L2 année 2018-2019

TD 2 – Méthodes

Partie 1 : bibliothèque de méthodes

Exercice 1 :

Nous voulons mettre en place un ensemble de méthodes qui implémentent des utilitaires de manipulation de tableaux d'une ou plusieurs dimensions de réelle et de chaîne de caractères.
Voici le descriptif (spécification) des méthodes qu'on veut déclarer.

1. La méthode **créerTabString** permet de saisir le nombre d'éléments du tableau, ensuite elle permet de saisir son contenu pour enfin renvoyer un tableau de **String** rempli.

2. La méthode **créerTabDouble** permet de faire la même chose que la méthode précédente mais pour le type double.

3. Deux méthodes **afficher** qui permettent respectivement d'afficher un tableau de string et un tableau de double au format suivant :
[e1,…..,en]

4. Deux méthodes **copierZone** qui prennent en entrée deux tableaux (disons tab1 et tab2) et trois entiers (i1deb, i1fin, i2deb) et qui permettent de copier le sous tableau du tab1 compris entre i1deb et i1fin dans le sous tableau tab2 à partir de i2deb. La première méthode concerne un tableau de String et la deuxième un tableau de double.

5. Deux méthodes **agrandir** qui permettent d'agrandir respectivement un tableau de String et un tableau de double par un entier spécifié en paramètre.

6. Deux méthodes **fusion** qui permettent de fusionner deux tableaux de String et deux tableaux de double.

7. Deux méthodes **insérer** qui étant donné un tableau de String (resp. de double), un entier représentant un indice et un String (resp. un double), permettent d'insérer le String (resp. un double) à l'indice précisé.

8. Quatre méthodes **supprimer** les deux premières permettent respectivement de supprimer, en réduisant la taille du tableau, un élément d'un tableau de String (resp. double) dont l'indice est donné en paramètre. Les deux autres méthodes font la même chose mais en prenant en paramètre, non pas l'indice, mais l'élément à supprimer c.à.d un String pour l'une et un double pour l'autre.

Question 1> Commencez d'abord par déduire à partir des descriptifs les signatures. En listant les différentes contraintes sur les valeurs des paramètres et en précisant le comportement de votre méthode en cas de violation.

Question 2> Sous ecplise créez un projet et ajouter un programme appelé **UtilTab.java**, Implémentez **l**es méthodes de la question 1. Exceptionnellement le programme **UtilTab** ne contiendra pas de méthode **main**.
NB : Certaines des méthodes peuvent être réalisé en utilisant d'autres, maximisez la réutilisation.

Partie 2 : Réutilisation

Considérons un projet java qui contient deux programmes **P1.java** et **P2.java**. Supposons que P1 contient une méthode qui porte le nom **m1** et déclaré avec le mot clé **static**. Dans le programme **P2.java** on peut écrire des expressions d'appels à la méthode **m1** simplement en précédant son nom par **P1.** Ce qui donne **P1.m1**.

Exercice 2 :

Toujours dans le même projet créé pour la question 1, ajoutez, un programme appelé **TriTab.java**.
Ce programme a pour objectif de saisir un tableau de double, ensuite le programme propose le menu suivant :
Choisissez votre algorithme de tri :
1 **>** bulle
2> fusion
Selon le choix le programme effectue un tri du tableau en appliquant l'algorithme de tri correspondant (voir annexe pour un rappelle sur les principes des algorithmes de tri) et affiche le résultat.

Exercice 3.

Écrivez un troisième programme **Etudiants.java** qui permet de saisir une liste d'étudiants et une liste de notes. Ensuite il affiche les étudiants classés de la meilleure note à la plus mauvaise note.

Annexe

Tri à bulle (source wikipédia)

L'algorithme parcourt le tableau et compare les éléments adjacents. Lorsque les éléments ne sont pas dans l'ordre, ils sont échangés
Après un premier parcours complet du tableau, le plus grand élément est forcément en fin de tableau, à sa position définitive. En effet, aussitôt que le plus grand élément est rencontré durant le parcours, il est mal trié par rapport à tous les éléments suivants, donc échangé à chaque fois jusqu'à la fin du parcours.
Après le premier parcours, le plus grand élément étant à sa position définitive, il n'a plus à être traité. Le reste du tableau est en revanche encore en désordre. Il faut donc le parcourir à nouveau, en s'arrêtant à l'avant-dernier élément. Après ce deuxième parcours, les deux plus grands éléments sont à leur position définitive. Il faut donc répéter les parcours du tableau autant de fois que d'éléments dans celui-ci en diminuant à chaque itération l'indice pour l'arrêt des comparaisons

Tri par fusion

À partir de deux tableaux triés, on peut facilement construire un tableau trié comportant les éléments issus de ces deux tableaux (leur *fusion*). Le principe de l'algorithme de tri fusion repose sur cette observation : le plus petit élément de la liste à construire est soit le plus petit élément de la première liste, soit le plus petit élément de la deuxième liste. Ainsi, on peut construire la liste élément par élément en retirant tantôt le premier élément de la première liste, tantôt le premier élément de la deuxième liste (en fait, le plus petit des deux, à supposer qu'aucune des deux listes ne soit vide, sinon la réponse est immédiate).

L'algorithme est naturellement décrit de façon récursive.

1. Si le tableau n'a qu'un élément, il est déjà trié.
2. Sinon, on sépare le tableau en deux parties à peu près égales.
3. On trie récursivement les deux parties avec l'algorithme du tri fusion.
4. On fusionne les deux tableaux triés en un tableau trié.