TP no 3

Préambule

But : Le but du TP est de programmer :

- une implémentation du type abstrait générique *Table de correspondance* selon la spécification donnée en cours ;
- un programme client qui pourra servir par exemple à gérer les résultats de sportifs qui cumulent des points lors d'épreuves sportives.

Spécifications : L'explorateur de paquetages (Package Explorer) permet d'explorer les spécifications des types abstraits du cours (bibliothèque types) ainsi que celles des opérations de lecture et écriture (bibliothèque lectureEcriture) : il faut pour cela naviguer dans la « JRE » associée à votre projet ; vous pouvez ouvrir les fichiers sources des spécifications.

1 Implémentation

1.1 Remarques

1. Il est recommandé de procéder à un bon découpage fonctionnel ; en particulier, *évitez de programmer plusieurs fois la même fonctionnalité* : il vous est demandé de créer des fonctions ou méthodes privées (propres à l'implémentation) qu'on veillera à spécifier correctement.
2. Toute recherche d'une clé dans la table doit se faire par *dichotomie* ; on réfléchira avec soin aux pré et post-conditions de cette recherche, sachant qu'elle doit :
   (a) permettre de déterminer la présence ou l'absence d'une clé,
   (b) déterminer la position auquel insérer un nouvel élément, si besoin.
3. Pour protéger vos méthodes, vous vérifierez systématiquement leurs pré-conditions avec des *assertions* ; la syntaxe d'une assertion est :

```
assert expression_booléenne : "Message d'erreur";
```

   Lors de l'exécution de l'instruction, si l'expression booléenne est vraie, le programme se poursuit normalement ; si elle est fausse, le programme s'arrête et affiche le message d'erreur donné ainsi que la ligne où est située l'assertion qui n'est pas vérifiée.

1.2 Représentation

On décide de représenter la table de correspondance ainsi :

- chaque association (clé, valeur) de la table sera représentée par un *couple générique* (classe Couple, de la bibliothèque types) ;
- les couples seront placés dans un tableau (*obligatoirement* classe Array, de la bibliothèque types), *trié par ordre croissant des clés.*

1.3 Plan

Programmez cette implémentation de la table de correspondance, en respectant l'interface fournie et en suivant le plan ci-dessous.

1. Créez un paquetage de nom tdc dans lequel vous placerez l'implémentation ;
2. Créez la classe de votre implémentation ; dans la fenêtre de création « Nouvelle Classe Java » :
   - rubrique « Nom » : indiquez le nom de votre implémentation sans oublier les types génériques ;
   - rubrique « Interfaces » : cliquez « Ajouter », indiquez le nom de l'interface implémentée, validez puis complétez le nom de l'interface avec celui des types génériques.
3. Déclarez les attributs ;
4. Programmez au moins un constructeur ;
5. Programmez les méthodes de parcours ;
6. *Testez les méthodes de parcours* : pour ce faire, écrivez une version *temporaire* de la méthode d'ajout qui ajoute le nouvel élément *en fin de table*.
   Le programme client de test sera placé dans le *paquetage par défaut* ; vous y programmerez une procédure (générique) d'affichage puis une procédure qui crée une table, y ajoute quelques éléments puis l'affiche ;
7. Programmez les autres méthodes ;
8. Testez votre implémentation en écrivant le programme décrit en section 2.

1.4 Comparaison des clés

Pour gérer un tableau trié par ordre croissant des clés, il faut pouvoir comparer les clés. Or, les opérateurs classiques ($<$, $\leq$, $==$, $>$, $\geq$) 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 x); telle que, si x1 et x2 désignent des instances du type T :

- $x1.compareTo(x2) < 0 \iff valeur(x1) < valeur(x2)$
- $x1.compareTo(x2) = 0 \iff valeur(x1) = valeur(x2)$
- $x1.compareTo(x2) > 0 \iff valeur(x1) > valeur(x2)$

De plus, tous les types non scalaires de java (Integer, String, etc...) implémentent cette interface, donc définissent la méthode **compareTo.**

Une solution consiste à exiger que le type des clés de la table de correspondance implémente l'interface **Comparable<T>**, c'est-à-dire définisse la méthode **compareTo**.

Ceci se réalise de la façon suivante :

```
/**
 * implémentation du TA Table de correspondance avec un tableau trié.
 * Tcle = type des clés et TVal = type des valeurs associées.
 * TCle doit définir les opérations de l'interface Comparable<T>
 */
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**).

2 Programme client

2.1 cas où la clé est d'un type Java

Complétez le programme de test précédent avec une procédure qui gère une table dont les clés sont des *chaînes de caractères* et les valeurs des *entiers* ; cette procédure devra lire des couples (clé, valeur) et les placer dans la table de la façon suivante :

- si la clé est *absente* dans la table, l'y *ajouter* avec la valeur associée.
- si la clé est *présente* et la valeur *positive*, *l'additionner* à la valeur actuellement associée à la clé dans la table.
- si la clé est *présente* et la valeur *négative*, *retirer* l'association de la table.

Affichez la table après chaque opération.

Pour tester le programme vous pourrez utiliser le fichier /share/diic/AP/ap1_tp3_tdc/donnees1 et comparer vos résultats avec ceux du fichier /share/diic/AP/ap1_tp3_tdc/resultats.

2.2 cas où la clé n'est pas d'un type Java

Complétez le programme précédent pour créer une table dont les clés sont des *rationnels* et les valeurs des chaînes ; on se contentera d'ajouter des éléments dans la table et d'afficher son contenu ; pour tester votre programme vous pourrez procéder comme ci-dessus, avec le fichier /share/diic/AP/ap1_tp3_tdc/donnees2.