OBSERVER PATTERN

En este ejemplo se aplica el patrón de diseño *Observer*, donde está como el elemento *Subject* el que va a entregar o actualizar información relevante a otros objetos *Observers*. Se tiene en cuenta en que estos dos elementos no deben de estar fuertemente acoplados, no deben de tener dependencia directa. Esto como resultado nos da flexibilidad y mantenibilidad del código.

La clase concreta que será *Subject* debe implementar una interfaz en la que se definen los métodos para registro, eliminación y notificación a sus observadores. Al igual aquí se definen los elementos que se van a compartir.

Las clases concretas que serán *Observers* deben implementar una interfaz *Observer,* en la cual se define el método que les dará las notificaciones del *Subject* y en estas clases se podrá esos datos para algún fin definido, y sin interferir o afectar al *Subject.*

En este ejemplo se presenta como *Subject (WeatherData)* a un objeto que entrega tres datos sobre el clima (*temperature, humidity, pressure)* y estos datos los usan diferentes tipos de pantalla para presentarlo de manera diferente (*Observers= displays*).

Al momento de crear un *Observer*, este en automático se registra. Y cada vez que se actualizan los datos, los tres *displays* imprimen sus datos. Si se elimina alguno, solo los registrados imprimen sus pantallas. De esta manera si se quieren usar los datos del *Subject* en otra clase diferente o para otro fin, solo te registras y ya.

A continuación, el diagrama UML: