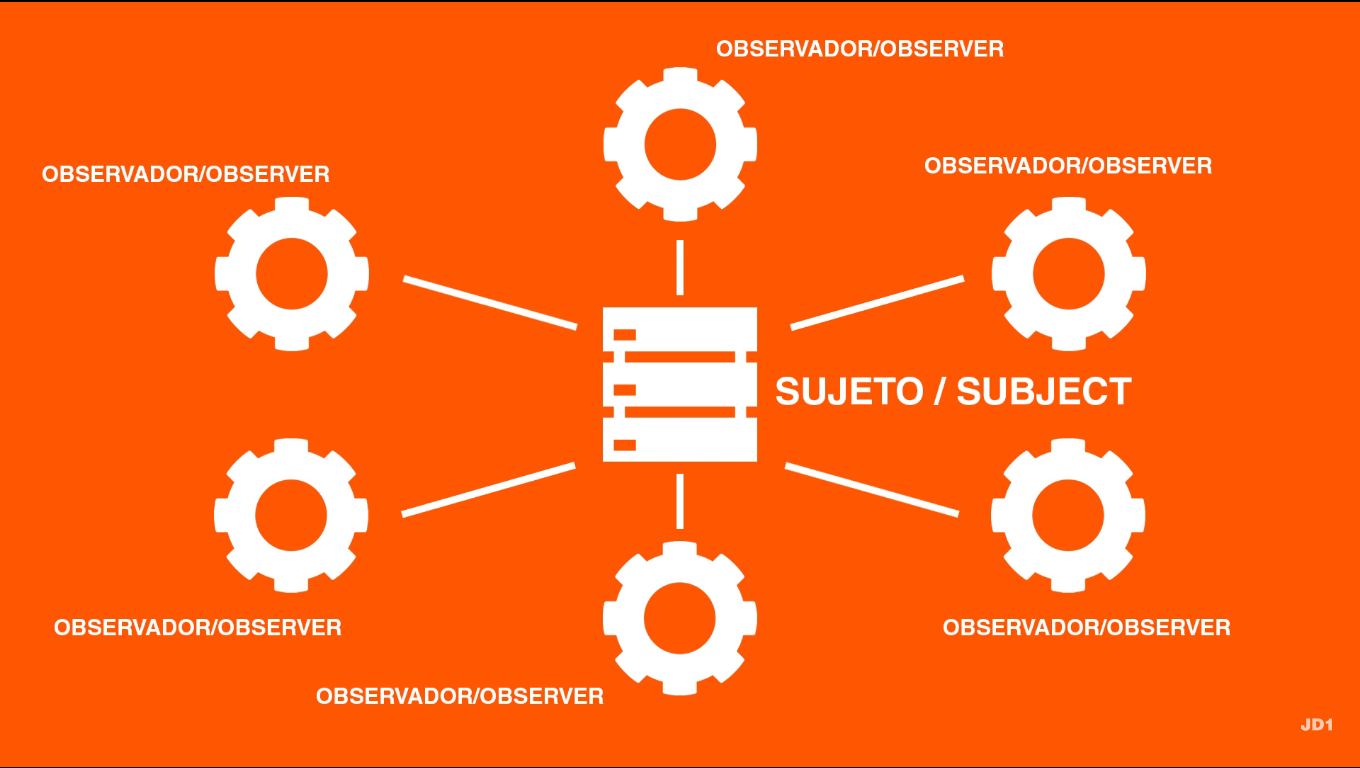
PATRON DE DISEÑO

OBSERVER

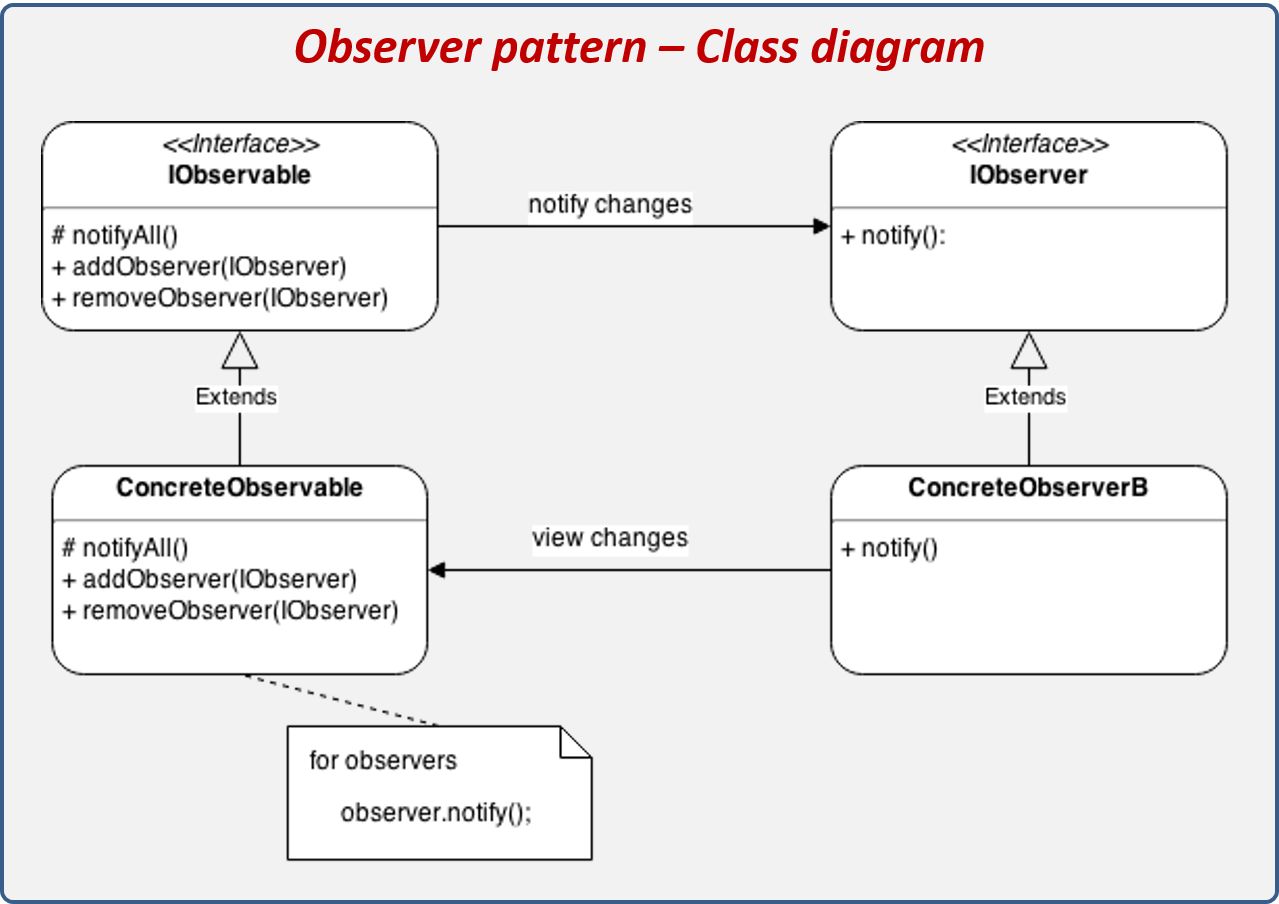


Este patrón permite observar los cambios producidos por un objeto, de esta forma, cada cambio que afecte el estado del objeto observado lanzará una notificación a los observadores; a esto se le conoce como Publicador-Suscriptor.

Cuando un objeto/clase quiere convertirse en observador del sujeto, tiene que solicitarlo, de esta forma el sujeto central añade el objeto a su lista de observadores.

De esta forma cada vez que haya algo relevante que notificar todos los observadores recibirán una actualización del estado de la clase sujeto, esta es una forma limpia y ordenada de que todo funcione.

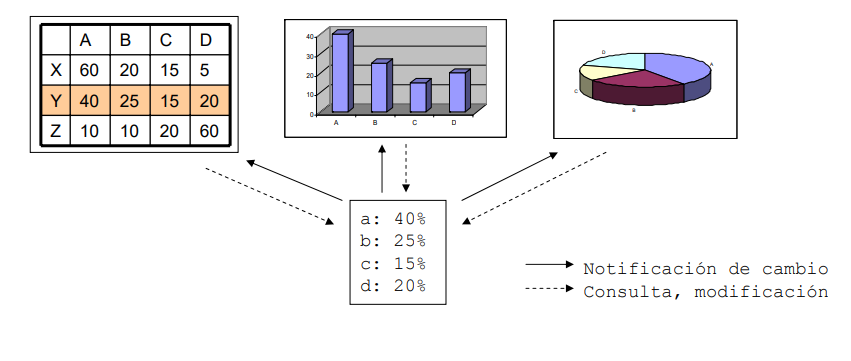
**ESTRUCTURA:**

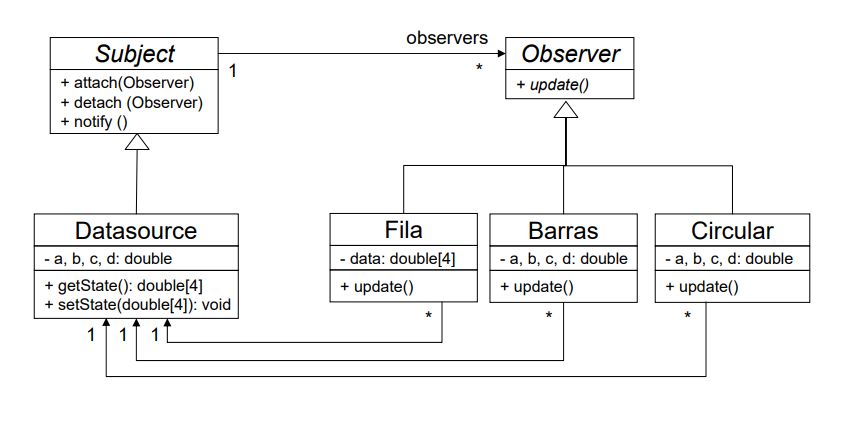


* **IObservable**: Interface que deben de implementar todos los objetos que quieren ser observados, en ella se definen los métodos mínimos que se deben implementar.
* **ConcreteObservable**: Clase que desea ser observada, ésta implementa IObservable y debe implementar sus métodos.
* **IObserver**: Interfaces que deben implementar todos los objetos que desean observar los cambios de IObservable.
* **ConcreteObserver**: Clase concreta que está atenta de los cambios de IObserver, esta clase hereda de IObserver y debe de implementar sus métodos.

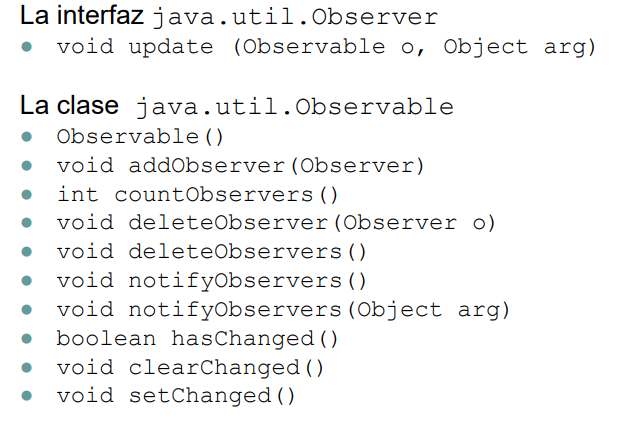
Debe ser utilizado cuando:

* Un objeto necesita notificar a otros objetos cuando cambia su estado. La idea es encapsular estos aspectos en objetos diferentes permite variarlos y reutilizarlos independientemente.
* Cuando existe una relación de dependencia de uno a muchos que puede requerir que un objeto notifique a múltiples objetos que dependen de él cuando cambia su estado.





En java:



Ejemplo:

<https://www.seas.es/blog/informatica/patrones-de-diseno-en-java-patron-observer/>