TD5 : Composite, Decorator

Exercice 1 - Composite et Decorator

Vous regrettez l'époque où vous aviez le temps de jouer à Command & Conquer ? Voici l'occasion de créer votre version du jeu ! Commencer par télécharger le fichier [wargame0.zip](http://igm.univ-mlv.fr/ens/IR/IR2/2012-2013/ProgOO/wargame0.zip) (<http://igm.univ-mlv.fr/ens/IR/IR2/2012-2013/ProgOO/wargame0.zip>) Importer-le dans le répertoire source de votre projet courant dans Eclipse.

Le but de cet exercice est de modéliser des unités militaires. Certaines unités (AircraftCarrier, ArmyTruck, Helicopter et Destroyer) peuvent jouer un rôle de transport de troupe et contenir d'autres unités.

Dans le paquetage `fr.umlv.poo.wargame0` se trouvent toutes sortes d'unités militaires sous forme de classes.

1. Organiser les classes pour que les unités effectuant du transport de troupe en utilisant le design pattern composite. Les transporteurs devront donc posséder des méthodes permettant d'ajouter ou de retirer d'autres unités et de visualiser la liste des unités.
2. Modifier votre implantation pour qu'il soit impossible de transporter une unité étant elle même un transporteur.
3. Faites en sorte que la puissance de feu, la vie et la vitesse d'un transporteur soit la somme des caractéristiques des unités de même armes qu'il transporte. Par exemple si un destroyer transporte un soldat et deux marines, ses caractéristique ne dépendront que des deux marines.
4. Nous souhaitons maintenant ajouter un système de bonus qui permet de booster les fonctionnalités de feu, de vie et de vitesse des soldats.
Sachant que l'on souhaite avoir des bonus qui ajoute de la puissance de feu OU de la vie OU augmente la vitesse ainsi que des bonus multipliant par un facteur l'ensemble des caractéristiques d'un soldat. Proposer une solution d'implantation en utilisant le design pattern Decorator.
5. faites en sorte que l'on puisse aussi appliquer le système de bonus aux transporteurs.

Exercice 2 – Decorator et Composite

Pour illustrer les possibilités de ces design patterns avec les flux, on veut :

- écrire des classes `UpperCaseWriter` et `LowerCaseWriter` qui permettront de forcer l'écriture en majuscule ou en minuscule
- faites le diagramme UML du Design Pattern que vous venez d'utiliser
- écrire une classe `CompositeWriter` qui permettra (entre autres) d'écrire de manière simultanée dans un fichier avec le texte en minuscule et dans un autre fichier avec le texte en majuscule.

Exemple du code client que l'on veut pouvoir écrire :

```
CompositeWriter cw = new CompositeWriter();
cw.add(new UpperCaseWriter(new FileWriter(new File("test-u"))));
cw.add(new LowerCaseWriter(new FileWriter(new File("test-l"))));
Reader isrf = new UpperCaseReader(
    new BufferedReader(
        new InputStreamReader(
            new FileInputStream(
                new File("src\\fr\\uml\\poo\\dp\\composite\\books.xml"))));
int c;
while ((c = isrf.read()) > 0) {
    cw.write(c);
}
cw.close();
```

Une fois que votre code fonctionne, que se passe-t-il si l'écriture dans le fichier est faite avec une des autres méthodes `write()`, par exemple `write(char[] cbuf)` ?

Exemple avec lecture et écriture par 100 chars

```
char[] t = new char[100];
CompositeWriter cw2 = new CompositeWriter();
cw2.add(new UpperCaseWriter(new FileWriter(new File("test-u2"))));
cw2.add(new LowerCaseWriter(new FileWriter(new File("test-l2"))));
Reader isrf2 = new BufferedReader(
    new InputStreamReader(
        new FileInputStream(
            new File("src\\fr\\uml\\poo\\dp\\composite\\books.xml"))));
while (isrf2.read(t) > 0) {
    cw2.write(t);
}
cw2.close();
```

Que faut-il faire pour résoudre le problème ?