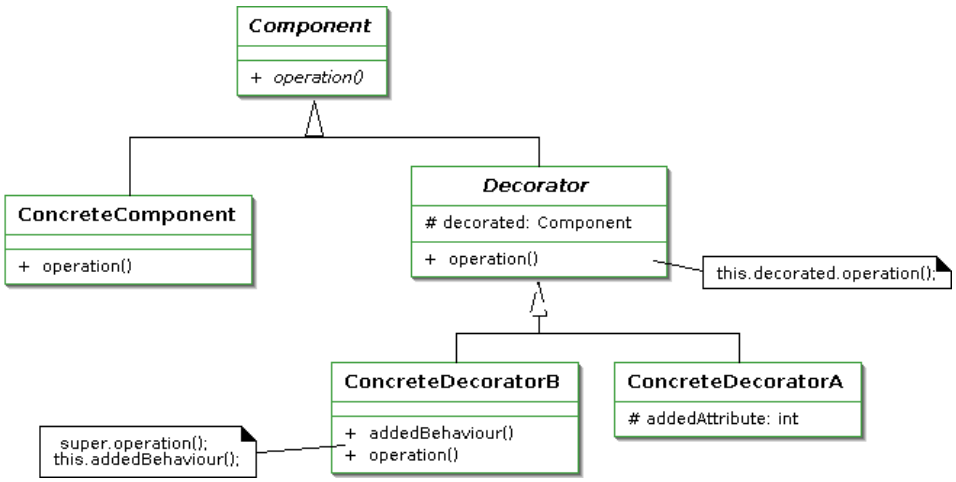
Design Pattern : décorateur

Intent

Attach additional responsibilities to an object dynamically. Decorators provide a flexible alternative to subclassing for extending functionality.

- utilisation de la composition en alternative à l'héritage
- évite la multiplication (voire l'explosion) des sous-classes à créer

Structure



Eléments caractéristiques

- un attribut du type de base (`Component`) dans le décorateur : l'élément **décoré**, fourni à la construction du décorateur le plus souvent.
- par défaut, dans ses méthodes le décorateur abstrait (`Decorator`) se contente de déléguer le travail au décoré, donc du point de vue du comportement on ne fait pas la différence avec l'objet décoré.
- la décoration (concrète) se caractérise par un ajout d'attributs ou de méthodes et en modifiant "à la marge" (subjectif) les comportements d'une ou des méthodes du décoré, les autres méthodes restant inchangées et reprenant donc le comportement de base du décoré.

Les objets *décorateur* reprenne et/ou s'appuie donc le plus souvent majoritairement sur le comportement de leur *décoré*.

Exemples rencontrés

Dans l'api java

- `java.io.BufferedReader` : décoration d'un `Reader` en ajoutant la possibilité de lire des lignes de texte.

  D'une manière plus générale dans le paquetage `java.io` on trouve de nombreux exemples de mise en œuvre du décorateur avec les "enveloppes" successives d'objets de type `Stream` qui ajoute successivement une "couche de décoration" pour ajouter des fontionnalités :

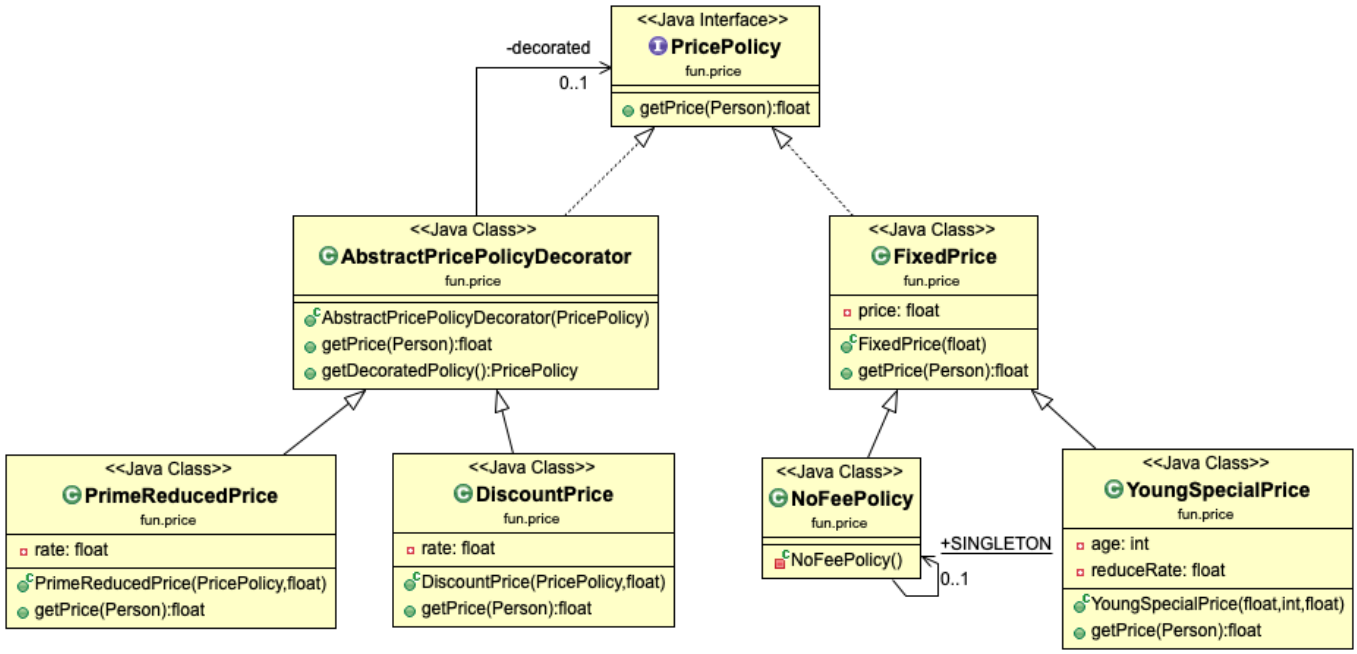
```
FileInputStream in = new FileInputStream("/tmp/exemple");
 // décoration 1 : ajout des readInt, readFloat, etc.
DataInputStream dataIn = new DataInputStream(in);
 // décoration 2 : ajout de la "lecture bufferisée"
BufferedInputStream bufDataIn = new BufferedInputStream(dataIn);
```

- `javax.swing.JScrollPane` : décoration d'un `java.awt.Component` en ajoutant les fonctionnalités liées aux ascenseurs (cf. TD Questionnaires).

```
Jpanel myBigPanel = new Jpanel();
...               // add components to myBigPanel
// décoration de myBigPanel : ajout des ascenseurs de défilement
JScrollPane paneLWidthScrollBar = new JScrollPane(myBigPanel);
```

TD – Attractions : politique de prix réduit

La politique de prix *discount* est obtenue par une décoration de la politique de prix « normale » en appliquant un pourcentage de réduction sur le prix du décoré.



La méthode `getPrice(Person)` dans *DiscountPrice* est redéfinie comme suit:

```
public float getPrice(Person person) {
    return super.getPrice(person) * this.rate;
}
```

Le taux de réduction vient donc “décorer” le prix normal.