***Patrón de diseño del observador***

El patrón de observador es una especie de patrón de comportamiento que se ocupa de la asignación de responsabilidades entre objetos.

Los patrones de comportamiento caracterizan flujos de control complejos que son difíciles de seguir en tiempo de ejecución. Cambiaron tu enfoque del flujo de control para que te concentres justo en la forma en que los objetos están interconectados.

El patrón de observador define una dependencia de uno a varios entre objetos para que cuando un objeto cambia de estado, todos sus dependientes se notifican y actualizan automáticamente.

El patrón observador describe estas dependencias. Los objetos clave de este patrón son sujeto y observador. Un sujeto puede tener cualquier número de observadores dependientes.

Todos los observadores son notificados cada vez que el sujeto sufre un cambio en su estado. En respuesta, cada observador consultará el asunto para sincronizar su estado con el estado del asunto. La otra manera de entender el patrón de observador es la forma en que funciona la relación editor-suscriptor.

Supongamos, por ejemplo, que se suscribe a una revista para su revista deportiva o de moda favorita. Cada vez que se publica un nuevo número, se le entrega. Si cancelas tu suscripción cuando ya no quieres la revista, no te la entregarán. Pero el editor sigue trabajando como antes, ya que hay otras personas que también están suscritas a esa revista en particular.

***Patrón de diseño del mediador***

El patrón de mediador define un objeto que encapsula cómo interactúan un conjunto de objetos.

El mediador promueve el acoplamiento flexible al evitar que los objetos se refieran entre sí explícitamente, y le permite variar su interacción de forma independiente.

En lugar de interactuar directamente entre sí, los objetos piden al Mediador que interactúe en su nombre, lo que resulta en reutilización y acoplamiento. Encapsula la interacción entre los objetos y los hace independientes de los demás.

Esto les permite variar su interacción con otros objetos de una manera totalmente diferente mediante la implementación de un mediador diferente.

El mediador ayuda a reducir la complejidad de las clases. Cada objeto ya no tiene que saber en detalle cómo interactuar con los otros objetos. El acoplamiento entre objetos va de apretado y frágil a suelto y ágil

***Patrón de diseño del proxy***

El patrón de proxy se utiliza para crear un objeto representativo que controla el acceso a otro objeto, que puede ser remoto, costoso de crear o que necesita estar protegido.

Una razón para controlar el acceso a un objeto es aplazar el costo total de su creación e inicialización hasta que realmente necesitemos usarlo.

Otra razón podría ser actuar como un representante local para un objeto que vive en una JVM diferente. El proxy puede ser muy útil para controlar el acceso al objeto original, especialmente cuando los objetos deben tener diferentes derechos de acceso.

En el patrón de proxy, un cliente no habla directamente con el objeto original, delega las llamadas al objeto proxy que llama a los métodos del objeto original. El punto importante es que el cliente no sabe sobre el proxy, el proxy actúa como un objeto original para el cliente.