Travaux Dirigés de Java n°4

Maîtrise d'Informatique

— *Etienne Duris* - `Etienne.Duris@inria.fr` —

▸ **Exercice 1. (Stuctures récursives)** *On veut représenter des listes en Java. Écrire une interface* `List` *qui déclare les méthodes suivantes :*

- `List addElement(Object elem);`
- `List insertElementAt(Object elem, int pos);`
- `List removeElement(Object elem);`
- `List removeElementAt(int pos);`
- `List append(List other);`
- `int length();`

Écrire une classe `Cons` *et une classe* `Nil` *qui implémentent cette interface, avec en plus pour* `Cons` *les méthodes* `getHead()`, `getTail()`.

Ajouter la méthode `Enumeration elements()` *à l'interface* `List`*, et implémenter cette méthode dans les classes* `Nil` *et* `Cons` *en utilisant les classes internes (inner classes).*

▸ **Exercice 2. (Programmation Dynamique)** *Le but de cet exercice est d'apréhender la programmation dynamique. Il s'agit d'avoir un programme qui soit capable dynamiquement, à l'exécution, d'utiliser différentes ressources. Nous allons prendre un cas simple. Considérons une interface* `Format` *qui définit des méthodes permettant de faire des conversions ou de sauvegarder des fichiers dans un format donné. Supposons que les classes* `FormatHTML`, `FormatWORD` *et* `FormatTEX` *implémentent cette interface. Nous voulons définir une classe* `LookForFormat` *qui permet de récupérer tous les fonctionalités correspondant à tous les formats existant (c'est à dire à toutes les classes* `Format<??>` *existantes). Notre programme doit être capable de récupérer toutes les classes dans un répertoire donné (disons, le répertoire courant) et de trier celles qui ont un certain nom (disons, qui commencent par* `Format`*). Il devra de plus créer des instances de ces classes, puis appeler sur ces objets des méthodes définies dans leur classe respective.*

À terme, dynamiquement, notre programme sera capable au cours de son execution d'utiliser les fonctionalités de tous les formats disponibles en recherchant toutes les classes de format et en appelant les méthodes définies sur des objets instances de ces classes (sans pour autant les connaître statiquement à la compilation).

1. *Écrire l'interface* `Format` *avec une méthode* `affiche()` *qui se contente d'afficher le format courant, et les classes* `FormatHTML`, `FormatWORD` *et* `FormatTEX` *implémentant cette interface, avec les définitions de méthodes correspondantes.*

2. *Écrire la classe* `LookForFormat`, *et une méthode* `formatFileNames()` *qui récupère dans le répertoire courant tous les noms de fichiers commençant par "Format" et ayant l'extension ".class" (définir pour cela une classe* `MyFormatFilter` *qui implémente l'interface* `java.io.FilenameFilter`*). Cette méthode retourne un tableau de String.*

3. *Écrire une méthode* `formatClassNames()` *qui retourne un tableau de classes associées à chaque nom de classe de format.*

4. *Écrire une méthode* `afficheFormats` *qui, pour chaque classe de format disponible, créé une instance de classes et appelle sur cet objet la méthode* `affiche()`.