TD n°7 : Patterns Factory Method et Abstract Factory

Exercice 1 (Factory Method : connexion à une BDD)

Supposons qu'un serveur dispose de deux types de base de données : une base Oracle et une base MySQL. Lorsqu'un client se connecte, le SGBD (Système de Gestion de Base de Données) crée un objet de connexion pour gérer la communication entre le client et la base de données. Nous aurons donc deux types de connexions à créer (selon le client) : les connexions Oracle et les connexions MySQL. Pour chaque type de connexions, le système doit être capable de proposer une connexion normale et une connexion sécurisée. Pour tout cela, nous allons utiliser le pattern Factory Method.

1. Faire le diagramme UML du programme. Identifiez les classes produits et les classes créatrices. N'oubliez pas qu'un serveur gère plusieurs communications à la fois, donc vous aurez besoin d'une `ArrayList` (par exemple) quelque part.
2. Écrire en Java les classes définies par le diagramme UML.

Exercice 2 (Abstract Factory : interfaces graphiques en fonction de l'OS)

Le but de cet exercice est de proposer une modélisation d'un système permettant la création de différents composants d'une interface graphique (boutons, menu, ...) en fonction du système d'exploitation (Linux, Windows, ...) auquel ils sont destinés.

Le modèle doit permettre de créer des composants sans devoir connaître leurs classes concrètes, afin d'être indépendant de la façon dont les composants évoluent. Le pattern Abstract Factory semble donc bien adapté.

1. Faire le diagramme UML de ce programme. On considère que les boutons ont une couleur, une largeur et une hauteur, et les menus simplement une largeur.
2. Écrire en Java les classes définies par le diagramme UML.
3. Rajouter un nouveau système d'exploitation (Mac OS X, par exemple).