Université Grenoble Alpes
Licence 2 MIASHS
Inff3 – Algorithmique et programmation par objets
Jérôme David
2023-2024

TD3 : Tableaux à une dimension

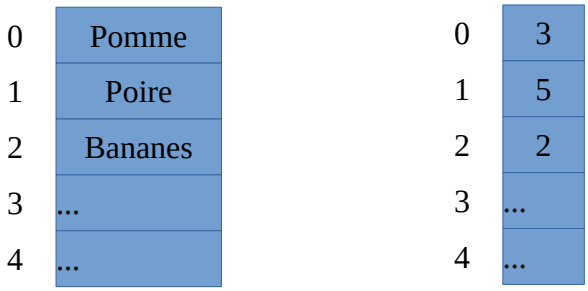
En programmation, un dictionnaire ou tableau associatif, est une structure de données permettant de stocker un ensemble de couples (clé ; valeur) de telle sorte qu'une clé n'apparaît au plus qu'une fois dans la structure (i.e. chaque objet « clé » est unique).
Les opérations que l'on peut classiquement faire sur un objet de type tableau associatif sont :

- l'ajout de couple (clé ;valeur) dans l'ensemble / modification de la valeur associée à une clé.
- la suppression d'un couple (clé ; valeur). Pour cela, seulement la clé est donnée en paramètre.
- la recherche de la valeur associée à une clé donnée.

Des exemples de tableaux associatifs sont les annuaires, ou encore les dictionnaires.

Dans ce TD nous nous intéresserons à créer une classe pour représenter des tableaux associatifs qui associent des clés de type `String` à des valeurs de type `int`. Nous nous servirons du travail fait durant ce TD pour réaliser en TP un compteur d'occurrences de mots dans un texte.

Il existe de nombreuses façons de programmer une telle classe. Nous choisirons dans ce TD de représenter notre tableau associatif avec deux tableaux : un tableau de `String` pour les clés, et un tableau de `int` pour les valeurs. La valeur d'une clé se trouvant à l'indice i du tableau des clés sera rangée à l'indice i du tableau des valeurs.

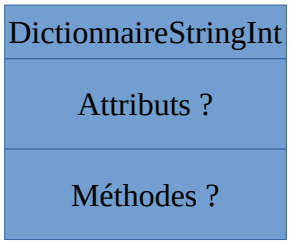


Dans cet exemple, Pomme est associé à 3, Poire à 5, etc.

Les tableaux ont une taille fixée à leur création. Une fois créé, un tableau ne peut pas être agrandi. Afin que nos tableaux aient une taille dynamique, une stratégie consiste à créer au départ des tableaux d'une taille fixée, par exemple à 10 éléments, et de stocker dans une variable le nombre d'éléments réellement présent dans les tableaux. A la création, cette variable sera initialisée à 0 et ensuite cette valeur sera incrémentée lorsque l'on ajoute un couple (clé ; valeur) et décrémentée lorsque l'on en supprime un.

Dans ce scénario, notre tableau associatif aura une capacité maximale de 10 couples (clé ; valeur). Lorsque l'on n'a plus de place, il suffit de créer deux nouveaux tableaux plus grands (avec, par exemple, une taille augmentée de 10 éléments), de recopier les éléments des anciens tableaux vers les nouveaux, puis de mettre à jour les variable (elles doivent référencer les nouveau tableaux). De manière inverse, lorsque l'on a supprimé beaucoup d'éléments, on pourrait réduire la taille des tableaux (non fait dans ce TD).

Question 1 : Identifier les attributs (et leur type) et les méthodes (type de retour et type des paramètres) de la classe `DictionnaireStringInt`.



Question 2 : Ecrire le squelette (i.e. la définition de la classe avec la déclaration et initialisation des attributs, et les définitions des méthodes mais sans le corps des méthodes) de la classe `DictionnaireStringInt`.

1 - Version simple

Question 3 :Ecrire la méthode `rechercherIdx` permettant de rechercher et retourner l'indice d'une clé passée en paramètre. Si la clé n'est pas présente la méthode retournera la valeur `-1-idx` (où `idx` est l'indice de la case où aurait du se trouver la clé).
Rappel : l'égalité entre deux objets est réalisée en utilisant la méthode « `equals` ».

Question 4 : En se servant de la méthode définie à la question précédente, écrire la méthode `rechercherValeur` retournant la valeur associée à une clé. Si la clé n'est pas présente, on retournera 0.

Question 5 : Ecrire la méthode `agrandir` qui permet de remplacer les tableaux si ils sont pleins. Les nouveaux tableaux auront une taille supérieure de 10 éléments (par rapport aux anciens tableaux).

Question 6 : Ecrire la méthode `ajouterModifier` qui permet d'ajouter une paire (clé ; valeur) si la clé n'est pas présente ou de remplacer valeur associée si la clé est déjà présente. Avant l'ajout, cette méthode appellera la méthode définie à la question précédente pour s'assurer qu'il y a la place suffisante dans le tableau.

Question 7 : Ecrire la méthode `supprimer` qui supprime couple (clé ; valeur). Pour cela, on remplacera la clé, resp. la valeur, à supprimer par le dernier élément du tableau.

Question 8 : Ecrire la méthode `String toString()` permettant de retourner une représentation du tableau associatif sous forme de chaîne de caractères. L'exemple du début sera représenté par la chaîne « `{ [Pomme : 3] , [Poire : 5] , [Banane : 2] }` »

2 - Version triée

Dans cette cette seconde partie, nous nous intéresserons à développer une version du tableau associatif où les clés sont triées (i.e. rangées en ordre alphabétique).
Pour comparer deux chaînes de caractères en Java, on utilise la méthode `compareTo`. Par

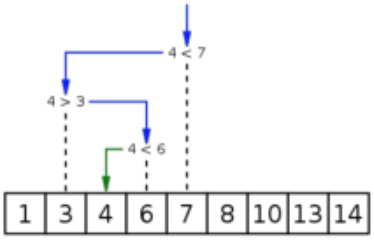
exemple, soit le code suivant :

```
String s1 = "abc";
String s2 = "bcd";
String s3 = "abc";

int comp1 = s1.compareTo(s2);
int comp2 = s2.compareTo(s1);
int comp3 = s1.compareTo(s3);
```

`comp1` sera un nombre négatif car `s1` est « plus petit » que `s2`, `comp2` sera un entier positif car `s2` est plus grand que `s1` et `comp3` sera égal à 0 car `s1` est égal à `s2`.

Question 9 : Proposer une nouvelle méthode de recherche (remplaçant celle de la Question 3) qui réalise une recherche dichotomique. La recherche dichotomique est un moyen efficace de recherche dans un tableau trié. Le principe consiste à comparer l'élément à recherche avec l'élément de la case du milieu du tableau. Si les éléments sont égaux alors c'est gagné sinon, on recommence la même chose dans la moitié du tableau pertinente (celle d'après si l'élément du milieu est plus petit que l'élément recherché, celle d'avant sinon).
Un exemple dans le cas ou recherche la valeur 4 dans un tableau d'entiers.



Question 10 : Proposer une nouvelle méthode d'ajout (remplaçant celle de la Question 6) qui insère un couple (clé ;valeur) à la bonne place selon l'ordre alphabétique des clés. Modifier également, la méthode de suppression, pour qu'elle préserve l'ordre dans le dictionnaire.