


Les patrons de *construction* – partie 2

 Le présent TP s'appuie sur celui de la semaine dernière. Il est nécessaire d'avoir terminé la modélisation et l'implémentation du TP de la semaine dernière pour réaliser l'exercice de cette fiche.

Exercice 1 – Chargement de dessins vectoriels depuis un flux XML

Vous en avez assez d'écrire des programmes de démonstration pour tester votre application graphique de dessin vectoriel. De plus vos utilisateurs auront besoin, tôt ou tard, d'un format d'entrée/sortie pour sauvegarder leur travail. On propose de créer un dialect XML pour sérialiser dans un flux nos dessins en mémoire.

Le schéma est décrit dans la DTD suivante (fournie avec ce sujet) :

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <!ELEMENT drawing (circle|line|rectangle)*>
3 <!ATTLIST drawing xmlns CDATA #FIXED "http://www.univ-rouen.fr/drawing">
4
5 <!ENTITY % color "(black|red|green|blue)">
6
7 <!ELEMENT circle (point,radius)>
8   <!ATTLIST circle color %color; #REQUIRED>
9
10 <!ELEMENT line (point,point)>
11   <!ATTLIST line color %color; #REQUIRED>
12
13 <!ELEMENT rectangle (point,point)>
14   <!ATTLIST rectangle color %color; #REQUIRED>
15
16 <!ELEMENT point EMPTY>
17   <!ATTLIST point x CDATA #REQUIRED>
18   <!ATTLIST point y CDATA #REQUIRED>
19
20 <!ELEMENT radius (#PCDATA)>
```

Vous trouverez également dans l'archive de ce TP quelques fichiers de test (`drawing.xml` et `pencil.xml`).

Dans le cadre de ce TP, on s'intéresse uniquement au processus de désérialisation (chargement), la sérialisation (sauvegarde) ne sera pas traitée. On souhaite ajouter dans notre application un module qui, à partir du nom d'un fichier, renvoie une liste de `Drawables` extraite à partir de ce fichier que l'on pourra donner au `GraphicViewer`. On souhaite en outre pouvoir toujours choisir entre les différents styles de dessin (normal ou à main levée).

Question 1.1 : Quel patron de conception proposez-vous d'utiliser pour répondre à notre problème ? Rappelez son schéma de principe (sans regarder le cours) ainsi que les responsabilités des différents acteurs.

Question 1.2 : Donnez la modélisation UML complète de votre solution. Vous décrirez vos interfaces de manière exhaustive !

Question 1.3 : Si l'on souhaite ajouter un nouveau format de données en entrée (ex : JSON, binaire, etc.), quelle modification faudra-t-il faire au programme ? Faut-il modifier l'architecture ?

Question 1.4 : Implémentez et testez votre solution à l'aide des fichiers fournis.

Note : on pourra s'inspirer du code donné en cours utilisant SAX. Attention, il est recommandé d'utiliser la validation à partir de la DTD (appel à `setValidating(true)` sur votre *ParserFactory*).

Note2 : la tâche étant complexe, on ne respectera pas tous les principes dans l'implémentation privée.