BPO
TP Calculette

Exercice 1 - artEoz et les tables de hachage

1. Demandez l'exécution du logiciel **artEoz** sur le code :

```
HashMap<String, Point> table = new HashMap<>() ;
table.put("un", new Point(1,1)) ;
```

2. Sur le schéma, retrouvez la clé et l'objet stocké dans la table. Où sont-ils? Pourquoi sont-ils placés ainsi?

3. Modifiez l'instruction d'instanciation de la table pour ajouter un argument au constructeur de la **HashMap** :

```
HashMap<String, Point> table = new HashMap<>(3) ;
```

   Comment est modifiée votre table? Où se trouvent à présent la clé et l'objet stockés? Concluez.

4. Dans la table, ajoutez un **Point** avec une clé différente du premier **Point** de la table. Regardez l'évolution de la table en mode pas à pas. Que se passe-t-il? Expliquez.

5. Dans la table, ajoutez un **Point** avec une clé identique à celle du premier **Point** de la table. Regardez l'évolution de la table en mode pas à pas et visualisez les objets morts. Que se passe-t-il ?. Expliquez.

6. La table a une capacité de 4 entrées. Combien faut-il ajouter de points pour que la structure de la table soit complètement réorganisée?

7. Écrivez un appel à la fonction de recherche **get** (voir la doc de **HashMap**).

8. Enlevez un par un, tous les points de la table (fonction **remove**). Est-ce que cette table est réorganisée?

9. Regardez la différence entre l'appel d'un constructeur de copie et l'appel de la fonction **clone()** :

```
HashMap<String, Point> table1 = new HashMap<>(5) ;
table1.put("un", new Point(1,1)) ;
table1.put("deux", new Point(2,3)) ;
HashMap<String, Point> table2 = new HashMap(table1) ;
HashMap<String, Point> table3 = (HashMap<String, Point>) table1.clone() ;
```

10. Enlevez le point de **table1**. Est-ce que les autres tables sont affectées?

11. La table ci-dessous possède 2 éléments. Que pouvez-vous conclure sur les clés ? Pouvez-vous ajouter un troisième élément avec cette même caractéristique ?

```
HashMap <String, Point> hm = new HashMap<>(3) ;
hm.put("FB", new Point(1,1)) ;
hm.put("Ea", new Point(2,2)) ;
```

Exercice 2 - Programmation et test de l'application Calculette

- Écrivez les classes de la hiérarchie d'**Expression** vue en cours (**calc.Expression**, **calc.Binaire**, *etc.*). N'oubliez pas les **assert** si nécessaire pour tester les paramètres. Dans un premier temps, vous pouvez vous contenter de l'opérateur **Somme** ; vous écrivez les autres quand celui-çi sera écrit et testé.

- Écrivez une classe de test qui teste l'opérateur **Somme**, sur la base d'**assert**. Une fois le test complet, écrivez et testez les classes pour les autres opérateurs.

- Complétez le diagramme de classes pour pouvoir intégrer des opérateurs n-aires (moyenne, min, max, etc.). Écrivez (et testez) une séquence d'instructions qui crée une expression n-aire, comme :

  min (moyenne (1 , - 2 , 3) , max (6 , 9))

  et affiche sa valeur sur la sortie standard.

- Pour tester votre application **calculette** avec **artEoz** :

  - Il faut construire une archive java avec votre application. Placez vous au-dessus du répertoire **calc** et construisez l'archive avec la commande **jar** :

    jar cvf calc.jar calc

    … puis chargez votre propre archive avec le bouton