TD 3 : Interfaces et architecture

Objectifs pédagogiques : l'interface comme type, `equals()`, `clone()`.

3.1 Interface « Réversible »

⇒ Dans le paquetage `pobj.util`, définissez une interface `Reversible` déclarant une unique méthode dont la signature est : `+ reverse() : void`.

3.2 Inversion de Chaînes

On souhaite implémenter l'interface `pobj.util.Reversible` pour permettre l'inversion de l'ordre des caractères dans une chaîne. On s'intéresse tout d'abord à la classe standard `java.lang.String`.

⇒ Peut-on implémenter l'interface directement dans la classe `String` ?

⇒ Peut-on étendre la classe `java.lang.String` pour implémenter l'interface ?

La solution consiste en l'écriture d'une classe `ChaineReversible` séparée et proposant à la fois les méthodes de `String` et l'implémentation de l'interface `Reversible`. On n'implémentera que deux ou trois méthodes de `String` en TD : `char charAt(int)`, `String toString()`, `int length()` et bien sûr `reverse()`.

3.3 Vecteur Réversible

3.3.1 Classe VecteurReversible

En utilisant un tableau, définissez une classe `pobj.util.VecteurReversible` qui implémente l'interface `Reversible` et qui propose de plus les méthodes suivantes :

```
+ VecteurReversible(int tailleMaximale)//constructeur avec taille maximale du tableau
+ size(): int //retourne la taille du vecteur
+ add(Object o): void //ajout d'un élément dans le vecteur
+ remove(Object o): void //enlever un élément dans le vecteur (avec equals)
+ get(int index): Object //retourne l'élément situéà l'index i ou null
```

3.3.2 Inversion en profondeur

⇒ Proposez une modification de votre classe `VecteurReversible` de sorte qu'elle ne puisse contenir que des objets eux-mêmes réversibles et que l'inversion (via la méthode `reverse()`) se fasse « en profondeur » récursivement : quand l'objet contenu est un `VecteurReversible`, on l'inverse à son tour jusqu'à atteindre un objet élémentaire (une `ChaineReversible`).

⇒ Codez un exemple d'inversion de vecteur contenant à la fois des chaînes et des sous-vecteurs.

3.3.3 Copies de surface et copies profondes

On considère le code suivant :

```
public static void main(String[] args) {
  ChaineReversible c1 = new ChaineReversible("reversible");
  ChaineReversible c2 = new ChaineReversible("complement");
  VecteurReversible vecteur1 = new VecteurReversible(2);
  vecteur1.add(c1);
  vecteur1.add(c2);
```

```
  VecteurReversible vecteur2 = vecteur1;                                    7
  vecteur2.reverse();                                                       8
  System.out.println(vecteur1);                                             9
  System.out.println(vecteur2);                                             10
}                                                                           11
```

⇒ Représenter un diagramme d'objets de l'objet `vecteur1` :

- après sa création,
- après appel de `vecteur2.reverse()`.

⇒ Que se passe-t-il ? Comment peut-on résoudre le problème ? Ajouter aux classes `VecteurReversible` et `ChaineReversible` les méthodes nécessaires pour résoudre proprement le problème.