

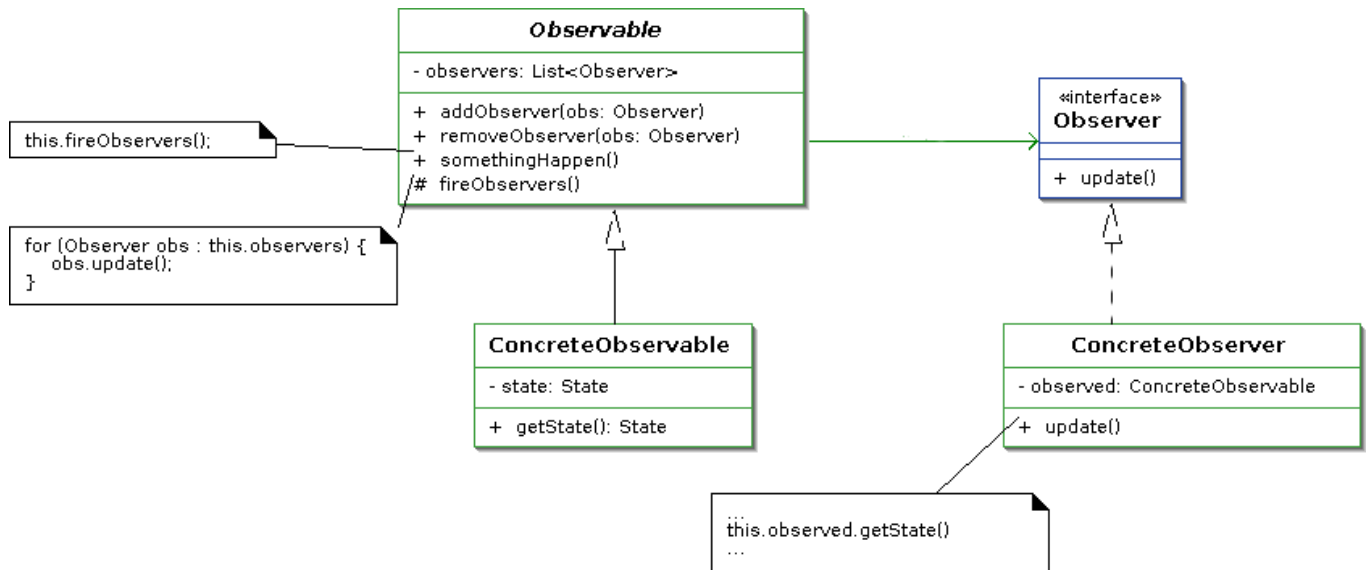
## Design Pattern : observer

### 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

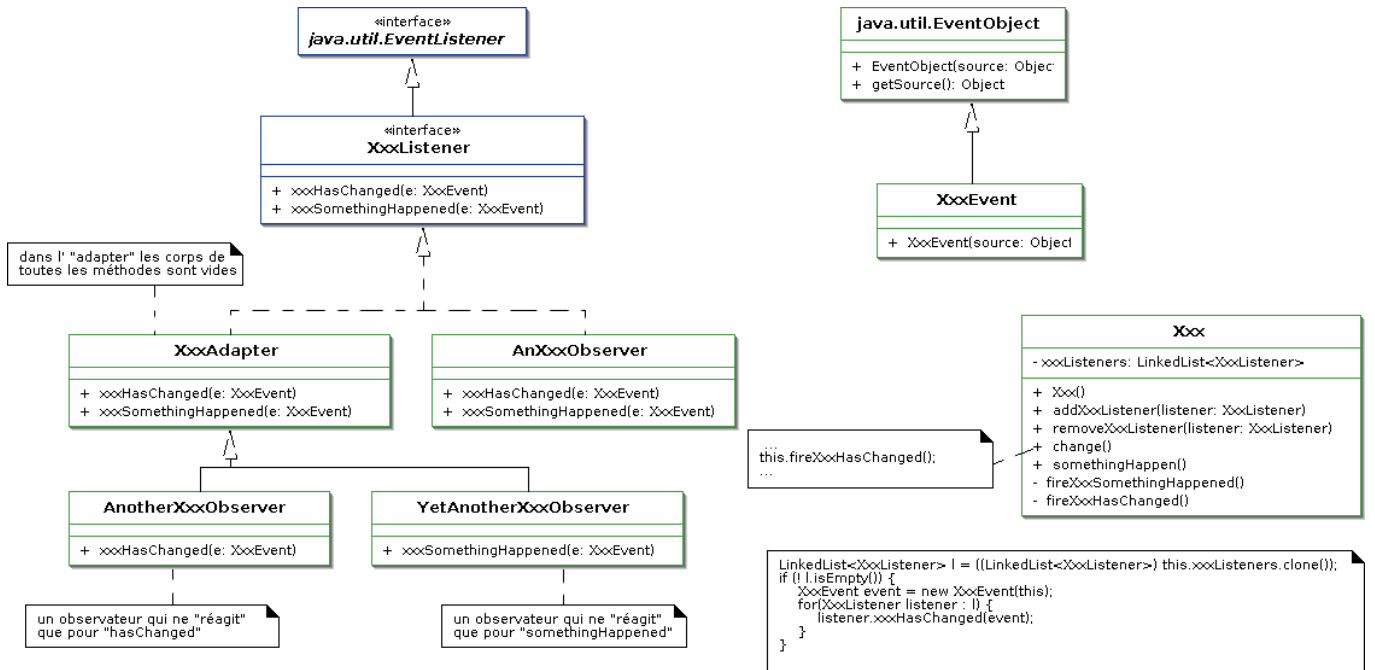
- Plusieurs objets peuvent être avertis des événements émis par une source
- Le nombre et la nature des objets avertis ne sont potentiellement pas connus à la compilation et peuvent changer dans le temps.
- L'émetteur de l'événement et le récepteur ne sont pas fortement liés.

### Exemples

#### Mise en place gestion d'événements.

1. définir la classe des événements
2. définir l'interface des listeners (*observers*)
3. définir la classe émettrice (génératrice des événements) (*observable*)
4. définir les classes réceptrices (implémentant l'interface « listener »)

(Voir les fichiers joints à la v3 du projet pour le détail des différentes étapes)



### API java

Voir gestion des évènements dans `javax.swing` : interface graphique dans les TD Questionnaires ou Plugins , ou utilisation du Timer dans le TD Plugins.

### Projet – v3

Les reporters et les bookmakers viennent observer les matchs