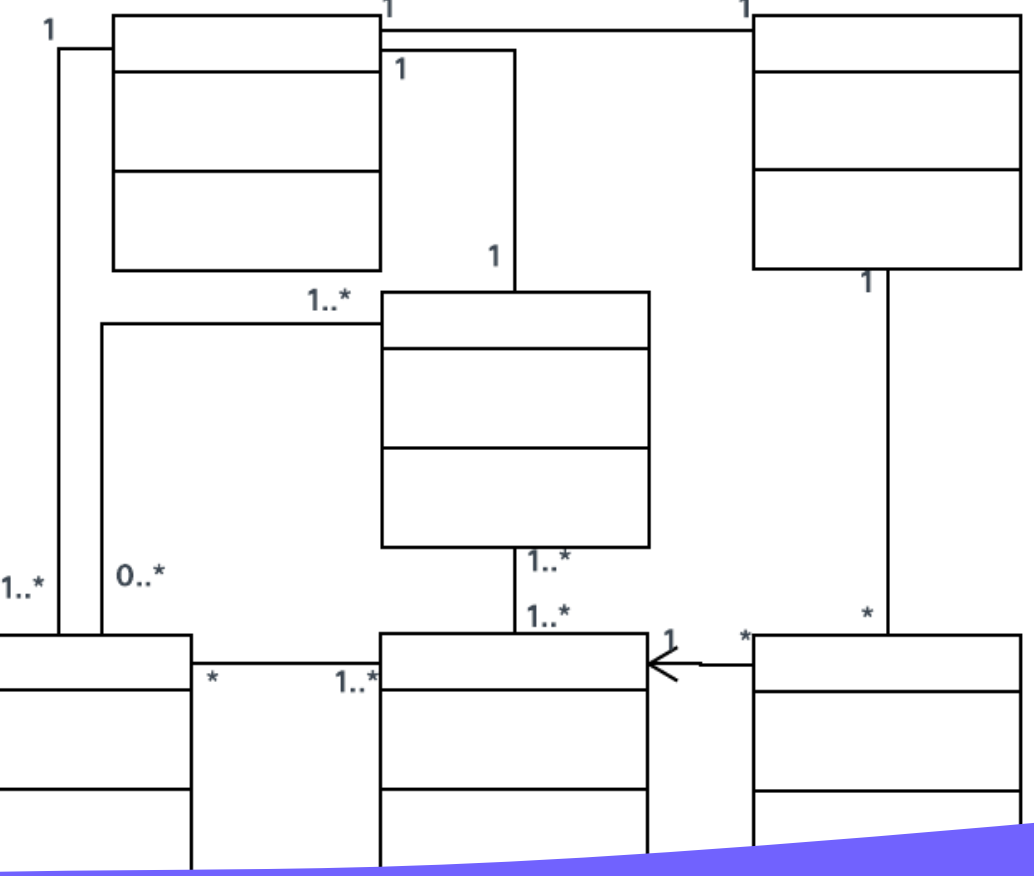
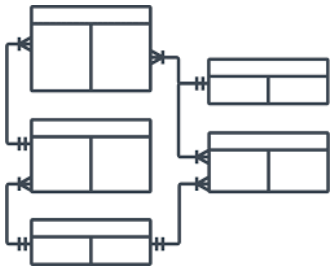


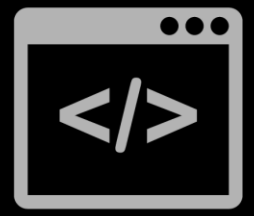


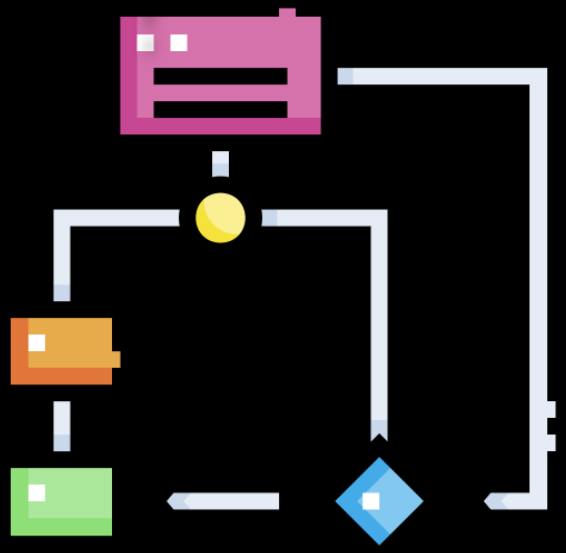
Patrones de Diseño

Agenda

- Patrones de Diseño y sus clasificaciones
- Patrón State
- Patrón Template Method

Patrones de Diseño





Patrones de Diseño

"Son descripciones de clases y objetos relacionados que están adaptados para resolver un problema de diseño general en un contexto determinado"

Erich Gamma, Richard Helm, John Vlissides y Ralph Johnson

"Consiste en un diagrama de objetos que forma una solución a un problema conocido y frecuente"

Laurent Debrauwer - Patrones de diseño en Java

Patrones de Diseño

"Soluciones conocidas a problemas conocidos y reiterados en el mundo del Desarrollo de Software"

Patrones de Diseño

El objetivo es reutilizar la experiencia de quienes ya se han encontrado con problemas similares y han encontrado una buena solución.

Patrones de Diseño - Estructura

- Propósito
- Motivación
- Participantes
- Colaboraciones
- Consecuencias
- Implementación Código de ejemplo
- Usos conocidos
- Patrones relacionados

Patrones de Diseño – Partes esenciales

- ***Nombre***: Comunica el objetivo del patrón en una o dos palabras. Aumenta el vocabulario sobre diseño.
- ***Problema***: Describe el problema que el patrón soluciona y su contexto. Indica cuándo se aplica el patrón.
- ***Solución***: Indica cómo resolver el problema en términos de elementos, relaciones, responsabilidades y colaboraciones. La solución debe ser lo suficientemente abstracta para poder ser aplicada en diferentes situaciones.
- ***Consecuencias***: Indica los efectos de aplicar la solución. Son críticas al momento de evaluar distintas alternativas de diseño.

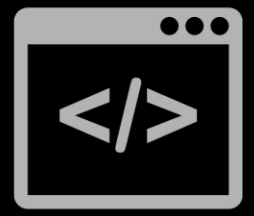
Patrones de Diseño – Clasificación

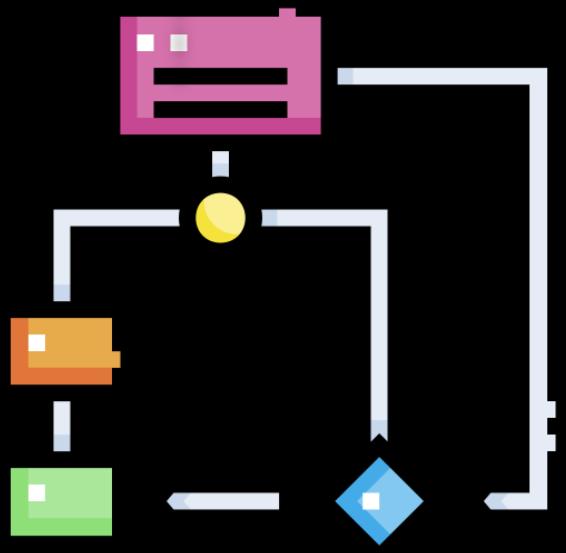
Creacionales	Comportamiento	Estructurales
• Abstraen el proceso de creación/instanciación de los objetos. Se los suele utilizar cuando debemos crear objetos, complejos o no, tomando decisiones dinámicas en momento de ejecución.	• Resuelven cuestiones, complejas o no, de interacción entre objetos en momento de ejecución.	• Resuelven cuestiones, generalmente complejas, de generación y/o utilización de estructuras complejas o que no están acopladas al dominio.

Patrones de Diseño

Creacionales	Comportamiento	Estructurales
• Factory Method • Simple Factory • Singleton • Abstract Factory • Builder • Prototype	• State • Strategy • Observer • Command • Template method • Iterator • Memento • Visitor • Interpreter • Chain of Responsability • Mediator	• Adapter • Composite • Facade • Decorator • Proxy • Flyweight • Bridge

Patrón Strategy





Patrón Strategy

Es un patrón de Comportamiento

¿Qué hace?

- Encapsula distintas formas (o algoritmos) de resolver el mismo problema en diferentes clases.
- Permite intercambiar en momento de ejecución la forma en que un tercero resuelve un problema.

Patrón Strategy

Se sugiere su utilización cuando:

- Se requiere que un objeto realice una misma acción pero con un algoritmo distinto o de una forma distinta.
- Existen muchas formas de realizar la misma acción (pero con distintos pasos) en el mismo objeto.
- Se requiere permitir configurar en momento de ejecución la forma en que un objeto realizará una acción.

Patrón Strategy

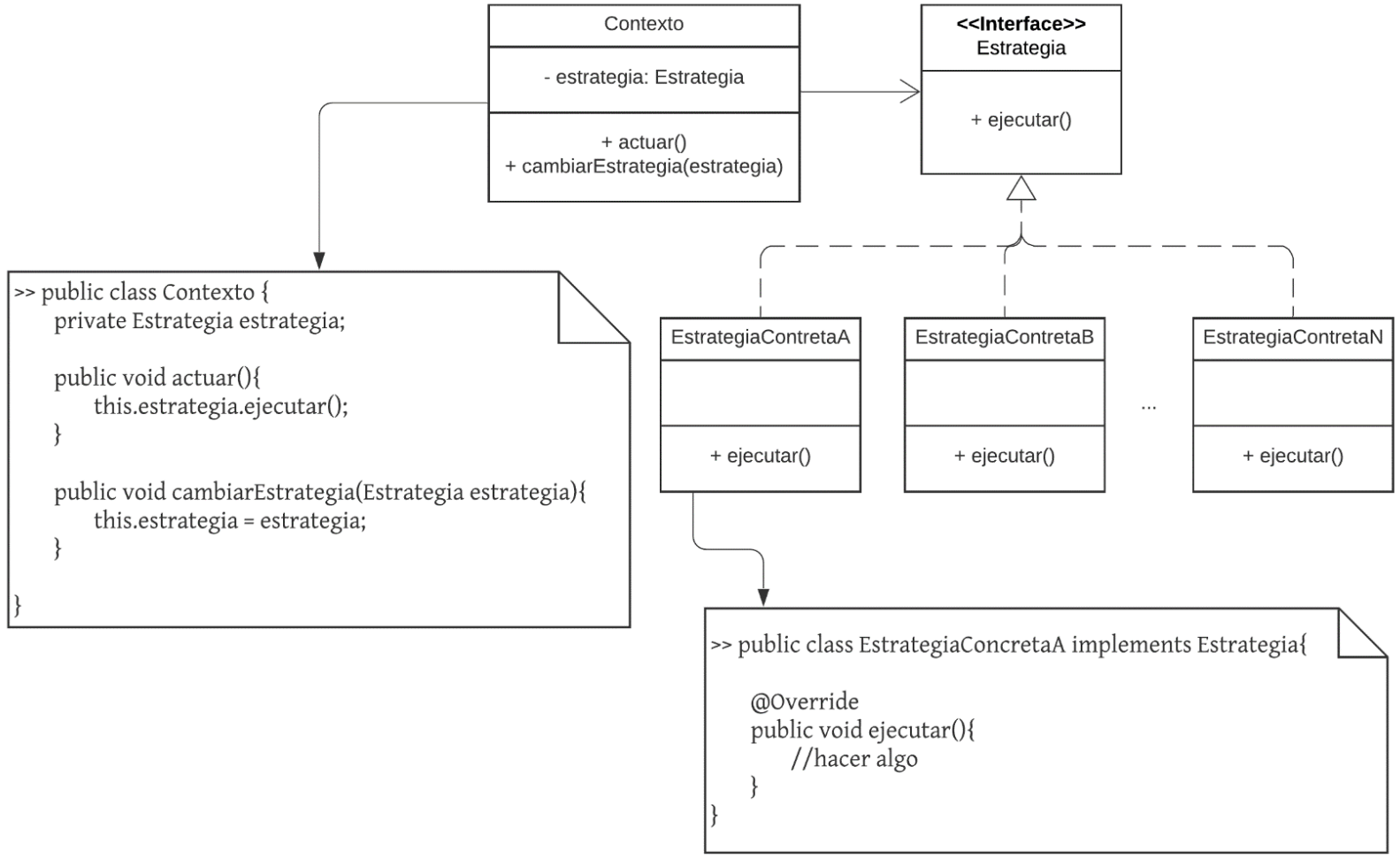
Componentes:

- ***Interface (o clase abstracta) Strategy***: Define la firma del método que se utilizará como estrategia de resolución de algún problema.

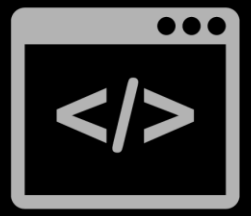
- ***Clases de estrategias concretas***: Clases que implementan la interface Strategy (o que heredan de ella, si ésta fuera clase abstracta), es decir, que tienen la implementación real del algoritmo.

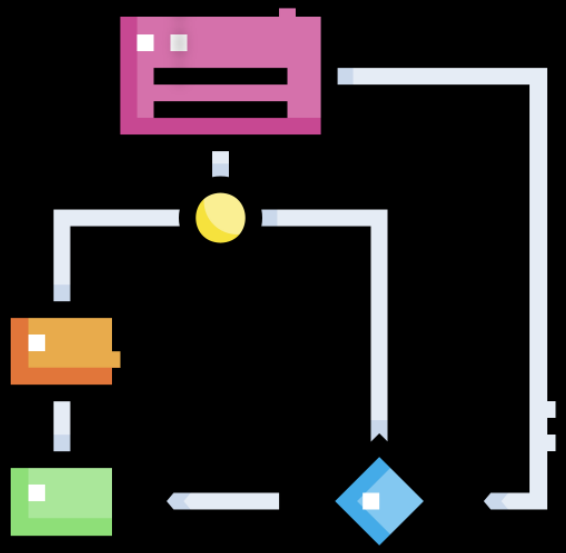
- ***Contexto***: Clase que tiene referencia a la interface/clase abstracta Strategy, cuyos objetos van a delegar la responsabilidad de resolución de algún problema en la estrategia. Estos objetos utilizarán de forma polimórfica a las estrategias concretas.

Patrón Strategy



Patrón Adapter





Patrón Adapter

Es un patrón Estructural

¿Qué hace?

- Encapsula el uso (llamadas/envío de mensajes) de la clase que se quiere adaptar en otra clase que concuerda con la interfaz requerida.

Patrón Adapter

Se sugiere su utilización cuando:

- Se requiere seguir adelante con el diseño y/o implementación (código) sin conocer exactamente cómo, quién y cuándo resolverá una parte necesaria; y solamente se conoce qué responsabilidad tendrá dicha parte.

- Se requiere usar una clase que ya existe, pero su interfaz no concuerda con la que se necesita.

Patrón Adapter

Componentes:

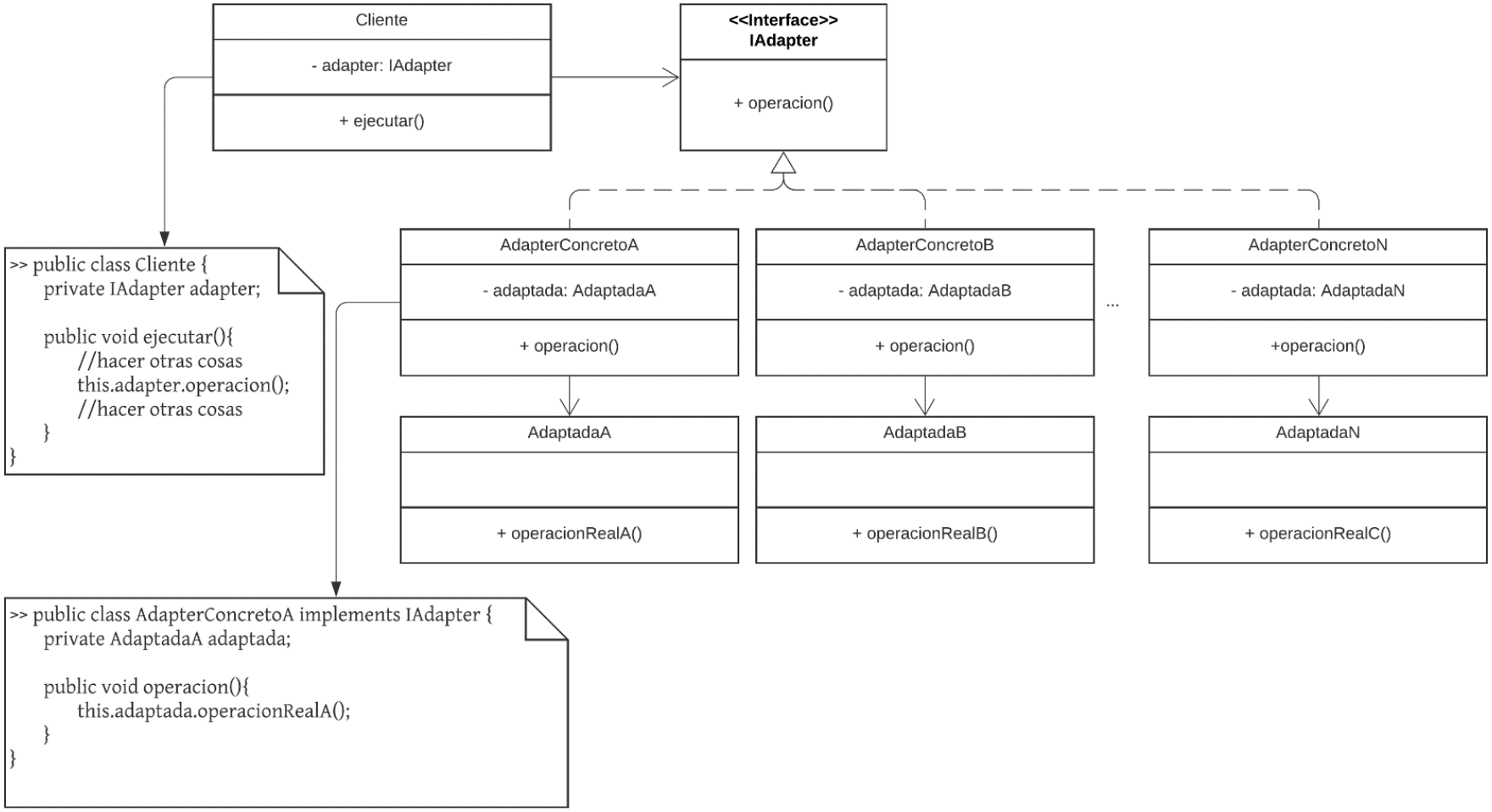
- ***Interface Adapter***: Define la firma del método que se utilizará como "puente" de acoplamiento a nuestro dominio.

- ***Clases de Adapters concretos***: Clases que implementan la interface Adapter, que se encargan de acoplar los componentes externos al dominio. Guardan referencia o hacen uso de las clases Adaptadas. Llaman a los métodos "reales" que resuelven el problema.

- ***Clases Adaptadas***: Clases externas al dominio, o no, que posee firmas incompatibles, o que todavía no conocemos (puede que todavía no estén creadas). Estas clases no se pueden/deben tocar.

- ***Cliente***: Clase que tiene referencia a la interface Adapter, cuyos objetos van a hacer uso de la funcionalidad que ésta les brinda.

Patrón Adapter



Gracias