



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 3

Patrones de Diseño

IIC2113 – Diseño Detallado de Software

Rodrigo Saffie

rasaffie@uc.cl

31 de agosto de 2018

Actividad 2

Principios SOLID

- Single Responsibility
- Open/Closed
- Liskov Substitution
- Interface Segregation
- Dependency Inversion

Tarea 1

- Objetivos

- Justificar decisiones de diseño en base a supuestos razonables.
- Aplicar sintaxis *UML 2.0* para definir un sistema de *software* a través de distintos tipos de diagramas.
- Plasmar el diseño de un software en un programa funcional.

Patrones de diseño

¿Qué es un patrón de diseño?

“Each pattern describes a problema which occurs over and over again in our environment, and then describes the core of the solution to that problema, in such a way that you can use this solution a million times over, without ever doing it the same way twice” [Christopher Alexander](#)

Este concepto fue acuñado por GoF en [Design Patterns: Elements of Reusable Object-Oriented Software](#) en 1994

Patrones de diseño

¿Qué es un patrón de diseño?

“Design patterns make it easier to reuse successful designs and architectures. Expressing proven techniques as design patterns makes them more accessible to developers of new systems. Design patterns help you choose design alternatives that make a system reusable and avoid alternatives that compromise reusability.” [GoF, 1994]

Patrones de diseño

¿Qué es un patrón de diseño?

“[...] allows the software engineering community to capture design knowledge in a way that enables it to be reused.” [Pressman, 2009]

Patrones de diseño

¿Qué es un patrón de diseño?

- Un patrón de diseño es una solución genérica a problemas recurrentes al momento de diseñar *software*.
- No son soluciones acabadas que se pueden aplicar directamente a un código en particular: son referentes al momento de diseñar.

Patrones de diseño

¿Qué caracteriza a un patrón de diseño?

Según [Pressman, 2009]:

- Un contexto
- Un problema
- Una solución

Patrones de diseño

¿Qué caracteriza a un patrón de diseño?

Según [GoF, 1994]:

- Un nombre
- Un problema
- Una solución
- Sus consecuencias

Patrones de diseño

- En este curso se los 23 patrones definidos en el libro de [GoF, 1994].
- Estos patrones son aplicados a OOP, pero existen muchos más en diferentes ámbitos:
 - Seguridad
 - Comunicación
 - Visualización
 - Diseño de interfaces gráficas

Patrones de diseño

¿Por qué estudiarlos?

- Permiten reutilizar soluciones
- Permiten establecer terminología común: facilitan el intercambio de ideas dentro de un equipo
- Nos dan una perspectiva de más alto nivel para el diseño de *software*
- Código es más fácil de modificar
- Permiten aprender estrategias generales de diseño

Patrones de diseño

Existen 3 clasificaciones:

- Estructurales
- Creacionales
- De comportamiento

Patrones de diseño

Estructurales:

- Se centran en cómo los objetos se organizan e integran en un sistema

Patrones de diseño

Creacionales:

- Se centran en la creación, composición y representación de los objetos

Patrones de diseño

De comportamiento:

- Se centran en las interacciones y responsabilidades entre objetos

Patrones de diseño

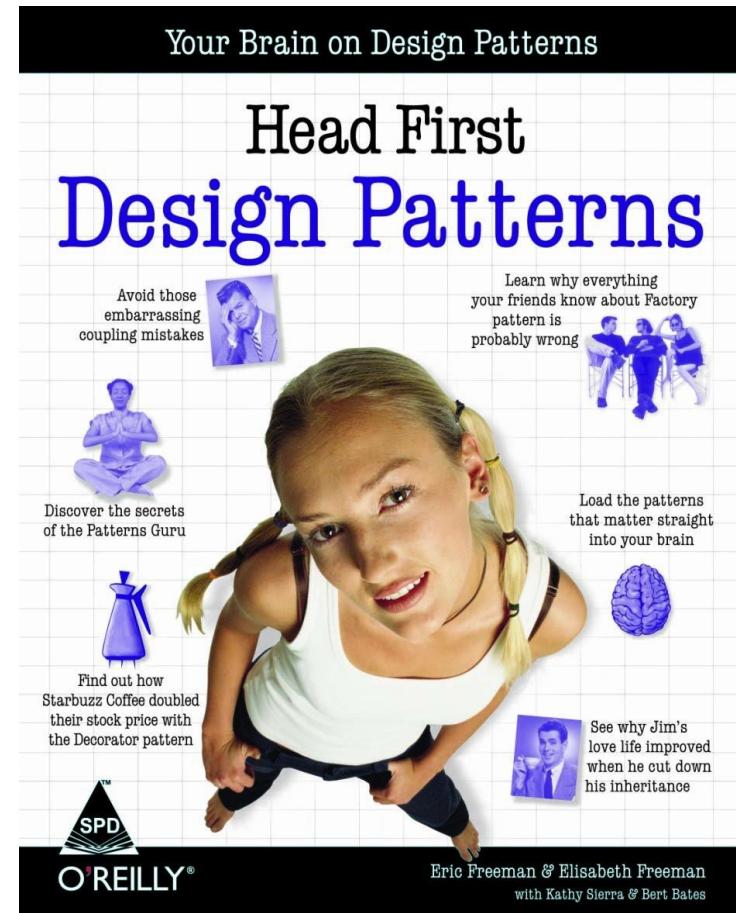
¿Cómo seleccionar un patron de diseño?

- Detectar el problema: causa del rediseño
- Analizar el propósito de cada patrón
- Identificar si hay relación entre el problema y un propósito
- Estudiar los patrones de la misma categoría
- Considerar qué podría variar en el diseño

Patrones de diseño



Patrones de diseño



Patrones de diseño

Referencias:

- [Source Making](#)
- Dofactory:
 - [.NET](#)
 - [JavaScript](#)

Patrones estructurales

Façade (Fachada)

Provee una interfaz unificada a un conjunto de interfaces en un sub-sistema. *Façade* define un interfaz de alto nivel que hace a un sistema más fácil de utilizar.

Patrones estructurales

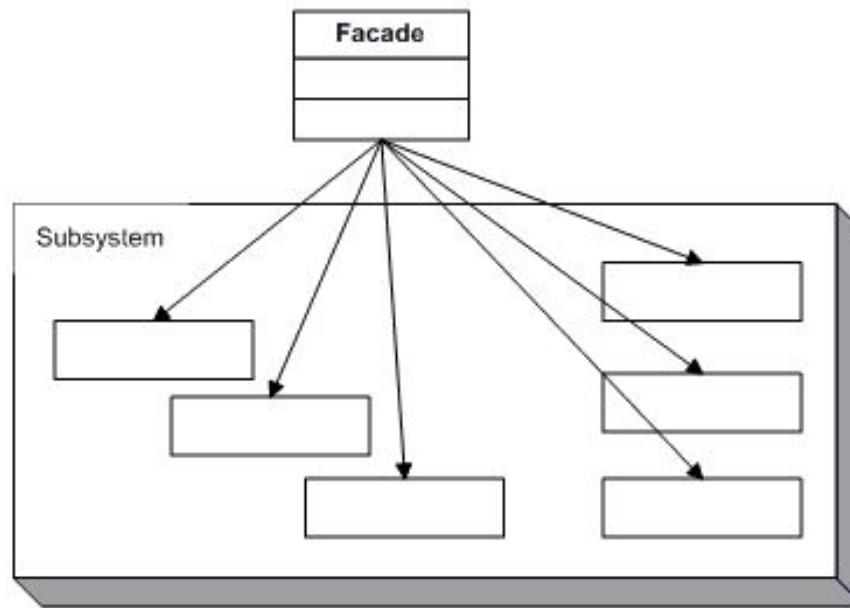
Façade (Fachada)

¿Cuándo se utiliza?

- Se quiere proveer una interfaz simple para un sistema complejo (altamente configurable)
- Desacoplar los clientes de los sub-sistemas
- Ordenar los sub-sistemas por capas, con una *façade* como punto de entrada para cada capa

Patrones estructurales

Façade (Fachada)



Ejemplo

Patrones estructurales

Adapter

Convierte la interfaz de una clase en otra interfaz que los clientes esperan. *Adapter* permite a ciertas clases interactuar, sin lo cual serían incompatibles.

Patrones estructurales

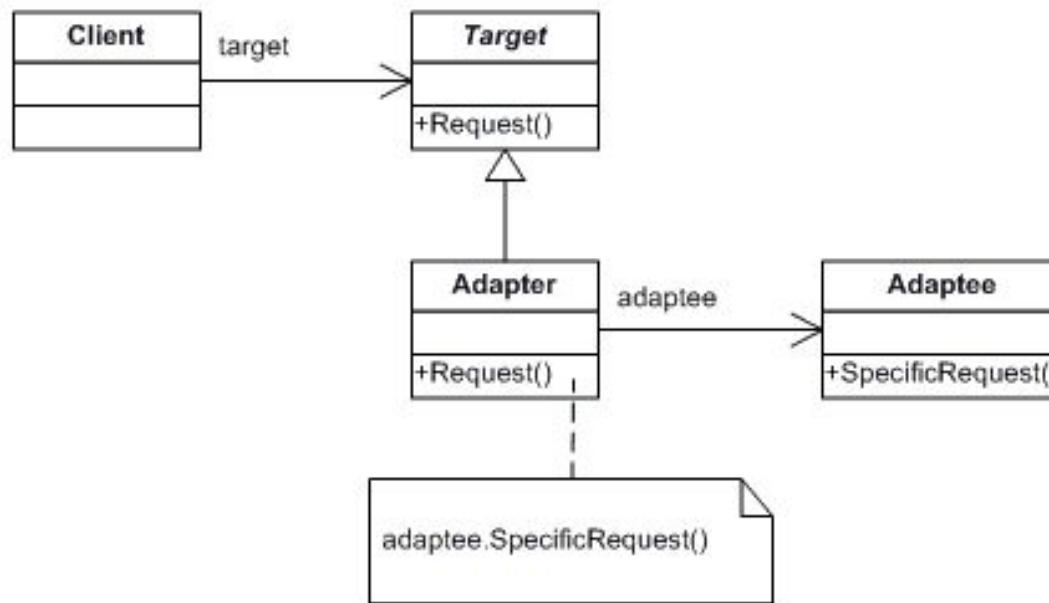
Adapter

¿Cuándo se utiliza?

- Se necesita utilizar una clase existente, pero su interfaz es incompatible con lo que se necesita
- Se quiere crear una clase que coopere con otras clases que utilizan interfaces incompatibles

Patrones estructurales

Adapter



Ejemplo

Patrones estructurales

Composite

Compone objetos en estructuras de árboles para representar jerarquías completas. *Composite* permite a los clientes tratar objetos y composiciones de objetos de igual manera.

Patrones estructurales

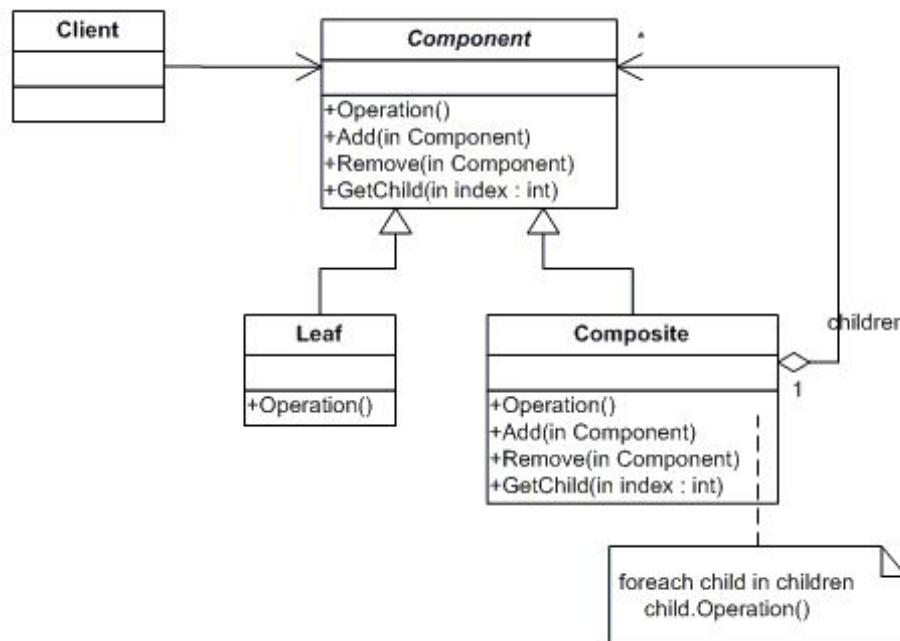
Composite

¿Cuándo se utiliza?

- Se quiere representar jerarquías completas de objetos
- Se quiere que los clientes puedan ignorar las diferencias entre objetos compuesto y objetos individuales. Los clientes tratarán a todos los objetos de la jerarquía de manera uniforme

Patrones estructurales

Composite



Ejemplo

Patrones estructurales

Decorator

Agrega responsabilidad adicional a un objeto de manera dinámica. Decoradores permiten una alternativa flexible para extender funcionalidad de sub-clases.

Patrones estructurales

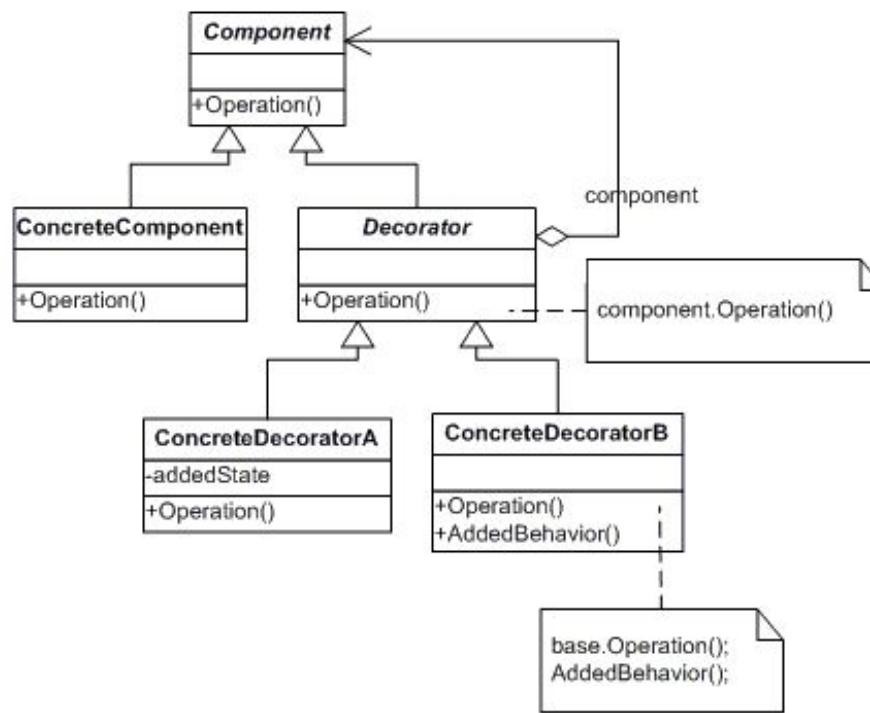
Decorator

¿Cuándo se utiliza?

- Para agregar responsabilidad a objetos individuales de manera dinámica y transparente (sin afectar otros objetos)
- Para responsabilidades que pueden ser retiradas
- Cuando extender con sub-clases es impracticable.
Puede ser por un gran número de extensiones independientes, o que no se pueden desprender sub-clases a partir de una clase

Patrones estructurales

Decorator



Ejemplo

Patrones estructurales

Bridge

Desacopla una abstracción de su implementación, para que ambas puedan variar independientemente.

Patrones estructurales

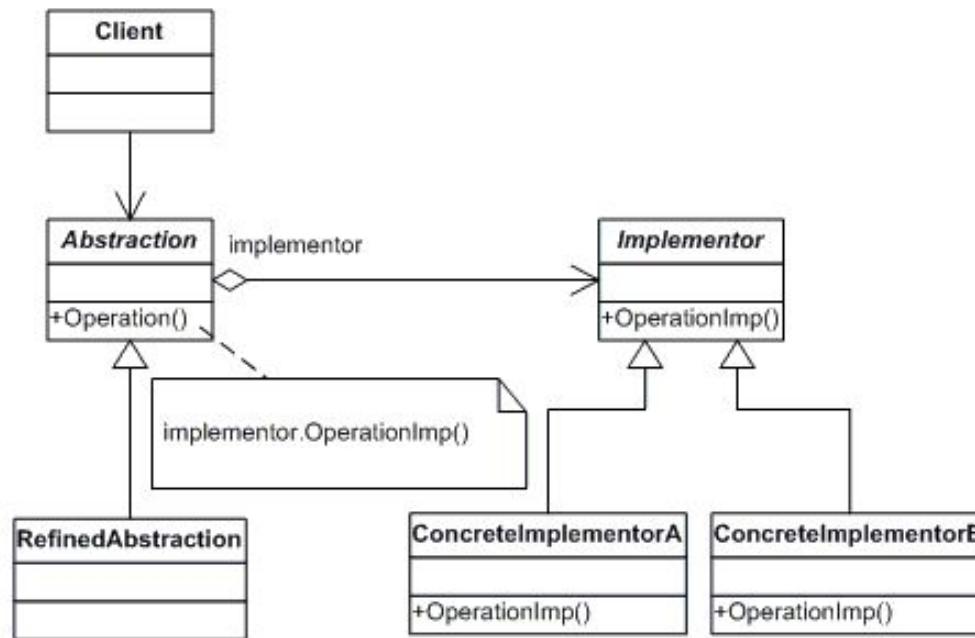
Bridge

¿Cuándo se utiliza?

- Se quiere evitar una unión permanente entre una abstracción y su implementación
- Una abstracción y su implementación deberían ser extensibles
- Cambios en la implementación de una abstracción no deberían tener impacto en los clientes

Patrones estructurales

Bridge



Ejemplo

Patrones estructurales

Proxy

Provee un sustituto para que ese objeto controle el acceso al original.

Patrones estructurales

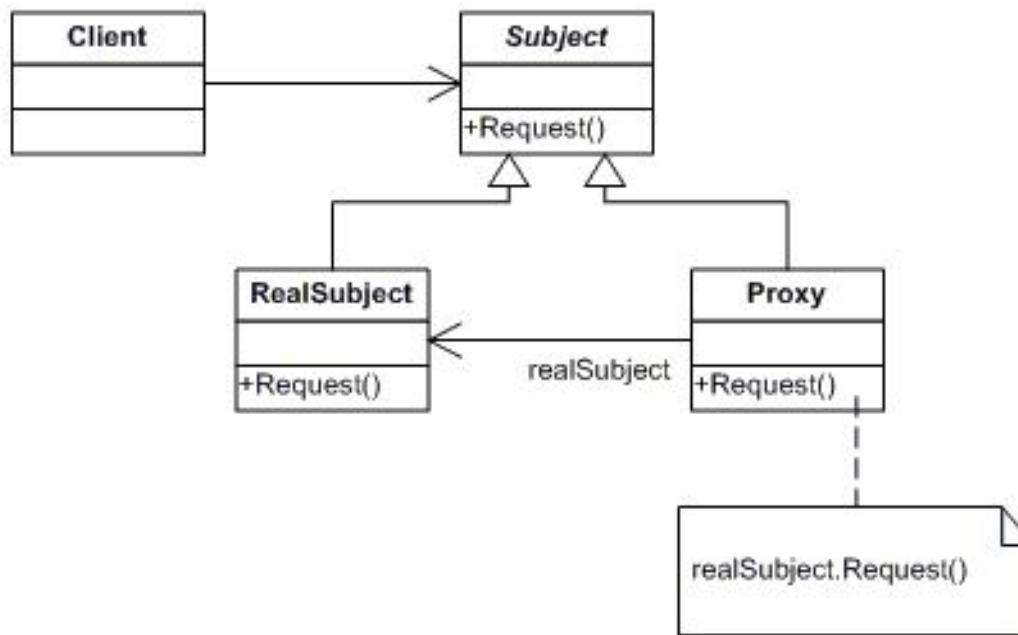
Proxy

¿Cuándo se utiliza?

- Se requiere una referencia más sofisticada que un puntero.

Patrones estructurales

Proxy



Ejemplo

Patrones estructurales

Flyweight

Utiliza valores por referencia para soportar grandes números de objetos.

Patrones estructurales

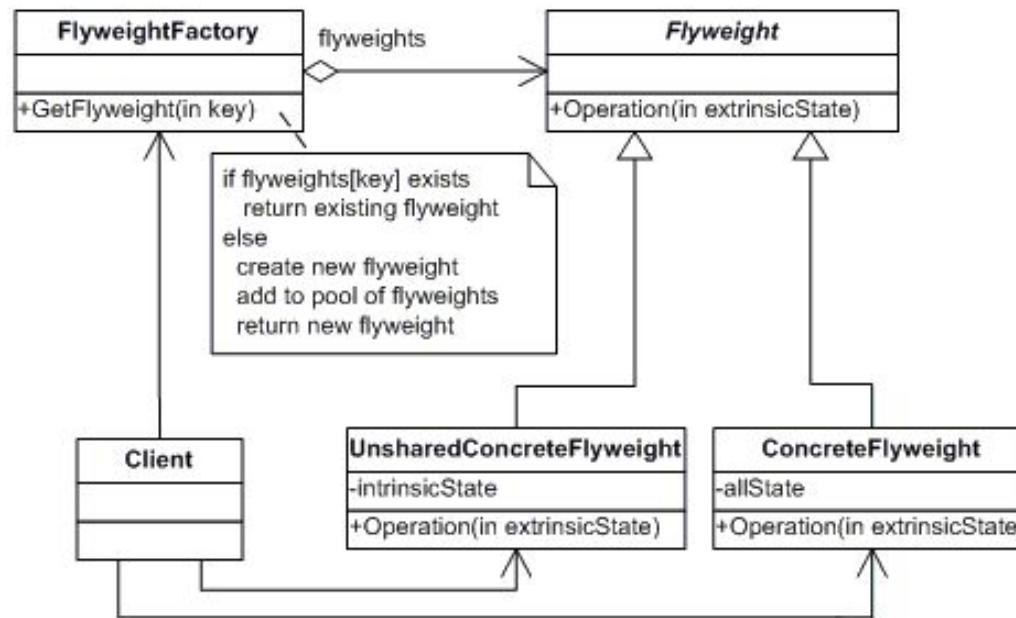
Flyweight

¿Cuándo se utiliza?

- Una aplicación utiliza un gran número de objetos.
- El costo de almacenamiento es caro dada la enorme cantidad de objetos.
- La aplicación no depende de la identidad exacta del objeto.
- La diferenciación de objetos es extrínseca.

Patrones estructurales

Flyweight



Ejemplo

Actividad 3



Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

Clase 3

Patrones de Diseño

IIC2113 – Diseño Detallado de Software

Rodrigo Saffie

rasaffie@uc.cl

31 de agosto de 2018