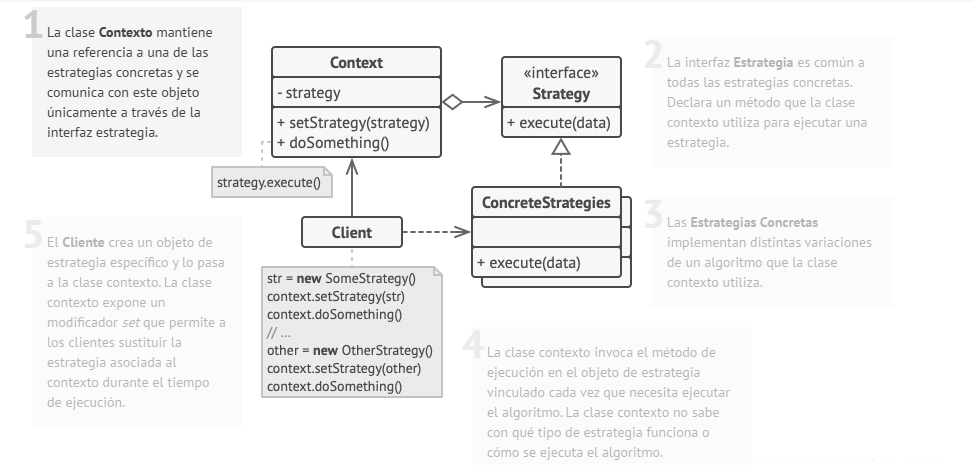
STRATEGY   


**Nombre del patrón:** Strategy  
**Tipo de patrón:** Comportamiento (Behavioral)

**Descripción:**  
El patrón Strategy permite definir una familia de algoritmos, encapsular cada uno de ellos y hacerlos intercambiables sin afectar el cliente que los utiliza. El algoritmo puede ser cambiado en tiempo de ejecución según el contexto. Esto favorece la flexibilidad y la reutilización del código.

**Características principales:**

* Separa el comportamiento de un algoritmo del objeto que lo usa.
* Facilita la adición de nuevos algoritmos sin modificar el código existente.
* Promueve el uso de la composición sobre la herencia.
* Evita el uso extensivo de estructuras condicionales como if o switch.
* Favorece el principio de abierto/cerrado: las clases están abiertas a la extensión pero cerradas a la modificación.

**Componentes:**

1. **Strategy:** Interfaz común para todos los algoritmos posibles.
2. **ConcreteStrategy:** Implementaciones específicas de los algoritmos.
3. **Context:** Clase que utiliza una estrategia concreta y puede cambiarla dinámicamente.

**Ventajas:**

* Facilita el mantenimiento del código.
* Aumenta la modularidad y la cohesión.
* Permite cambiar el comportamiento del programa en tiempo de ejecución.

**Desventajas:**

* Puede aumentar el número de clases en el sistema.
* El cliente debe conocer las diferentes estrategias para seleccionarlas adecuadamente.

**Aplicaciones comunes:**

* Selección dinámica de algoritmos de ordenamiento, búsqueda o compresión.
* Cálculo de tarifas, descuentos o impuestos según condiciones específicas.
* Implementación de comportamientos configurables en sistemas complejos.