TD 3 : Interfaces et clonage

Objectifs pédagogiques :

- **utilisation d'interfaces comme types**
- **clonage d'objets**

3.1 Interface « Réversible »

Question 1. Dans le paquetage `pobj.reversible`, définissez une interface `Reversible` déclarant une unique méthode dont la signature est : `+ reverse() : void`.

3.2 Chaînes réversibles

Nous souhaitons implémenter l'interface `pobj.reversible.Reversible` pour permettre l'inversion de l'ordre des caractères dans une chaîne.

Nous nous intéressons tout d'abord à la classe `java.lang.String`.

Question 2. Pouvons-nous implémenter l'interface directement dans la classe `String` ?

Question 3. Pouvons-nous 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`. Vous n'implémenterez que quelques méthodes de `String` en TD : `char charAt(int)`, `String toString()`, `int length()`, `boolean equals(Object)` et bien sûr `reverse()`.

3.3 Vecteur réversible

Question 4. En utilisant une `ArrayList` capable de contenir tout type d'objets, définissez une classe `pobj.reversible.VecteurReversible` qui implémente l'interface `Reversible` et qui propose de plus le constructeur et les méthodes suivantes :

```
+ VecteurReversible() // constructeur
+ 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
+ get(int i) : Object // retourne l'élément situé à l'index i
+ toString() : String // retourne une représentation textuelle du vecteur (la concaté
    nation des représentations textuelles des éléments, séparées par des virgules)
```

Question 5. Implémentez la classe `DeepVecteurReversible` 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.

3.4 Clonage

Considérons le code suivant :

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

  DeepVecteurReversible vecteur1 = new DeepVecteurReversible();
  vecteur1.add(c1);
  vecteur1.add(c2);

  DeepVecteurReversible vecteur2 = new DeepVecteurReversible();
  vecteur2.add(c1);
  vecteur2.add(c2);
  vecteur2.reverse();
}
```

Question 6. Représentez un diagramme d'objets de l'objet `vecteur1` :

- juste avant l'appel à `vecteur2.reverse()`,
- juste après l'appel à `vecteur2.reverse()`.

Que se passe-t-il ?

Question 7. Comment peut-on résoudre le problème ?

Question 8. Modifiez en conséquence l'interface `Reversible` et les classes qui l'implantent.