**PATRON STRATEGY**

**Propósito**

Este es un patrón de diseño de comportamiento que te permite definir una familia de algoritmos, colocar cada uno de ellos en una clase separada y hacer sus objetos intercambiables.

**Dominios de aplicación**

El patrón se utiliza en los siguientes casos:

* El comportamiento de una clase puede estar implementado mediante distintos algoritmos siendo alguno de ellos más eficaz en términos de ejecución o de consumo de memoria o incluso contener mecanismos de decisión.
* La implementación de la elección del algoritmo mediante instrucciones condicionales se vuelve demasiado compleja.
* Un sistema posee numerosas clases idénticas salvo una parte correspondiente a su comportamiento.

**Participantes**

Son los siguientes:

* Strategy es la interfaz común a todos los algoritmos. Esta interfaz se utiliza en Context para invocar al algoritmo.
* ConcreteStrategyA, ConcreteStrategyB y ConcreteStrategyC son las subclases concretas que implementan los distintos algoritmos.
* Context es la clase que utiliza uno de los algoritmos de las clases que implementan la clase Strategy. Por consiguiente, posee una referencia hacia una de estas clases. Por último, si fuera necesario, puede exponer sus datos internos a las clases de implementación.

**Estructura**

