CNAM Bretagne **2005-2006**

TP Algorithmique programmation

TP 9 : Interface Java générique

Exercice 1 (Table de correspondance). On se propose de réaliser une structure de donnée "Table de correspondance" qui permet d'associer des valeurs à des clés. Les clés pourront par exemple être des chaînes de caractères et les valeurs des nombres ce qui permettra de gérer les résultats de sportifs qui cumulent des points lors d'épreuves sportives. : `<("Jyvé", 50), ("Michou", 750), ("Barrique", 120), ...>`. Dans une table de correspondance, chaque clé ne peut être présente qu'une fois.

La classe `TableDeCorrespondance` doit offrir les fonctionnalités suivantes :
- constructeur qui crée une table vide de taille maximum donnée.
- déterminer si une clé est présente dans la table.
- donner la valeur associée à une clé ; (précondition : la clé est présente dans la table).
- ajouter une association (clé, valeur) ; laisse la table inchangée si la clé est présente ou si la table est pleine.
- modifier la valeur associée à une clé ; laisse la table inchangée si la clé est absente.
- retirer une association dont on donne la clé ; laisse la table inchangée si la clé est absente.

On veut en plus disposer de méthodes qui permettent de parcourir séquentiellement les éléments de la table :
- méthode qui initialise un parcours
- méthode qui détermine si le parcours est terminé
- méthode qui donne la clé de l'association courante (précondition : le parcours n'est pas terminé)
- méthode qui donne la valeur de l'association courante (précondition : le parcours n'est pas terminé)
- méthode qui fait passer à l'association suivante (précondition : le parcours n'est pas terminé)

Cette spécification est disponible sous forme d'interface java (voir web des TP).

On décide de représenter une table de correspondance au moyen d'un tableau trié par ordre croissant des clés ; un élément de la table (une association (clé, valeur)) sera représenté par un Couple de T1, T2 (T1 est le type des clés, T2 celui des valeurs) ; la classe Couple possède les fonctionnalités vues en cours et sera implémentée dans son propre fichier.

Programmer l'implémentation de ces deux classes, en respectant le plan suivant :
- implémentation de la classe couple (dans son propre fichier)

- implémentation de la classe table de correspondance générique en respectant l'interface fournie :
  - déclarer les attributs
  - programmer le constructeur
  - programmer les méthodes de parcours
  - tester les méthodes de parcours : pour ce faire, écrire une version temporaire de la méthode d'ajout qui ajoute simplement en fin de table ; dans un autre fichier (programme de test), programmer une fonction (générique) d'affichage puis une procédure qui crée une table, y ajoute quelques éléments puis l'affiche.
  - programmer les autres méthodes, qui devront accéder aux éléments de la table à l'aide des méthodes de parcours exclusivement.

Pour gérer un tableau trié par ordre croissant des clés, il faut pouvoir comparer les clés. Or, les opérateurs classiques (`<, <=, ==, >, >=`) ne sont définis que pour les types numériques et ne peuvent en aucun cas s'appliquer à un type générique comme celui des clés (`Tcle`). Par ailleurs, dans la bibliothèque java existe une interface générique `Comparable<T>` qui définit une unique méthode : `int compareTo(T)` ; telle que, si x1 et x2 désignent des instance du type T :

- `x1.compareTo(x2) < 0 () valeur(x1) < valeur(x2)`
- `x1.compareTo(x2) = 0 () valeur(x1) = valeur(x2)`
- `x1.compareTo(x2) > 0 () valeur(x1) > valeur(x2)`

De plus, tous les types de java (`Integer`, `String`, etc...) implémentent cette interface, donc définissent la méthode `compareTo`. La solution consiste donc à exiger que le type des clés de notre table de correspondance implémente obligatoirement l'interface `Comparable<T>`, c'est-à-dire définisse la méthode `compareTo`. Ceci se réalise de la façon suivante :

```
public class TableDeCorrespondanceTabloTrie<TCle extends Comparable<TCle>, TVal>
  implements TableDeCorrespondance<TCle, TVal> {
```

Pour gérer une table dont les clés sont d'un type java (`Integer`, `String`, etc...), il n'y a rien d'autre à faire ; pour gérer une table dont les clés ne sont pas d'un type java prédéfini, il faut que ce type implémente l'interface `Comparable<T>` en plus de sa spécification propre (donc définisse la méthode `compareTo`).