El patrón de diseño **Observer** (o **Observable**) es utilizado en situaciones donde un objeto (llamado sujeto u observable) necesita notificar a otros objetos (llamados observadores) sobre cambios en su estado sin necesidad de que los objetos estén estrechamente acoplados. Este patrón promueve un diseño más flexible y modular, permitiendo la comunicación entre objetos de manera desacoplada.

El Observable va a ser una interfaz donde principalmente tendremos 3 métodos (Administra los Observer)  
-Añadir un observer  
-Eliminar un observer  
-Notificar que algo ha pasado a todos los observers

El Observer es una interfaz donde se va a llamar al método update, donde se hará algún tipo de lógica

**Interfaces de Usuario (UI):**

* **Model-View-Controller (MVC):** En una arquitectura MVC, el modelo notifica a las vistas cuando hay cambios en los datos. Las vistas actúan como observadores del modelo.
* **Eventos de UI:** Los componentes de la interfaz de usuario, como botones y menús, pueden notificar a los controladores sobre eventos como clics y selecciones.

 **Sistemas de Publicación/Suscripción:**

* **Noticias y Alertas:** Un sistema de noticias puede notificar a los suscriptores cuando hay una nueva noticia. Los suscriptores pueden ser aplicaciones, usuarios u otros servicios.
* **Notificaciones Push:** Aplicaciones móviles que utilizan notificaciones push para informar a los usuarios sobre nuevos mensajes, actualizaciones de aplicaciones, etc.

 **Sistemas de Registro y Monitoreo:**

* **Monitoreo de Aplicaciones:** Un sistema de monitoreo puede notificar a diferentes servicios o aplicaciones cuando detecta un problema o un cambio en el estado del sistema.
* **Sistemas de Log:** Un componente que registra eventos puede notificar a diferentes manejadores de log para que procesen o almacenen los eventos de diferentes maneras.

**Ejemplo con Interfaz Gráfica (UI)**

Supongamos que se tiene una aplicación de clima que muestra la temperatura actual y las condiciones meteorológicas. Cada vez que el modelo de datos del clima se actualiza, las vistas correspondientes (por ejemplo, widgets de temperatura y condiciones) deben actualizarse.

En el ejemplo, WeatherData es el sujeto (observable) que notifica a los observadores (CurrentConditionsDisplay) sobre los cambios en el estado del clima. Cada vez que las mediciones cambian, los observadores son notificados y actualizan su visualización