Patrón de Diseño Estrategia - Ejemplo en Java

# ¿Qué es el Patrón Estrategia?

El patrón de diseño Estrategia (Strategy) es un patrón de comportamiento que permite definir una familia de algoritmos, encapsular cada uno de ellos y hacerlos intercambiables. Este patrón permite que el algoritmo varíe independientemente de los clientes que lo utilizan.

# Características del Patrón Estrategia

- Permite cambiar el algoritmo o comportamiento de una clase en tiempo de ejecución.

- Favorece la composición sobre la herencia.

- Encapsula los comportamientos relacionados en clases separadas.

- Facilita la escalabilidad y el mantenimiento del código.

# Explicación del Código Proporcionado

## 1. Interfaz EstrategiaTransporte

Define el contrato que todas las estrategias concretas deben implementar. Contiene el método `irAlTrabajo()` que define el comportamiento común que debe ser implementado por todas las clases que representen una estrategia de transporte.

## 2. Clases que Implementan EstrategiaTransporte

Estas clases representan estrategias concretas de transporte y cada una implementa el método `irAlTrabajo()` de manera diferente:  
- `IrCaminando`: Simula ir al trabajo caminando.  
- `IrEnAuto`: Simula ir al trabajo en auto.  
- `IrEnBicicleta`: Simula ir al trabajo en bicicleta.

## 3. Clase Viajero

Representa el contexto que utiliza una estrategia de transporte. Tiene un atributo de tipo `EstrategiaTransporte` que puede cambiarse dinámicamente mediante el método `setEstrategia()`. El método `irAlTrabajo()` delega el comportamiento a la estrategia actual.

## 4. Clase Main

Contiene el método `main` donde se crea un objeto `Viajero` y se le asignan diferentes estrategias de transporte en tiempo de ejecución. Esto demuestra la flexibilidad del patrón Estrategia para cambiar el comportamiento de un objeto sin modificar su código interno.

# Conclusión

El patrón Estrategia es útil cuando se desea definir múltiples comportamientos intercambiables para una tarea común. Este ejemplo demuestra cómo se puede aplicar en Java para representar diferentes formas de transporte que un viajero puede utilizar para ir al trabajo, promoviendo así un diseño limpio, flexible y fácilmente extensible.