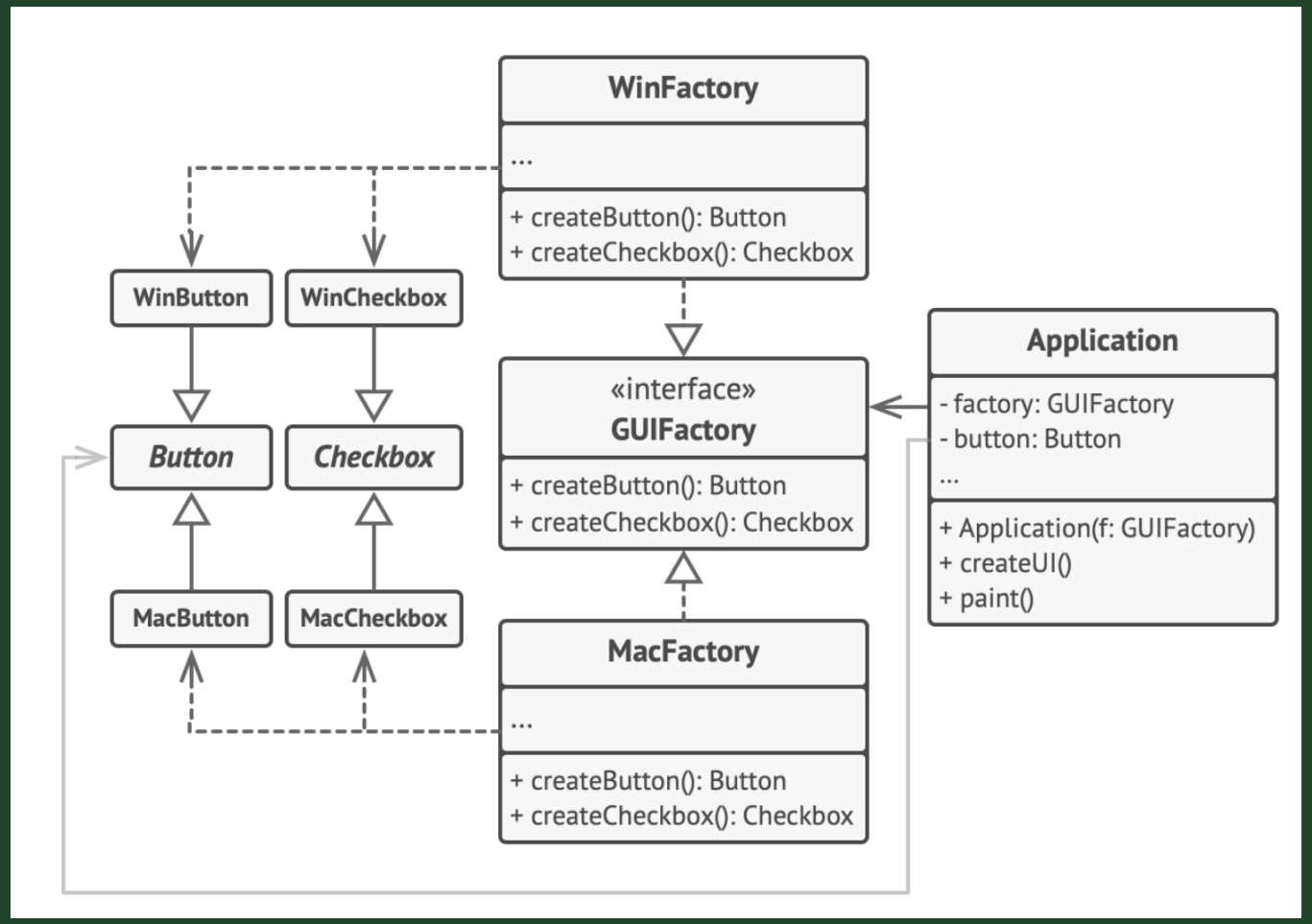
Crea una variedad de objetos complejos desde un objeto fuente, que se compone de una variedad de partes que contribuyen individualmente a la creación.

Patrón Factory y Abstract Factory

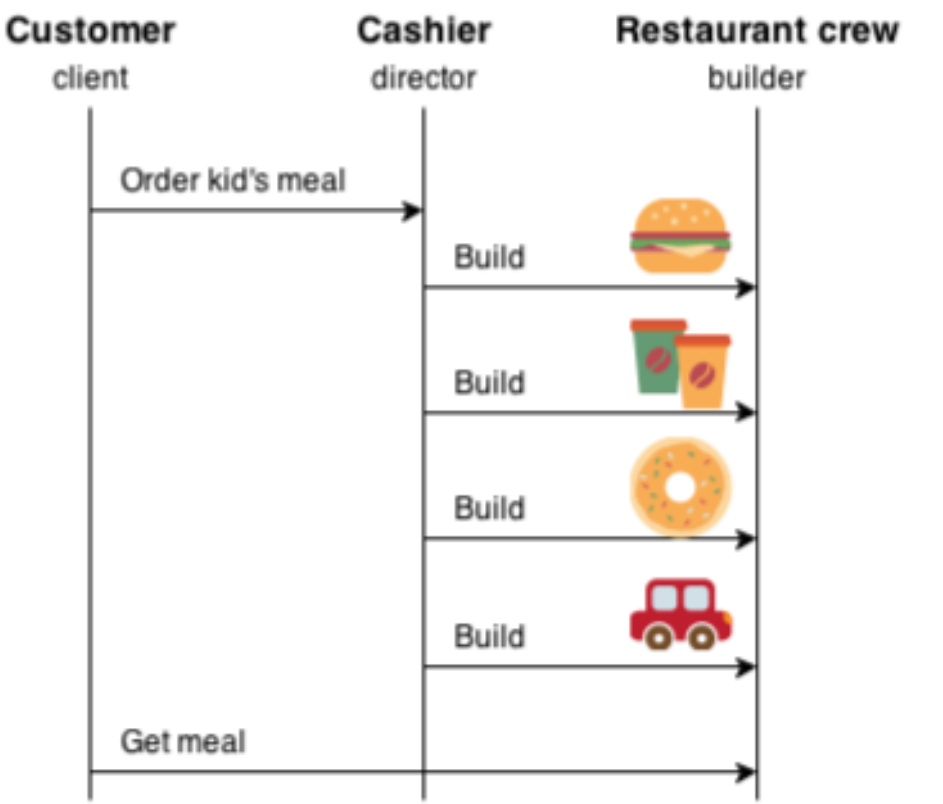
- Factory elimina la necesidad de unir clases específicas de la aplicación dentro de tu código.
- El objetivo del patrón de diseño Abstract Factory es formar un grupo de clases con funcionalidades compartidas, más conocidas como familias, creadas a partir de Factory.

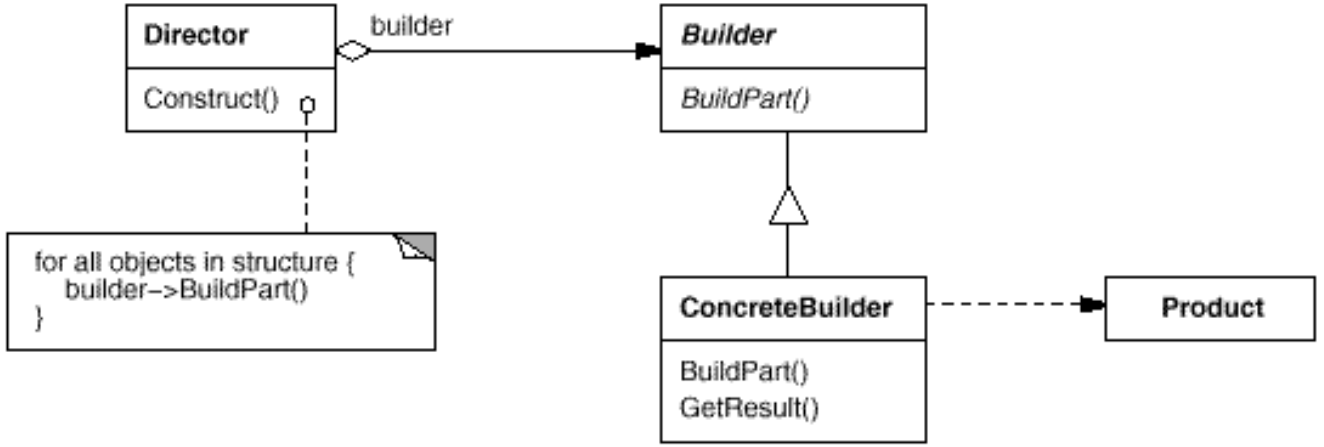
- Factory es usado cuando tenemos una clase o interfaz con muchas subclases o implementaciones y según algún input necesitamos devolver una de estas subclases concretas.





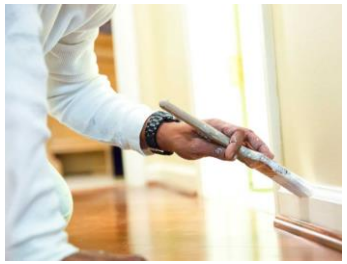
Patrón Builder





- El patrón de diseño Builder permite crear objetos que habitualmente son complejos, utilizando otro objeto más simple que los construye paso por paso.
- Se utiliza en situaciones en las que debe construirse un objeto repetidas veces o cuando este objeto tiene gran cantidad de atributos y objetos asociados, y en donde usar constructores para crear el objeto no es una solución cómoda.
- Se trata de un patrón de diseño bastante útil también en la ejecución de pruebas en donde debemos crear el objeto con atributos válidos o por defecto.

Patrones de Diseño Estructurales



Decorator

Permite añadir funcionalidad extra a un objeto (de forma dinámica o estática) sin modificar el comportamiento del resto de objetos del mismo tipo.

Adapter

Permite a dos clases con diferentes interfaces trabajar entre ellas, a través de un objeto intermedio con el que se comunican e interactúan.

Composite

Facilita la creación de estructuras de objetos en árbol, donde todos los elementos emplean una misma interfaz.

Bridge

Desacopla una abstracción de su implementación, para que las dos puedan evolucionar de forma independiente.

Facade

Una facade (o fachada) es un objeto que crea una interfaz simplificada para tratar con otra parte del código más compleja, de tal forma que simplifica y aísla su uso.

Proxy

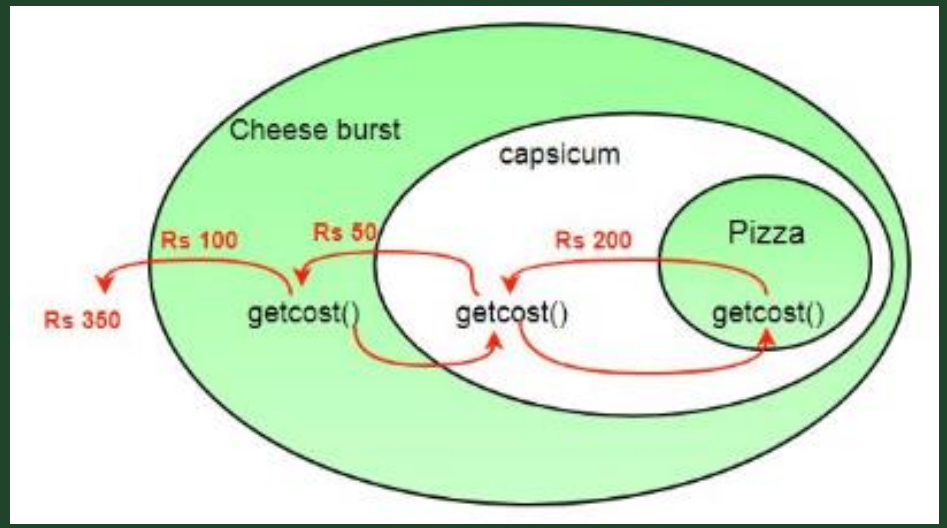
Es una clase que funciona como interfaz hacia cualquier otra cosa, para cualquier recurso que sea costoso o imposible de duplicar.

Flyweight

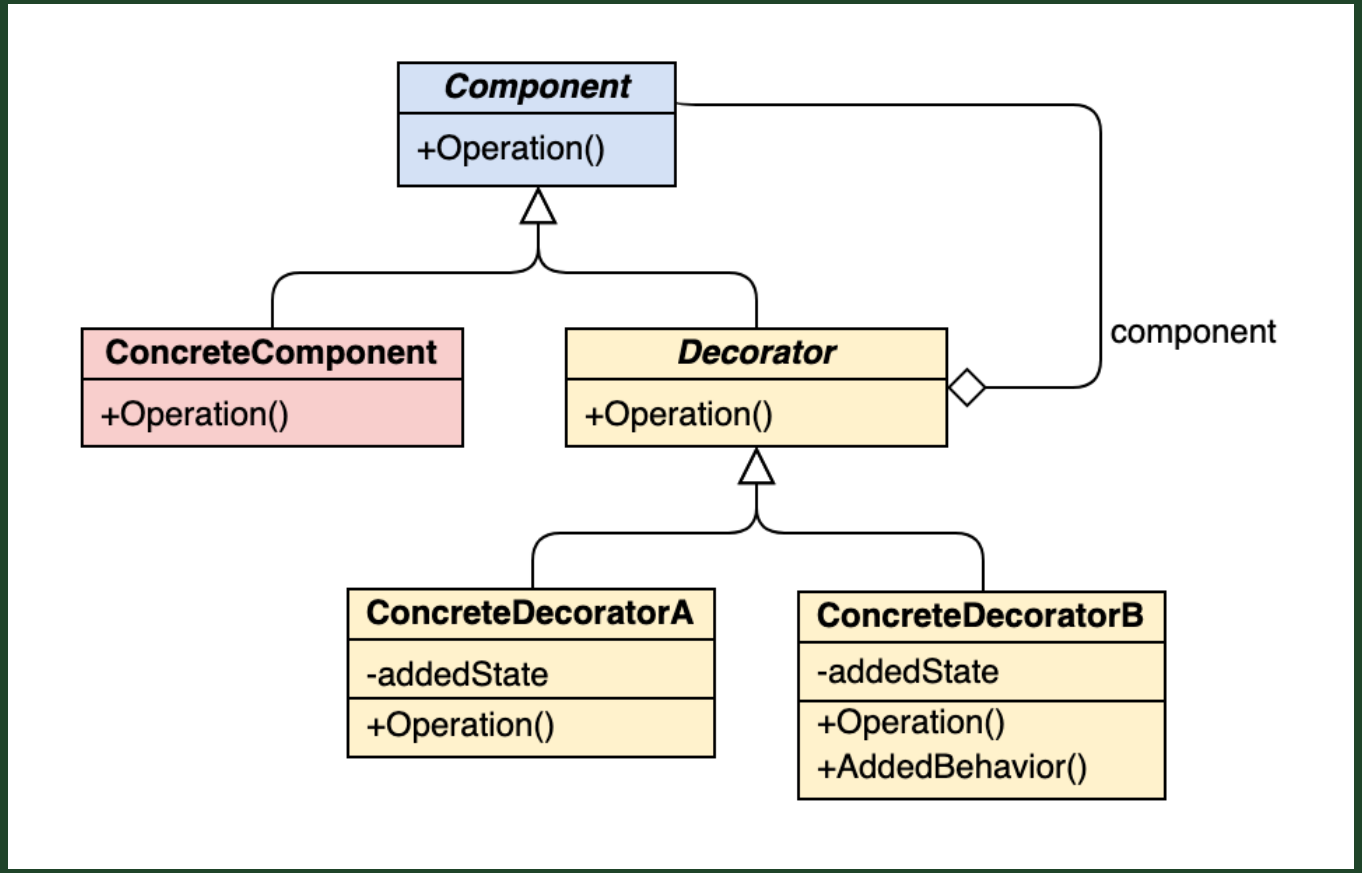
Una gran cantidad de objetos comparte un mismo objeto con propiedades comunes con el fin de ahorrar memoria.

Patrón Decorator

- Permite agregar nuevas funcionalidades a las clases sin modificar su estructura.
- Ejemplo, tenemos una pizza básica y cada vez que agregamos algo, “decoramos” nuestra pizza. Para calcular el costo total, en lugar de la herencia, utilizamos la agregación

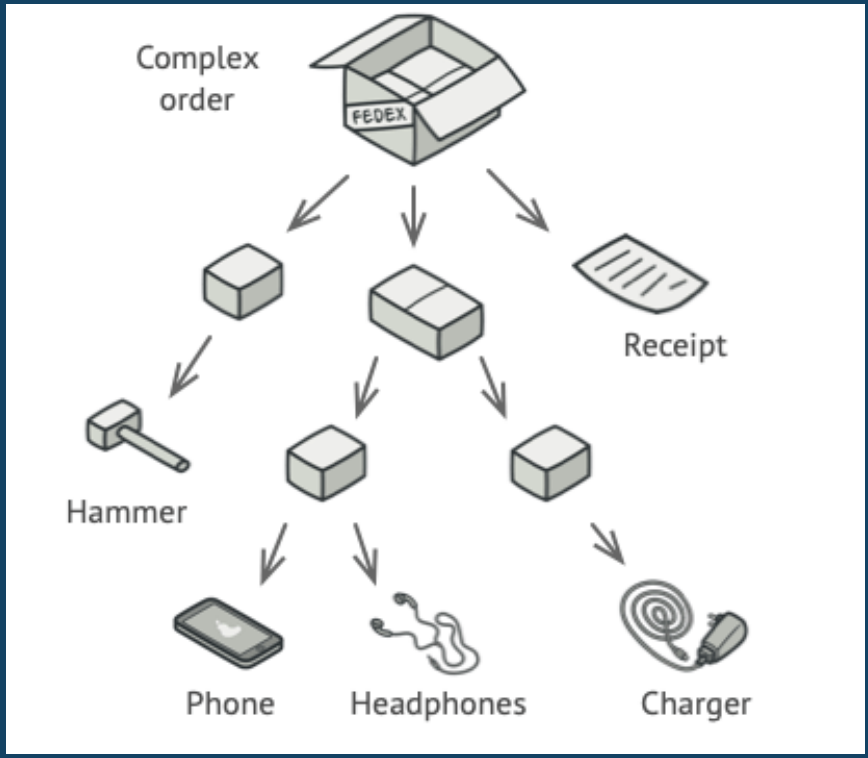


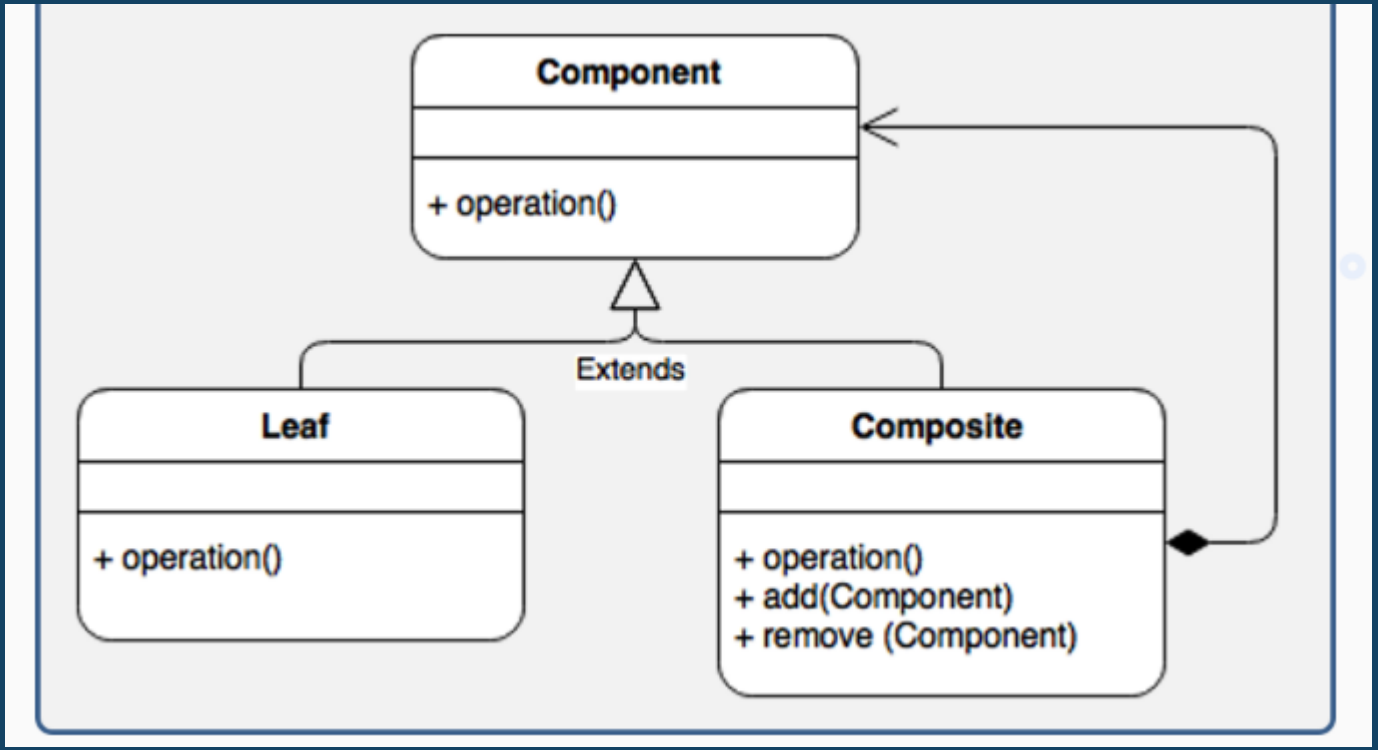
- El concepto de este patrón es agregar de forma dinámica nuevo comportamiento o funcionalidades a la clase principal. La definición oficial dice que “permite el ajuste dinámico de objetos para modificar sus responsabilidades y comportamientos existentes.”
- Los participantes para el patrón son:



Patrón Composite

- Sirve para construir objetos complejos a partir de otros más simples y similares entre sí, gracias a la composición recursiva y a una estructura en forma de árbol.

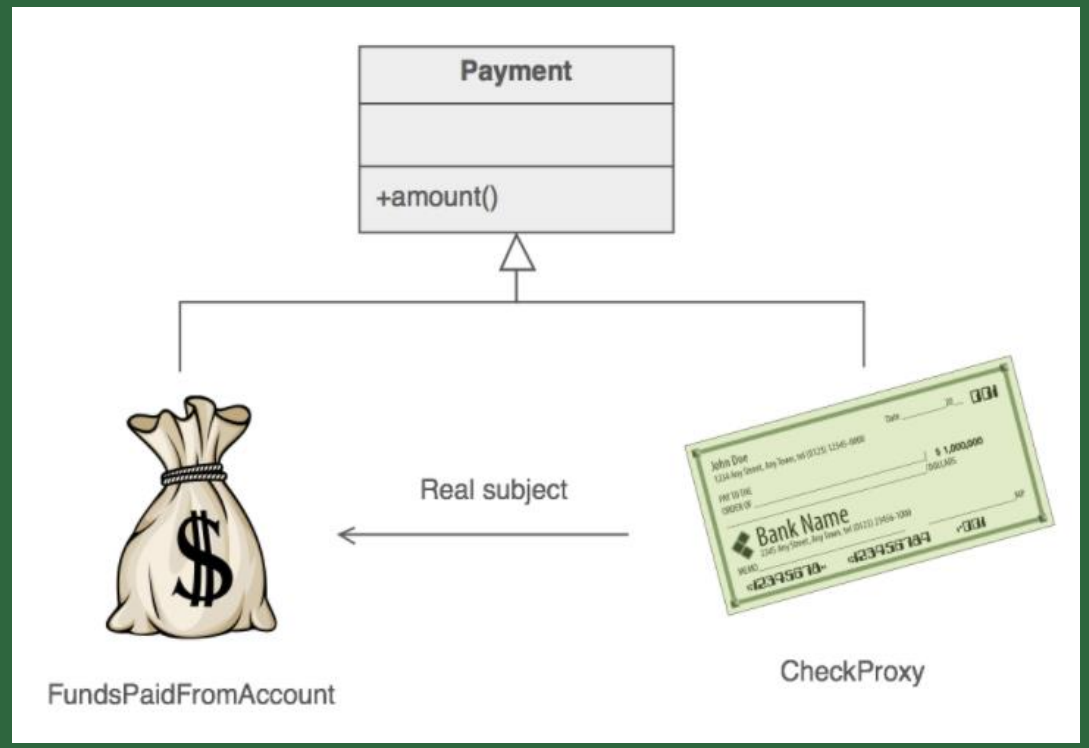




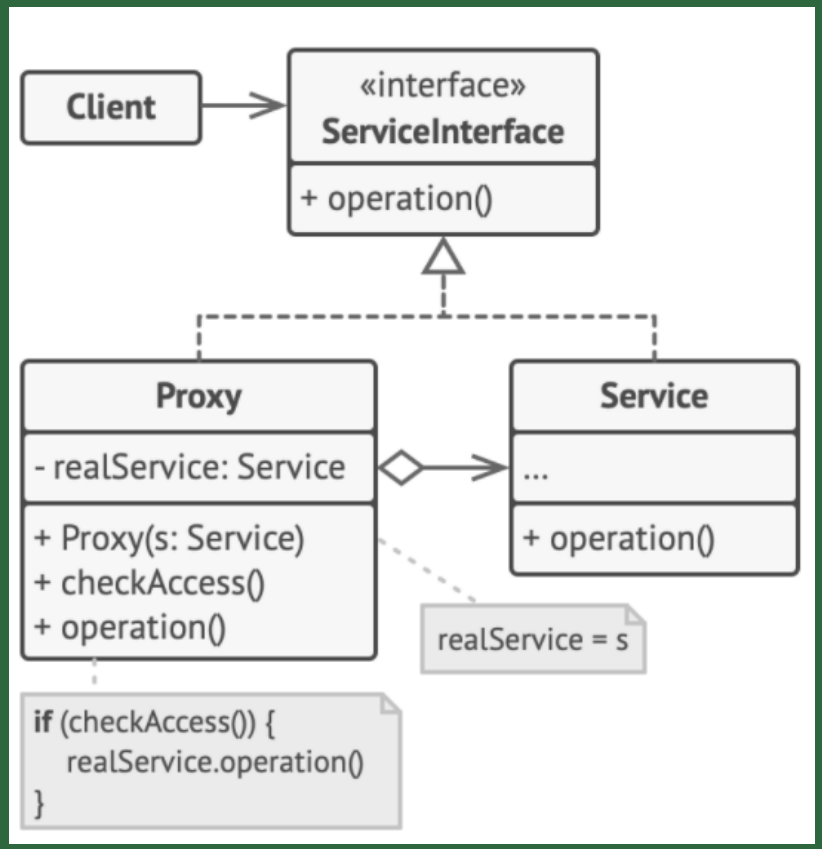
- Permitir ejecutar un comportamiento de forma recursiva sobre todos los componentes de un árbol de objetos.
- El mayor beneficio de este enfoque es que no necesita preocuparse por las clases concretas de objetos que componen el árbol.
- No necesita saber si un objeto es un simple o sofisticado. Puede tratarlos a todos de la misma manera a través de la interfaz común.

Patrón Proxy

- Proporciona el control para acceder al objeto original. Simplemente, proxy significa un objeto que representa a otro objeto.



- El rasgo principal del patrón Proxy es que mantiene al usuario inconsciente de la mediación que se está llevando a cabo en segundo plano, porque el cliente recibe un objeto estructurado como se esperaba, todo ello interactuando, sin saberlo, con un proxy.
- El proxy está dedicado entonces a la mediación entre dos objetos, que incluye la serie de acciones que se ejecutan antes y después de realizar la tarea solicitada.



Llamada a la acción…

Investigue mayor información sobre dos patrones e inclúyalos en desarrollo (código) para su proyecto.

Los patrones deben ser de las clasificaciones de creación o estructura.

Adicionalmente realice el diagrama de clases de participantes para los patrones que ha implementado.

Correo electrónico
hcorderos@unmsm.edu.pe

X / Threads
@hugorcorderos

Gracias