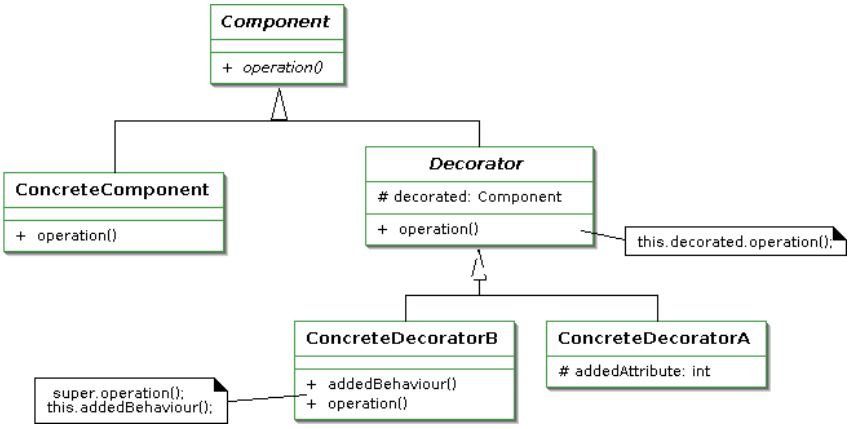
Design Pattern : décorateur

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 dans le décorateur : l'élément décoré
- par défaut dans les méthodes, le décorateur se contente de déléguer le travail au décoré, donc par défaut, du point de vue du comportement on ne fait pas la différence avec l'objet décoré.x
- la décoration se caractérise par un ajout d'attributs ou de méthodes et en modifiant "à la marge" (subjectif) les comportements des méthodes du décoré

Exemples rencontrés en TD

Dans l'api java

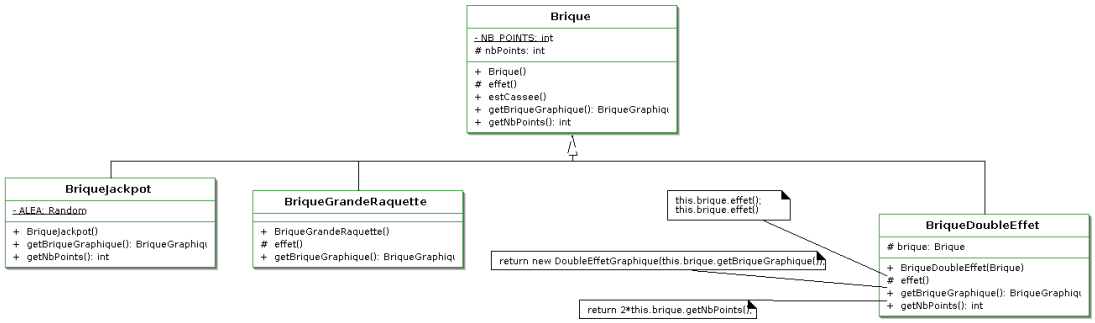
- `java.io.BufferedReader` : décoration d'un `Reader` en ajoutant la possibilité de lire des lignes de texte.

  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);
```

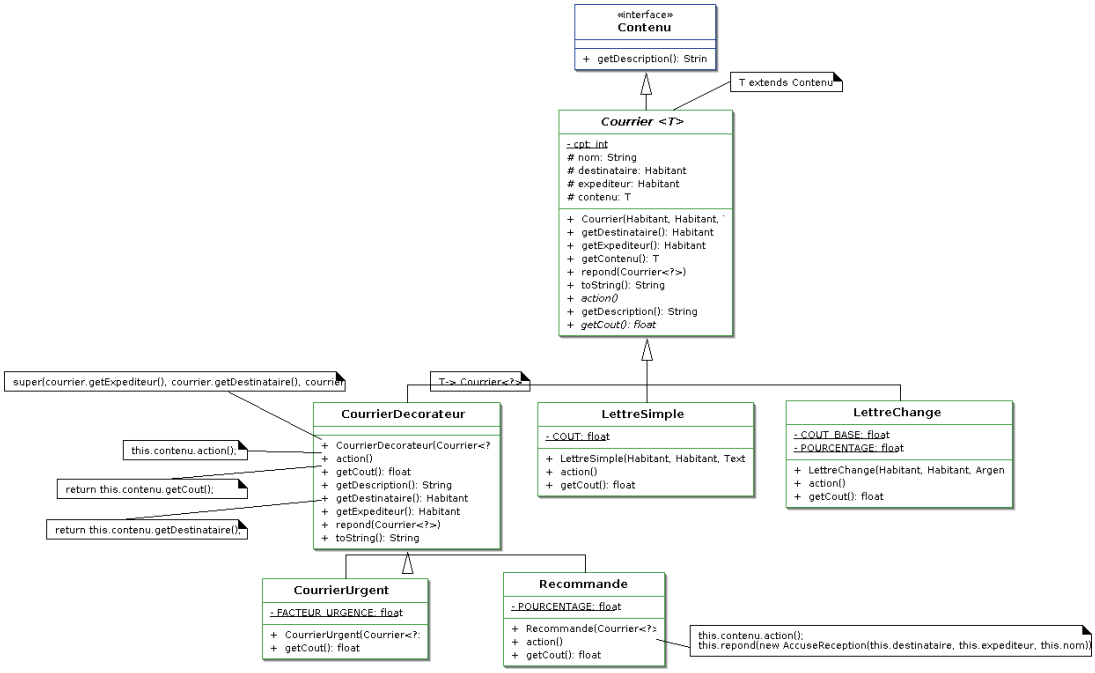
- `javax.swing.JScrollPane` : décoration d'un `java.awt.Component` en ajoutant les fonctionnalités liées aux ascenseurs.

Casse-briques : briques double effet.



Pour obtenir une brique double jackpot :`new BriqueDoubleEffet(new BriqueJackpot())`

Les courriers recommandés et urgents.



Pour créer une lettre de change envoyée en recommandé urgent :
`new CourrierUrgent(new CourrierRecommande(new LettreChange(exp,dest,new Argent(10.5))`
On peut cumuler les décorations sans avoir à multiplier les sous-classes.