Exercices sur les ArrayList

Maria-Virginia Aponte, François Barthélémy

NFA031

Exercice 1 notes

On veut gérer les notes obtenues par un élève dans une matière en utilisant un objet de la classe `ArrayList`. Les notes seront de type `double`.

Question 1 *opérations de base*

Écrivez les méthodes suivantes. Chaque méthode prendra en paramètre un `ArrayList<Double>` contenant les notes. Vous choisirez vous-même les signatures appropriées pour ces méthodes.

1. Une méthode pour ajouter une nouvelle note à la liste
2. Une méthode pour afficher toutes les notes
3. Une méthode pour calculer et retourner la moyenne des notes

Le programme principal devra réaliser les opérations suivantes dans l'ordre :
— Créer un `ArrayList` vide pour les notes
— Ajouter les notes 12, 14 et 9
— Calculer et afficher la moyenne
— Ajouter la note 13
— Calculer et afficher à nouveau la moyenne
— Afficher toutes les notes

Question 2 *correction des erreurs*

Une nouvelle méthode permettra de modifier une note dans la liste (utile en cas d'erreur de saisie ou de correction). Cette méthode prendra en paramètre la liste des notes, l'indice de la note à modifier et la nouvelle valeur.

Testez cette méthode dans le programme principal en modifiant une des notes puis en affichant à nouveau la moyenne et toutes les notes.

Question 3 *coefficients*

On souhaite maintenant associer un coefficient (de type `int`) à chaque note pour calculer une moyenne pondérée. Pour cela, on utilisera **deux ArrayList coordonnés** :
— Un `ArrayList<Double>` pour les notes
— Un `ArrayList<Integer>` pour les coefficients
Le lien entre les deux se fait par l'indice : la note à l'indice `i` a le coefficient à l'indice `i`.
Travail à faire :

1. Modifiez la méthode d'ajout pour qu'elle prenne en paramètre une note ET son coefficient, et les ajoute aux deux listes respectives
2. Modifiez la méthode d'affichage pour afficher chaque note avec son coefficient
   Format suggéré : `Note 1 : 12.0 (coefficient 2)`

3. Modifiez la méthode de calcul de moyenne pour calculer la moyenne pondérée
   Rappel : `moyenne pondérée = somme(note[i] * coef[i]) / somme(coef[i])`
4. Testez avec les données suivantes :
   — Note 12, coefficient 2
   — Note 14, coefficient 3
   — Note 9, coefficient 1
   — Note 13, coefficient 2

Question 4 *menu (bonus)*

Modifiez le programme principal pour que les différentes opérations soient proposées au moyen d'un menu interactif permettant à l'utilisateur de choisir l'opération à effectuer.

Exercice 2 ensembles mathématiques

On va utiliser des objets de la classe `ArrayList<Integer>` pour représenter des ensembles mathématiques de nombres entiers.

Rappel : Un ensemble mathématique ne peut pas contenir deux fois le même élément. Il faudra donc vérifier avant d'ajouter un élément qu'il n'est pas déjà présent (utilisez `contains` ou `indexOf`).

Question 1 *opérations de base*

Écrivez les méthodes suivantes (choisissez vous-même les signatures appropriées) :

1. **creerSingleton** : crée et retourne un nouvel ensemble contenant un seul élément (donné en paramètre)
2. **ajouterElement** : ajoute un élément à un ensemble existant, seulement s'il n'est pas déjà présent
3. **afficherEnsemble** : affiche tous les éléments d'un ensemble
   Format suggéré : `{1, 3, 5, 7}`

Le programme principal créera quelques ensembles et testera ces opérations.

Exemple de test suggéré :

```
// Créer un singleton {5}
// Ajouter 3, puis 7, puis 3 ànouveau (ne doit pas être ajouté)
// Afficher l'ensemble : doit afficher {5, 3, 7}
```

Question 2 *propriétés des ensembles*

Ajoutez les deux méthodes suivantes :

1. **contient** : teste si un élément appartient à un ensemble (retourne un booléen)
2. **taille** : retourne le nombre d'éléments dans un ensemble

Si vous avez bien implémenté la méthode `ajouterElement` de la question précédente (en vérifiant les doublons), la méthode `taille` peut simplement utiliser `size()` de l'`ArrayList`.

Question 3 *opérations ensemblistes*

Implémentez les deux opérations mathématiques suivantes, qui créent et retournent chacune un **nouvel** ensemble (donc un nouvel objet `ArrayList`) :

1. **union** : retourne un ensemble contenant tous les éléments présents dans au moins l'un des deux ensembles

2. **intersection** : retourne un ensemble contenant uniquement les éléments présents dans les deux ensembles

Exemple :

```
ensemble1 = {1, 2, 3, 4}
ensemble2 = {3, 4, 5, 6}
union(ensemble1, ensemble2) = {1, 2, 3, 4, 5, 6}
intersection(ensemble1, ensemble2) = {3, 4}
```

Algorithme suggéré pour l'union :

1. Créer un nouvel ensemble vide
2. Ajouter tous les éléments du premier ensemble
3. Parcourir le deuxième ensemble et ajouter chaque élément (la méthode `ajouterElement` vérifiera automatiquement les doublons)

Algorithme suggéré pour l'intersection :

1. Créer un nouvel ensemble vide
2. Parcourir le premier ensemble
3. Pour chaque élément, vérifier s'il est aussi dans le deuxième ensemble
4. Si oui, l'ajouter au nouvel ensemble

Ce chapitre a été rédigé avec l'assistance d'une intelligence artificielle, Claude 4.5 Sonnet de la société Anthropic