Algorithmique
TP 5 : Fonctions de hachage

Le premier exercice de ce TP doit être déposé sur l'espace "Rendu TP5".

1 Fonctions de hachage

On définit un numéro d'étudiant par une séquence de chiffres formée comme suit[1] :

$$s\ aa\ mm\ \ xxx$$

où s est un code d'identification (on suppose $1 \leq s \leq 3$), aa dénote les dernières deux chiffres de l'année de la première inscription, mm dénote le mois de naissance de l'étudiant ($1 =$ janvier, $\ldots$, $12 =$ décembre) et xxx est un nombre aléatoire tel que $1 \leq xxx \leq 999$. Dans le fichier `NoEtud.java` complétez l'implémentation de la classe `NoEtud`, qui encode un numéro d'étudiant. Cette classe possède les quatre attributs privés (entiers) `s`, `aa`, `mm` et `xxx`. On fournit également les accesseurs pour ces attributs.

1. Définir constructeur `NoEtud(int s, int aa, int mm, int xxx)` qui construit le numéro d'étudiant dont les composants sont donnés en paramètre. Si les paramètres actuels ne satisfont pas les contraintes ci-dessus, le numéro d'étudiant créé sera 2 15 01 000.

2. Définir le constructeur `NoEtud(int n)` qui construit le numéro d'étudiant à partir de l'entier n (compris entre 10 000 000 et 40 000 000). Si le paramètre n ne satisfait pas cette contrainte et celles de validité des nombres, le numéro d'étudiant créé sera 1 00 01 000.

3. Coder la méthode `public String toString()` qui renvoie la chaîne de caractères correspondant au numéro d'étudiant.

4. Coder une méthode `public int toInteger()` qui renvoie l'entier qui représente le numéro d'étudiant.

La classe `NoEtud` hérite de la classe `Objet` une méthode `public int hashCode()`. Dans les exercices suivants, on va coder d'autres fonctions de hachage pour les comparer entre elles.

7. Coder une méthode `public int hashCodeInt()` qui renvoie comme code de hachage l'entier qui représente le numéro d'étudiant.

8. Coder la méthode `public int hashCodeUniform(int m)` qui renvoie comme code de hachage $k \times m$ où k est l'entier qui représente le numéro d'étudiant et m est un entier positif[2].

[1] Combien de numéros d'étudiant distincts peuvent être créés en utilisant ce format ?

[2] Il sera fixé par la taille de la table de hachage dont les éléments sont des numéros d'étudiant.

9. Coder la méthode `public int hashCodeMod(int m)` qui renvoie comme code de hachage $k\%m$ avec k et m comme ci-dessus.

10. Coder la méthode `public int hashCodeGold(int m)` qui renvoie comme code de hachage $\lfloor m \times frac(k \times A) \rfloor$ avec $A = (\sqrt{5}+1)/2$, $frac(f)$ la partie fractionnaire de f, et k et m comme ci-dessus.

11. Redéfinir la méthode `public int hashCode()` de la classe `Object` pour qu'elle soit une des méthodes ci-dessus ; on utilisera $m = 100$.

2 Utilisation de `HashSet`

Cet exercice a comme but de vous familiariser avec la classe `HashSet`[3] qui utilise une table de hachage pour implémenter un ensemble.
Dans le fichier `InscritsUE.java` complétez l'implémentation de la classe `InscritsUE` qui encode l'ensemble des inscrits à une unité d'enseignement (UE). Cet ensemble sera implémenté en utilisant la classe `HashSet`.

1. Définir l'attribut privé `inscrits` de type `HashSet<NoEtud>`.

2. Définir le constructeur par défaut `InscritsUE()` qui construit un ensemble vide d'inscrits dont la capacité initiale est de 100 inscrits. Comme décrit dans la documentation de la classe `HashSet`, le taux de remplissage par défaut est de 0,75.

3. Coder la méthode `public boolean add(NoEtud n)` qui ajoute, s'il n'y est pas encore, l'étudiant dont le numéro est n.

4. Coder la méthode `toString` qui renvoie la chaîne de caractères avec la liste des inscrits, un numéro par ligne.

5. Tester dans la méthode `main` que l'ajout de plusieurs numéros à l'ensemble vide se passe correctement. Changez la fonction de hachage dans la classe `NoEtud` et réaffichez l'ensemble. Obtenez-vous le même affichage pour des fonctions de hachage différentes ?

6. Coder la méthode `public void loadFromFile(String fname)` qui ajoute à l'ensemble `inscrits` les numéros d'étudiant stockés dans le fichier `fname` (contenant un numéro d'étudiant par ligne).

7. Dans le `main`, appelez la méthode `loadFromFile` avec le fichier `tp05.dat` qui contient 105 numéros d'étudiant. Affichez l'ensemble ainsi obtenu. Changez la fonction de hachage dans la classe `NoEtud` et réaffichez l'ensemble. Obtenez-vous le même affichage pour des fonctions de hachage différentes ?

[3] Cette API est disponible à l'adresse `https://docs.oracle.com/javase/8/docs/api/java/util/HashSet.html`.