

## TP Design Pattern :

<https://github.com/Hugo-Demangeat/TemperatureSensor>

Le couplage : .....	1
Problème : .....	1
Solution : .....	1
L'Open/Close : .....	1
Problème : .....	1
Solution : .....	1

### Le couplage :

#### Problème :

```
public void setTemperature(int temperature) {  
    this.temperature = temperature;  
    // Mauvaise pratique : appel direct et rigide  
    displayTemperature();  
    triggerAlarm();  
}
```

Il manipule directement displayTemperature et triggerAlarm, il est donc fortement couplé.

#### Solution :

Afin de ne pas manipuler directement, il faut créer une interface observer qui sera implémenté par TemperatureSensor qui ne connaît alors plus les classes concrètes, il se contente de notifier une liste d'observateurs via cette interface.

Ainsi, on peut ajouter de nouveaux comportements sans modifier le capteur, ce qui réduit le couplage et respecte le principe OCP.

```
private List<TemperatureObserver> observers = new ArrayList<>();  
  
public void addObserver(TemperatureObserver obs) {  
    observers.add(obs);  
}  
  
private void notifyObservers() {  
    for (TemperatureObserver obs : observers) {  
        obs.update(temperature);  
    }  
}
```

## L'Open/Close :

### Problème :

```
public void setTemperature(int temperature) {  
    this.temperature = temperature;  
    // Mauvaise pratique : appel direct et rigide  
    displayTemperature();  
    triggerAlarm();  
}
```

Il appelle directement displayTemperature et triggerAlarm donc pour tout ajout il modifier le capteur lui-même.

### Solution :

Pour respecter le principe Open/Closed, le capteur ne doit plus appeler directement les actions comme l'affichage ou l'alarme. J'ai donc introduit une interface TemperatureObserver, que tous les comportements à exécuter implémentent.

Le capteur notifie seulement cette interface, sans connaître les classes concrètes. Ainsi, on peut ajouter de nouveaux observateurs sans modifier la classe TemperatureSensor.

```
public class DisplayTemperature implements TemperatureObserver {  
    @Override  
    public void update(int temperature) {  
        System.out.println("Température actuelle : " + temperature + "°C");  
    }  
}  
  
public class TriggerAlarm implements TemperatureObserver {  
    @Override  
    public void update(int temperature) {  
        if (temperature > 30) {  
            System.out.println("Alerte : température trop élevée !");  
        }  
    }  
}
```

```
sensor.addObserver(new DisplayTemperature());  
sensor.addObserver(new TriggerAlarm());
```