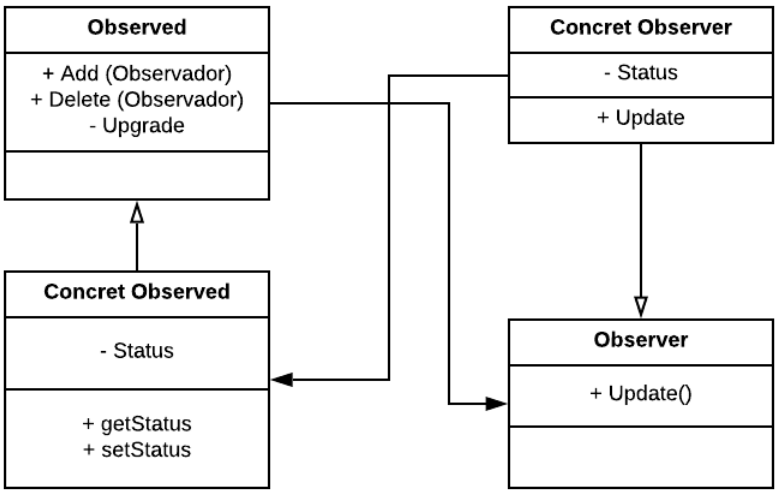
Observer:

***Propósito:** Generalmente la utilización de este patrón se da cuando se requiere estar pendiente de un elemento o del estado de este sin estar realizando una constante consulta a este, que pueda generar grandes dependencias o bucles, por lo que se declara un observador que reciba una notificación cuando su observado o observados infringen un cambio, ya sea en ellos o en el de estado de ellos.

***Estructura:**



Según el diagrama anterior las clases concret, son las que poseen el estado del observado, pero solo se limitan a esa información, estas a su vez heredan de Observer y Observed, estas dándole un manejo al dado de las concrect, es decir; notificándolo ya sea Observed a Observer o en su defecto la consulta, a través de una actualización de Observer a Observed.

***Contexto:**

Caso 1: En una plataforma donde se da la interacción de diversos usuarios entre sí, se puede dar la implementación de un observador, con el cual los usuarios (observados) que se conectan y desconectan a dicha plataforma, al entrar en línea, estos actualizan su estado y el observador es notificado que la plataforma posee “y” cantidad de usuarios en línea y “x” desconectados.

Caso 2: En un sistema cliente – servidor, se puede dar la implementación de observer, para ver el estado de las peticiones que realiza el cliente al servidor, o el estado de esta petición cuando va de vuelta del servidor al cliente.