# TD5 : Composite, Decorator, Flyweight

## 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 td1war.zip ( http://igm.univ-mlv.fr/ens/Master/M1/2007-2008/GenieLog/td1war.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.gl.td2.war 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 possèder 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 sortes 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, de la vite 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.

Enfin, faites en sorte que l'on puisse aussi appliquer le système de bonus aux transporteurs. A vos claviers !

## Exercice 2 - Flyweight

Le centre ville va être refait. Le logiciel de conception doit pouvoir représenter les rues et les différents pavés qui vont être posés. 3 types de pavés vont être posés :

1. le standard, décrit par ses dimensions, sa couleur, l'endroit où il va être posé (supposons une position terrestre donné par une paire de  Double), son poids, etc.
2. les pavés de bordures, un peu plus jolis, qui seront mis en bordure des routes. Leur description est donc quasi identique à celle des pavés standards.
3. les pavés fantaisies, avec des initiales ou des dessins et qui seront disséminés dans la ville. Les pavés fantaisie sont décrits comme les pavés standards avec, en plus, une indication d'initiale ou un numéro de dessin.

1. Rappeler le principe du design pattern flyweight
2. On souhaite implémenter, en utilisant le pattern Flyweight, le coeur de ce logiciel de conception :
   1. on doit pouvoir positionner des pavés à des positions terrestres
   2. on doit pouvoir afficher la description du pavage prévu

## Exercice 3 - Proxy Pattern

Voici quelques classes permettant de gérer un album de photo td4photo.zip (http://igm.univ-mlv.fr/ens/IR/IR2/2008-2009/POO/td4photo.zip). On cherche pour déboguer à afficher un message lorsque l'on entre ou sort d'une méthode particulière d'un objet.

1. Créez un objet `java.util.logging.Logger` et afficher un message avec le niveau (`Level`) `ALL`.
   Affichez en utilisant la méthode `log` un message de niveau `WARNING`.
2. Rappelez le principe du design pattern `Proxy`.
   Écrivez la méthode `createPhotoProxy(Photo photo,Logger logger)` dans la classe `PhotoFactory` qui prend un objet de type `Photo` et un logger et qui renvoie un objet de type `Photo`. Lorsqu'elle est appelée, elle doit afficher sur le logger, lors de l'entrée dans une méthode le message `enter method` suivi du nom de la méthode, et lors de la sortie de la méthode le message `exit method` suivi du nom de la méthode.
   Pour tester le proxy, changez le code de la méthode `createPhoto(File file)` de sorte que, si l'on crée la factory avec un logger, on utilise le proxy.
3. On souhaite écrire un proxy générique qui affiche les messages d'entrée et de sortie quel que soit l'objet.
   Pour cela nous allons utiliser la classe `java.lang.reflect.Proxy`.
   Écrivez une méthode `createProxy()` générique et correctement typée créant un proxy générique.
4. Modifiez votre code pour que ne soient affichées que les méthodes ayant l'annotation `Log`.