M1 Informatique - HMIN102 - Ingenierie Logicielle

Schémas de conception (*Design Patterns*)

Ayant réalisé tout ou partie d'un logiciel il est important d'avoir su le rendre réutilisable, c'est ce que nous étudié avec les TDs précédents. Il est également important de savoir décrire et généraliser les solutions logicielles que l'on a découvertes. C'est le rôle des schémas de conception. Le but de ce TD, exercice de spécification puis d'abstraction, est de vous faire redécouvrir et décrire une solution logicielle pour un problème récurrent. L'exercice du TP est de programmer un exemple du schéma défini en TD.

1 Un problème : une simulation d'un système de mémoire secondaires (fichiers et répertoires)

Considérons l'exemple des éléments de stockage en mémoire secondaire fournis par un système d'exploitation : fichiers, liens symboliques et dossiers (répertoires). Soit à réaliser une représentation objet de ces éléments en vue d'en programmer une **simulation**. On a les éléments de spécification et de conception suivants :

- un dossier (*Directory*) peut contenir des fichiers (*File*), des liens symboliques (*Link*) et des dossiers,
- un fichier possède un contenu que l'on représentera dans la simlulation par une *String*.
- tout élément de stockage est contenu dans un dossier parent. Exception : le parent du dossier racine est vide.
- tout élément de stockage possède un attribut `basicSize` intiquant l'espace de base qu'il occupe en mémoire quand il est vide (0 pour un fichier, 0 pour un lien, 4 pour un dossier).
- tous les éléments de stockage possèdent les méthodes :
  - `int size()` qui rend la taille occupée par l'élément au moment de l'invocation de la méthode.
  - `String absoluteAddress()` qui donne l'adresse absolue de l'élément. Dans notre simulation, cette méthode rend une string équivalente au résultat de la commande unix *pwd*.
- chaque élément de stockage possède des attributs et méthodes spécifiques, comme par exemple :
  - pour un fichier ou un lien, `void cat()` qui affiche son contenu à l'écran,
  - `void ls()` pour un répertoire qui affiche à l'écran la liste des éléments qu'il contient,
  - `int nbElem()` pour un répertoire ou pour un fichier rendant respectivement le nombre d'éléments du répertoire ou de mots du fichier,
  - `Collection find(String name)` pour un répertoire qui rend la collection des adresses absolues des éléments de nom `name` qu'il contient,
  - `Collection findR(String name)` pour un répectoire qui rend la collection des adresses absolues des éléments de nom `name` qu'il contient directement ou par transitivité.

Proposez un schéma UML d'une solution au problème énoncé. En le faisant, réflechissez aux méthodes nécessaires au fonctionnement de l'exemple, celles citées dans ce texte et aussi celles induites par la structure des données.

2 Un autre problème : la programmation avec des composants graphiques

Réaliser un schéma UML de classes décrivant des composants permettant de réaliser des interfaces graphiques. Certains sont simples comme des *boutons*, des *Canvas*, des *boites de choix* ou des *Scrollbars*, d'autres sont des conteneurs comme les *Panels*. Il est possible de placer n'importe lequel des composants graphiques précédemment listés dans un *Panel*. Tous ces composants peuvent être affichés grace à une méthode nommée *draw()*).

3 Généralisation

Etablissez la similitude entre les deux exemples précédents, énoncez le problème informatique commun qui les unifie. Déduisez en un schéma de conception, solution à ce problème. Trouvez lui un nom, trouvez ses "participants" et établissez le schéma UML de sa solution générique. Essayez d'être le plus abstrait possible afin que la description de cette solution couvre le plus de cas possibles.

4 TP : Instantiation et Implantation

Implanter en Java le schéma de conception que vous avez élaboré en l'appliquant à l'exemple des fichiers et des dossiers.