Université Paris Diderot **Programmation Orientée Objet**
L2, EIDD et M1 linguistiques **Année 2015-2016**

TP n° 9

Shéma Observateur-Observé

Exercice 1 Implémentez la structure d'arbre de recherche vu en td (et testez la).

Exercice 2 Implémentez la modélisation de la Banque vu en td.

Exercice 3 On va modéliser ici le fonctionnement d'un réseau social semblable à Twitter. On va utiliser pour cela le shéma de conception Observateur/Observé. Java fournit une implémentation de la classe `Observable`, et une interface `Observer`, que vous êtes encouragés à utiliser. Effecuer la modélisation UML, puis implémentez là. L'idée générale est la suivante :

- Chaque utilisateur dispose d'une liste de messages correspondant aux messages qu'il émet.
- Chaque utilisateur a une (autre) liste de messages qui correspond au message qu'il reçoit.
- Chaque utilisateur peut décider de suivre un certain nombre d'autres utilisateur.
- Si un utilisateur en suit un autre, il doit être notifié de chaque message écrit par cet utilisateur. En pratique, cela signifie que le message émis est ajouté à sa liste de message reçus. L'utilisateur peut ensuite décider (ou non) de le retweeter(c'est à dire de le rajouter à sa liste de message émis).

Il s'agit en fait d'un cas particulier du shéma de conception Observateur/Observé, où l'utilisateur est à la fois l'observateur et l'observé :

- Un utilisateur observe tous les utilisateurs qu'il suit.
- Il peut observer les messages qu'ils émettent.

On vous demande :

1. D'écrire une classe `Utilisateur`, qui hérite de la classe `Observable` et implémente l'interface `Observer`.
2. D'écrire une classe `Reseau`, qui contient l'ensemble des utilisateursdu réseau.
3. De permettre la création et la gestion (envoyer des messages, les réemmettre, suivre des gens) intéractive d'utilisateurs. (On pourra uttiliser les classes entrées et sorties du tp précédent...)

Exercice 4 Décorateur.
On va ici utiliser un autre Shéma de conception : le décorateur.
On a une situation de départ où l'on connait le menu d'un restaurant :

1. Ecrivez une classe abstraite dessert, avec un attribut nom et un attribut prix (et les getteurs et setteurs associés).

2. Programmez des classes `Crepes`, `Gateaux`... qui hérite de la classe `Dessert`

Maintenant, on veut considérer la possibilité de créer des nouveaux dessert à partir des anciens en utilisant des supléments : chocolat, chantilly...On va développez ici une solution correspondant au shéma de conception décorateur.
Pour cela, on crée une classe abstraite `Supplément`, qui :

— hérite de la classe Dessert
— a un attribut protected `Dessert dessertOriginel` (qui correspond au dessert auquel on ajoute le supplément)
— a un attribut `int prix` qui correspond au cout suplémentaire induit par le supplément.

On peut remarquez que cette solution uttilise à la fois l'héritage et la composition...

1. Ecrivez la classe `Supplement`
2. Ecrivez des classes `Chocolat`, `Chantilly`... qui héritent de `Supplement`.
3. Ecrivez une class Test, qui crée un objet de la classe `Dessert` correspondant à une crepe avec du chocolat, et qui afiche son prix.
4. On peut également spécifier que certains suppléments ne peuvent etre ajoutés qu'à certain desserts. Implémentez un exemple.