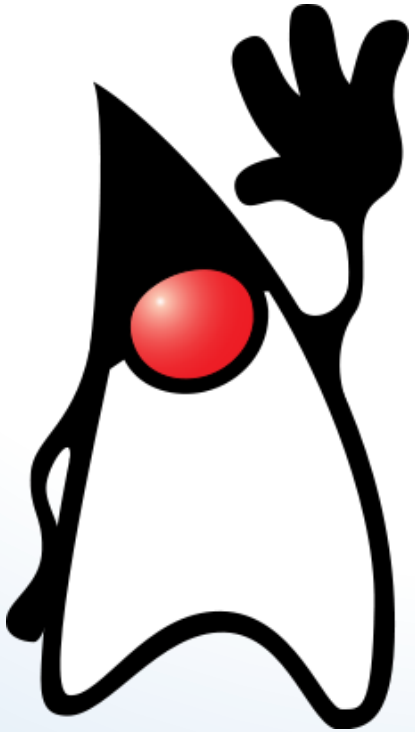
75.07 / 95.02 ALGORITMOS Y PROGRAMACIÓN III

.UBAfiuba
FACULTAD DE INGENIERÍA

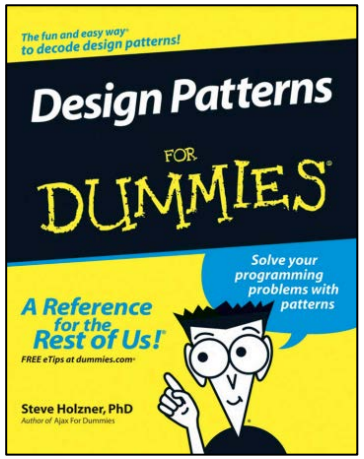
Patrones de diseño

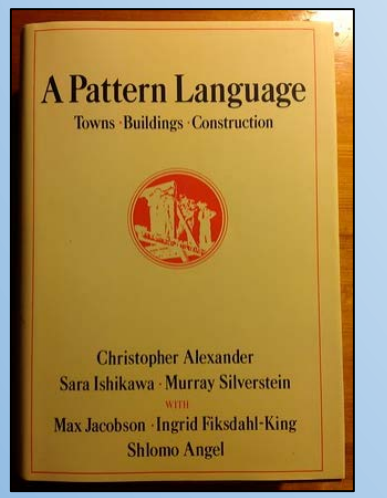


Cátedra: Corsi - Essaya - Maraggi

Congratulations, Your Problem Has Already Been Solved!

Título del capítulo 1 del libro “Design Patterns for Dummies”





Christopher Alexander (4/10/1936 – 17/3/2022)

“Cada **patrón** describe un **problema** que ocurre una y otra vez en nuestro entorno, así como la **solución** a ese problema, de tal modo que se pueda aplicar esta solución un millón de veces, sin hacer lo mismo dos veces” (Christopher Alexander, 1977).

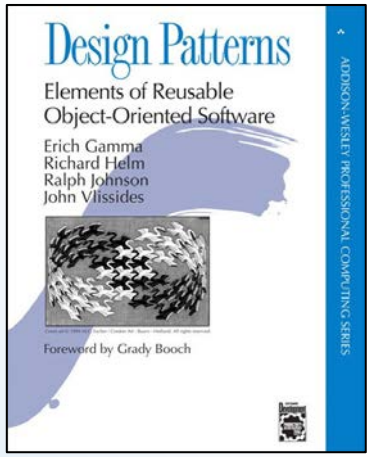
La banda de los 4 (The GoF)

Ralph Johnson

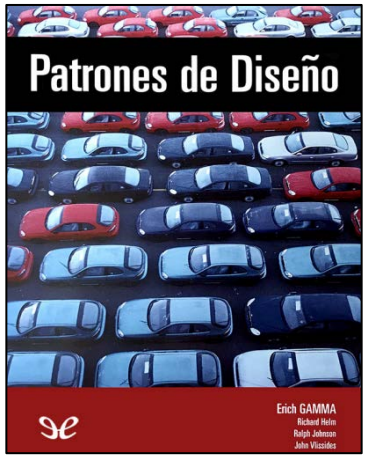
Erich Gamma

Richard Helm

John Vlissides



(1995)



(2002)

Cada patrón **nomina**, **explica** y **evalúa** un diseño importante y recurrente en los **sistemas orientados a objetos**. Nuestro objetivo es representar esa experiencia de diseño de forma que pueda ser reutilizada de manera efectiva **por otras personas**. Para lograrlo, hemos documentado algunos de los patrones de diseño más importantes y los presentamos como un **catálogo**. . . . Al expresar como patrones de diseño **técnicas que ya han sido probadas**, las estamos haciendo más accesibles para los desarrolladores de nuevos sistemas. Los patrones de diseño nos ayudan a elegir las **alternativas de diseño** que hacen que un sistema sea **reutilizable**, y a evitar aquéllas que dificultan dicha reutilización.

Plantilla con las secciones usadas por *The GoF* para describir los patrones

<Nombre del patrón> **<Clasificación>**

Propósito

Describe lo que hace el patrón en una breve oración.

También conocido como

Otros nombres, si existen, por los que se conoce al patrón.

Motivación

Brinda un escenario concreto que describe el problema y cómo la solución resuelve el problema.

Aplicabilidad

Describe las situaciones en las cuales se puede aplicar.

Estructura

Proporciona diagramas UML con las clases involucradas y sus relaciones, así como las interacciones entre los objetos.

Participantes

Son las clases y objetos del diseño. Esta sección describe sus responsabilidades y roles en el patrón.

Colaboraciones

Se indica cómo los participantes actúan juntos en el patrón.

Consecuencias

Describe los efectos que obtenemos al aplicar el patrón. Esto implica tanto los efectos deseados (*buenos*) como los no deseados (*malos*).

Implementación

Provee las técnicas que se necesita usar al implementar el patrón y menciona los problemas que se deben considerar.

Código de ejemplo

Fragmentos de código como ejemplo que puedan facilitar la implementación.

Usos conocidos

Ejemplos del patrón encontrados en sistemas reales.

Patrones relacionados

Describe la relación entre este patrón y otros.

Organización del catálogo de patrones de *The GoF*

		Propósito		
		De Creación	**Estructurales**	**De comportamiento**
Ámbito	**Clase**	Factory Method	Adapter (de clases)	Interpreter Template Method
	Objeto	Abstract Factory Builder Prototype Singleton	Adapter (de objetos) Bridge Composite Decorator Facade Flyweight Proxy	Chain of Responsibility Command Iterator Mediator Memento Observer State Strategy Visitor

https://github.com/bethrobson/Head-First-Design-Patterns/tree/master/src/headfirst/designpatterns

El **propósito** refleja qué hace un patrón. Los patrones **de creación** tienen que ver con la creación de objetos. Los patrones **estructurales** tratan con la composición de clases u objetos. Los **de comportamiento** caracterizan el modo en que las clases y objetos interactúan y se reparten la responsabilidad.

El **ámbito** especifica si el patrón se aplica principalmente a clases o a objetos. Los **patrones de clases** se ocupan de las relaciones entre las clases y sus subclases, que son relaciones estáticas —fijadas en tiempo de compilación—. Los **patrones de objetos** tratan con las relaciones entre objetos, que pueden cambiarse en tiempo de ejecución y son más dinámicas.

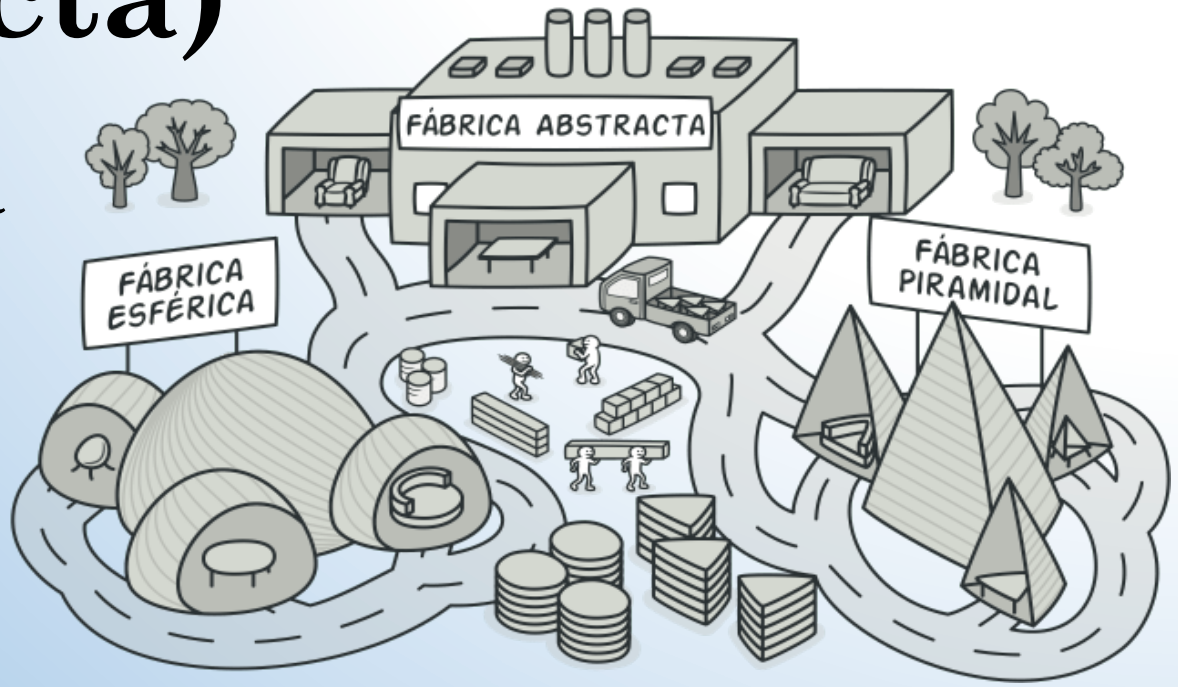
.UBAfiuba
FACULTAD DE INGENIERÍA

Los 23 patrones de *The GoF*

1. Abstract Factory (Fábrica Abstracta)

De Creación

Proporciona una interfaz para crear familias de objetos relacionados o que dependen entre sí, sin especificar sus clases concretas.



https://refactoring.guru/es/design-patterns/abstract-factory

Los 23 patrones de *The GoF*

2. Builder (Constructor)

De Creación

Separa la construcción de un objeto complejo de su representación, de forma que el mismo proceso de construcción pueda crear diferentes representaciones.

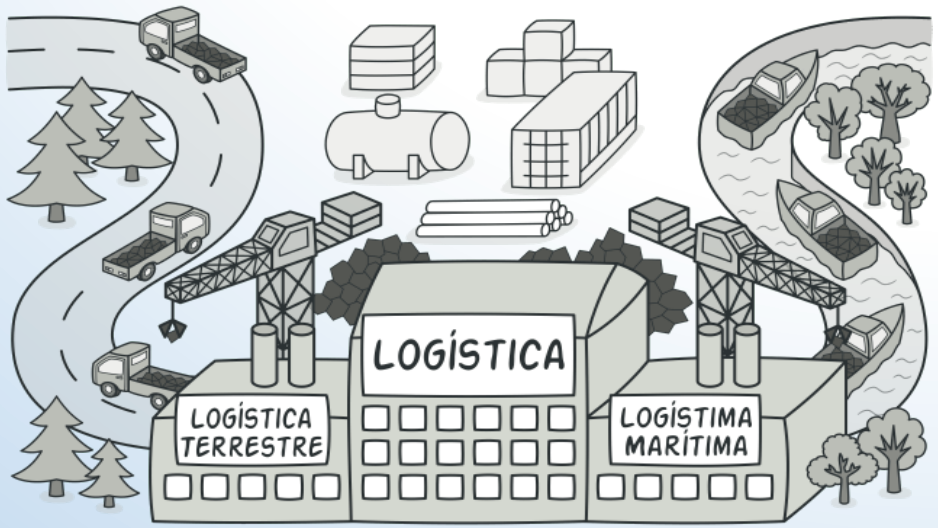
https://refactoring.guru/es/design-patterns/builder

Los 23 patrones de *The GoF*

3. Factory Method (Método de Fabricación)

De Creación

Define una interfaz para crear un objeto, pero deja que sean las sub-clases quienes decidan qué clase instanciar. Permite que una clase delegue en sus subclases la creación de objetos.



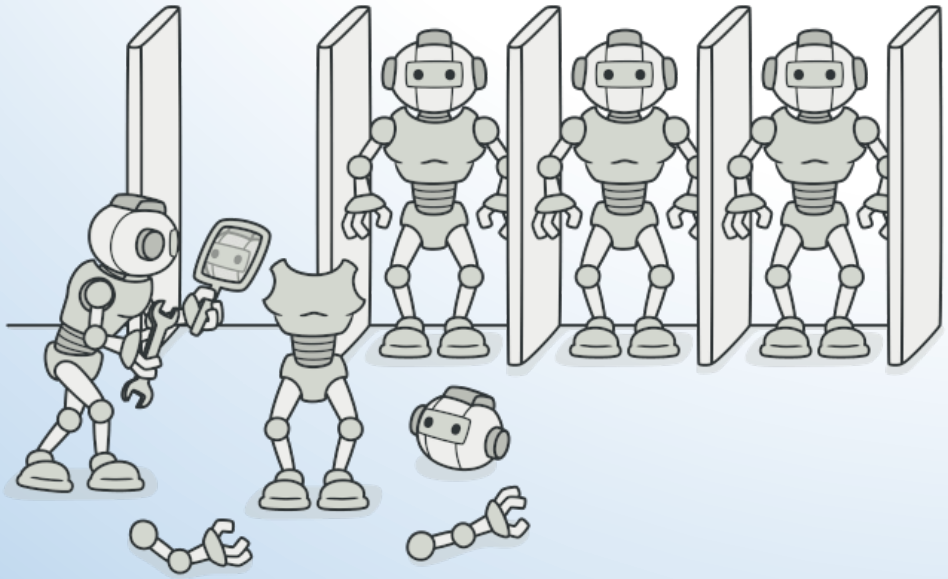
https://refactoring.guru/es/design-patterns/factory-method

Los 23 patrones de *The GoF*

4. Prototype (Prototipo)

De Creación

Especifica los tipos de objetos a crear por medio de una instancia prototípica, y crea nuevos objetos copiando de este prototipo.



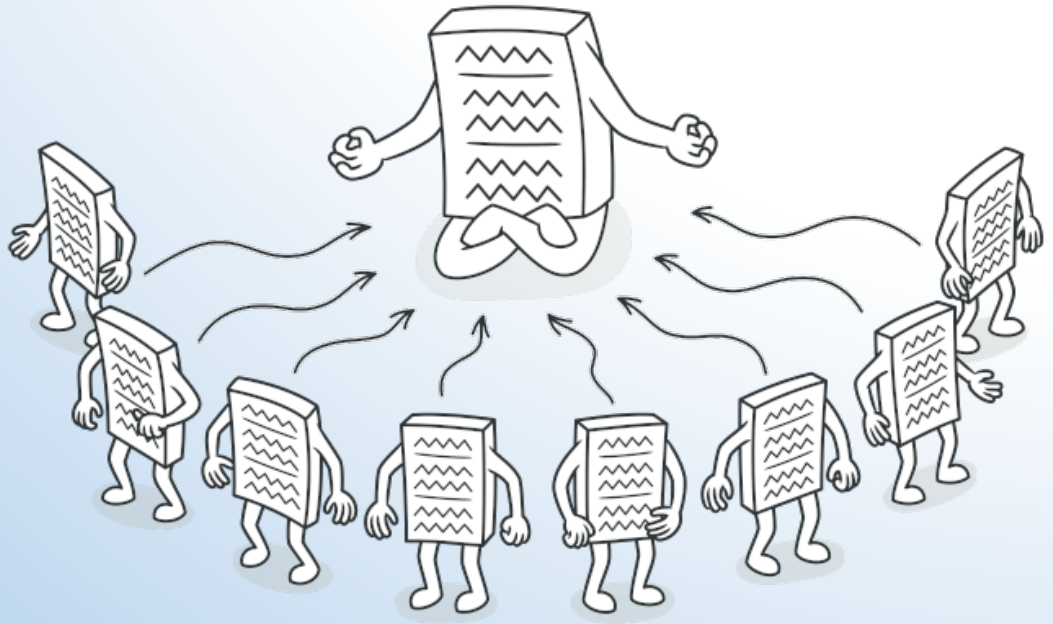
https://refactoring.guru/es/design-patterns/prototype

Los 23 patrones de *The GoF*

5. Singleton (Único)

De Creación

Garantiza que una clase sólo tenga una instancia, y proporciona un punto de acceso global a ella.



https://refactoring.guru/es/design-patterns/singleton

Los 23 patrones de *The GoF*

6. Adapter
(Adaptador)

Estructural

Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permite que cooperen clases que de otra manera no podrían por tener interfaces incompatibles.

https://refactoring.guru/es/design-patterns/adapter

Los 23 patrones de *The GoF*

7. Bridge (Puente)

Estructural

Desacopla una abstracción de su implementación, de manera que ambas puedan variar de forma independiente.

https://refactoring.guru/es/design-patterns/bridge

Los 23 patrones de *The GoF*

8. Composite (Compuesto)

Estructural

Combina objetos en estructuras de árbol para representar jerarquías de parte-todo. Permite que los clientes traten de manera uniforme a los objetos individuales y a los compuestos.

https://refactoring.guru/es/design-patterns/composite

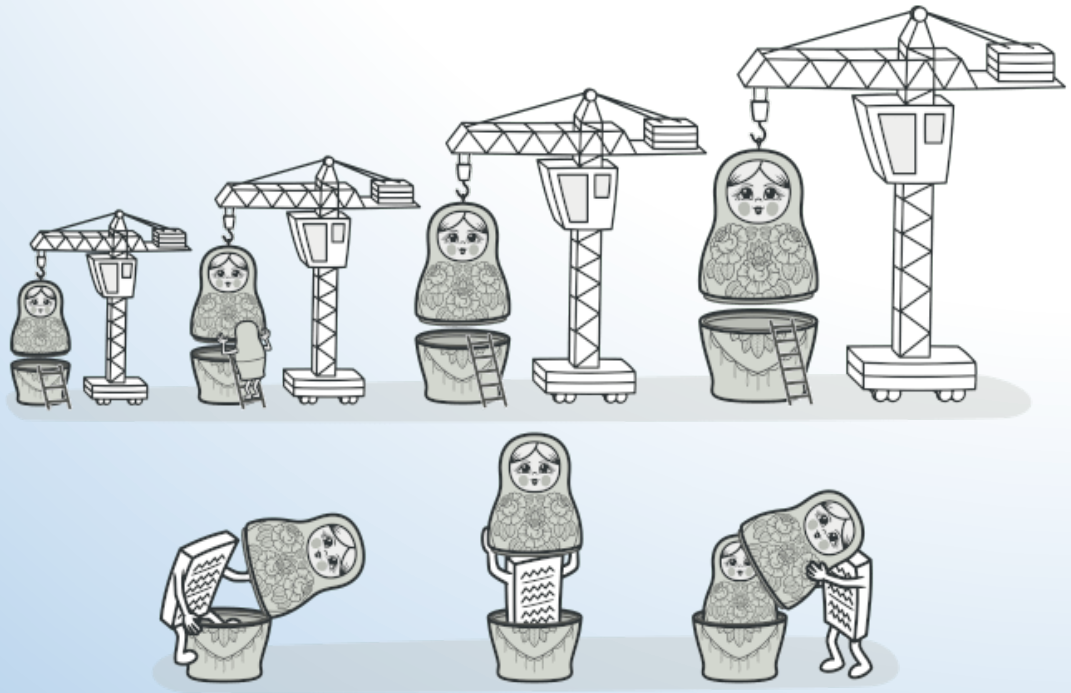
Los 23 patrones de *The GoF*

9. Decorator (Decorador)

Estructural

Añade dinámicamente nuevas responsabilidades a un objeto, proporcionando una alternativa flexible a la herencia para extender la funcionalidad.

https://refactoring.guru/es/design-patterns/decorator



Los 23 patrones de *The GoF*

10. Façade (Fachada)

Estructural

Proporciona una interfaz unificada para un conjunto de interfaces de un subsistema. Define una interfaz de alto nivel que hace que el subsistema sea más fácil de usar.

https://refactoring.guru/es/design-patterns/facade

Los 23 patrones de *The GoF*

11. Flyweight (Peso Ligero)

Estructural

Usa el compartimiento para permitir un gran número de objetos de grano fino de forma eficiente.

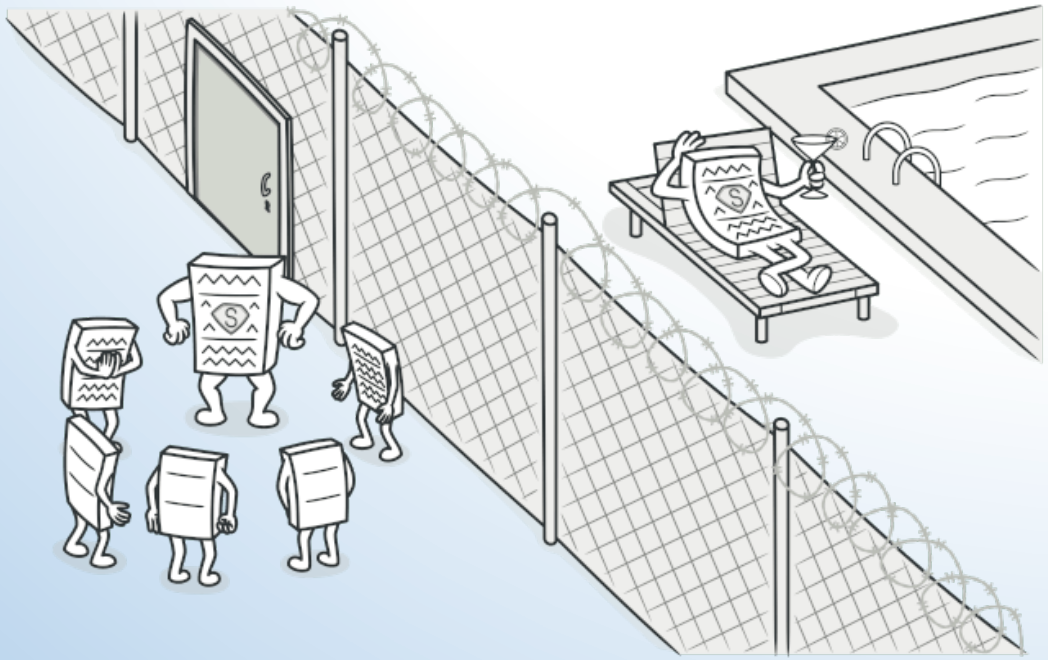
https://refactoring.guru/es/design-patterns/flyweight

Los 23 patrones de *The GoF*

12. Proxy (Apoderado)

Estructural

Proporciona un sustituto o representante de otro objeto para controlar el acceso a éste.



https://refactoring.guru/es/design-patterns/proxy

Los 23 patrones de *The GoF*

De Comportamiento

13. Chain of Responsibility (Cadena de Responsabilidad)

Evita acoplar el emisor de una petición a su receptor, al dar a más de un objeto la posibilidad de responder a la petición. Crea una cadena con los objetos receptores y pasa la petición a través de la cadena hasta que ésta sea tratada por algún objeto.

https://refactoring.guru/es/design-patterns/chain-of-responsibility

Los 23 patrones de *The GoF*

14. Command (Orden)

De Comportamiento

Encapsula una petición en un objeto, permitiendo así parametrizar a los clientes con distintas peticiones, encolar o llevar un registro de las peticiones y poder deshacer las operaciones.

https://refactoring.guru/es/design-patterns/command

Los 23 patrones de *The GoF*

15. Interpreter (Intérprete)

De Comportamiento

Dado un lenguaje, define una representación de su gramática junto con un intérprete que usa dicha representación para interpretar sentencias del lenguaje.

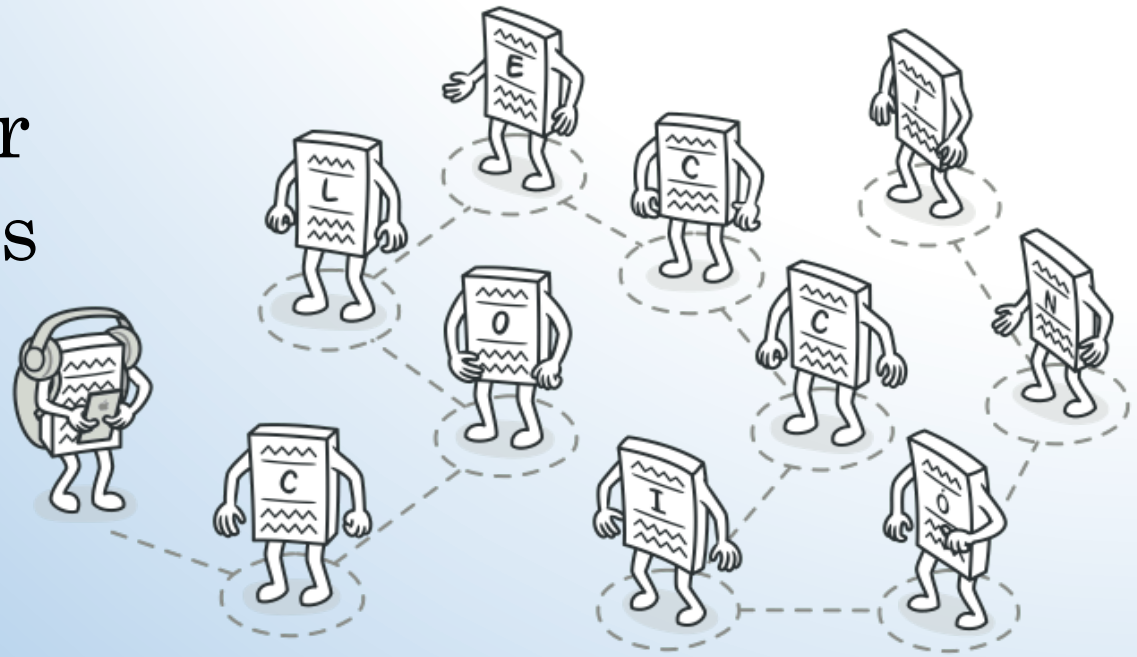
Los 23 patrones de *The GoF*

16. Iterator
(Iterador)

De Comportamiento

Proporciona un modo de acceder secuencialmente a los elementos de un objeto agregado sin exponer su representación interna.

https://refactoring.guru/es/design-patterns/iterator



Los 23 patrones de *The GoF*

17. Mediator (Mediador)

De Comportamiento

Define un objeto que encapsula cómo interactúan un conjunto de objetos. Promueve un bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente, y permite variar la interacción entre ellos de forma independiente.

https://refactoring.guru/es/design-patterns/mediator

Los 23 patrones de *The GoF*

18. Memento (Recuerdo)

De Comportamiento

Representa y externaliza el estado interno de un objeto sin violar la encapsulación, de forma que éste puede volver a dicho estado más tarde.

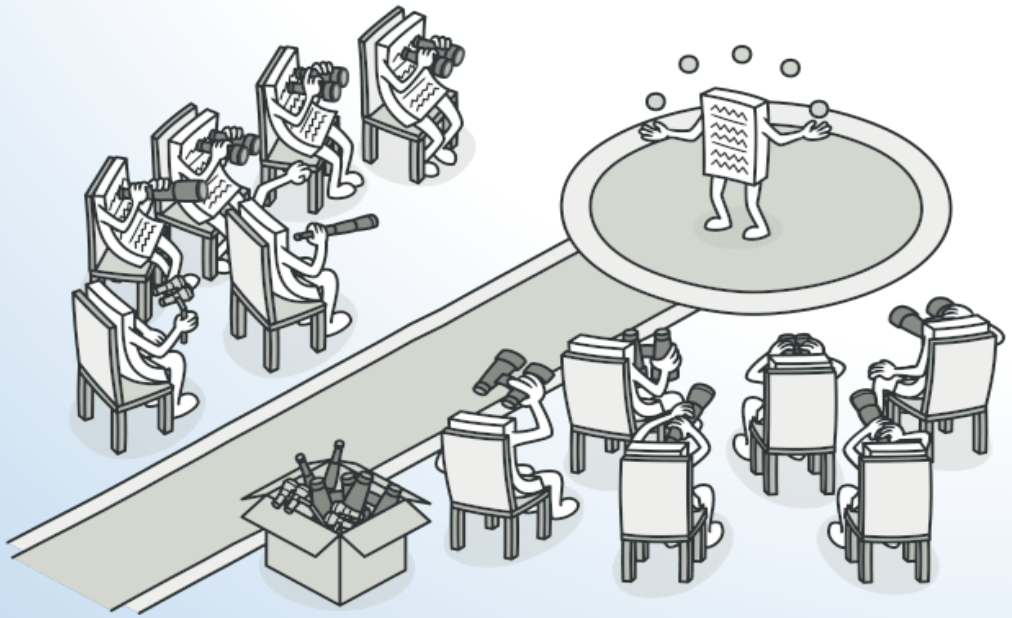
https://refactoring.guru/es/design-patterns/memento

Los 23 patrones de *The GoF*

19. Observer (Observador)

De Comportamiento

Define una dependencia de uno-a-muchos entre objetos, de forma que cuando un objeto cambie de estado se notifica y se actualizan automáticamente todos los objetos que dependen de él.



https://refactoring.guru/es/design-patterns/observer

.UBAfiuba
FACULTAD DE INGENIERÍA

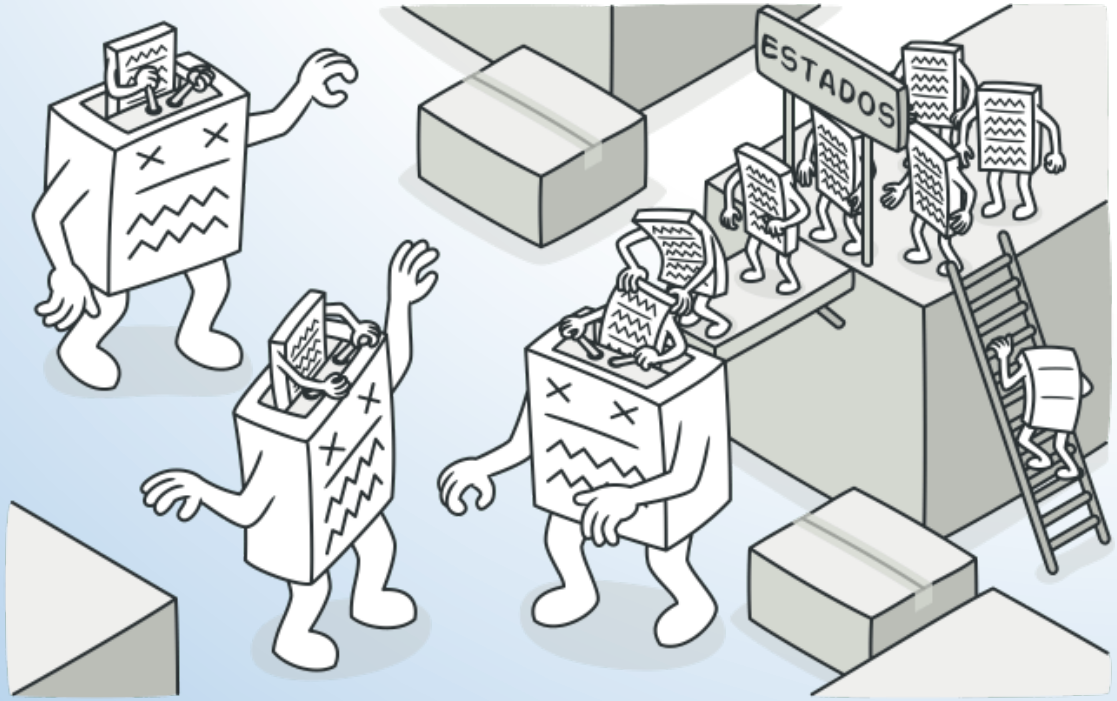
Los 23 patrones de *The GoF*

20. State (Estado)

De Comportamiento

Permite que un objeto modifique su comportamiento cada vez que cambie su estado interno. Parecerá que cambia la clase del objeto.

https://refactoring.guru/es/design-patterns/state



Los 23 patrones de *The GoF*

21. Strategy (Estrategia)

De Comportamiento

Define una familia de algoritmos, encapsula cada uno de ellos y los hace intercambiables. Permite que un algoritmo varíe independientemente de los clientes que lo usan.

https://refactoring.guru/es/design-patterns/strategy

Los 23 patrones de *The GoF*

22. Template Method (Método Plantilla)

De Comportamiento

Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos de sus pasos. Permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar su estructura.

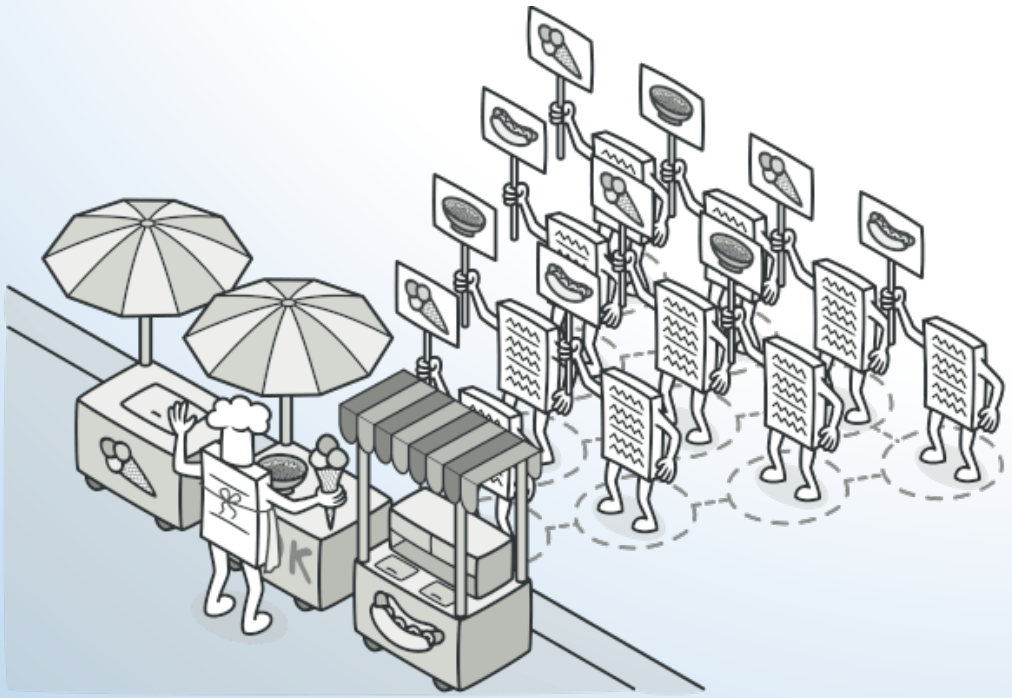
https://refactoring.guru/es/design-patterns/template-method

Los 23 patrones de *The GoF*

23. Visitor (Visitante)

De Comportamiento

Representa una operación sobre los elementos de una estructura de objetos. Permite definir una nueva operación sin cambiar las clases de los elementos sobre los que opera.



https://refactoring.guru/es/design-patterns/visitor

BULLET POINTS

- Conocer los conceptos básicos de OO no hace de uno un buen diseñador.
- Los buenos diseños son reutilizables, extensibles y mantenibles.
- Los patrones muestran cómo construir sistemas con buenas cualidades de diseño.
- Los patrones son **experiencia con orientación a objetos** probada.
- Los patrones no dan código, dan soluciones. Uno los aplica al producto específico.
- Los patrones **no se inventan**, **se descubren**.
- La mayoría de los patrones y principios abordan cuestiones de **cambio** del software.
- La mayoría de los patrones permiten que alguna parte de un sistema varíe independientemente de todas las demás partes.
- A menudo tratamos de tomar lo que varía en un sistema y encapsularlo.
- Los patrones proporcionan un **lenguaje compartido** que puede maximizar el valor de la comunicación con otros desarrolladores.

Conclusión

Hay que concentrarse en el buen **diseño**, no en los patrones.

Solo deberían usarse patrones cuando haya una necesidad real para ellos.

Si algo más simple funciona, entonces es mejor no usar un patrón.