TD5 : Flyweight, Proxy

Exercice 1 - 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 1 - 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.

2. 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`.
3. 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.
4. 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.
5. Modifiez votre code pour que ne soient affichées que les méthodes ayant l'annotation `Log`.