UE Conception Orientée Objet

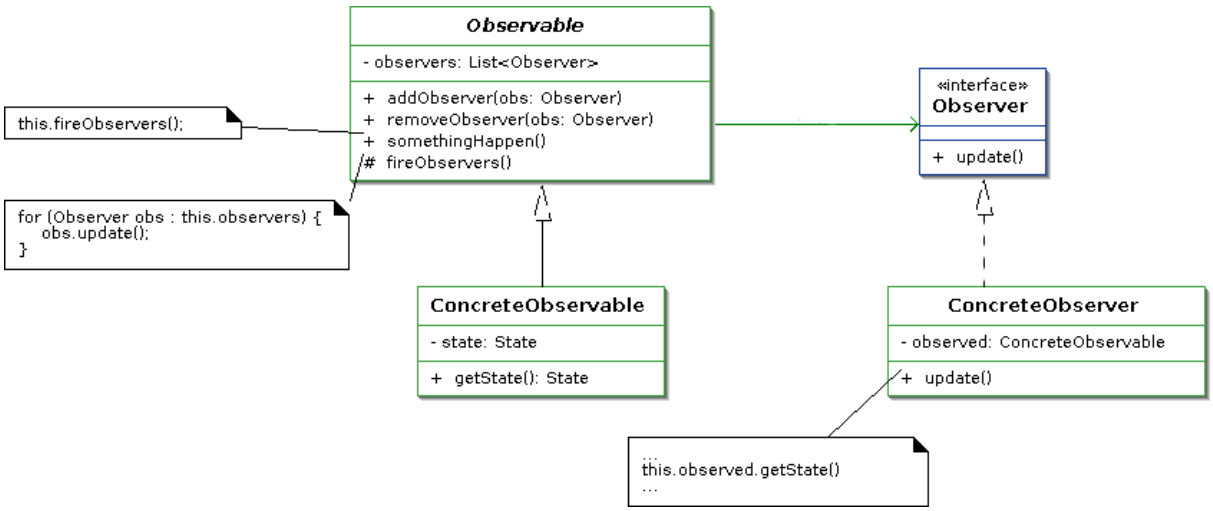
Design Pattern : observer

Intent

> *Define a one-to-many dependency between objects so that when one object changes state, all its dependents are notified and updated automatically.*

aussi appelé : *Abonneur/Abonné* ou *Event Idiom*

Structure



Eléments caractéristiques

- Plusieurs objets peuvent être avertis des événements émis par une source
- Le nombre et la nature des objets avertis ne sont potentiellement pas connus à la compilation et peuvent changer dans le temps.
- L'émetteur de l'événement et le récepteur ne sont pas fortement liés.

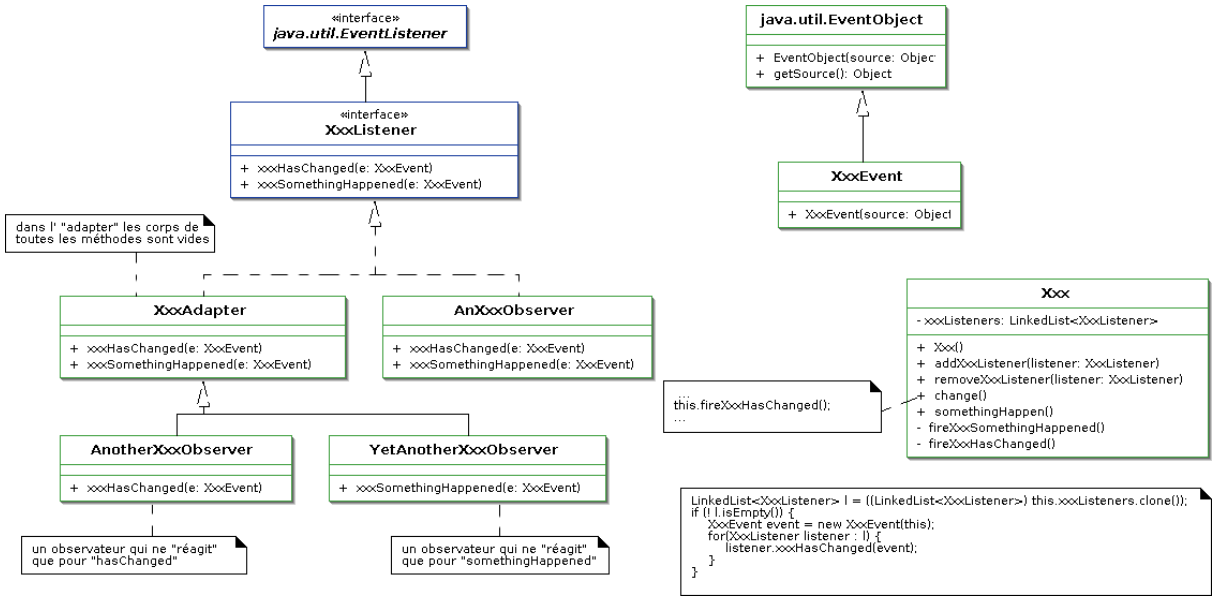
Exemples

Mise en place gestion d'événements.

1. Définir les classes d'événements
2. Définir les interfaces des `listeners`

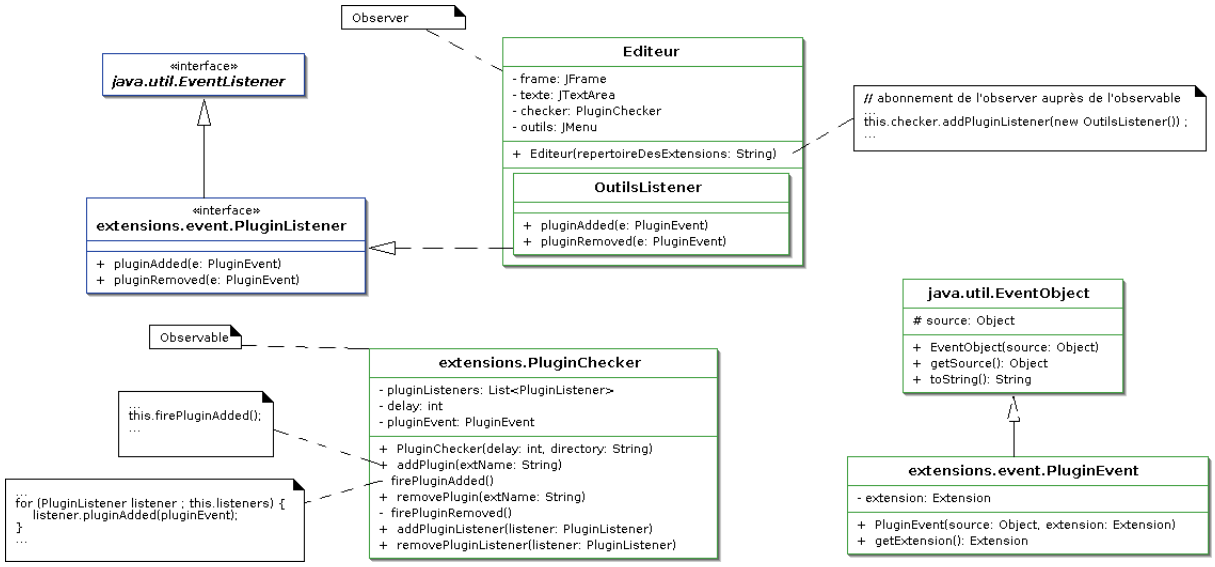
   2'. Définir les classes des `adapters` (optionnel)
3. Définir la classe émettrice (génératrice des événements)
4. Définir les classes réceptrices (les `listeners`)

(Voir les notes de cours pour le détail des différentes étapes)



API java
Voir gestion des évènements dans `javax.swing`.

TP sur les plugins Observation du répertoire contenant les plugins et déclenchement d'un évènement lors de l'apparition d'un nouveau plugins, l'application `Editeur` (qui gère un listener via sa classe interne `OutilsListener`) réagit aux évènements (`PluginEvent`) en prenant en compte l'apparition ou la disparition d'un plugin créés par `PluginChecker` qui joue le rôle de l'émetteur d'évènement `Xxx`.



(N'apparaissent sur le schéma que les éléments pour illustrer le design pattern.)