

Programmation Orientée Objet

TP d'introduction, deuxième partie : espaces de rationnels

Ensimag 2^{ème} année 2015-16

Résumé

L'objectif de ce TP est de créer des classes représentant des ensembles de **Rational**. Les notions de Java abordées sont principalement la composition d'objets, la copie profonde ou légère, la délégation, les tableaux en Java, ...

5 Des vecteurs de rationnels

Question 8 Notre classe **Rational** est sympathique, mais bien seule pour l'instant. Nous allons l'utiliser pour manipuler des vecteurs de nombres rationnels (par exemple $\langle 1/2, 3/2 \rangle$, ou encore $\langle 1/6, 6/4, 33/1 \rangle$).

8.a - Créez une classe **Vector** représentant un vecteur de rationnels (de dimension fixée à la création), et donc ayant pour attribut un tableau d'éléments de type **Rational**. Réfléchissez bien en particulier au(x) paramètre(s) dont vous avez besoin pour l'initialisation d'un objet de cette classe, et aux visibilités des attributs.

8.b - Ajoutez à votre classe une méthode **toString()**, sans paramètre et renvoyant une chaîne de caractères (type **String**) représentant le vecteur. Par exemple, cette méthode appelée sur le vecteur $\langle 1/2, 3/2 \rangle$ devrait renvoyer la chaîne de caractères "**(1 / 2, 3 / 2)**".

8.c - Compilez et testez sur un vecteur de dimension 2 par exemple.

Question 9 Intéressons-nous maintenant à l'accès aux composantes du vecteur.

9.a - Ajoutez à votre classe une méthode **void set(int i, Rational r)** qui modifie le $i^{\text{ème}}$ rationnel du vecteur de telle sorte qu'il vaille désormais **r** (une erreur est levée si **i** excède la dimension du vecteur).

9.b - Écrivez un programme de test qui exécute le pseudo-code suivant :

- $\vec{v} \leftarrow \langle 0/1, 0/1 \rangle$;
- $a \leftarrow 2/3$ (**a** est un **Rational**)
- $v_1 \leftarrow a$ (affecte **r** au premier élément du vecteur)
- Afficher \vec{v} ;
- $b \leftarrow 3/2$ (**b** est un **Rational**)
- $a \leftarrow a * b$ (grâce à la méthode **mult** de la classe **Rational**. **a** vaut désormais $3/3 == 1$)
- Afficher \vec{v} ;

Compilez et exécutez. Que constatez-vous ? Est-ce un problème ? Si oui comment l'expliquez-vous et comment le résoudre ?

9.c - Écrivez une méthode **Rational get(int i)** prenant en paramètre un entier **i** et renvoyant le rationnel correspondant à la $i^{\text{ème}}$ composante du vecteur (ou une erreur si **i** excède la dimension du vecteur).

9.d - Écrivez un programme de test qui exécute le pseudo-code suivant :

- $\vec{v} \leftarrow \langle 1/2, 1/2 \rangle$;
- Afficher \vec{v} ;
- $a \leftarrow v_1$ (récupérer la première composante de \vec{v}) ;
- $a \leftarrow a * 1/3$ (grâce à la méthode **mult** de la classe **Rational**.)
- Afficher \vec{v} .

Compilez et exécutez. Que constatez-vous ? Est-ce un problème ? Si oui comment l'expliquez-vous et comment le résoudre ?

Question 10 Nous allons maintenant doter notre classe `Vector` quelques opérations basiques d'un espace vectoriel sur \mathbb{Q} :

10.a - Ajoutez à la classe `Vector` une méthode de multiplication par un rationnel. Cette méthode prend en paramètre un `Rational`, et multiplie toutes les composantes du vecteur par ce rationnel.

10.b - Ajoutez une méthode `add`, additionnant un `Vector` donné en paramètre à l'objet de type `Vector` sur lequel elle est invoquée. Pour rappel, l'addition de deux vecteurs *de même taille* se fait en additionnant leurs composantes respectives. La méthode échoue si le vecteur passé en paramètre n'est pas de même dimension que le vecteur sur lequel la méthode est appelée.

6 Des vecteurs de taille dynamique (optionnel)

Question 11 Nous voulons maintenant enrichir notre modèle de vecteur pour rendre sa dimension dynamique. Concrètement, nous désirons créer une classe `ExtensibleVector` modélisant un vecteur de rationnels de dimension variable, et possédant au moins les méthodes suivantes :

- une méthode `void add(Rational r)` qui ajoute le rationnel passé en paramètre en fin de vecteur : par exemple, cette méthode, appelée sur le vecteur $\langle 1/2, 1/3 \rangle$ avec le paramètre $1/4$ modifie le vecteur en $\langle 1/2, 1/3, 1/4 \rangle$.
- une méthode `int getDimension()` renvoyant la dimension (la longueur) du vecteur.

Imaginez une solution simple pour développer une telle classe. Développez cette solution et analysez la complexité des méthodes `get`, `add` et `getDimension`.