TECNOLÓGICO NACIONAL DE MÉXICO
INSTITUTO TECNOLÓGICO DE TIJUANA

SUBDIRECCIÓN ACADÉMICA

DEPARTAMENTO DE SISTEMAS Y COMPUTACIÓN

Tarea

Clasificación de Patrones de diseño GOF

Docente
José de Jesús Parra Galaviz

SEMESTRE agosto – diciembre 2021

INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN Y COMUNICACIONES

Patrones de diseño

Sergio Alberto Garza Aguilar 15211700

29 de agosto del 2021

Clasificación de los patrones de diseño

Catálogo de patrones de diseño

Los patrones varían en su nivel de abstracción y dado que existen numerosos patrones de diseño es necesario organizarlos por familias.

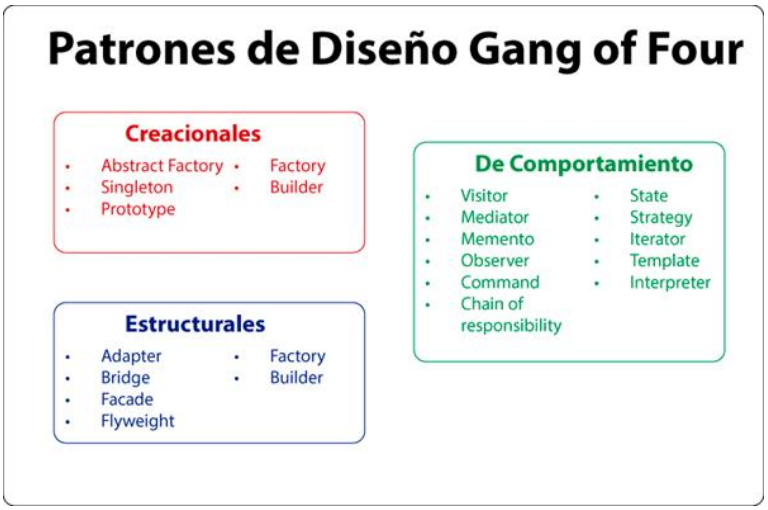
El **Propósito** refleja que hace un patrón y puede ser:

1. **De Creación**: Están relacionados con el proceso de creación o instanciación de objetos.
2. **Estructural**: Están relacionados con la composición de clases y objetos.
3. **De Comportamiento**: Están relacionados con el modo en que las clases y objetos interactúan y el modo en que se reparten las responsabilidades.

El **Ámbito** especifica a quién aplica el patrón y puede ser:

Clases:
Se ocupan de las relaciones entre las clases y subclases. Las relaciones entre clase se establecen a través de la herencia de modo que son relaciones estáticas establecidas en tiempo de compilación.

Objetos:
Se ocupan de las relaciones entre objetos que son relaciones dinámicas y pueden cambiar en tiempo de ejecución.
En la siguiente tabla podemos ver los patrones organizados por dos criterios “propósito” y “ámbito”.



Propósito de Creación

- **Factory Method (Método de Fabricación)**

Define una interfaz para crear un objeto, pero deja que las subclases decidan que clase se instancia. Permite que una clase delegue en sus subclases la creación de objetos

- **Abstract Factory (Fábrica Abstracta)**

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

- **Builder (Constructor)**

Separa la construcción de un objeto completo de su representación. De este modo el mismo proceso de construcción puede crear diferentes representaciones.

- **Prototype (Prototipo)**

Especifica los tipos de objetos a crear por medio de una instancia prototípica y crea nuevos objetos haciendo copias del objeto prototipo.

- **Singleton (Único)**

Garantiza que sólo exista una instancia de la Clase y proporciona un punto de acceso global a esa instancia.

Propósito de Estructura

- **Adapter (Adaptador)**

Convierte la interfaz de una clase en otra distinta que es la que esperan los clientes. Permite integrar clases con interfaces incompatibles.

- **Bridge (Puente)**

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

- **Composite (Compuesto)**

Combina objetos en estructuras de árbol para representar jerarquías. Permite que los clientes traten de la misma manera a los objetos individuales de los compuestos.

- **Decorator (Decorador)**

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

- **Facade (Fachada)**

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

- **Flyweight (Peso Ligero)**

Usa el comportamiento para permitir un gran número de objetos pequeños de forma eficiente.

- **Proxy (Apoderado)**

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

Propósito de Comportamiento

- **Interpreter (Intérprete)**

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

- **Template Method (Método de plantilla)**

Define en una operación el esqueleto de un algoritmo, delegando en las subclases algunos pasos. Es decir, permite que las subclases redefinan ciertos pasos del algoritmo sin cambiar la estructura.

- **Chain of Responsibility (Cadena de Responsabilidad)**

Desacopla el emisor de una petición del receptor. Crea una cadena de objetos receptores que tienen la posibilidad de responder a la petición y la petición pasa a través de la cadena hasta que uno de los receptores la trata.

- **Command (Orden)**

Encapsula una petición en un objeto, permitiendo parametrizar a los clientes con distintas peticiones.

- **Iterator (Iterador)**

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

- **Mediator (Mediador)**

Define un objeto que encapsula cómo interactúan un conjunto de objetos. Consigue bajo acoplamiento al evitar que los objetos se refieran unos a otros explícitamente.

- **Memento (Recuerdo)**

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

- **Observer (Observador)**

Define una dependencia de uno a muchos entre objetos, de forma que cuando un objeto cambie de estado se notifican todos los objetos que dependen de él.

- **Strategy (Estrategia)**

Define una familia de algoritmos, encapsula cada uno de ellos y los hace intercambiables. Permite que un algoritmo cambie independientemente del cliente que lo usa.

- **Visitor (Visitante)**

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.

Otra posibilidad interesante para categorizar los patrones es agruparlos en función de cómo unos patrones hacen referencia a otros.

Es importante tener diferentes puntos de vista al pensar en patrones de diseño, de esta manera lograremos comprender mejor lo que hacen, compararlos entre ellos y elegir cual resuelve de una manera más eficaz el problema concreto que queremos resolver.

Fuente.

Miguel Gomez Cuesta, (06 de abril del 2016), Aprende Java en español, Agrupacion Gof (Gang of Four) (http://javaespanol.blogspot.com/2016/04/patrones-de-diseno-agrupacion-gof-gang.html)

Erich Gamma,(21 de octubre de 1994), Design Patterns: Elements of Reusable Object Oriented Software, (https://profeuttec.yolasite.com/resources/Patrones%20de%20dise%C3%B1o%20-%20Erich%20Gamma.pdf)