TD n°3 sur les patterns

Il s'agit de modéliser en UML les trois problèmes soumis en s'appuyant le plus possible sur des patterns.

Dans la solution finale on intégrera les trois problèmes en un seul (figures composites avec représentations différentes et proposant la forme et le cri et on bâtira une modélisation unique faisant intervenir tous les patterns.

Enoncé 1

- Une figure géométrique est composée de figures simples ou composées; Une figure composée est constituée de plusieurs figures et une figure simples peut être un point, une ligne ou un cercle. Une figure peut être dessinée ou translatée.

1. Donnez la modélisation des classes de cette situation.
2. De manière plus générale, la notion d'objet composite se retrouve dans beaucoup d'applications. De ce fait, il est intéressant d'utiliser un patron de conception qui modélise les objets composites et de les instancier en fonction des objets considérés.

- Etendez la solution précédente afin de modéliser n'importe quel ensemble d'objets décrits par une hiérarchie d'agrégation d'objets ayant le même comportement.

Enoncé 2

- On souhaite proposer aux clients un outils de dessin des figures géométriques qui s'adapte à l'environnement informatique. Les interfaces, la qualité des dessins dépendent de l'environnement.

1. Proposez une modélisation qui isole le dessin d'une figure géométrique et qui la spécialise en fonction de l'environnement.
2. Selon la même principe et d'une manière plus générale, on souhaite concevoir une structure qui permet à un client de voir une classe et ses opérations de haut niveau. Cette structure doit séparer complètement la définition abstraite des classes et leur implémentation.

- Proposez un schéma de modélisation.

Enoncé 3

- Un éditeur de jeux possède un jeu permettant aux enfants de connaître les animaux. Les enfants peuvent, en particulier, apprendre la forme et le cri des animaux parmi lesquels le chat et la vache. Le chat est modélisé par la classe *LeChat* possédant au moins les deux méthodes *formeChat()* et *criChat()* et la vache est modélisée par la classe *LaVache* possédant les deux méthodes *criVache()* et *formeVache().*
- Comme le montrent les noms des méthodes, la première spécification de ce jeu est propre aux animaux modélisés. L'éditeur souhaite améliorer ce jeu en créant une interface commune à tous les animaux qui lui permette d'en ajouter de nouveaux, sans modifier l'interface avec le client, et d'utiliser le polymorphisme dans la gestion des animaux (manipuler des troupeaux …).

1. Proposez une modélisation des classes pour cette nouvelle version du jeu en faisant apprendre le client.
2. On souhaite réutilisez tout le code développé dans la version précédente? Proposez une modélisation permettant d'incorporer les anciennes méthodes pour éviter de les réécrire.
3. Est-il possible de généraliser ce raisonnement pour des applications de même type ? Si c'est le cas, proposez l'utilisation d'patron générique correspondant.