Utilisation et implantation des dictionnaires associatifs avec hachage

Travaux dirigés et pratiques - (FMIN 220)

1 Préliminaires

Un dictionnaire est un ensemble d'associations, c'est-à-dire de couples (clé, valeur), où la clé est d'un type donné $TypeCle$ et la valeur d'un type donné $TypeValeur$.
Une association est typiquement représentée par une classe `Assoc` à deux attributs.

Dans leur version Java, les dictionnaires sont spécifiés par l'interface `Map`, de l'anglais (*to map*, qui représentent les objets qui font correspondre des clés à des valeurs, à une clé correspondant une unique valeur. Dans ce TD, nous nous intéressons aux dictionnaires implantés avec une fonction de hachage (voir plus loin). Cette implantation garantit un temps d'accès constant (indépendant de la taille du dictionnaire) à une association. Nous utiliserons à partir d'ici le terme dictionnaire pour parler de "dictionnaire haché".

2 Modélisation

Un dictionnaire, paramétré par les types K et V possède entre autres les méthodes suivantes (inspirée par l'interface `Map` de *Java)* :

- `V put(K key, V value)` : place une clé et une valeur associée dans le dictionnaire ;
- `get(Object key)` : prend une clé et renvoie la valeur associée ;
- `boolean isEmpty()` : dit si le dictionnaire est vide ;
- `int size()` : retourne le nombre d'associations clé-valeur effectivement présentes dans le dictionnaire ;
- `boolean containsKey(Object key)` : dit si une clé est présente dans le dictionnaire (pourquoi le type n'est-il pas K ?) ;
- `String toString()` : fabrique une chaîne représentant le receveur.

Écrivez la partie "signatures des méthodes" dans une interface.

3 Implémentation par un tableau dynamique d'associations

Le tableau d'associations est créé avec une taille de 10, et augmente de 5 en 5 en cas de débordement, c'est en ce sens qu'on dit que c'est un tableau dynamique. L'indice d'une association est calculé grâce à une fonction de hachage appliquée à la clé (une méthode `hashcode()` est définie sur la classe Object. Voir la section "Annexe" pour un algorithme de calcul de la position d'une association dans un dictionnaire haché. Quand il faut agrandir le tableau, si l'algorithme dépend de la taille du tableau, il faut bien sûr redistribuer les associations existantes dans le nouveau tableau.

En utilisant les exemples de résultats de la fonction de hachage, faites quelques essais d'insertions "à la main" pour voir comment les choses se passent : par exemple `put("abricot",235); put("amande",1023); put("ananas",242); put ("pomme",83); ....`

Ecrivez la classe `Dico` implantant l'interface.

4 Utilisation

Instanciez vos classes pour avoir un dictionnaire dont les clés sont des chaînes et les valeurs des entiers.

Écrivez un programme simple qui teste le fonctionnement de ce dictionnaire (ajoutez des couples, affichez le dictionnaire, rajoutez des couples pour tester l'agrandissement, affichez, etc...).

FAites un programme utilisant un dictionnaire pour compter le nombre d'occurences des mots d'une chaîne de caractères.

5 Annexe

Gestion des conflits de hachage. Lorsque la fonction de hachage produit une même valeur pour deux clés différentes, on est dans une situation de collision. La méthode de résolution des collisions la plus simple à implémenter est la méthode "recherche séquentielle simple" : on recherche, à partir de la case qui aurait du être la bonne, la première case libre (si on arrive en fin de tableau, on recommence au début), et on place l'association dedans. Cette méthode n'est pas excellente (elle crée des "grappes"), mais elle suffit pour tester cet exercice.

Agrandissement du tableau. Une bonne gestion d'un tel tableau consiste à l'agrandir dès qu'il est plein à 75%, de façon à éviter au maximum les collisions et à garantir qu'il y a toujours une place libre pour la recherche séquentielle locale réalisée en cas de conflit de hachage.