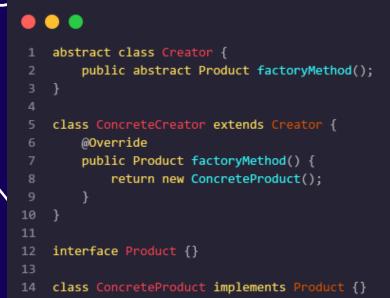


Factory

- Intención: Proporcionar una interfaz para crear objetos en una superclase, pero permite que las subclases alteren el tipo de objetos que serán creados.
- Conocido como: Factory Method, Método Fábrica.
- Motivo: Permite a una clase delegar en sus subclases la creación de objetos, lo que facilita la extensión de la funcionalidad de la aplicación modificando únicamente las subclases involucradas.



Adapter-(Wrapper)

- **Intención:** Convertir la interfaz de una clase en otra interfaz que esperan los clientes. Adapter permite que clases con interfaces incompatibles trabajen juntas.
- Motivo: Permite la colaboración entre clases que de otro modo no podrían trabajar juntas debido a sus interfaces incompatibles.
- Aplicaciones: Cuando se quiere utilizar una clase existente, y su interfaz no se corresponde con la que necesitamos.

```
// Interfaz objetivo
    interface Target {
        void request();
    // Clase a adaptar
    class Adaptee {
        void specificRequest() {
            // Implementación específica
11
    // Adaptador
    class Adapter implements Target {
        private Adaptee adaptee = new Adaptee();
        @Override
        public void request() {
            adaptee.specificRequest();
```



Strategy-(Policy)

- Intención: Definir una familia de algoritmos, encapsular cada uno de ellos y hacerlos intercambiables. Strategy permite que el algoritmo varíe independientemente de los clientes que lo utilizan.
- Motivo: Permite seleccionar el algoritmo de comportamiento en tiempo de ejecución, proporcionando una alternativa flexible a la herencia para seleccionar comportamientos.
 - **Aplicaciones**: Cuando hay varios algoritmos para realizar una tarea y se desea seleccionar dinámicamente cuál utilizar.

```
interface Strategy (
        void algorithmInterface();
    class ConcreteStrategyA implements Strategy
        public void algorithmInterface() {
            // Implementación del algoritmo
11 class Context {
        private Strategy strategy;
        // Método para cambiar la estrategia en tiempo de ejecución
        public void setStrategy(Strategy strategy) {
            this.strategy = strategy;
        public void executeStrategy() {
            strategy.algorithmInterface();
```



