TP Design patterns

TELECOM Nancy 2016

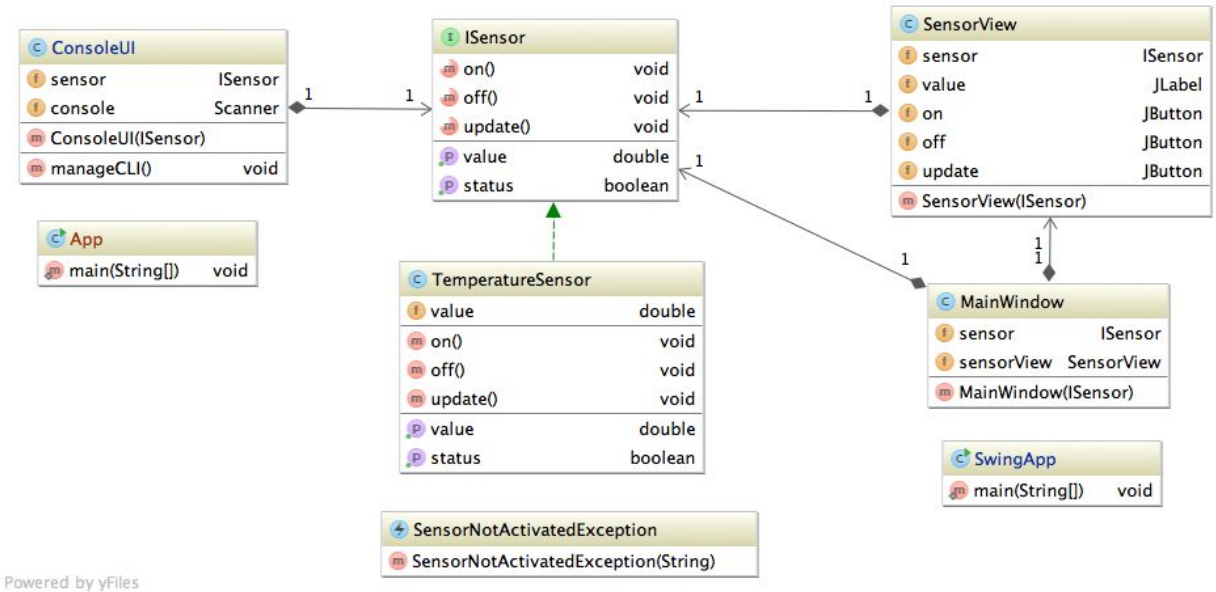
L'objectif de ce TP est de mettre en oeuvre différents design patterns.

Vous devrez utiliser github pour faire votre développement et créer un fork à partir de cette adresse
https://classroom.github.com/assignment-invitations/002633a03de4615adb799d22b56c86a0

Clonez ce dépot et faites le test de la classe principale `App`.

Pour chaque exercice, vous ferez un commit tagué avec le nom de l'exercice
Par exemple
```
git tag adapteur
```

L'application `SwingApp` ne fonctionne pas, c'est normal …



1 Adapteur

Les applications de gestion des capteurs proposées (`App` et `SwingApp`) manipulent des

capteurs réalisant l'interface `ISensor`. On souhaite que ces applications soient capables de manipuler un ancien capteur de température (`LegacyTemperatureSensor`) qui ne réalise pas cette interface, et ce, sans modifier ni le code des clients, ni le code du capteur de température. Appliquer le patron Adapteur (*Adapter*) pour répondre à ce besoin. En vous basant sur le code fourni de l'application `App`, écrivez un nouveau client mettant en oeuvre l'adapteur.

2 Observateur

Appliquez le pattern Observateur (*Observer*) pour que la vue `SensorView` soit mise à jour lorsque la valeur du capteur est rafraîchie et ainsi rendre fonctionnelle l'application `SwingApp`.

3 Etat

Ecrivez une implantation d'un nouveau capteur réalisant l'interface `ISensor` en utilisant le patron Etat (*State*). La mise à jour de la valeur se fait quand on appelle la méthode `update`. Vous pouvez partir de l'implantation `TemperatureSensor` fournie.

4 Proxy

Utilisez le patron Proxy pour permettre d'enregistrer un journal (*log*) de l'utilisation du capteur. Chaque appel à une méthode du capteur donne lieu à une écriture sur la console (Date, méthode appelée, valeur de retour)

Quel est le principe commun utilisé par les patrons Décorateur, Etat et Proxy ?

5 Décorateur

Utilisez le pattern Décorateur (*Decorator*) pour permettre de changer à la demande l'unité de la valeur affichée par le capteur. La valeur retournée est en Celsius. Faites un décorateur avec une conversion en Fahrenheit et un décorateur qui arrondit la valeur retournée.
La formule de conversion est disponible sur cette page wikipedia.

6 Commande

Dans la vue `SensorView`, mettez en place le pattern Commande (*Command*) pour remplacer les appels au capteur par des exécutions de commandes.
Question optionnelle :
Vous ajouterez un menu créé dynamiquement à partir du fichier de properties qui contiendra les noms de classes implantant les commandes sur le modèle du fichier `commande.properties`

7 Fabrique

Implantez une fabrique (*Factory Method*) qui permette de découpler l'instanciation du capteur et donc le choix de son implantation de son utilisation. Vous pouvez envisager que l'implantation choisie fasse partie d'un fichier de properties.