Respuestas ejercicio 16

16a)

Inciso 3:

📌 Template Method

- **¿El objetivo se distingue?**
  No. El objetivo del patrón Template Method es definir el esqueleto de un algoritmo en una clase base y dejar que las subclases reemplacen ciertos pasos. En este caso, no hay una estructura fija que cada filtro complete; cada uno aplica una transformación distinta.
- **¿Coincide con la estructura?**
  No. No hay una clase abstracta con pasos predefinidos y métodos que las subclases llenan. Cada filtro parece ser independiente y autónomo.

📌 Strategy

- **¿El objetivo se distingue?**
  Parcialmente. El patrón Strategy define una familia de algoritmos (en este caso, filtros), encapsula cada uno y los hace intercambiables. Creo que para este ejemplo (solo este inciso a) se podria llegar a usar un estrategy, ya que tengo entendido que no permite los filtros encadenados/anidados.
  La desventaja de usar el patron Strategy es que no podemos tener filtros anidados.
  Si quisieramos tener filtros anidados, la mejor opcion es utilizar el patron Decorator.

📌 Decorator

- **¿El objetivo se distingue?**
  Sí. El patrón Decorator permite aplicar funcionalidades adicionales (filtros) de forma dinámica, envolviendo objetos. Cada filtro transforma una imagen y luego pasa al siguiente, lo cual se asemeja a cómo funcionan los decoradores.
- **¿Coincide con la estructura?**
  Sí. Si cada filtro recibe una imagen, la modifica y la pasa a otro filtro, se están "encadenando" decoradores. Cada filtro decora la imagen con una nueva transformación.

Ejercicio 16 b)

Inciso 3:

✅ Decorator

¿El objetivo del patrón se distingue en el diseño?
Sí, claramente. El patrón Decorator tiene como objetivo agregar responsabilidades adicionales a un objeto de forma dinámica. En este diseño, cada filtro encapsula a otro filtro, formando una cadena en la que cada uno modifica o enriquece la imagen antes de pasarla al siguiente. Esto permite componer comportamientos complejos (varias transformaciones) de forma flexible y modular, que es exactamente lo que se busca en un sistema de filtros encadenados.

¿La estructura del proyecto coincide con la estructura y los participantes del patrón?
Sí. El diseño típicamente incluye una interfaz o clase base común para todos los filtros (por ejemplo, `Pipe` o `Filter`), y cada implementación concreta puede almacenar una referencia a otro filtro del mismo tipo. Esto permite anidar filtros, como lo hacen los decoradores. Cada filtro aplica su transformación y luego delega la siguiente operación al siguiente "decorado", exactamente como se espera en el patrón Decorator.

🚫 Template Method

¿El objetivo del patrón se distingue en el diseño?
No. Template Method define el esqueleto de un algoritmo en una clase base y permite que las subclases personalicen ciertos pasos. En un sistema de pipes/filtros encadenados no se está siguiendo un algoritmo común con pasos fijos, sino componiendo diferentes pasos de manera dinámica. Por tanto, el objetivo de Template Method no se refleja en este tipo de diseño.

¿La estructura del proyecto coincide con la estructura y los participantes del patrón?
No. Template Method requiere una clase abstracta con un método que define el flujo del algoritmo, y métodos "hook" o abstractos que se implementan en las subclases. El diseño de pipes, en cambio, está más enfocado en la composición de objetos en tiempo de ejecución, lo cual no encaja con la estructura jerárquica del Template Method.

🚫 Strategy

¿El objetivo del patrón se distingue en el diseño?
Parcialmente. Strategy permite definir una familia de algoritmos e intercambiarlos en tiempo de ejecución, lo cual puede parecer similar a aplicar distintos filtros. Sin embargo, Strategy se enfoca en elegir una única estrategia (un filtro, en este caso), mientras que en el diseño de pipes se combinan varios filtros de manera estructural y progresiva, lo cual excede el objetivo del patrón Strategy.

¿La estructura del proyecto coincide con la estructura y los participantes del patrón?
No completamente. En Strategy hay un contexto que utiliza un algoritmo (estrategia) y permite intercambiarlo. En cambio, en tu diseño, no hay un "contexto" que selecciona un filtro único,

sino que se construye una cadena dinámica de transformaciones, lo que implica una estructura recursiva o compuesta, mucho más alineada con Decorator.

Ejercicio 16 c)

No lo hice, es un quilombo este.