Premières classes

Alice Jacquot (d'après Guillaume Wisniewski)
jacquot@lri.fr

octobre 2019

Objectifs :

— Mettre en œuvre la syntaxe permettant de définir des classes en Java.

1 Une classe pour représenter les vecteurs

Écrire une classe `Vector` permettant de manipuler des vecteurs de $\mathbb{R}^3$. Les composantes du vecteur seront représentées par des `double`. L'interface de cette classe comporte :

① un constructeur à trois arguments ;

② un constructeur permettant de créer un vecteur nul ;

③ d'une méthode retournant une représentation du vecteur sous forme d'une chaine de caractères de la forme : `<composante n° 1, composante n° 2, composante n° 3>` ;

④ d'une méthode fournissant la norme du vecteur ;

⑤ d'une méthode renvoyant *une nouvelle instance* représentant la somme de deux vecteurs

⑥ d'une méthode renvoyant le produit scalaire de deux vecteurs.

Le code suivant permettra de tester le code développé :

```
public class TestVector {

    public static void main(String[] s) {
        Vector v1 = new Vector(1, 2, 3);
        Vector v2 = new Vector(3, 4, 5);
        Vector v3 = v1.add(v2);

        System.out.println(v3.toString());
        System.out.println(v1.dot(v2));
        System.out.println(v1.norm());
    }
}
```

2 Une classe pour représenter les ensembles

Cet exercice porte sur la création d'une classe permettant de stocker une liste non redondante d'entiers (c.-à-d. qu'elle ne peut pas contenir deux entiers identiques).

2.1 Classe StaticSet

⑦ Implémentez la classe `StaticSet` correspondant à l'interface décrite ci-dessous :

- `boolean add(int el)` : ajoute un élément à l'ensemble ;
- `boolean contains(int el)` : teste si l'élément appartient à la collection ;
- `String toString()` renvoie une chaîne de caractères contenant tous les éléments de l'ensemble classés par ordre croissant et séparés par un tiret (p. ex. si l'ensemble contient les éléments 3, 7 et 5, la procédure renverra `"3-5-7"`.

Les éléments seront stockés dans un tableau dont la taille est fixée lors de la création de la classe. Si on essaye d'ajouter un élément à un ensemble « plein », la méthode `add` renverra `false`. On pourra utiliser la classe `Arrays`.

Le code suivant permettra de tester le classe développée :

```
  StaticSet s = new StaticSet(10);
  s.add(3);
  s.add(2);
  s.add(1);
  s.add(2);

  System.out.println("Test contains: " + (s.contains(1) && !s.contains(5)));
  System.out.println("Test toString: " + s.toString().equals("1-2-3"));
}
```

2.2 Ensemble dynamique

L'implémentation précédente ne permet pas d'avoir une collection dynamique. On souhaite créer une nouvelle classe `DynamicSet` qui utilisera, pour stocker les données, un tableau dont la taille est adaptée au fur et à mesure des ajouts :

- lors de la création de l'objet, on crée un tableau d'entiers de taille donnée ;
- lors d'un ajout :
  - si le nombre d'éléments de la collection est plus petit que la taille du tableau, il suffit d'ajouter l'élément à la fin du tableau ;
  - dans le cas contraire, on crée un nouveau tableau plus grand (à vous de choisir la nouvelle taille de manière à limiter la complexité de l'ajout d'un élément) et on y copie tous les éléments de l'ancien tableau avant d'y ajouter le nouvel élément.

L'interface de cette classe est identique à l'interface de classe `StaticSet`. Mais elle comporte une méthode supplémentaire :

— `boolean del(int el)` : supprime un élément (renvoie `false` si on essaye de supprimer un élément qui n'appartient pas à l'ensemble).

⑧ Créez une nouvelle classe implémentant cette méthode.