



Patrones de Diseño:
Patrones de
Comportamiento.
Tema 5-10:
Strategy

Descripción del patrón

- **Nombre:**
 - Estrategia
 - También conocido como Policy (política)
- **Propiedades:**
 - Tipo: comportamiento
 - Nivel: objeto, componente
- **Objetivo o Propósito:**
 - Definir un grupo de clases que representan un conjunto de posibles comportamientos. Estos comportamientos pueden ser fácilmente intercambiados en una aplicación, modificando la funcionalidad en cualquier instante.

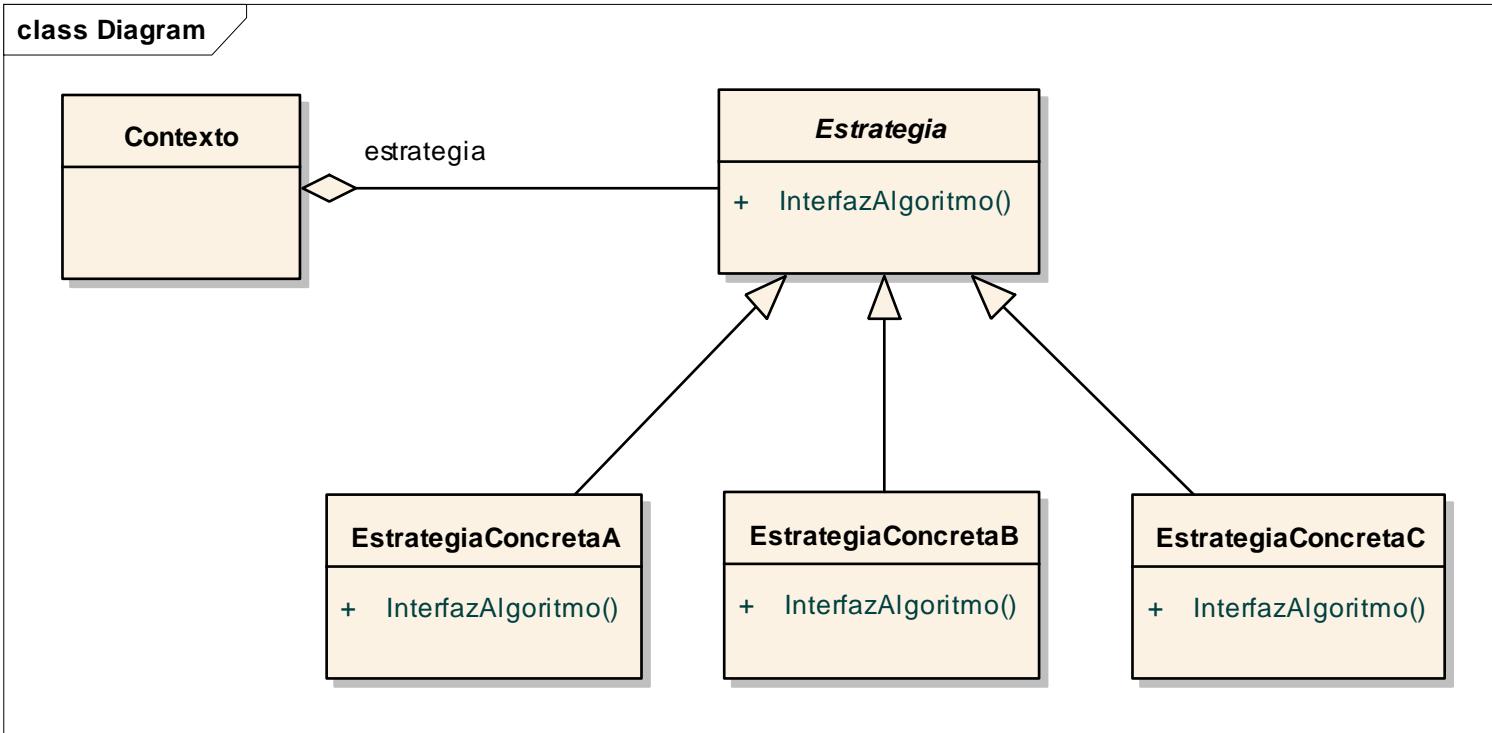


Aplicabilidad

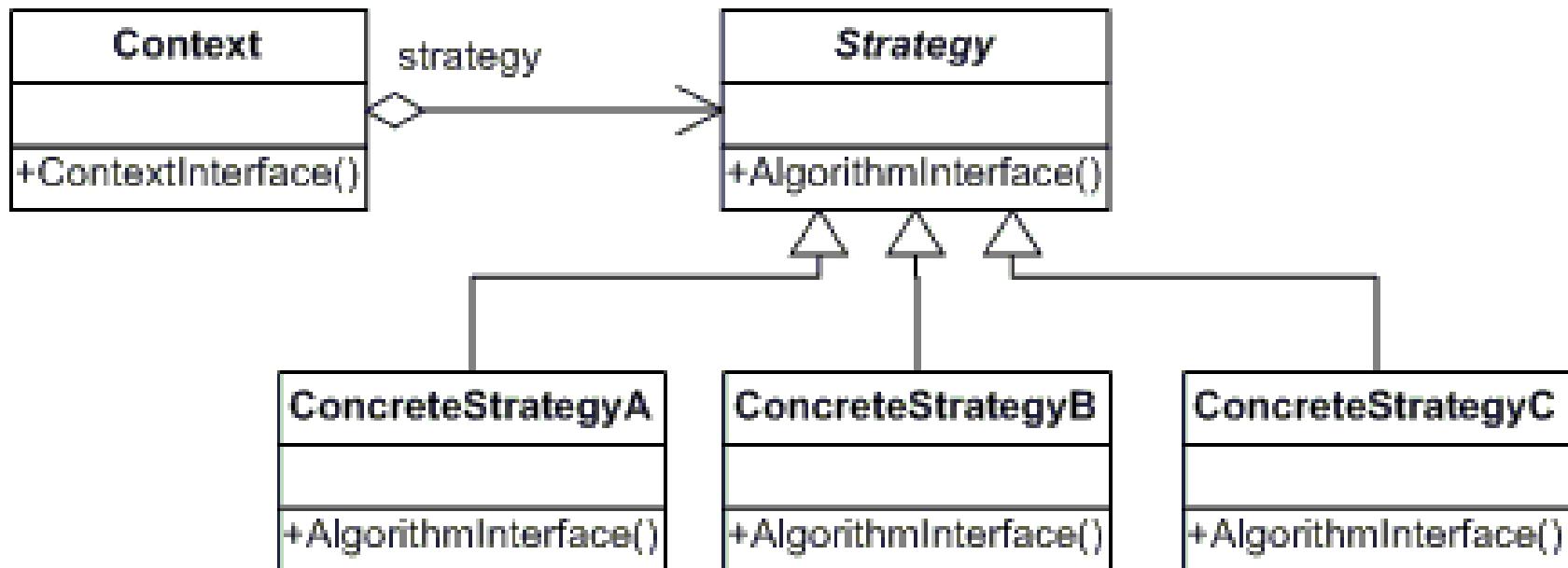
- Use el patrón Strategy cuando:
 - Muchas clases relacionadas sólo se diferencian en su comportamiento.
 - Se necesitan distintas variantes de un algoritmo.
 - Una clase tiene distintos comportamientos posibles que aparecen como instrucciones condicionales en sus métodos.
 - No se sepa qué aproximación utilizar hasta el momento de ejecución.



Estructura



Estructura



Estructura. Participantes

- **Contexto:** Clase que utiliza las diferentes estrategias para las distintas tareas. Mantiene una referencia a la instancia Estrategia que usa, y tiene un método para reemplazar la actual instancia de Estrategia.
- **Estrategia:** Interfaz en la que se definen todos los métodos disponibles para ser manejados por Contexto.
- **EstrategiaConcreta:** Clase que implementa la interfaz Estrategia utilizando un conjunto específico de reglas para cada uno de los métodos de la interfaz.



Consecuencias

- Cada comportamiento se define en su propia clase, por lo que se consigue que sean más fáciles de mantener.
- Resulta sencillo ampliar el modelo para incorporar nuevos comportamientos sin el alto coste de modificar todo el código de la aplicación o la utilización de la herencia.
- Las estrategias pueden proporcionar distintas implementaciones del mismo comportamiento.
- El principal problema es que cada estrategia debe tener la misma interfaz. Debe identificar una interfaz suficientemente genérica como para ser aplicada a distintas implementaciones, pero que, al mismo tiempo, sea suficientemente específica para ser utilizada por las distintas estrategias concretas.



Patrones relacionados

- **Flyweight:** Como las estrategias aumentan el número de objetos de una aplicación se puede utilizar el patrón flyweight para construir estrategias compartidas como objetos sin estado y reducir el número de objetos. Las estrategias compartidas no deberían mantener el estado entre invocaciones.

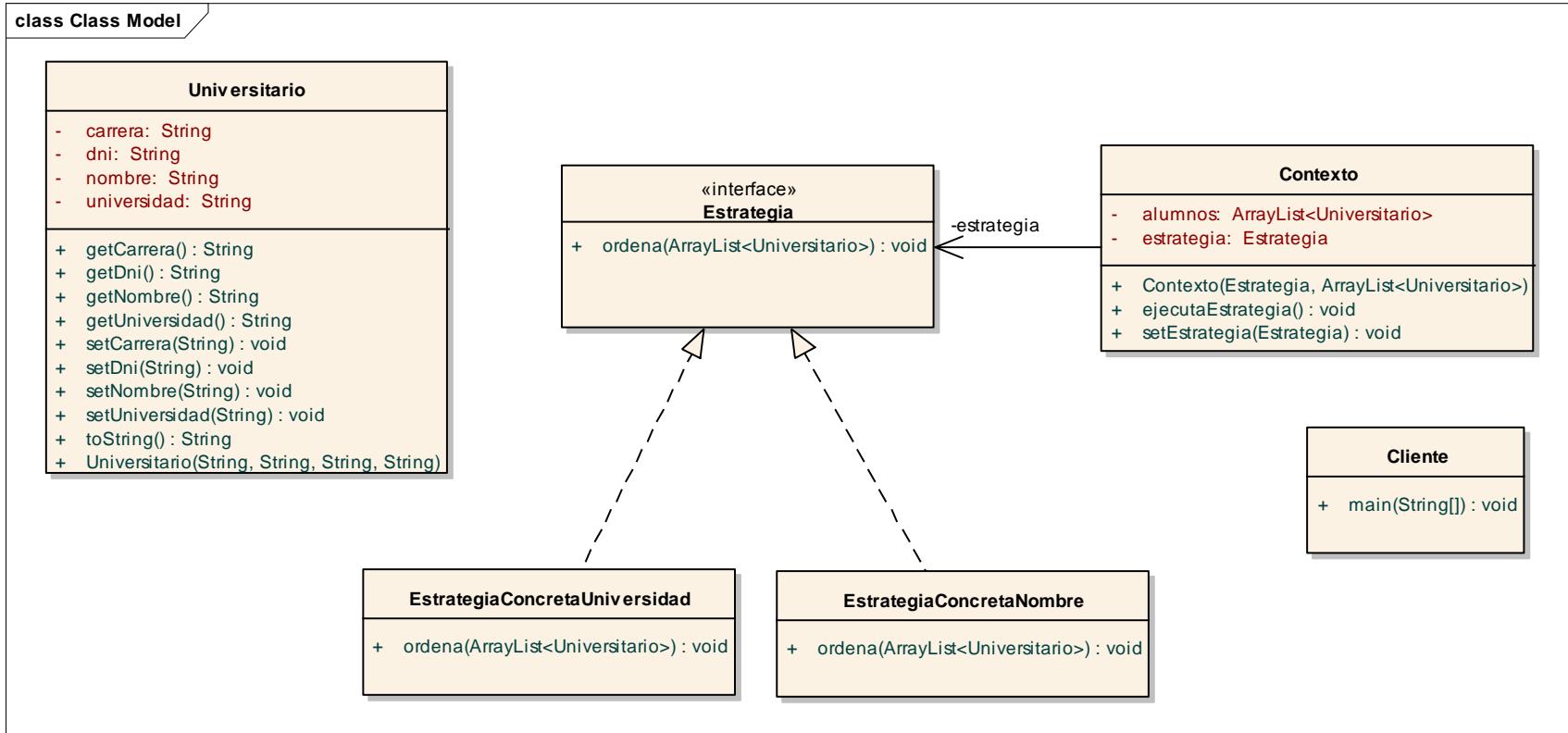


Código de ejemplo

Ordenación



Código de ejemplo



Código de ejemplo

- Identificamos a continuación los elementos del patrón:
 - Contexto: Contexto.
 - Estrategia: Estrategia.
 - EstrategiaConcreta: EstrategiaConcretaUniversidad, EstrategiaConcretaNombre.

