1. **Notificaciones a los clientes.**

   En tu empresa están desarrollando una plataforma de comercio electrónico. Actualmente, cuando un pedido es completado, se le notifica al cliente por correo electrónico.

   El equipo de producto ha solicitado una nueva funcionalidad: quieren poder enviar notificaciones por múltiples canales. Es decir, a futuro, un cliente podría recibir su notificación por:

   - Correo electrónico
   - SMS
   - WhatsApp
   - Slack
   - ...u otros canales en el futuro

   La clave es que los canales de notificación pueden variar por cliente, y también podrían combinarse (por ejemplo: un cliente recibe el mensaje por Correo + WhatsApp, otro por Slack + SMS, etc.).

   Inicialmente el sistema se había diseñado con una clase NotificadorCorreo, pero agregar canales adicionales mediante herencia (NotificadorCorreoSMS, NotificadorCorreoSMSWhatsApp, etc.) resulta en una explosión de clases y una estructura difícil de mantener.

   ¿Cómo rediseñarías el sistema para permitir una combinación flexible y extensible de canales de notificación, evitando duplicación de código y sin depender de una jerarquía rígida de clases?

   Diseña la solución, permitiendo que:

   - Se pueden combinar múltiples canales
   - El comportamiento se pueda extender fácilmente con nuevos tipos de notificación
   - No se cree una clase por cada combinación de canales

   Criterios que debe cumplir tu solución:

- Código abierto a extensión, cerrado a modificación (principio OCP)
- No se debe duplicar lógica común de envío de notificaciones
- Los canales deben poder agregarse o quitarse en tiempo de ejecución
- La solución debe ser limpia y mantenible

2. **Constructor de Reportes Personalizados**

Estás trabajando en un sistema que genera reportes PDF personalizados para los clientes de una entidad financiera.

Cada cliente puede pedir un reporte con diferentes secciones opcionales, como:

- Portada con logotipo y nombre del cliente
- Gráficos estadísticos (de inversión, ahorro, etc.)
- Tabla de movimientos recientes
- Análisis de tendencias
- Pie de página con contacto del asesor

Actualmente, hay una clase ReportePDF con un constructor que recibe muchos parámetros (unos obligatorios y otros opcionales), y se ha vuelto difícil de mantener, porque:

- Algunos parámetros no se usan siempre
- El orden de los parámetros es confuso
- A veces los desarrolladores crean reportes con configuraciones inválidas
- Hay muchos null al llamar al constructor
- Agregar una nueva sección al reporte obliga a modificar varias partes del código existente

  Además, algunos clientes requieren que los reportes se generen de forma progresiva, dependiendo de condiciones dinámicas en tiempo de ejecución. Por ejemplo, si el usuario tiene datos suficientes, se incluye el análisis de tendencias; si no, se omite.

¿Cómo rediseñarías el sistema para construir objetos ReportePDF de forma clara, flexible y segura, evitando constructores complejos y errores de configuración?

Diseña la solución, permitiendo:

- Crear reportes personalizados con secciones opcionales
- Asegurar que los reportes se construyan solo cuando estén correctamente configurados
- Mantener el código limpio y fácil de extender si se agregan nuevas secciones

Criterios que debe cumplir tu solución:

- El código debe ser legible y fluido
- Debe ser seguro frente a mal uso (por ejemplo, no permitir construir un objeto incompleto)
- Debe permitir agregar o quitar secciones fácilmente
- El objeto final debe ser inmutable una vez construido

3. Sistema de Gestión de Tareas

Imagina un sistema de gestión de tareas en el que los usuarios pueden crear, editar, eliminar y completar tareas. Cada acción realizada por el usuario corresponde a una acción que debe ser ejecutada. Además, es importante mantener un registro de todas las acciones realizadas para permitir la reversión de las mismas si es necesario.

Aplicación del Patrón:
En este escenario, el patrón será aplicado para encapsular cada una de las acciones que el usuario puede realizar sobre una tarea.
El patrón que seleccione debe tener los siguientes beneficios:

- Desacopla el invocador de los objetos que realizan las acciones.
- Permite la extensión de nuevas operaciones sin modificar el código existente.
- Facilita el registro de acciones para realizar operaciones de reversión.

4. Sistema de Gestión de Archivos y Carpetas

Estás desarrollando un módulo para un sistema operativo educativo que simula una estructura de archivos y carpetas, como el explorador de archivos de Windows o Linux.

Se necesita representar:

- Archivos individuales, con un nombre y un tamaño en bytes.
- Carpetas, que pueden contener archivos y otras carpetas dentro de ellas (de forma recursiva).

El objetivo es que se puedan realizar operaciones como:

- Obtener el tamaño total de una carpeta (suma del tamaño de todos sus archivos y subcarpetas)
- Imprimir el contenido de una carpeta de forma jerárquica
- Aplicar acciones como borrar, renombrar o mover un elemento (sin importar si es archivo o carpeta)

Actualmente, el equipo tiene dos clases distintas: Archivo y Carpeta, y cada una tiene métodos diferentes. Esto ha llevado a:

- Mucho código duplicado en los métodos que recorren el árbol de archivos
- Estructuras condicionales (if o instanceof) para saber si se está tratando con un archivo o carpeta

- Dificultades al agregar nuevas operaciones comunes a ambos tipos de elementos

¿Cómo rediseñarías este modelo para que archivos y carpetas puedan tratarse de forma uniforme, y se puedan recorrer y operar sobre ellos recursivamente y sin condicionales?

Diseña la solución , permitiendo:

- Tratar archivos y carpetas con una interfaz común
- Calcular el tamaño de una carpeta sin importar cuántos niveles de subcarpetas haya
- Extender la estructura fácilmente con nuevos tipos de elementos (como accesos directos o enlaces)

Criterios que debe cumplir tu solución:

- El diseño debe permitir composición recursiva
- Las operaciones deben funcionar sin necesidad de saber si el objeto es archivo o carpeta
- Debe ser abierto a extensión, pero sin modificar las clases existentes
- El cliente del sistema no debe tener que usar instanceof o if para saber cómo interactuar con cada tipo de elemento