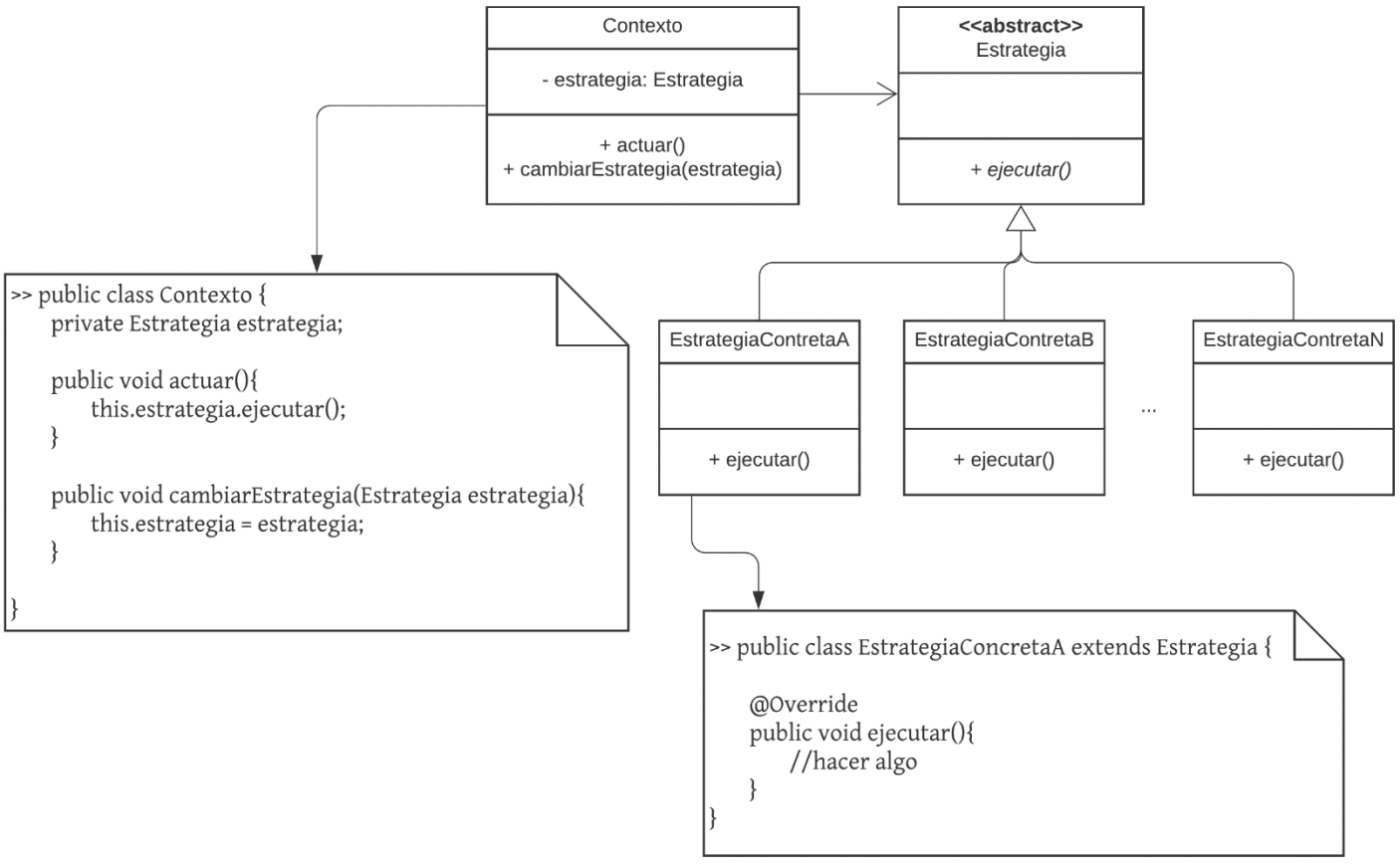
Patrón Strategy

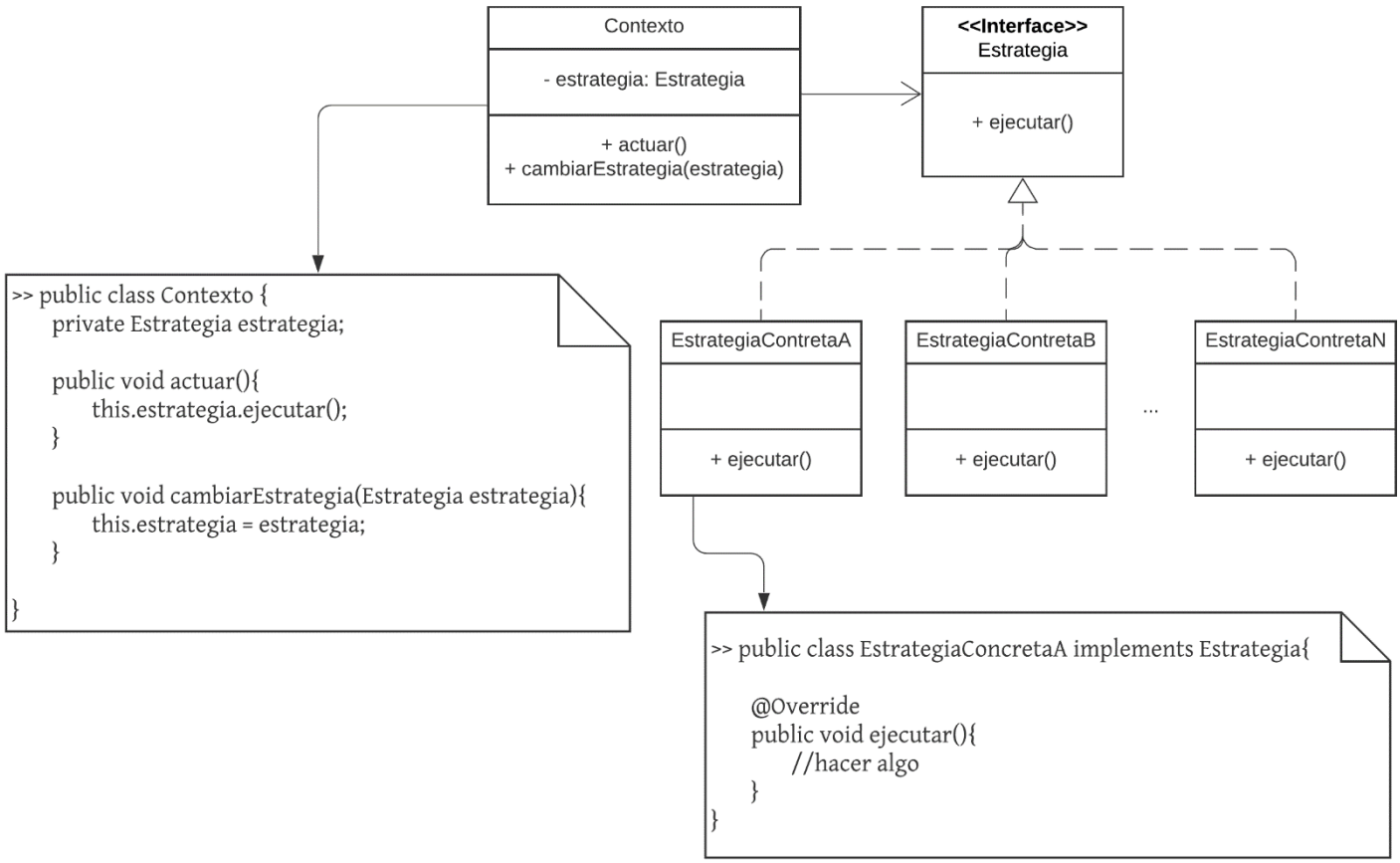
- **Clasificación**: Patrón de comportamiento
- **¿Qué hace?**
  - ✓ Encapsula distintas formas (o algoritmos) de resolver el mismo problema en diferentes clases.
  - ✓ Permite intercambiar en momento de ejecución la forma en que un tercero resuelve un problema.
- **Se sugiere su utilización cuando**:
  - ✓ 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.
- **Estructura genérica**:
  - ✓ *Primera posibilidad*: Estrategia como clase abstracta con relación de asociación unidireccional con el Contexto.



Consideraciones:

- ✓ La clase abstracta Estrategia podría declarar el método *ejecutar* como abstracto, por lo que cada clase hija estaría obligada a implementar dicho método.
- ✓ Si la clase abstracta Estrategia no declarara el método *ejecutar* como abstracto, podría definir un comportamiento por defecto.
- ✓ Las estrategias concretas implementan su propio algoritmo/lógica en el método *ejecutar*.
- ✓ El método *ejecutar* podría recibir parámetros si el problema lo amerita.
- ✓ Las estrategias concretas podrían necesitar una configuración extra para funcionar.
- ✓ Las estrategias concretas no se conocen.
- ✓ No existen transiciones entre las estrategias.
- ✓ Las instancias de cada estrategia pueden ser reutilizables.

✓ *Segunda posibilidad*: Estrategia como interface.



Consideraciones:

- ✓ Las estrategias concretas implementan su propio algoritmo/lógica en el método *ejecutar*.
- ✓ El método *ejecutar* podría recibir parámetros si el problema lo amerita.
- ✓ Las estrategias concretas podrían necesitar una configuración extra para funcionar.
- ✓ Las estrategias concretas no se conocen.
- ✓ No existen transiciones entre las estrategias.
- ✓ Las instancias de cada estrategia pueden ser reutilizables.

- ¿Qué proporciona su uso?
  - ✓ Mayor cohesión a la clase Contexto.
  - ✓ Mayor mantenibilidad debido a que el comportamiento por cada algoritmo es fácilmente localizable.
  - ✓ Extensibilidad para incoporar nuevos algoritmos/formas de realizar las acciones.
- Code smells que soluciona/evita de forma directa:
  - ✓ Métodos largos
  - ✓ Código duplicado
  - ✓ Clase Dios