

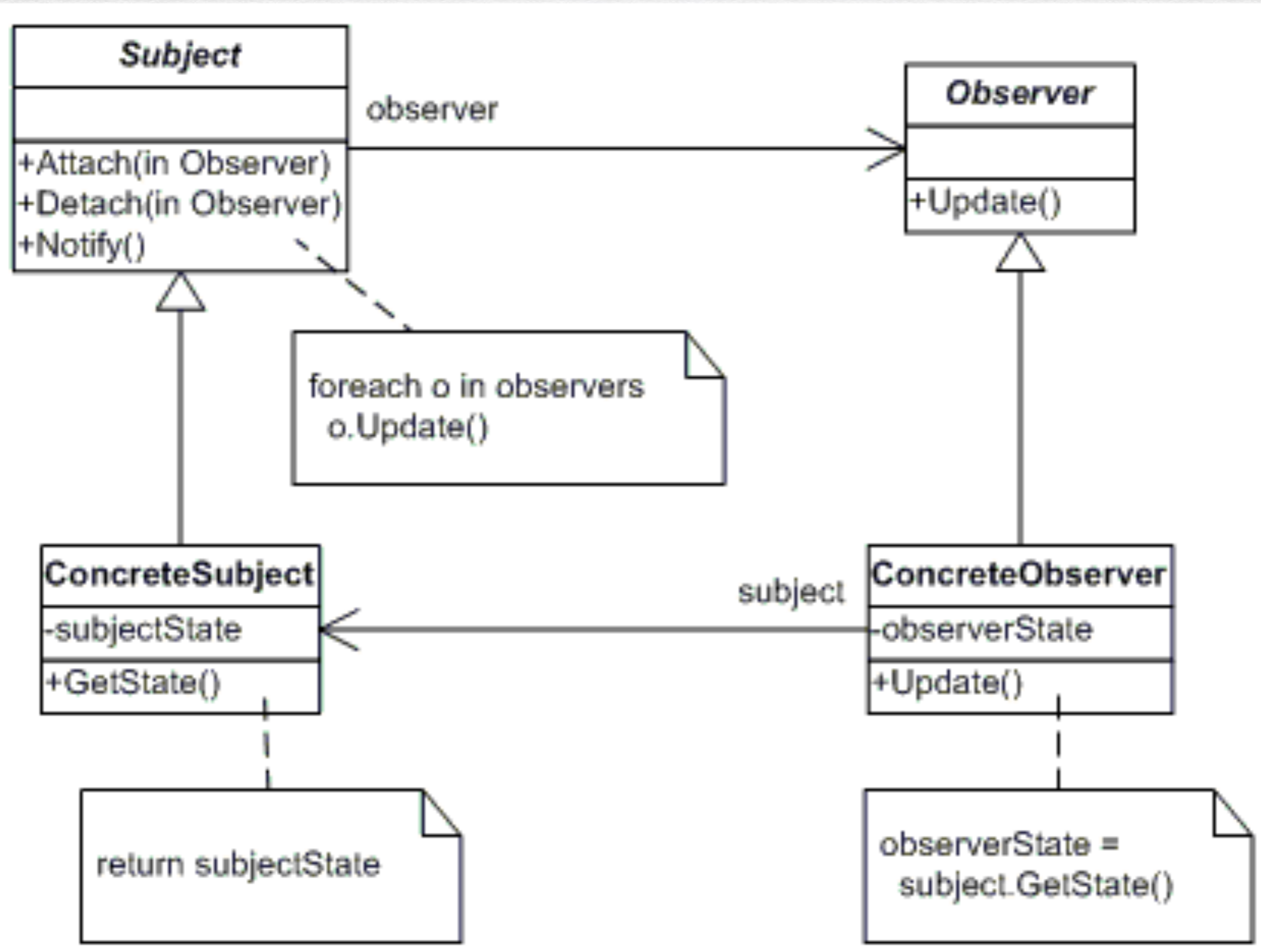
Design Pattern Observer

«Pattern Observer» : le problème

✓ Problème : Mettre en oeuvre une relation de «un vers plusieurs» objets afin que plusieurs objets puissent être notifiés du changement d'état d'un objet et puisse réagir.

Il est très utilisé en IHM, mais peut être appliqué dans bien d'autres cas.

Pattern Observer : Solution



«Pattern Observer» : les rôles

✓ Rôles : Un sujet et des «observers» (le sujet doit devenir «observable»)

✓ Responsabilités :

➡ Le sujet :

- ▶ notifie les observeurs quand il «change»
- ▶ permet aux observeurs de s'(de-)enregistrer.

➡ Les observeurs

- ▶ acceptent les notifications

Pattern Observer : Solution

➔ Sujet Abstrait (Observable) :

- ▶ gère les observeurs (*AddObserver(Observer)*)
- ▶ notifie les observeurs (*notifyObservers*)

➔ Sujet (Observable) :

- ▶ A chaque changement d'état, il «notifie» les observeurs (*notify*)
- ▶ Il peut donner son état (*getState*)

➔ Observateur

- ▶ Se met à jour quand il est notifié (*update*)

➔ Observateur (Abstrait)

- ▶ Peut être notifié (*update*)

«Pattern Observer» en java

✓ Le sujet abstrait : classe abstraite
`java.util.observable`

✓ Le sujet concret :

➔ Votre classe qui hérite de observable

➔ *C'est à vous d'appeler* `notifyObservers`

✓ L'observeur abstrait : Interface
`java.util.Observer`

✓ L'observeur concret :

➔ Votre classe qui «implémente» `Observer`

➔ *doit implémenter* `update`

```
Observable
  changed
  obs
  Observable()
  addObserver(Observer) : void
  clearChanged() : void
  countObservers() : int
  deleteObserver(Observer) : void
  deleteObservers() : void
  hasChanged() : boolean
  notifyObservers() : void
  notifyObservers(Object) : void
  setChanged() : void
```

```
Observer
  update(Observable, Object) : void
```



«Pattern Observer» en action

✓ Un forum

➔ On peut poster des messages dans le forum : un message à un titre.

✓ Des abonnés

➔ Un abonné peut recevoir des messages dans ses boites de messages.

✓ Dès qu'un message est posté sur le forum, tous les abonnés sont notifiés.

➔ Certains abonnés enregistrent le message dans leur boîte.

➔ (2) Certains abonnés n'enregistrent que les messages dont le titre contient «IUT» ;-)

«Pattern Observer» en java exemple

```
import java.util.Observable;

public class ObservableObject extends Observable
{
    private int n = 0;
    public ObservableObject(int n)
    {
        this.n = n;
    }
    public void setValue(int n)
    {
        this.n = n;
        setChanged();
        notifyObservers();
    }
    public int getValue()
    {
        return n;
    }
}
```

<http://www.javaworld.com/article/2077258/learn-java/observer-and-observable.html>

«Pattern Observer» en java exemple

```
import java.util.Observer;
import java.util.Observable;
public class TextObserver implements Observer
{
    private ObservableObject ov = null;
    public TextObserver(ObservableObject ov)
    {
        this.ov = ov;
    }
    public void update(Observable obs, Object obj)
    {
        if (obs == ov)
        {
            System.out.println(ov.getValue());
        }
    }
}
```

«Pattern Observer» en java exemple

```
public class Main
{
    public Main()
    {
        ObservableValue ov = new ObservableValue(0);
        TextObserver to = new TextObserver(ov);
        ov.addObserver(to);
    }
    public static void main(String [] args)
    {
        Main m = new Main();
    }
}
```

DP Observateur : Résumé

✓ Intention :

➡ Définit une interdépendance de type un à plusieurs, de telle façon que quand un objet change d'état, tous ceux qui en dépendent en soient notifiés et automatiquement mis à jour.

✓ Applicabilité : Utilisez l'Observateur dans les situations suivantes :

➡ Quand un concept a deux représentations, l'une dépendant de l'autre. Encapsuler ces deux représentations dans des objets distincts permet de les réutiliser et de les modifier indépendamment.

➡ Quand la modification d'un objet nécessite de modifier les autres, et que l'on ne sait pas combien sont ces autres.

➡- Quand un objet doit être capable de faire une notification à d'autres objets sans faire d'hypothèses sur la nature de ces objets. En d'autres termes, quand ces objets ne doivent pas être trop fortement couplés.

http://www.goprod.bouhours.net/?page=pattern&pat_id=16