Programmation Java
A. Bourmaud, J. Maitre, P. Rodriguez
Ch. Sempé, J-F. Viaud

Introduction à la programmation et à la modélisation en Java

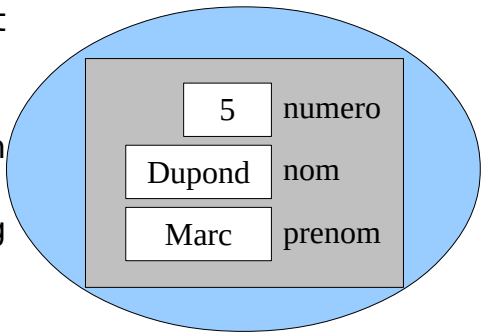
TD n°4 & 5 – Objets et Collections

Dans la suite, plusieurs classes vont intervenir. Utilisez une feuille de papier différente par classe (en machine, le code serait présent dans autant de fichiers que vous utilisez de feuille)

I. Les personnes

A. Contexte

On dispose de la classe *Personne*. Une personne est définie par son numéro, son nom et son prénom.



Pour manipuler les objets de la classe *Personne*, on dispose :

- d'un constructeur : public Personne(int num, String nom, String prenom)
- des getters getNum, *getNom* et *getPrenom*
- d'un méthode *toString* pour l'affichage.

Révisions : Ecrivez cette classe Personne.

On manipule des groupes de personnes en les stockant dans des *ArrayList*.

Les méthodes applicables à un *ArrayList* sont les suivantes :

- *add(Object ob)* : place *ob* dans l'ArrayList
- *get(int n)* : renvoie le n^{ième} élément de l'ArrayList
- *remove(int n)* : supprime le n^{ième} élément de l'ArrayList
- *size()* : renvoie la taille (le nombre d'éléments) de l'ArrayList
- *isEmpty()* : renvoie vrai si l'ArrayList est vide

B. Mise en place de la collection

1. Créez une classe *CollectionPersonnes* dotée : d'un attribut public groupe qui est un *ArrayList* de personnes et d'un constructeur sans paramètre instanciant groupe comme *ArrayList* vide.
2. Créez une classe App*Scores* qui contiendra la méthode *main*.
3. Dans la méthode *main,* écrivez les instructions permettant de créer deux personnes, une collection de personne et de stocker ces deux personnes dans la collection.

4. Dans la classe *CollectionPersonnes*, écrivez le sous-programme *afficherPersonnes* capable d'afficher les noms de tous les éléments du groupe. Est-ce une fonction ou une procédure ?
5. Dans la classe *CollectionPersonnes,* écrivez la fonction *personneDeNumero*, qui reçoit *n* (int) et qui renvoie la personne de collection portant ce numéro. **Dans le cas où la personne serait absente, l'objet *null* sera retourné.**
6. Dans la classe *CollectionPersonnes,* écrivez la fonction *personneDeNom*, qui reçoit *s* (String) qui renvoie la personne de collection portant ce nom. **Attention, la comparaison == n'est pas utilisable avec les String ; vous devez utiliser la méthode *equals,* par exemple avec la syntaxe *s.equals(nom);* où s et nom sont des String.**
7. Dans la méthode main, utilisez les trois méthodes *afficherPersonnes, personneDeNumero* et *personneDeNom*, afin de les tester.

Remarque : dans la vraie vie, la méthode *personneDeNom* requiert beaucoup de tests, par exemples : la collection est vide, ne contient qu'un élément ou plusieurs, la personne est présente au début, à la fin, à un emplacement autre, la personne est absente, le nom apparaît plusieurs fois, le nom apparaît une seule fois.

II. Gestion des scores obtenus durant un jeu

1. Les scores

Les personnes jouent à un jeu et on veut pouvoir gérer les scores obtenus par chaque personne.

Nous disposons de la classe *Score*. Un score est défini par deux informations : le numéro de la personne qui a obtenu le score et son résultat (c'est à dire le nombre de points obtenus).

Dans la classe *Score* sont définis les getters *getNumPers()* et *getResultat()*.

On dispose du constructeur : *s = new Score(2, 1345);*

Révisions : Ecrivez cette classe Score.

2. Les collections de scores

Exemple :

groupe

numero	nom	prenom
7	Dupond	Marc
9	Durand	Robert
2	Allais	Magali
10	Gomez	Annie

scores

numPersonne	resultat
9	1 235
7	512
7	787
2	1 220
7	1 583

On voit ici que la personne numéro 7 (donc Marc Dupond), a obtenu un résultat de 1 583 points.

8. Créez une classe *CollectionScores* dotée : d'un attribut public resultats qui est un *ArrayList* de scores et d'un constructeur sans paramètre instanciant scores comme *ArrayList* vide.
9. Dans la méthode *main,* écrivez les instructions permettant de créer deux scores, une collection de scores et de stocker ces deux scores dans la collection.

Les méthodes des questions 10 à 14 sont à ajouter dans la classe *CollectionScores*

10. Ecrivez la fonction *meilleurScore*, qui renvoie le meilleur score de la collection.
11. Ecrivez la fonction *meilleurJoueur*, qui renvoie le numéro du joueur possédant le meilleur score de la collection.
12. Ecrivez la fonction s*coresDuJoueur* qui reçoit un numéro de joueur et qui renvoie la CollectionScores contenant les scores de ce joueur.
13. Ecrivez la fonction s*coreMoyenDuJoueur*, qui reçoit un numéro de joueur et renvoie le nombre de points moyen obtenu par ce joueur.
14. Créez une classe *GestionScores* dotée : d'un attribut public collecScores qui est une CollectionScores, d'un attribut public collecJoueur qui est une CollectionPersonnes et d'un constructeur sans paramètre instanciant les collections.

```
Robert Durand a obtenu 1235 points
Marc Dupond a obtenu 512 points
...
```

15. Dans cette classe *GestionScores,* ajoutez la méthode *afficherScores* qui pour chaque élément de scores, affiche les informations sous la forme ci-dessus.
    Pour écrire cette procédure, posez vous les questions suivantes : combien de lignes ce sous-programme doit-il afficher ? Et donc, quel tableau faut-il prioritairement parcourir ?
16. Toujours dans la classe *GestionScores,* ajoutez la méthode scoresMoyens qui retourne une nouvelle CollectionScores contenant pour chaque joueur son score moyen. Réfléchissez bien à votre stratégie de gestion des doublons (il ne doit pas y en avoir dans la collection retournée).