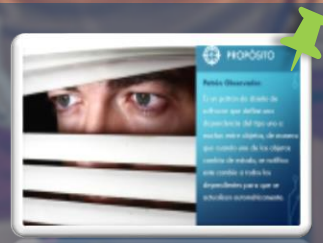
PATRONES DE DISEÑO

PATRONES DE COMPORTAMIENTO - OBSERVER

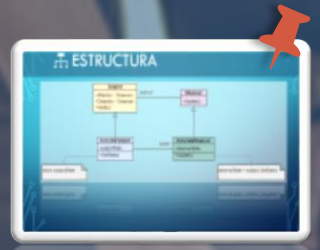
PATRON DE COMPORTAMIENTO - OBSERVER



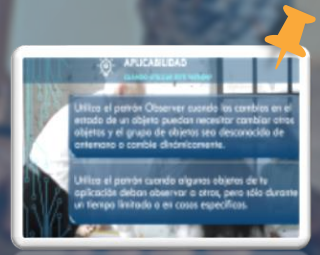
Propósito

Problema

Solución



Estructura



Aplicabilidad

Consecuencias

Ejercicio

Tarea

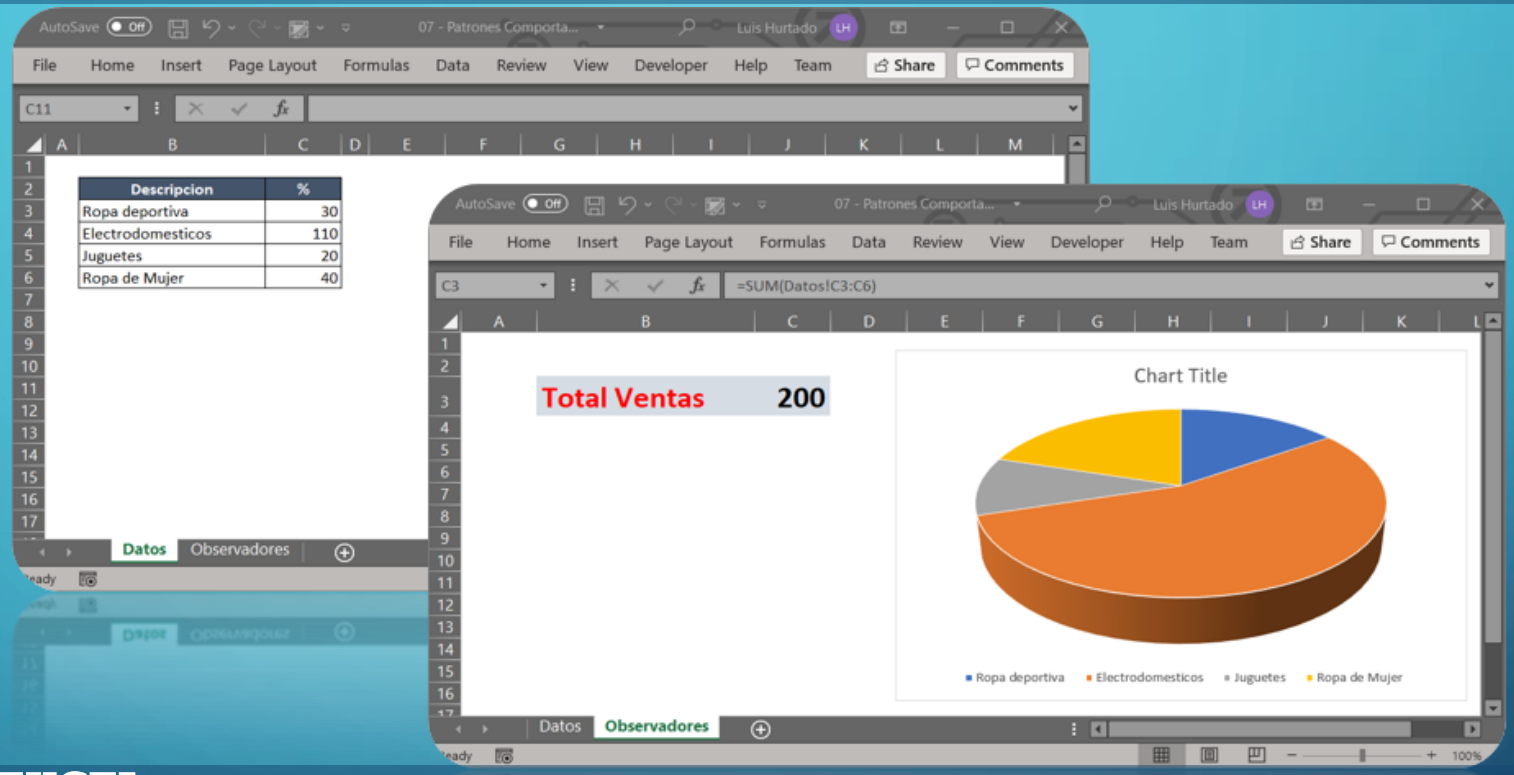
Terminado

PROPÓSITO

Patrón Observador:

Es un patrón de diseño de software que define una dependencia del tipo uno a muchos entre objetos, de manera que cuando uno de los objetos cambia de estado, se notifica este cambio a todos los dependientes para que se actualicen automáticamente.

PROBLEMA



HOJA DE CÁLCULO TIPO EXCEL

Supongamos que queremos simular el comportamiento de una hoja de datos en Excel, en la cual tenemos un conjunto de datos que representan las ventas del mes.

Asociados a ellos tenemos otros dos tipos de objetos que actuaran como observadores:

- una celda calculada con el Monto total
- un grafico tipo pie.

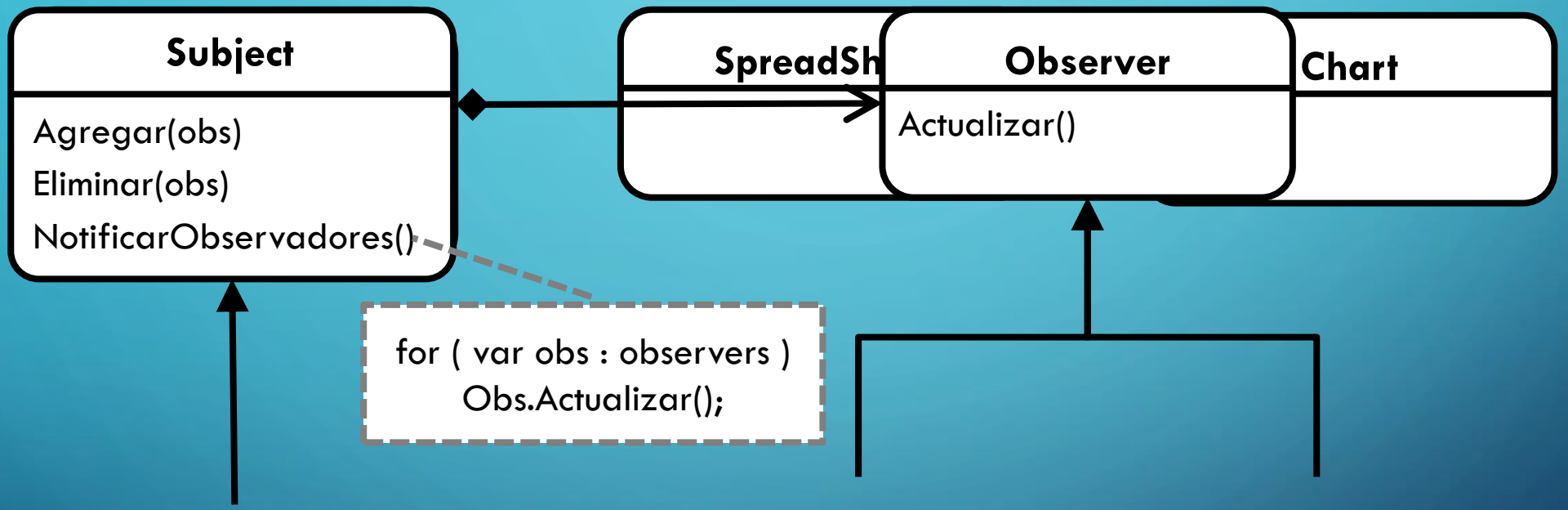
DEMO

REVISEMOS EN CÓDIGO LA IMPLEMENTACIÓN DEL PROBLEMA

SOLUCIÓN

CLASE ABSTRACTA

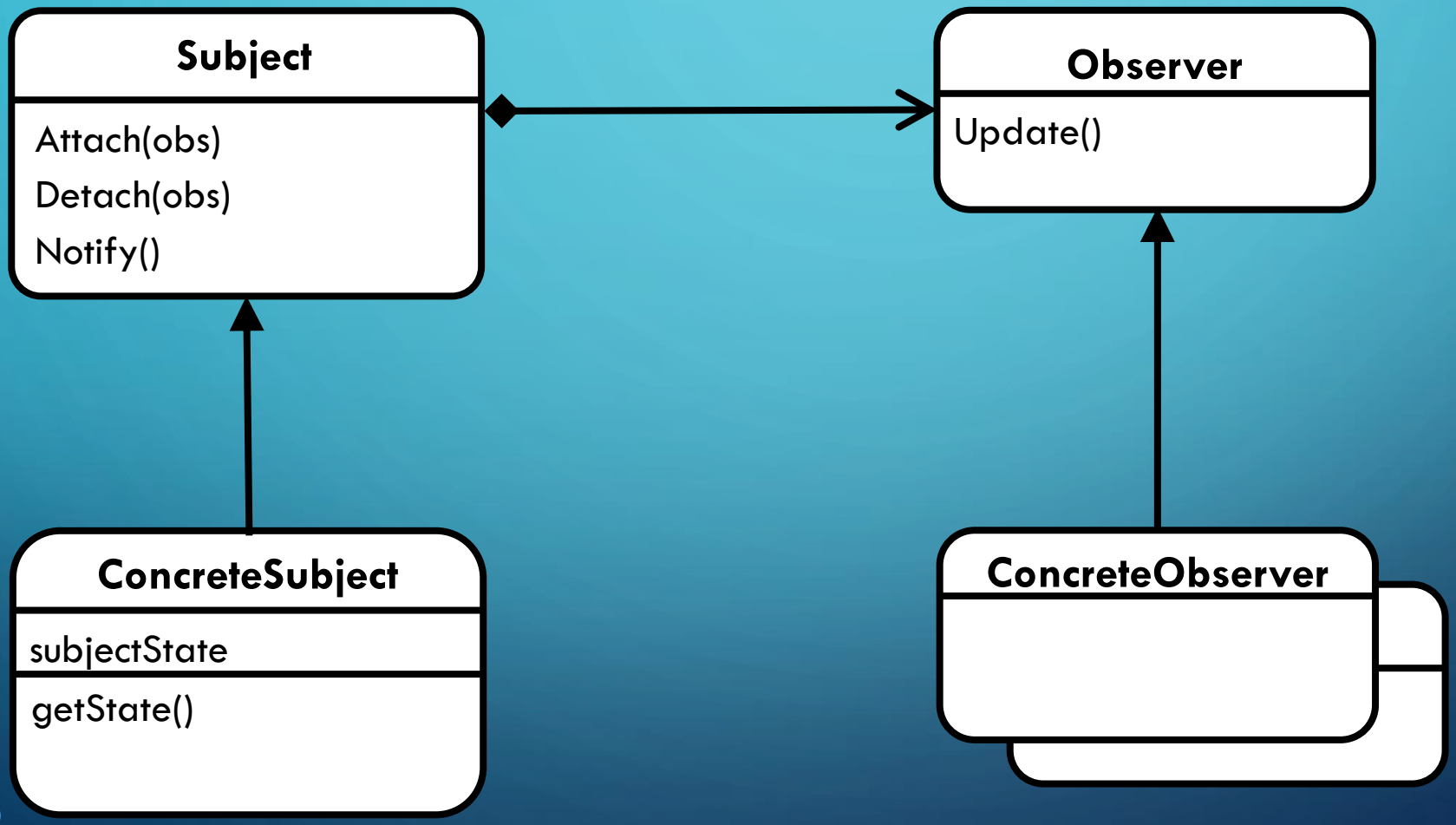
INTERFACE



SOLUCIÓN

CLASE ABSTRACTA

INTERFACE

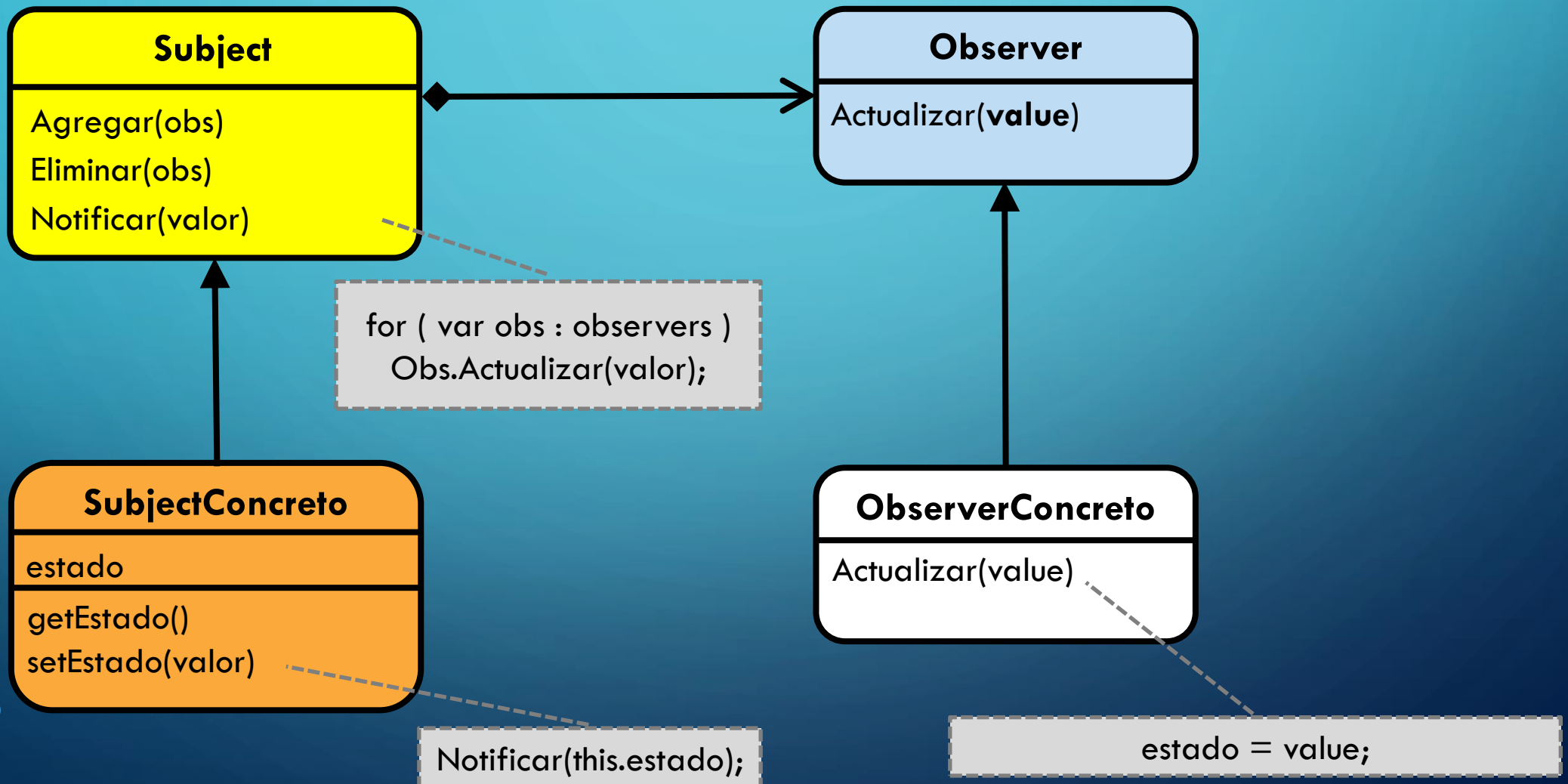


SOLUCIÓN

PUSH STYLE

CLASE ABSTRACTA

INTERFACE

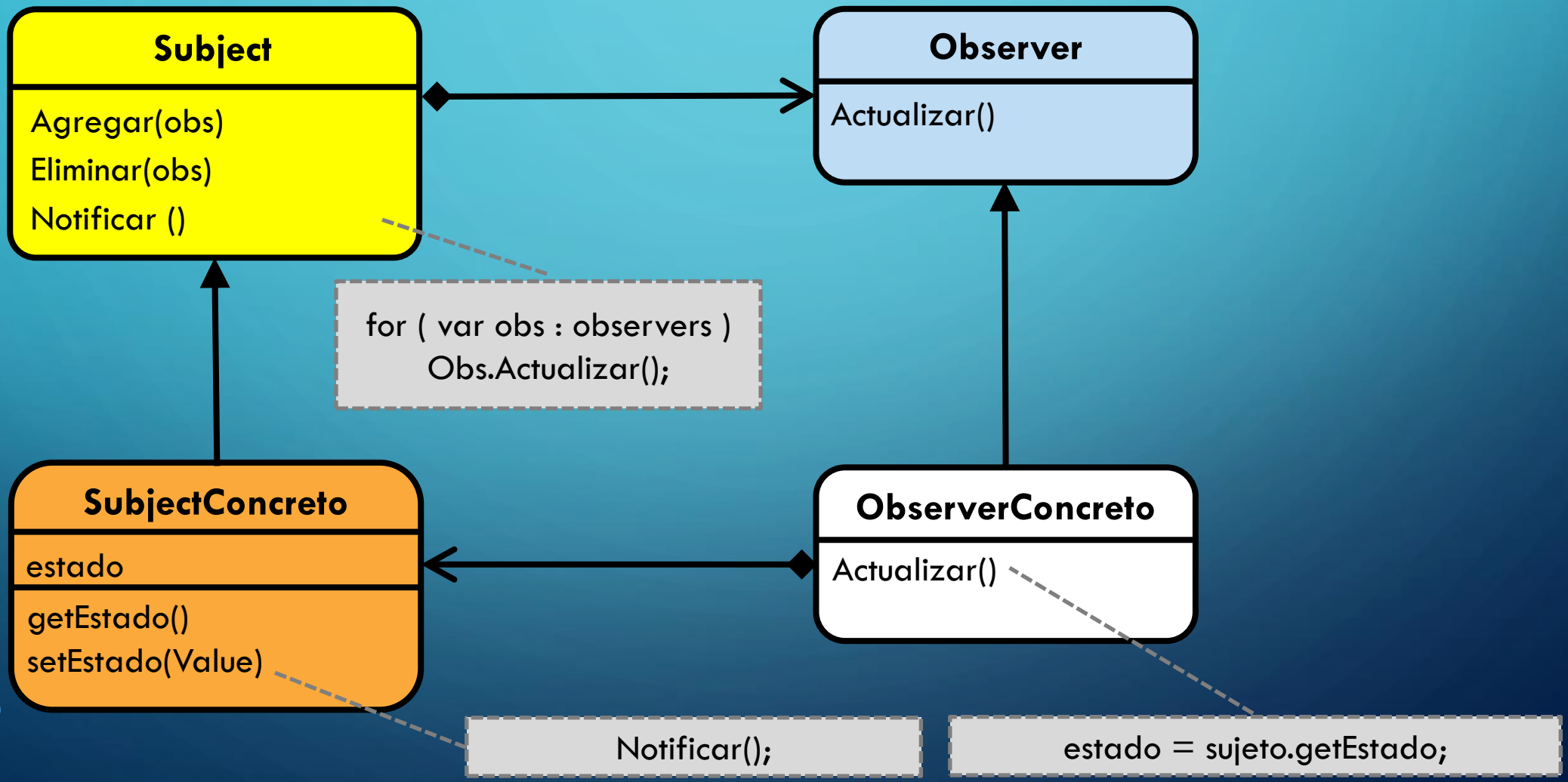


SOLUCIÓN

PULL STYLE

CLASE ABSTRACTA

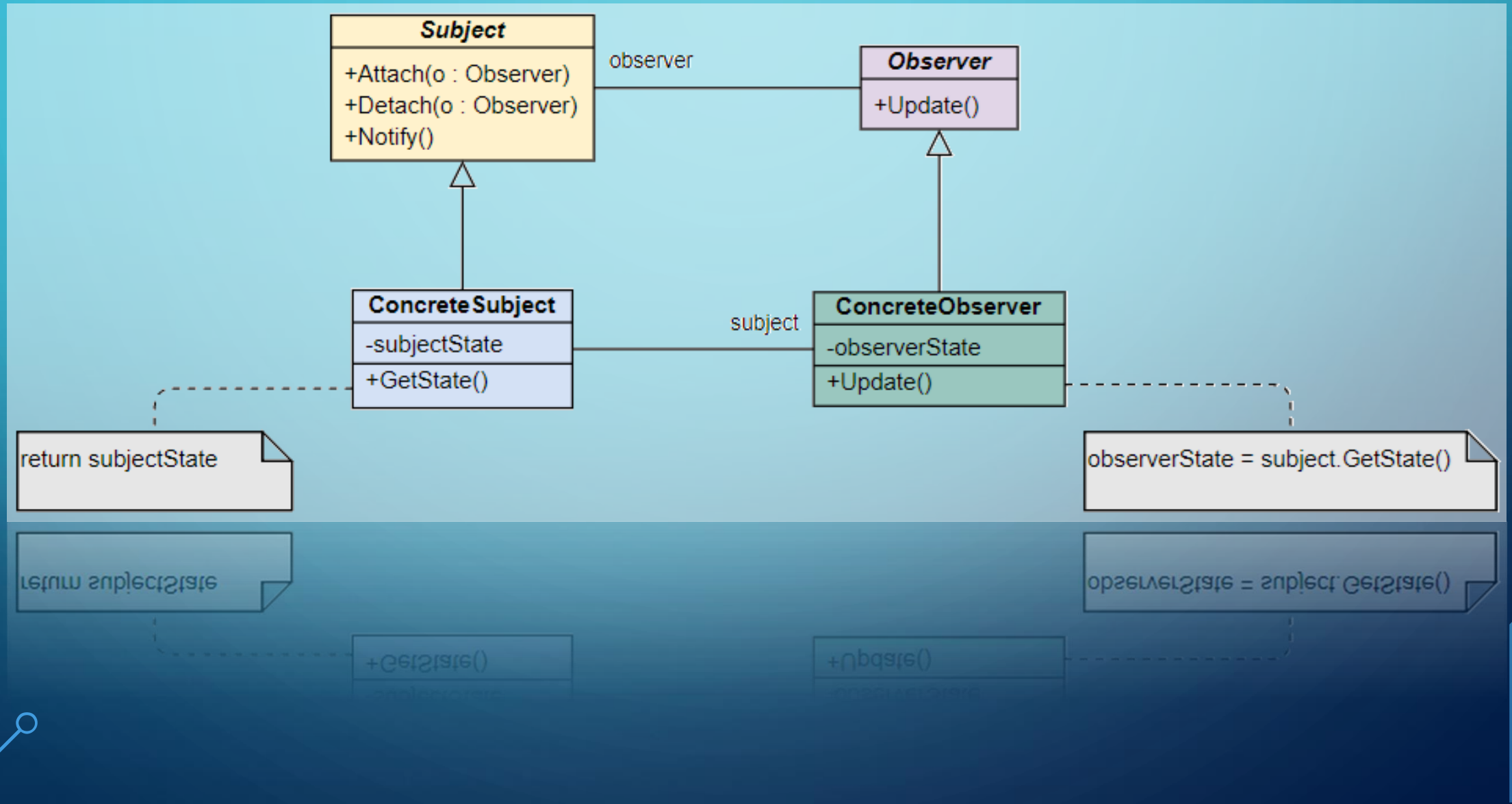
INTERFACE



DEMO

REVISEMOS EN CÓDIGO LA IMPLEMENTACIÓN DE LA SOLUCIÓN

ESTRUCTURA


Subject
+Attach(o : Observer)
+Detach(o : Observer)
+Notify()
observer
Observer
+Update()
ConcreteSubject
-subjectState
+GetState()
subject
ConcreteObserver
-observerState
+Update()
return subjectState
observerState = subject.GetState()

APLICABILIDAD

CUÁNDO UTILIZAR ESTE PATRÓN?

Utiliza el patrón Observer cuando los cambios en el estado de un objeto puedan necesitar cambiar otros objetos y el grupo de objetos sea desconocido de antemano o cambie dinámicamente.

Utiliza el patrón cuando algunos objetos de tu aplicación deban observar a otros, pero sólo durante un tiempo limitado o en casos específicos.

CONSECUENCIAS

VENTAJAS:

- ✓ Principio de abierto/cerrado. Puedes introducir nuevas clases suscriptoras sin tener que cambiar el código de la notificadora (y viceversa si hay una interfaz notificadora).
- ✓ Puedes establecer relaciones entre objetos durante el tiempo de ejecución.

DESVENTAJAS:

- ✗ Los suscriptores son notificados en un orden aleatorio.

EJERCICIO

BOLSA DE VALORES

- Cree una pequeña aplicación de la bolsa de valores para unas 3 empresas importantes del rubro tecnológico.
- Permita que varios sujetos se puedan añadir a la lista de notificados y mostrar los valores que recibieron
- Algunos de estos subscriptores podrían abandonar nuestra bolsa de valores.
- Realice el ejemplo utilizando el patrón Observer

DEMO

REVISEMOS EN CÓDIGO LA SOLUCION DE LA PRACTICA

TAREA

Aplicación para ver acciones

Hay dos lugares en nuestra aplicación donde necesitamos mostrar las acciones:

- **StatusBar**: muestra las acciones populares
- **StockListView**: muestra la lista completa de acciones

Cuando el precio de una acción cambia, las vistas correspondientes(*StatusBar* y/o *StockListView*) deben actualizarse para reflejar el último precio.

Nuestra aplicación no tiene actualmente la capacidad de comunicar el cambio de precios de las acciones a las vistas correspondientes.

Utilice el patrón observador para resolver este problema.

Muchas Gracias...

los espero la siguiente clase