Lic.Info 2ème année Université d'Orléans

SOL3IF02 – POO

TP n°4 – Interfaces et classes internes

Exercice 1. On souhaite manipuler des listes de n-uplets de manière à pouvoir trier les n-uplets contenus dans la liste.

Créer une classe `ListeNuplets` caractérisée par un attribut `lesNuplets` de type un tableau de `Nuplet`.

On devra donc créer une classe `Nuplet` pour manipuler les n-uplets du tableau.

Cette classe pourra être interne à la classe `ListeNuplets`. On la déclarera statique et dans un premier temps, on lui attribuera une visibilité privée.

1) Construction de la classe `Nuplet` :

1. On définit un n-uplet $(u_1, u_2, \ldots, u_k)$ de taille quelconque `k` dans la classe `Nuplet` par un attribut privé `private int[] content` contenant les valeurs des u_i.
2. Définir un constructeur `public Nuplet (int k)` qui construit un n-uplet de taille k. Les valeurs u_i doivent être saisies par lecture interactive et doivent être positives. Gérer la possibilité de saisir des valeurs non entières ou négatives.
3. Redéfinir la méthode `public String toString()` qui retourne l'état de l'objet sous la forme : "$(u_1, u_2, \ldots, u_k)$".
4. Définir deux autres méthodes :
   - `public int nbElements()` qui retourne le nombre d'éléments du n-uplet.
   - `public int getElement(int index)` qui retourne l'élément du n-uplet correspondant à l'indice passé en paramètre si l'indice est valide et -1 sinon.

2) Construction de la classe `ListeNuplets` :

1. Définir le constructeur `public ListeNuplets(int...lesTailles)` permettant de passer en paramètre une suite variable de valeurs entières correspondant à la taille de chacun des n-uplets de la liste que l'on veut créer.

   Exemple : `ListeNuplets l = new ListeNuplets(3,3,4,2);`

   permet de créer une liste de 4 n-uplets, le premier et le second de taille 3, le troisième de taille 4 et le dernier de taille 2.
2. Redéfinir la méthode `toString()`.
3. Définir les méthodes suivantes :
   - `public Nuplet getNuplet (int index)` qui retourne le `Nuplet` présent dans la liste à l'indice indiqué en paramètre si l'indice est valide et `null` sinon.

- `public void trier()` qui trie la liste des n-uplets.

  Quelle classe de l'API Java devons-nous utiliser pour réaliser cela ?

  Comment doit-on modifier la classe `Nuplet` pour que cela fonctionne ? Implémentez-le sachant que l'ordre de comparaison de deux n-uplets se base sur la définition suivante[1] :

  Soient a = $(a_1, a_2, \ldots, a_i)$ et b = $(b_1, b_2, \ldots, b_j)$ deux n-uplets.

  a < b si et seulement si

  i. $i < j$ et $\forall k \leqslant i, a_k = b_k$ ou

  ii. jusqu'à l'indice $k < i$ et $k < j, a_k = b_k$ et $a_{k+1} < b_{k+1}$.

  Exemples :

  (4, 3) < (4, 3, 5, 7) par i.

  (4, 3, 2) < (4, 3, 5, 7) par ii.

  (4, 1, 2) < (4, 3) par ii.

3) Créer une classe `TestTriNuplets` contenant la méthode `main` et permettant de créer une liste de n-uplets puis de la trier. Afficher l'état de la liste avant et après le tri.

Récupérer le `Nuplet` de la liste correspondant à l'indice de votre choix et afficher son nombre d'éléments et tous ses éléments. La visibilité privée de la classe `Nuplet` permet-elle de le faire ? Modifier cette visibilité si nécessaire.

Exercice 2. Consulter la Javadoc de l'interface `Predicate<T>` dans l'API Java.

Créer une classe `Main` contenant le `main`.

Définir le tableau d'entiers `int tab = {1 , 13 , 4 , 15 , 32 , 25};`

Définir une variable de type `Predicate<Integer>` nommée `PlusGrandQue10` qui permet de tester si un entier est strictement supérieur à 10.

Définir une variable de type `Predicate<Integer>` nommée `PlusPetitQue20` qui permet tester si un entier est inférieur ou égal à 20.

1. Parcourir le tableau `tab` de manière à afficher les entiers strictement supérieurs à 10.
2. En utilisant une des méthodes par défaut de l'interface `Predicate`, parcourir le tableau `tab` de manière à afficher les entiers strictement supérieurs à 10 **et** inférieurs ou égaux à 20.

Exercice 3. Reprendre l'exercice 3 du TD4.

1 - Utiliser la classe `EnsTransformable` pour transformer un tableau de chaînes de caractères en majuscule ou en minuscule.

Construire une classe `MajusculeMinuscule` contenant la méthode `main`. Créer un ensemble transformable à partir du tableau de `String` suivant : `String[] lesMots = {"Bonjour", "HIER","Aujourd'hui"}.`

1 Vous remarquerez que cet ordre est l'ordre lexicographique utilisé pour comparer deux chaînes de caractères.

Interagir avec l'utilisateur de manière à lui proposer de choisir le type de transformation qu'il souhaite appliquer à l'ensemble : **`a`** pour majuscule et **`i`** pour minuscule. Appliquer la transformation demandée à l'ensemble et afficher le résultat.

La classe `String` fournit la méthode `public String toUpperCase()` qui convertit tous les caractères de la chaîne en majuscule et la méthode `public String toLowerCase()` qui convertit tous les caractères de la chaîne en minuscule.

2 - Construire la classe `TransformationType` contenant la méthode `main`. Définir un tableau de type `Double`. En utilisant la classe `EnsTransformable`, transformer les éléments du tableau en changeant leur type. On pourra transformer les éléments sous la forme d'une chaîne de caractères.
Exécuter le programme. Est ce que cela fonctionne ? Modifier le code si besoin pour que cela fonctionne.