Observer

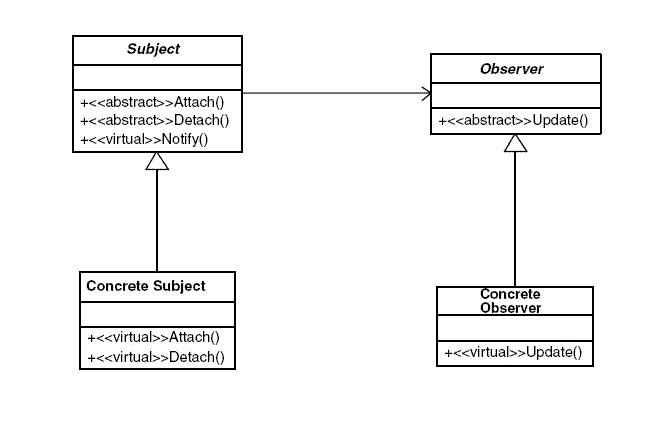
El patrón de comportamiento Observer define una interacción entre objetos, de manera que cuando uno de ellos cambia su estado, el Observer se encarga de notificar este cambio a los demás.  
Por tanto, la razón de ser de este patrón es desacoplar las clases de los objetos, aumentando la modularidad del lenguaje y evitando bucles de actualización.  
La idea básica del patrón es que el objeto de datos (o sujeto) contenga atributos mediante los cuales cualquier objeto observador (o vista) se pueda suscribir a él pasándole una referencia a sí mismo. De este modo, el sujeto mantiene así una lista de las referencias a sus observadores.

Dadas estas propiedades, el patrón Observer suele emplearse en el desarrollo de frameworks de interfaces gráficas orientados a objetos, enlazando 'listeners' a los objetos que pueden disparar eventos.

Las clasese participantes en el esquema general de este patrón de comportamiento son:

* Subject: Es el que conoce a sus observadores, proporcionando una Interfaz para que se suscriban los objetos de tipo *Observer*.
* Observer: Define la interfaz para actualizar los objetos a los que se deben notificar los cambios en el objeto Subject.
* ConcreteSubject: Guarda el estado de interés para los objetos ConcreteObserver y envia una notificación a sus observadores cuando cambia su estado.
* ConcreteObserver: Mantiene una referencia a un objeto ConcreteSubject, guardando el estado que debería permanecer sincronizado con el objeto observado, además de implementar la interfaz Observer para mantener su estado consistente con el objeto observado.

Este sería el diagrama de clases general del patrón de comportamiento Observer:



*Esquema del patrón Observer*