Travaux Dirigés de Java n°4

—Maîtrise d'informatique—

▸ **Exercice 1. (Interface et Implémentation)** *Définir une interface* `Stack` *déclarant les trois méthodes classiques d'une pile :* `void push(Object o)`, `Object pop()` *et* `isEmpty()`.

Définir une classe `ArrayStack` *qui implémente l'interface* `Stack` *en utilisant un tableau.*

▸ **Exercice 2. (Classe interne)** *Implémenter dans la classe* `ArrayStack` *une méthode* `Iterator iterator()` *(cf. package* `java.util`*).*

▸ **Exercice 3. (Classe abstraite)** *Toujours dans le domaine des piles, on désire disposer d'une méthode qui permet d'empiler une pile sur une autre. Par exemple, si* `p1` *est du type* `StackableStack` *et s'est vu empiler successivement* `a`, `b` *et* `c` *et que* `p2`*, du type* `Stack`*, s'est vu empiler successivement* `d`, `e` *et* `f` *on souhaite que l'appel à* `p1.push(p2)` *aboutisse à ce qu'on puisse dépiler successivement* `f`, `e`, `d`, `c`, `b` *et* `a` *de* `p1`.

Proposer une architecture de classes, classes abstraites et/ou interfaces permettant d'offrir cette fonctionnalité.

▸ **Exercice 4. (Listes récursives avec une classe)** *Écrire une classe* `Liste` *représentant une liste récursive d'objets. Cette classe contient deux champs,* `head` *et* `tail` *ainsi qu'une constante* `NIL` *représentant une liste vide. Écrire une méthode* `int length()` *qui retourne la longueur de la liste.*

Quels constructeurs définir pour cette classe ? Avec quelles visibilités ?

Redéfinir la méthode `toString()` *pour cette classe* `Liste`.

▸ **Exercice 5. (Interface `IList` et son implantation)**

Définir une interface `IList` *avec sa méthode* `length()`*. Définir deux classes* `Nil` *et* `Cons` *implantant l'interface* `IList` *et représentant les deux types de constituants d'une liste. Ces classes doivent naturellement implanter la méthode* `length()`*, mais également fournir chacune au moins un constructeur et redéfinir la méthode* `toString()`.

Que faire de la constante `NIL` *? Que deviennent les visibilités des constructeurs ?*

Ajouter à l'interface `IList` *une méthode* `IList append(IList l)` *permettant de concaténer de manière applicative (c'est-à-dire sans effets de bord) la liste* `l` *à la liste courante et de retourner la concaténation des deux listes.*

▸ **Exercice 6. (Expressions arithmétiques)** *On désire manipuler des expressions arithmétiques sur des* `double` *et, en particulier, on souhaite pouvoir les évaluer.*

Définir l'interface `Evaluable` *et sa méthode* `double eval()` *que devront implanter les classes d'expressions arithmétiques.*

Définir la hierarchie des classes d'expressions. On pourra considérer que :

- *une expression générale est représentée par une classe abstraite* `Expr` *;*
- *un opérateur général est représenté par une classe abstraite* `Operator` *qui définit une priorité et une représentation sous la forme* `String` *de cet opérateur, ainsi qu'un constructeur permettant d'initialiser ces deux champs ;*
- *les opérateurs unaires (*`Unary`*) et binaires (*`Binary`*) sont des opérateurs particuliers, définissant respectivement comme champs une opérande ou bien deux. Pour chacun de ces deux types, on doit être capable de définir un constructeur initialisant le ou les champs ainsi que la méthode* `toString()` *;*
- *les opérateurs "concrets" habituels (*`Plus`*,* `Div`*,* `UMinus`*, etc...) héritent des opérateurs binaires ou unaires. La méthode* `eval()` *peut être implantée dans chacun d'eux, conformément à leur sémantique ;*
- *les constantes et les variables sont des expressions (*`Expr`*) particulières qui associent une valeur à un nom. Seules les variables peuvent être affectées à une nouvelle valeur après leur création. On pourra discuter des différentes hierarchies de classes permettant de représenter les constantes et les variables.*

Créer différents objets de type `Expr` *et tester les méthodes définies. On ne demande pas ici d'analyser une expression saisie au clavier (pas d'analyse lexicale ou parsing), mais simplement de la créer dans une méthode* `main()` *et de l'évaluer ou de l'afficher.*

Que penser de l'affichage obtenu. Revoir la définition des méthodes `toString()` *afin d'ajouter des parenthèses dans la représentation des expressions, mais seulement là ou c'est indispensable (en fonction des priorités des opérateurs).*