

T1

II

# PATRONES ESTRUCTURALES

## Conceptos Básicos

**Objetivo:** Solucionar problemas de composición (agregación) de clases y objetos

Se acuerda de como se combinan las <sup>clases</sup> objetos para formar estructuras más <sup>grandes</sup> complejas

- **Patrones de clase:** utilizan la herencia
  - **Patrones de objeto:** describen formas de compartir objetos para obtener nueva funcionalidad

## Adapter & Wrapper | adaptador & Envoltorio

**Propósito:** inventar de una clase una o otra interfaz (esperan los clientes)

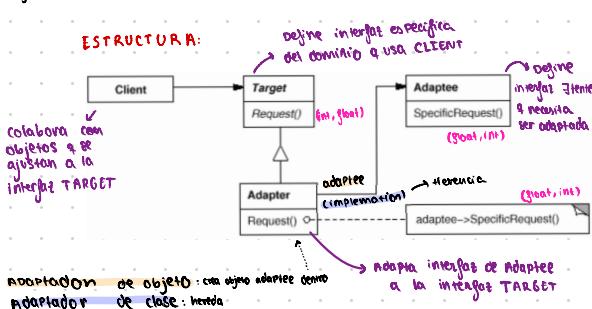
→ Permite q. cooperen clases q. de otra forma no podrían por tener interfaces incompatibles

### **Aplicabilidad:**

- ④ clase existente + interfaz no recomienda con la q necesita
  - ⑤ crear clase reutilizable  $\Rightarrow$  copiar con clases no previstas
  - ⑥ necesidad utilizar subclases existentes  $\Rightarrow$  adaptar interfaz con herencia

### Ejemplo Guía de:

## **ESTRUCTURA:**



# BRIDGE

# Puente

↳ Hanote / Body  
Manejador / cuero

• **PROPOSITO:** desarrollar una abstracción de su implementación

Luis Peña variar de forma indep.

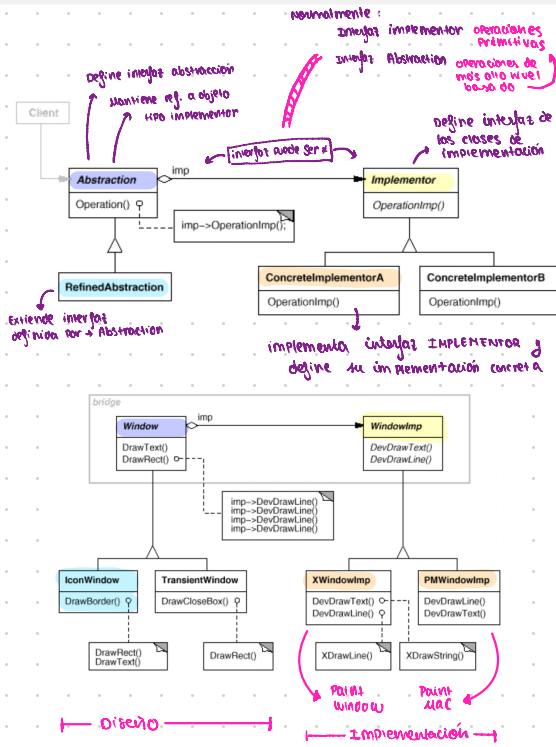
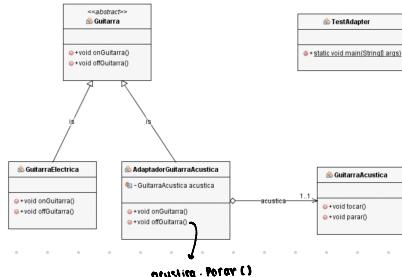
### - Aplicabilidad:

- ④ Evitar enlace permanente abstracción - implementación
  - ⑤ Abstracciones + Implementaciones  $\Rightarrow$  extensible por medio subclases
  - ⑥ Cambios implementación  $\Rightarrow$  NO impiden clientes

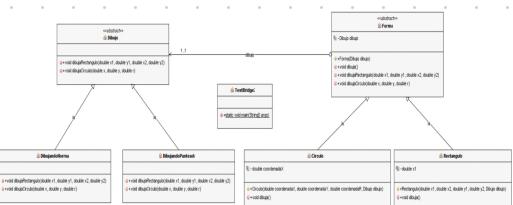
#### • Patrones Relacionados:

- ④ **Abstract Factory** → crea configuración un bridge
  - ⑤ **Adapter** → aplica a sist. ya diseñados  
Bridge → usa al comenzar un diseño

## ESEMPIO ADAPTER



## — Disney



# PATRONES ESTRUCTURALES

## COMPOSITE

## completo

**Propósito:** Componer objetos en estructura del árbol para representar jerarquías de parte-todo.

Permiten q. clientes -trotén = forma → obj

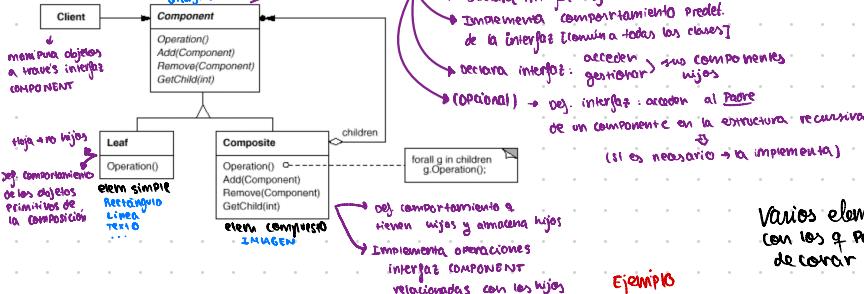
## indiv. compuestos

Objetos primitivos  
contenedores composición recursiva

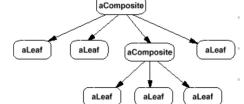
## Aplicabilidad:

- representar jerarquías de objetos parte-ítem
  - clientes,  DB ≠ entre composiciones de objetos individuales

## Estructura:



## Estruct. típica Obj. compuestos:



• como puedo componer objetos → tienen otros objetos

se pueden aplicar = llamadas a objetos → simples  
compuestos

## Decorator

## Decorador e Wrapper (envoltório)

**Propósito:** Asigna responsabilidades adicionales a un objeto → Proporcionando una alternativa flexible a la herencia para extender su funcionalidad

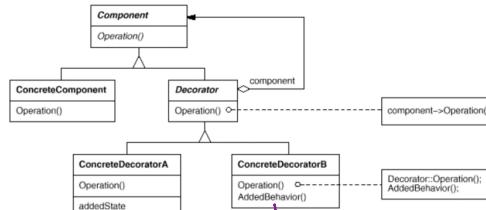
### Aplicabilidad:

- ④ Añadir obj individuales de forma dinámica transparente [reflejar a otros objetos]
  - ⑤ Responsabilidades ↗ pueden ser retiradas
  - ⑥ Extensión mediante herencia → no viable

en vez de  
macro clase  
q maneja todo

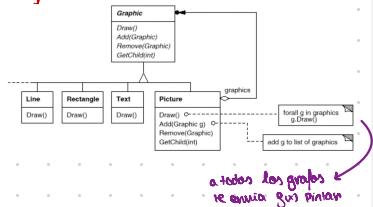
Esta clase va y añade más  
funciones además de la  
básica [operation (.)]

## Estructura:



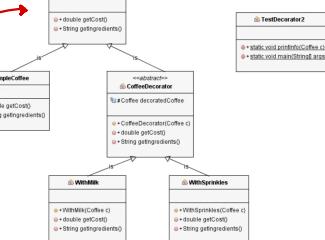
Varios elementos  
con los q Poder  
de fortalecer

## Ejemplo



• Solo un elemento  
con el q. Poder  
decorar

Ejemplo: Haciendo café con ingredientes adicionales



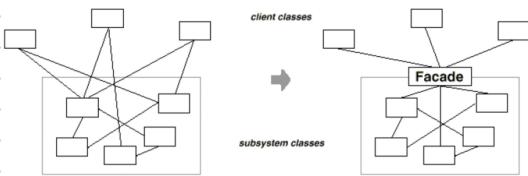
# PATRONES ESTRUCTURALES

## Facade

## Fachada

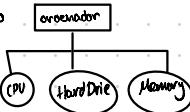
**Propósito:** Proporciona una interfaz unificada para un conjunto de interfaces de un sistema. Dej. int. de alto nivel

↓  
sub-sistema más fácil de usar



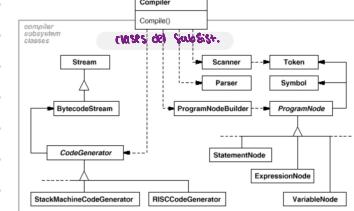
### Aplicabilidad:

- ① Interfaz simple → para Subsistema. Comprimo
- ② Muchas dependencias clientes ↔ clases q implementan una abstracción
- ③ En capas nuestro sistema.



### Estructura:

Delega peticiones en los objetos apropiados del Subsistema



- no convierte la fortuna (no ref. a ella)
- Implementa funcionalidad del sistema
- Realizan trabajos encomendados

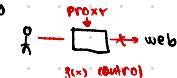
## Proxy

Representante

## Apoderado o Surrogate (Sustituto)

**Propósito:** Proporciona un representante/sustituto de otro objeto

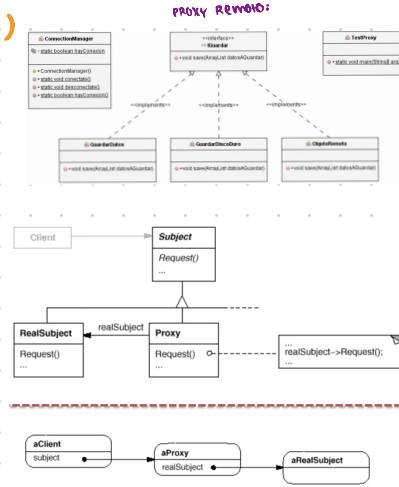
Para → controlar acceso a este  
Optimizar rendimiento del sist.



### Aplicabilidad:

necesidad de una referencia a un obj. más versatil → sofisticado → q. un simple proxy

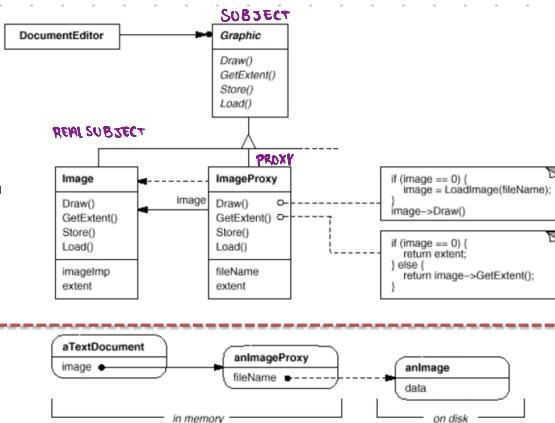
1. Proxy Remoto: Proporciona un representante local de un objeto situado en otro lugar
2. Proxy Lugar: Crea objetos estéticos por encargo
3. Proxy de Protección: controla acceso al objeto original
4. Proxy Inteligente: proporciona operaciones adicionales cuando se accede a un objeto
  - a) contar el nº de referencias al objeto real
  - b) cargar objeto persistente en memoria cuando es diferenciado por 1º vez
  - c) Cometer box → se bloquea al obj real → Para → garantizar q. no pueda ser modif. por ningún obj



### Participantes:

#### ② Proxy (Image Proxy)

- Mantiene referencias q. permite acceder al Real Subject
- Proporciona interfaz idéntica a la de Subjects
- Controla acceso RealSubject → puede controlarlo
- otros responsabilidades en función:
  - Proxies remotos: codificar una petición y sus argumentos para enviarla al RealSubject que está en otro lugar
  - Proxies virtuales: guardar información adicional sobre RealSubject para retardar el acceso al mismo
  - Proxies de protección: comprobar que el llamador tiene permisos para realizar la petición



#### ④ Subject (Graphic)

- Define interfaz común para el Real Subject y Proxy de modo q. pueda usarse un Proxy en cualquier sitio en el q. se exprese un Real Subject

#### ⑤ Real Subject (Image)

- define el obj. real representado