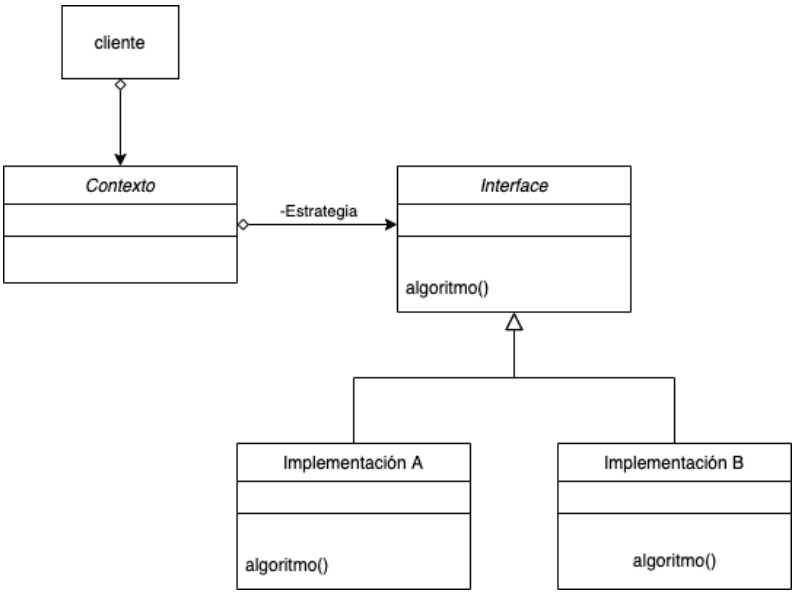
Patrón de Diseño: Estratégico
Nombre Alumno: Angel Odiel Treviño Villanueva

Nombre: Estratégico
Clasificación: Patrón de Comportamiento

Intención: Esta estrategia te permite definir una familia de algoritmos, la cual logran el mismo objetivo pero de diferentes maneras, y ponerlas en una clase independiente para poder ser reutilizada.

Aplicabilidad: Este patrón es recomendado utilizar cuando se utilizan diferentes variantes de un algoritmo dentro de un objeto. También se recomienda utilizarlo cuando se tienen clases similares y estas solo se diferencian en como realizan un cierto comportamiento. También es utilizado para aislar la lógica de negocios de una clase de los detalles de implementación de algoritmos que puedan no ser tan importantes en el contexto lógico.

Estructura:



La entidad de interfaz podría representar una clase base abstracta o las expectativas de firma del método por parte del cliente. En el primer caso, la jerarquía de herencia representa polimorfismo dinámico. En el último caso, la entidad de interfaz representa el código de plantilla en el cliente y la jerarquía de herencia representa el polimorfismo estático. El contexto mantiene una referencia hacia las implementaciones y solo se comunica con ellas a través de la interfaz de la estrategia.

Consecuencias: Este patrón trae consigo el beneficio de que puedes cambiar el algoritmo que se debe de implementar durante el run time. También puedes aislar la implementación detallada de un algoritmo en el código que lo utiliza y puedes remplazar la herencia con composition. Otro benefició que trae es que aplica el principio de Open/Closed, debido a que puedes incluir nuevas estrategias sin cambiar el contexto.

Implementación: Para implementar este patrón se necesitan aplicar los siguientes 4 pasos

1. En la clase del contexto, identificar un algoritmo que sufre cambios frecuentemente o una condicional masiva que ejecuta una variante de un algoritmo al ejecutar.
2. Declarar la interfaz de estrategia para todas las variantes del algoritmo.
3. Extraer todos los algoritmos en su propia clase y que implementen la interfaz de estrategia.
4. Conecta los clientes del algoritmo hacia la interfaz.

Usos Conocidos: Esto se puede observar gracias a las aplicaciones de mapas y que te indican como llegar a tu dirección. Esto es debido a que existen muchas estrategias para transportarte, las cuales pueden ser coche, caminando, bicicleta, etc. Puedes escoger cualquiera de estas estrategias, pero a final siempre va a realizar la tarea de calcular la ruta mas rápida con tu estrategia.

Patrones relacionados: Este patrón esta relacionado con Bridge, State, Adaptar ya que se parecen en estructura. También esta relacionado con Command, Decorator y Template Method.

Ejemplo: En el ejemplo que veremos para la implentación de este patrón de diseño, crearemos nuestra estrategia para implementar diferentes algoritmos de ordenamiento. Esto es debido a que queremos llegar un objetivo común, el cual es ordenar nuestro conjunto, pero lo queremos de hacer diferentes formas distintas porque en diferentes casos, unos son mas eficientes que otros. Para esto creamos nuestra clase de contexto, el cual será nuestro manejador de colección y el cual establecerá que método usar.

Video: https://youtu.be/XTkrXeoceOc