2. Feuille de TP

semaine 15 :

Préliminaires

L'objectif de ce TP est d'améliorer la classe **MapIUT** vue au TD précédent pour qu'elle soit plus efficace algorithmiquement. On va en fait s'approcher de ce que fait la classe *HashMap* de la bibliothèque standard.

Dans l'implémentation du TD précédent, les couples étaient stockés dans l'ordre d'insertion, de sorte que lorsqu'on cherchait une clef, il fallait parcourir tous les couples jusqu'à trouver celui qui correspondait à cette dernière.

- Pour l'implémentation du TD précédent, quelle est la complexité de la fonction **get** ? C'est cette complexité qui ne nous satisfait pas ...

1. **Hachage** : pour la solution algorithmique que nous allons voir, nous allons utiliser la méthode *hashCode()*, existant pour tout objet. Cette méthode est une fonction de hachage : elle calcule une *empreinte* de l'objet sous la forme d'un entier *positif ou négatif*.

Par exemple, sur la String bonsoir = "bonsoir"

```
bonsoir.hashCode() vaut 64006598.
```

2. **Le rapport avec HashMap** : chaque couple (clef, valeur) est stocké dans un tableau de taille donnée, l'indice de placement du couple étant donné par le résultat de la fonction de hachage modulo la taille du tableau pour ne pas déborder.

- Pour un tableau de taille 10, dans quelle case serait stocké le mot "bonsoir" ?
- Quelle complexité espère-t-on obtenir pour la méthode get ?

Il se peut que deux clés différentes aient des indices de placement identiques, on appelle ce phénomène une **collision**. Il va de soi que dans les HashMap les problèmes de collisions sont parfaitement gérés.

Version 0

Dans cette première version d'implémentation, on se contentera d'un tableau de couples de taille fixe (5 ou 10 pour faire des tests). La gestion des collisions n'est absolument pas traitée. Pour cela, vous simulerez le tableau en utilisant une liste de 5 ou 10 Couples de valeur null.

- Reprenez la classe *Couple* du TD.
- Définissez la classe **HashMapVersion0** dont l'attribut est une liste de couples et une taille fixe. Vous définirez les méthodes *put(clef, valeur)*, *get(clef)* et *containsKey(clef)*.
- Définissez une classe **Executable** reprenant les couples donnés en exemple, ci-dessous.

Version 1

Dans cette seconde version, on va tenter de minimiser les collisions (que nous ne traitons toujours pas). Pour cela, on garde en mémoire le **nombre** de couples ayant été ajoutés à la map (qu'il y ait ou non collision). Lorsque ce nombre est plus grand que la moitié de la taille du tableau, on multiplie sa taille par deux. Dans ce cas, vous ne **recopiez pas** les couples dans ce tableau agrandi aux mêmes indices que dans le premier tableau, mais en recalculant les indices grâce à la fonction de hachage.

- Définissez la classe **HashMapVersion1** qui implémente l'agrandissement de tableau.
- Enrichissez votre précédent exécutable et testez. Que notez vous ? qu'avez vous essayé ? (modification de taille ...).

Version 2

Il est temps maintenant de traiter les collisions. Pour chaque entrée du tableau, il faut conserver tous les couples en collision. À chaque entrée ne correspond plus un couple mais un tableau (liste) de couples (celui-ci peut ne contenir qu'un couple s'il n'y a pas de collision). Ce tableau de collisions a également une taille fixe. On ne traitera pas ici le fait de diminuer le nombre de collisions en agrandissant le tableau (version 1), ce sera une amélioration de la version 2). En revanche, lorsqu'une clef est déjà présente (avec une autre valeur) on la remplacera (i.e. au même endroit) par la nouvelle version du couple.

- On suppose l'existence des couples (clef, valeur) suivants avec leur valeur de hachage :

```
("bonjour", 6) : 1245
("bonjour", 33) : 1245
("demain", 1) : 3421
("lundi", 7) : 4533
("travail", 8) : 8766
("vacances", 22) : 5371
("IUT", 1) : 4212
("soleil", 80) : 6543
("ciel", 4) : 56324
("nuages", 5) : 6754
("bleu", 11) : 345620
("weekend", 6) : 45335
("semaine", 23) : 44365
```

- Donnez une représentation graphique de la hashMap selon la version 2, avec un tableau de taille 10. Le tableau des collisions sera fixé à 3. Combien y a-t-il de collisions ? Sont-elles toutes gérées ?
- Implémentez la map version 2, selon le principe ci-dessus, en ne définissant que les méthodes **put()** et **toString()**. Pensez à tous les cas pour la méthode put, vous pouvez (recommandé) définir des méthodes annexes (privées). Vous implémenterez chez vous les méthodes *get()* et *containsKey()*.
- Modifiez votre exécutable afin de tester cette nouvelle version. Faites varier les différentes tailles des tableaux. Que constatez vous ?