Patrones de diseño

Sesión 11:

PATRONES ESTRUCTURALES: PATRÓN PROXY, PATRÓN BRIDGE, PATRÓN DECORATOR Y PATRÓN COMPOSITE.

Logro de la sesión

Al finalizar la sesión, el estudiante elabora su programación usando patrones estructurales: Patrón proxy, Patrón bridge, Patrón decorator y Patrón composite.

Universidad Tecnológica del Perú

Patrones de Diseño

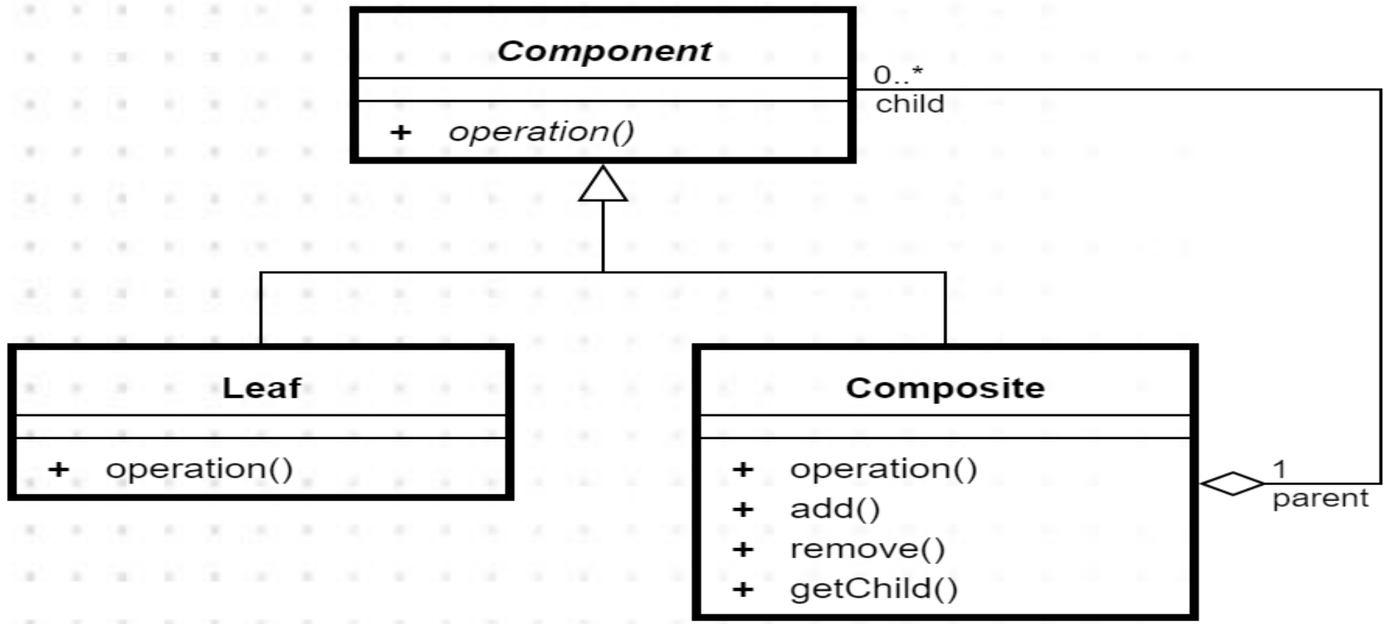
		PROPÓSITO			
		CREACIONAL	ESTRUCTURAL	COMPORTAMIENTO	
ALCANCE	CLASE	Factory Method	Adapter	Interpreter Template Method	
	OBJETO	Abstract Factory Builder Prototype Singleton	Adapter Bridge Composite Decorator Facade Proxy	Chain of Responsibility Command Iterator Mediator Memento Flyweight Observer	State Strategy Visitor

Universidad Tecnológica del Perú

PATRON ESTRUCTURAL

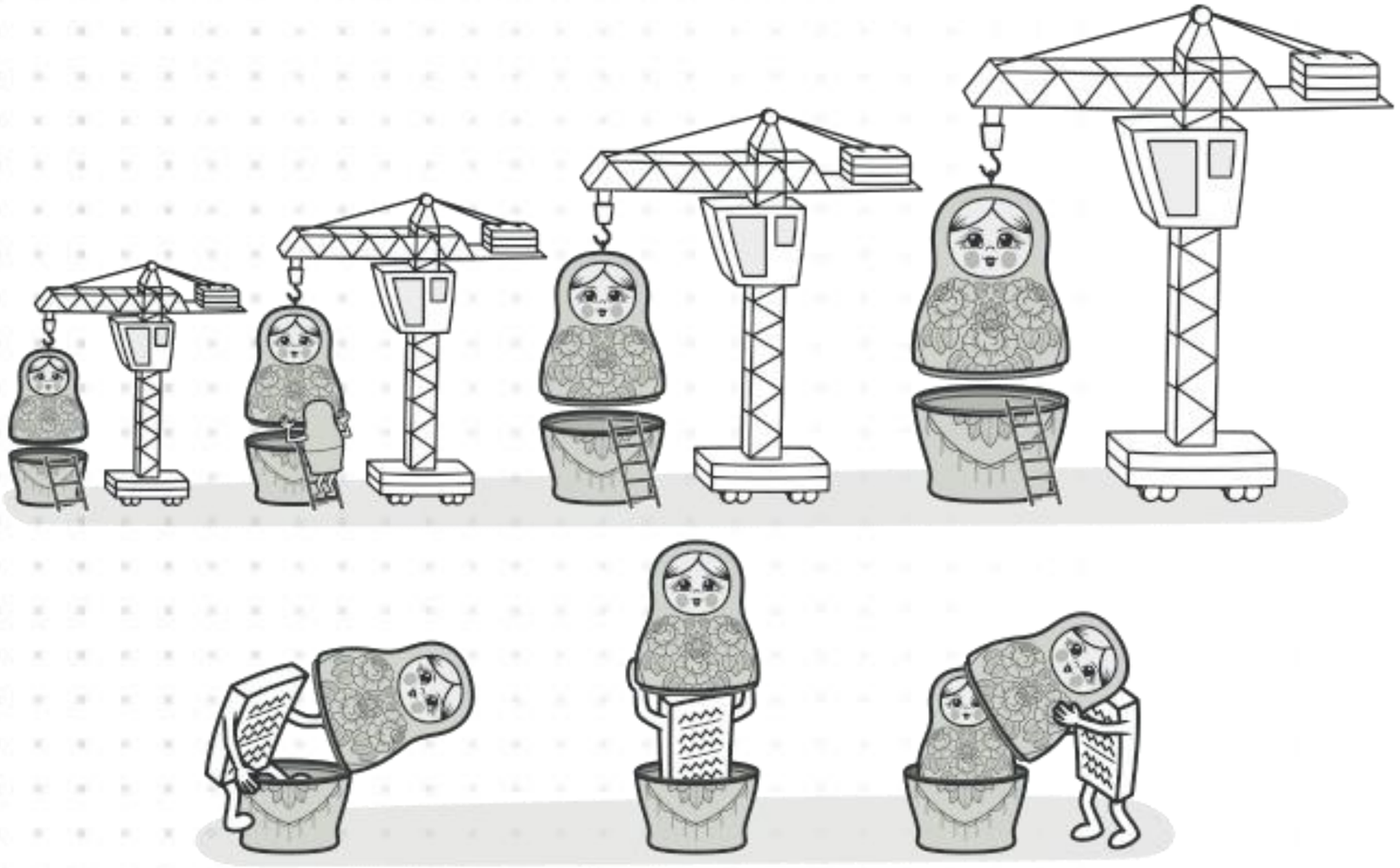
Los patrones estructurales se centran en la composición de clases y objetos para crear estructuras más flexibles y eficientes.

Estos patrones son la clave para organizar y gestionar grandes sistemas de software, proporcionando soluciones elegantes a desafíos comunes en el diseño de software.



Universidad Tecnológica del Perú

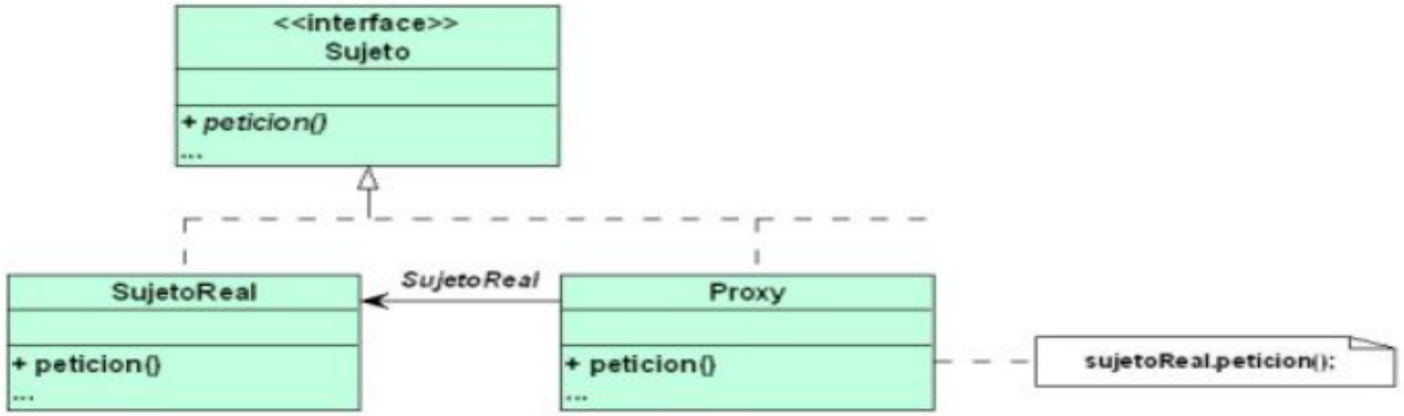
PATRON ESTRUCTURAL



Universidad Tecnológica del Perú

PATRÓN PROXY

Este patrón se basa en proporcionar un objeto que haga de intermediario (proxy) de otro, para controlar el acceso a él.



Existen diferentes tipos de proxy:

- **Proxy remoto**: proporciona un representante local de un objeto situado en otro espacios de direcciones (en otro dispositivo conectado en red).
- **Proxy virtual**: usados para crear objetos costosos sólo cuando se soliciten.
- **Proxy de protección**: permiten controlar el acceso a un objeto cuando es accesible o no, dependiendo de determinados permisos.
- **Referencia inteligente**: un sustito de un puntero, que realiza operaciones adicionales en el momento de accederse al objeto.

Material de apoyo:

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

Universidad Tecnológica del Perú

PATRÓN PROXY

1. Create a “wrapper” for a remote, or expensive, or sensitive target
2. Encapsulate the complexity/overhead of the target in the wrapper.
3. The client deals with the wrapper
4. The wrapper delegates to the target
5. To support plug-compatibility of wrapper and target, create an interface.

Ejemplos:

https://www.youtube.com/watch?v=cwfuydUHZ7o&list=PLvimn1Ins-41Uiugt1WbpyFo1XT1WOquL

https://informaticapc.com/patrones-de-diseno/proxy.php

https://github.com/mitocode21/patrones-diseno

Universidad Tecnológica del Perú

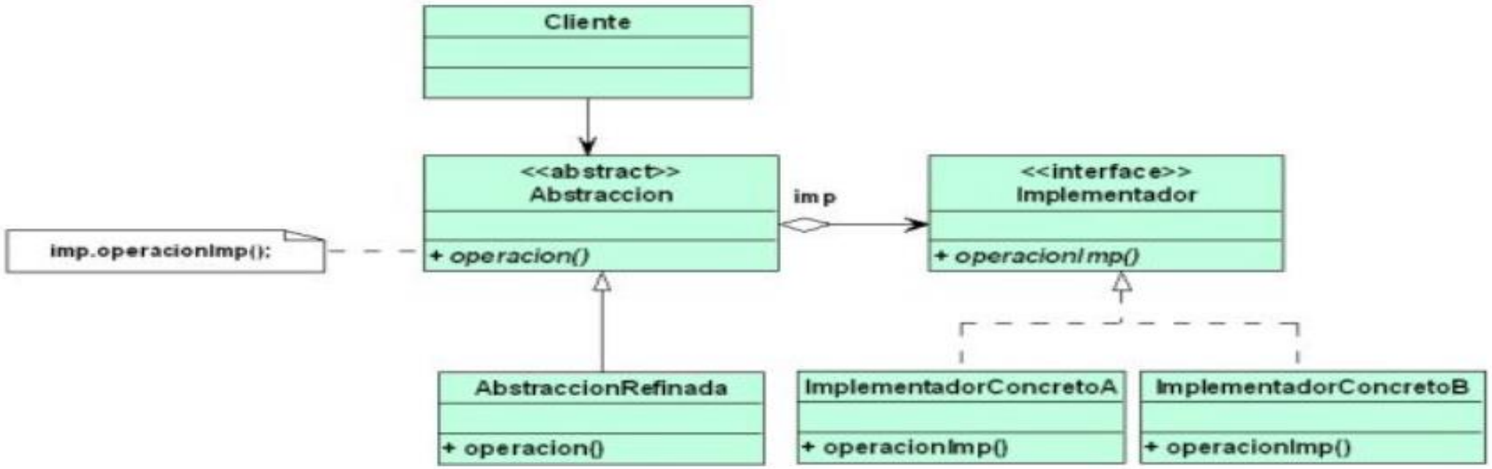
PATRÓN BRIDGE

Según el libro de **GoF** este patrón de diseño permite desacoplar una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.

Supongamos que tenemos una clase abstracta en la que se define un método que deberá implementar cada clase que herede de ella: ¿cómo haríamos si una clase hija necesitase implementarlo de forma que realizase acciones diferentes dependiendo de determinadas circunstancias?.

En dichos casos nos resultaría útil el patrón **Bridge** (puente) ya que 'desacopla una abstracción' (un método abstracto) al permitir indicar (durante la ejecución del programa) a una clase qué 'implementación' del mismo debe utilizar (qué acciones ha de realizar).

El **diagrama UML** de este patrón es el siguiente:



Veamos un ejemplo de uso de este patrón en el que creamos un sistema para elaborar lasagna, pudiendo ésta ser de carne o de verduras.

Universidad Tecnológica del Perú

Ejemplos:

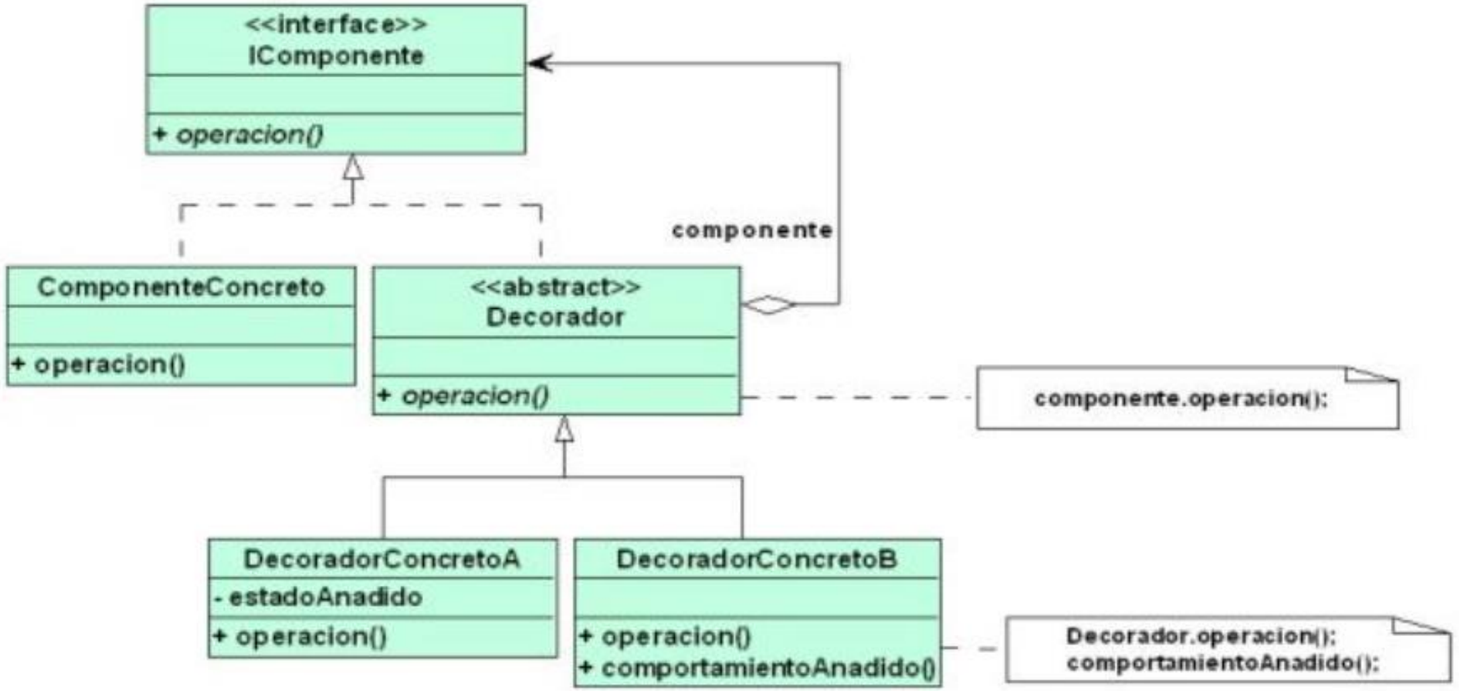
https://informaticapc.com/patrones-de-diseno/bridge.php

https://github.com/luisburgos/design-patterns

Universidad Tecnológica del Perú

PATRÓN DECORATOR

Sencillo e interesante patrón que permite añadir funcionalidades a un objeto en aquellos casos en los que no sea necesario o recomendable hacerlo mediante **herencia**:



A continuación mostramos uno de los ejemplos clásicos usados para explicarlo: crear una clase que dibuje ventanas básicas, y utilizar el patrón para agregar barras de desplazamiento sólo a algunas de ellas.

Universidad Tecnológica del Perú

Ejemplos:

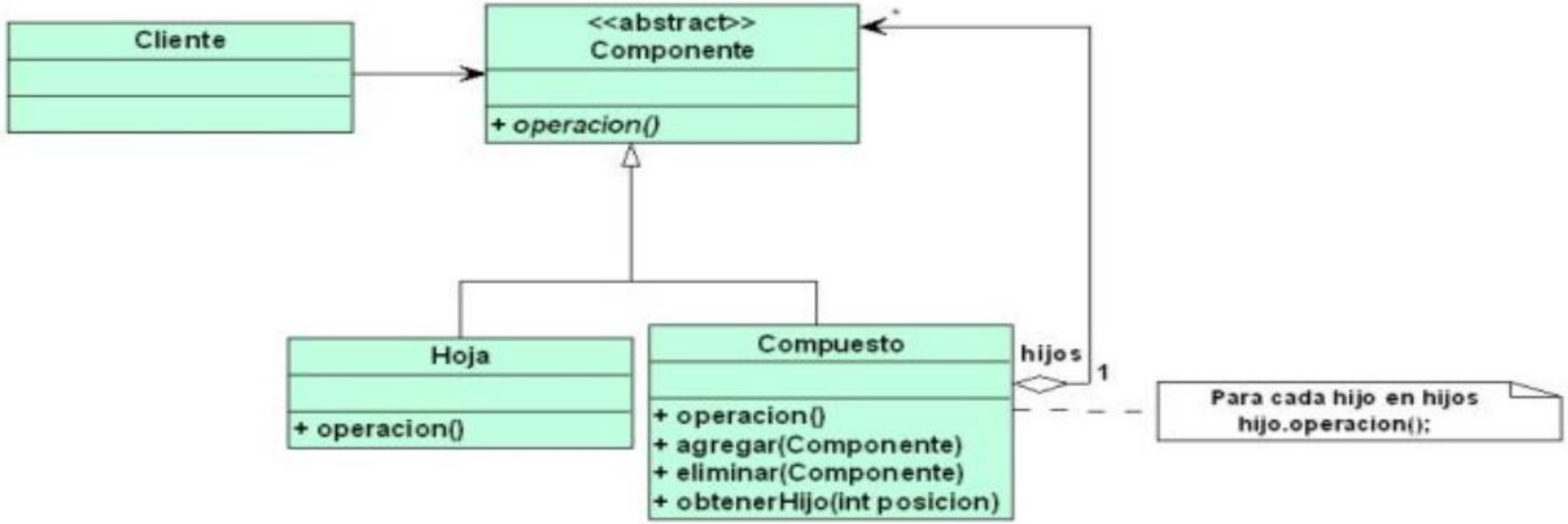
https://informaticapc.com/patrones-de-diseno/decorator.php

https://github.com/mitocode21/patrones-diseno

Universidad Tecnológica del Perú

PATRÓN COMPOSITE

Este útil patrón permite crear y manejar estructuras de objetos en forma de árbol, en las que un objeto puede contener a otro(s).



En este punto cabe aclarar que las estructuras de este tipo se componen de **nodos** (un objeto que a su vez contiene otros objetos) y **Hojas** (objetos que no contienen otros), y que ambos comparten una misma **Interface** que define métodos que deben implementar.

Para ilustrar cómo funciona este patrón de diseño veamos a continuación un programa de ejemplo en el que gestionamos archivos y carpetas.

Universidad Tecnológica del Perú

Ejemplos:

https://informaticapc.com/patrones-de-diseno/composite.php

https://github.com/luisburgos/design-patterns

Universidad Tecnológica del Perú

¿Consultas o dudas?

Universidad Tecnológica del Perú

Actividad

Resolver la actividad planteada en la plataforma.

Universidad Tecnológica del Perú

Cierre

¿Qué hemos aprendido hoy?

Elaboramos nuestras conclusiones sobre el tema tratado

Universidad Tecnológica del Perú

Universidad
Tecnológica
del Perú