Programmation Java

TP°4 – Collections d'objets

I. Introduction

Téléchargez le projet fourni via *Moodle* et ouvrez-le avec *NetBeans*.

Nous disposons d'un fichier de données concernant des personnes, fourni avec le projet.

Nous allons effectuer des calculs concernant ces données.

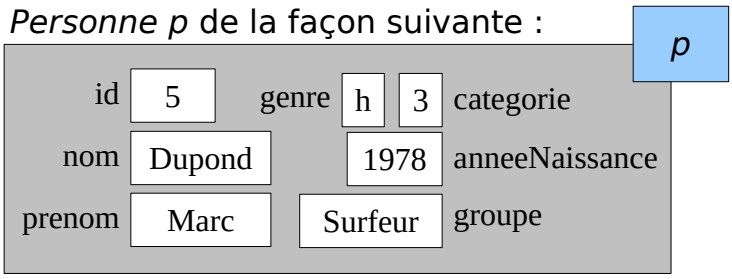
II. La classe *Personne*

A. Prise en main

On dispose de la classe *Personne* qui sert à définir le nouveau type de données *Personne*.

Une *Personne* est une donnée regroupant un identifiant, un nom, un prénom, un genre ('f', 'h', ...), un numéro de catégorie, une année de naissance et le nom du groupe auquel elle appartient.

On peut représenter une *Personne p* de la façon suivante :



Le constructeur d'une personne s'utilise de la manière suivante :

p = new Personne (5, "Dupond", "Marc", 'h', 1978, "Surfeur", 3) ;

La classe *Personne* est fournie avec des getters sur tous les attributs sauf le genre.

1. Ajoutez deux getters supplémentaires : *estHomme* qui retourne vrai si et seulement si le genre est 'h' et *estFemme* qui retourne vrai si et seulement si le genre est 'f'.

La classe *Personne* est fournie avec deux setters sur les attributs groupe et catégorie.

On dispose également d'une procédure *afficher* qui génère un affichage dans la console.

2. Dans la méthode *main* de la classe *App :*
   - instanciez deux personnes *marc* et *paul*.

- afficher le nom, l'année de naissance et le groupe des deux personnes
- modifier le groupe de la première personne, afficher à nouveau son groupe
- afficher les caractéristiques de la seconde personne en utilisant sa méthode *afficher*
- utilisez le débugger pour visualiser le contenu des variables de votre programme.

B. Amélioration de la classe Personne et notion de référence

3. Essayer d'afficher *marc* en utilisant uniquement l'instruction *System.out.println(marc)*.
4. Vous obtenez un affichage de type `Personne@1fc4bec`, à votre avis, que représente cette valeur ?
5. Instanciez une nouvelle personne *marcBis* possédant exactement les mêmes attributs que *marc.*
6. Testez l'égalité *marc == marcBis.* « Affichez » *marcBis* avec l'instruction *System.out.println(marcBis)*. Interprétez.
7. Testez l'égalité avec l'instruction *marc.equals(marcBis)*. Interprétez.
8. Créez un « nouveau » marc avec l'instruction *Personne marcTer = marc;*
9. Testez (avec les deux approches) l'égalité de *marcTer* avec *marc* et *marcBis*. Interprétez.
10. Changez la catégorie de *marcTer*. Affichez (avec la méthode *afficher*) *marc.* Interprétez.

Améliorons maintenant la situation en résolvant ces problèmes :

11. Dans la classe *Personne,* ajoutez une méthode *public string toString()* qui s'inspire de la méthode *afficher* mais qui retourne une chaine de caractères plutôt que de l'afficher.
12. Testez à nouveau l'instruction *System.out.println(marc).* Interprétez.
13. Dans la classe *Personne,* ajoutez une méthode *public boolean equals(Object p)* dans laquelle :
    - Vous comparez la classe de p et celle de la personne (avec la méthode *getClass())*, puis en cas d'égalité
    - Vous comparez les identifiants.
    - La méthode retourne vrai si et seulement les classes sont les mêmes et les identifiants aussi.
14. Testez à nouveau les égalités entre les trois *marc* avec les deux stratégies (== et *equals*). Interprétez.

III. Manipulation d'une collection de personnes

A. Prise en main

Le fichier *desPersonnes.txt*, situé à la racine du projet, contient les données concernant 25 personnes.
Ouvrez ce fichier avec un éditeur de texte pour connaître son contenu.

Le projet contient une classe *CollectionPersonnes* et une classe *LecteurFichier.*

15. Prenez cinq minutes pour lire le code fourni.
16. Dans la classe *CollectionPersonnes*, ajoutez une procédure *afficher* qui affichera toutes les personnes du groupe. Vous utiliserez la procédure *afficher* de la classe *Personne.*
17. Dans la classe *CollectionPersonnes, é*crivez la fonction *effectifDeLAnnee*, qui reçoit une année de naissance, et qui renvoie le nombre de personnes de la collection nées cette année là. Testez le fonctionnement de ce sous-programme.

B. Fonction de proximité – Matching

On cherche à déterminer le taux de ressemblance (score de matching) entre deux personnes.

Les règles de calcul du taux peuvent être basées sur celles-ci :

- Appartenance au même groupe : 3 points ; groupes différents : 0 point.
- Différence de catégorie : 0 → 4 points, 1 → 2 points, 2 → 0 point.
- Différence d'âge : 0 à 2 → 4 points, 3 à 5 → 3 points, 6 à 8 → 2 points, 9 à 11 → 1 point, au-delà → 0 point.

Remarque : Pour comparer deux variables *s1* et *s2* de type *String*, il faut effectuer le test *if(s1.equals(s2)).*

18. Dans la classe *Personne,* ajoutez une fonction qui reçoit une personne et qui renvoie ce taux avec cette personne.
19. Testez le fonctionnement de ce sous-programme. Pour cela, dans le *main,* créez plusieurs personnes pour lesquelles vous calculerez le score à la main, puis comparerez avec votre fonction. Gardez la trace de tous ces tests. Il doit y en avoir au moins 10, correspondant à des situations toutes différentes.

C. Recherche d'extremum

20. Dans la classe *CollectionPersonnes, é*crivez la fonction *laPlusProche* qui reçoit une personne *p* et qui renvoie la personne qui est la plus proche de p dans la collection. Testez cette fonction dans la méthode *main.* **Il faudra éviter de comparer une personne à elle-même, en utilisant la méthode *equals*.**

D. Construction d'un ensemble de personnes

21. Dans la classe *CollectionPersonnes,* écrivez la fonction *lesPlusProches* qui reçoit *p* (une personne) et qui renvoie le *ArrayList* de *Personne* contenant toutes les personnes de la collection qui sont les plus proches de *p*, c'est-à-dire toutes les personnes qui possèdent la proximité maximale avec *p*. La fonction devra trouver la personne la plus proche de *p*, déterminer son score de matching, puis utiliser cette données pour trouver toutes les personnes qui possèdent ce score, et les regrouper dans un *ArrayList*. Testez cette fonction dans le *main.*

E. Liste des groupes

Une dernière question plus facile pour finir :

22. Dans la classe CollectionPersonnes, écrivez une méthode *personnesGroupe qui recoit un groupe (String) et* qui renvoie un *ArrayList* de *Personne* contenant toutes les personnes de ce groupe. Testez votre méthode dans le *main.*