Université d'Aix-Marseille III - Licence de math-info 2ème année

I4 : Programmation objet - TP n°6 : classes abstraites

1. *Ecrire la classe abstraite* `Array` *contenant les méthodes suivantes* :

- `public Array(Object[])` : le constructeur.
- `abstract public boolean sup(Object x, Object y)` : méthode abstraite dont l'implémentation ultérieure permettra d'indiquer si x et supérieur ou égal à y.
- `public Object maximum()` : retourne le plus grand objet du tableau en utilisant la méthode `sup`.

Toute classe héritant de `Array` et implémentant la méthode `sup` peut permettre d'obtenir le maximum de ce tableau sans avoir à réécrire la méthode `maximum`.

2. *Ecrire la classe* `ArrayRationnel` *héritant de la classe* `Array` *et implémentant la méthode* `sup` *pour les instances de la classe* `Rationnel`. La classe `Rationnel` contient les deux méthodes `public int getNumerateur()` et `public int getDenominateur()` (dont le sens est explicite). *Compléter cette classe pour qu'elle permette d'exécuter un programme qui affiche le plus grand élément d'un tableau de 5 rationnels.*

3. Maintenant, on veut définir une classe encore plus générique que `Array` : la classe abstraite `Agregat` qui représente un ensemble d'objets dont on ne connaît pas a priori le type de la structure de données qui les regroupe. Pour pouvoir parcourir les éléments de cet agrégat, on a besoin de ce qu'on appelle un *itérateur*. Voici la classe abstraite définissant un itérateur :

```
abstract class Iterator{
    abstract void init(); // se positionne sur le premier élément
    abstract Object suivant(); // passe à l'élément suivant et le retourne
    abstract boolean fini(); // indique si on se trouve sur le dernier élément
    abstract Object courant(); // retourne l'élément courant
}
```

Ecrire la classe abstraite `Agregat` *qui contient les méthodes suivantes* :

- `abstract public boolean sup(Object x, Object y)` : même chose qu'avant.
- `abstract protected Iterator getIterator()` : méthode abstraite destinée à retourner un itérateur sur un objet de type `Agregat`.
- `public Object maximum()` : retourne le plus grand objet du tableau en utilisant la méthode `sup` et un itérateur récupéré grâce à l'appel de la méthode `getIterator`.

Une fois définie une classe concrète (disons `IteratorConcret`) héritant de la classe `Iterator`, la classe héritant d'`Agregat` implémentera la méthode `getIterator` ainsi :

```
protected Iterator getIterator(){
    return new IteratorConcret(x);
}
```

où `x` est le nom de l'attribut (défini dans la classe héritant d'`Agregat`) mémorisant l'agrégat.

4. *Ecrire la classe concrète* `IteratorArray` *qui hérite d'*`Iterator` *pour le parcours de tableau.* Cette classe contiendra notamment un attribut de type `Object []` pour mémoriser le tableau. Un constructeur prenant un paramètre de type `Object []` se contentera de l'affecter à cet attribut.

5. *Réécrire la classe* `ArrayRationnel` *en la faisant cette fois-ci hériter d'*`Agregat` *et utiliser un itérateur de type* `IteratorArray`.

6. *Ecrire la classe concrète* `IteratorListe` *qui hérite d'*`Iterator` *pour le parcours de liste chaînée. Ecrire la classe* `ListeRationnel` *en la faisant hériter d'*`Agregat` *et utiliser un itérateur de type* `IteratorListe`. *Compléter la classe* `ListeRationnel` *pour qu'elle permette d'exécuter un programme qui affiche le plus grand élément d'un tableau de 5 rationnels.*