Patrones de Diseño: Patrones de Comportamiento. Tema 5-8: Observer

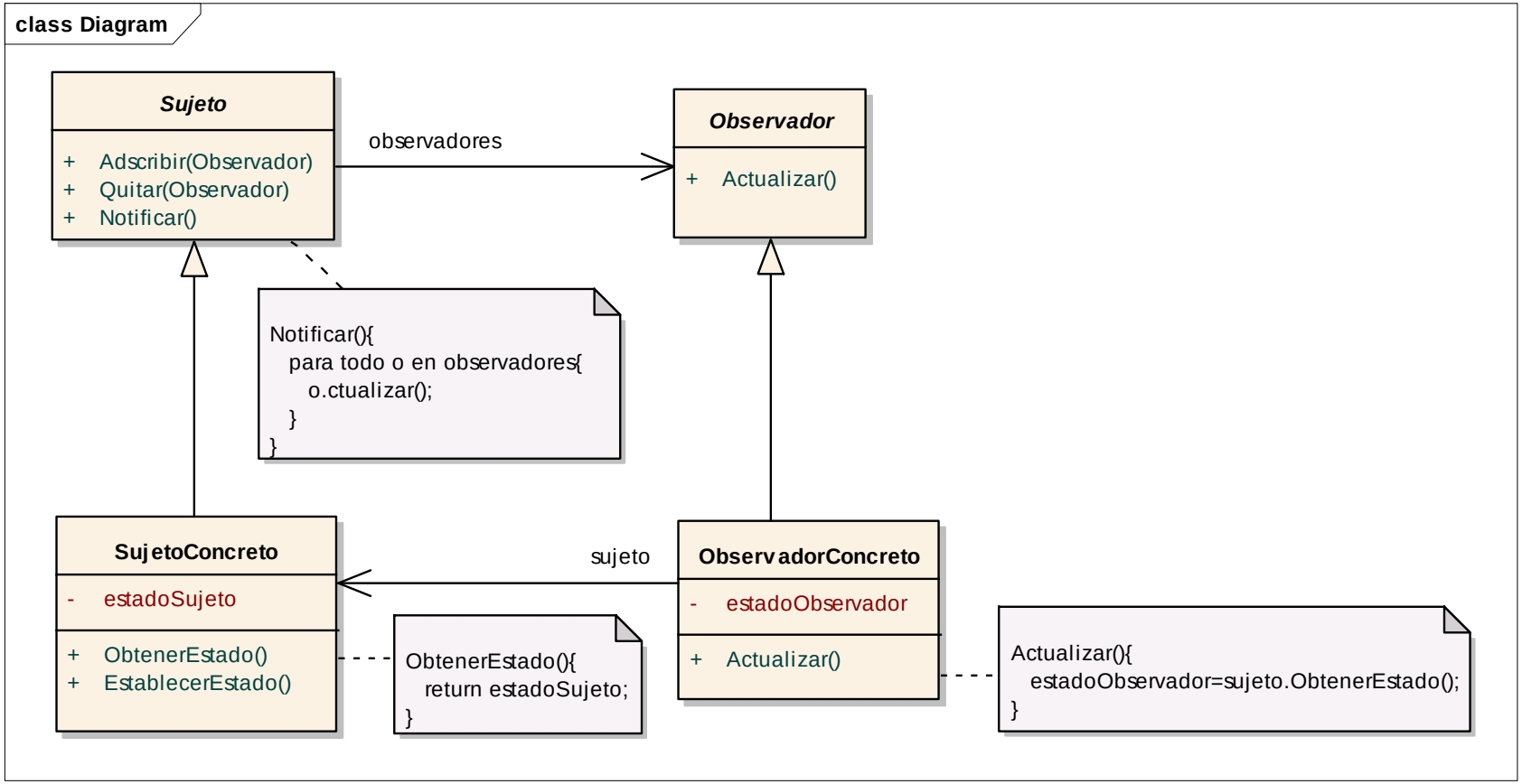
Descripción del patrón

- **Nombre:**
  - Observador
  - También conocido como Publisher-Subscriptor (editor-subscriptor), Dependents (dependientes)
- **Propiedades:**
  - Tipo: comportamiento
  - Nivel: objeto, componente
- **Objetivo o Propósito:**
  - Permite definir dependencias uno-a-muchos de forma que los cambios en un objeto se comuniquen a los objetos que dependen de él.

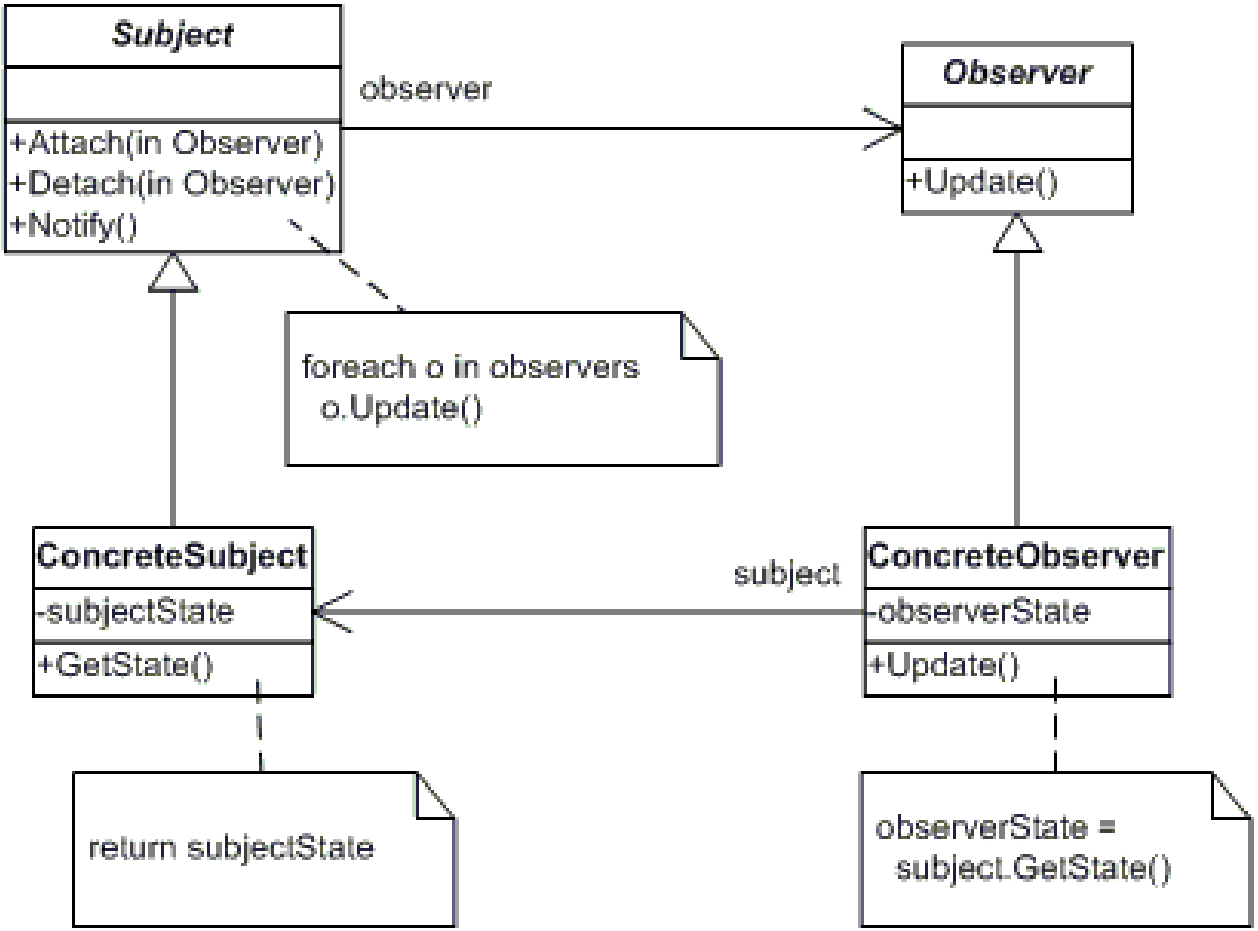
Aplicabilidad

- Use el patrón Observer cuando:
  - Existe al menos un emisor de mensajes.
  - Uno o más receptores de mensajes podrían variar dentro de una aplicación o entre aplicaciones.
  - Si se produce un cambio en un objeto se requiere el cambio de otros y no se sabe cuantos se necesitan cambiar. No queremos que estén fuertemente acoplados.
- En el API de Java tenemos la interfaz Observer y la clase Observable. Esto permite que muchos objetos reciban eventos de otro objeto en lugar de los sistemas de eventos básicos que sólo permiten notificar a un único objeto.

Estructura


class Diagram
Sujeto
+ Adscribir(Observador)
+ Quitar(Observador)
+ Notificar()
observadores
Observador
+ Actualizar()
Notificar(){
para todo o en observadores{
o.ctualizar();
}
}
SujetoConcreto
- estadoSujeto
+ ObtenerEstado()
+ EstablecerEstado()
sujeto
ObservadorConcreto
- estadoObservador
+ Actualizar()
ObtenerEstado(){
return estadoSujeto;
}
Actualizar(){
estadoObservador=sujeto.ObtenerEstado();
}

Estructura


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

Estructura. Participantes

- **Sujeto (Observado):** Interfaz que define como pueden interactuar los observadores con el sujeto. Define métodos para añadir y quitar observadores y avisarles de que se han producido cambios en el sujeto.
- **SujetoConcreto:** Implementa la interfaz Sujeto. Contiene una lista de observadores a los que avisa cuando cambia su estado.
- **Observador:** Interfaz para actualizar los objetos ante cambios en un sujeto.
- **ObservadorConcreto:** Mantiene una referencia a un objeto SujetoConcreto. Implementa la interfaz Observador y define los métodos para responder a los mensajes recibidos del sujeto.

Consecuencias

- Desacoplamiento entre sujetos y observadores, convirtiéndolos en entidades reutilizables por separado.
- Es un medio muy flexible de distribuir la información desde un objeto a muchos, de forma dinámica en tiempo de ejecución y sin que las clases implicadas sean conscientes del resto.
- El sujeto puede incluir cierta información en el mensaje de actualización de forma que cada observador pueda decidir si el cambio de estado le afecta o no.
- Un sujeto puede ser a su vez observador respecto de otros.
- Un problema es que un pequeño cambio en el sujeto puede provocar mucho procesamiento en los observadores.

Patrones relacionados

- **Adapter:** Puede ser útil para permitir que clases que no implementen la interfaz *Observador* puedan recibir notificaciones.
- **Mediator:** Se puede utilizar para coordinar múltiples cambios de estado de un mismo Sujeto provocados por diferentes objetos (con la finalidad de reducir el número de notificaciones).

Código de ejemplo

Valores Bolsa

Código de ejemplo