

Concept. et prog. objet


\1 TP n°2 : polymorphisme

Dépublié

Détails

Écrit par stéphane Domas

Catégorie : M3105 - Concept. et prog. objet avancée (/index.php/menu-cours-s3/menu-mmi3-test)

 Publication : 23 octobre 2014

 Affichages : 2092

1°/ Rencontre universelle

- Dans le TP n°1, quand deux humains h1 et h2 sont pris dans la population, on doit connaître leur type d'instance afin d'appeler la bonne méthode de rencontre. On a donc un code qui ressemble à :

```
1 | h1 = pop.getHumain(index1);
2 | h2 = pop.getHumain(index2);
3 |
4 | if (h1.isHomme()) {
5 |     if (h2.isFemme()) {
6 |         h = (Homme)h1;
7 |         f = (Femme)h2;
8 |         bebe = h.rencontre(f);
9 |     }
10 | }
11 | ...
```

- En s'inspirant des exemples du TD n°2, modifiez les classes Humain, Homme et Femme afin de pouvoir écrire directement `h1.rencontre(h2)` sans qu'il y ait d'erreurs à la compilation ni à l'exécution.

2°/ Classement

2.1°/ Classement mono-critère

- Modifiez l'entête de Humain pour que la classe implémente l'interface Comparable :

```
1 | class Humain implements Comparable<Humain> {
2 |     ...
```

- Dans Humain.java, ajoutez la méthode suivante et complétez le code sachant que :
 - la méthode renvoie -1 si l'age de this est < à l'age de h
 - la méthode renvoie 0 si les ages sont égaux
 - la méthode renvoie 1 sinon.

```
1 | public int compareTo(Humain h) {
2 |
3 |     // à compléter
4 | }
```

- On suppose que le programme principal s'appelle TP2.java (copier/coller de TP1.java)

- Modifiez TP2.java ainsi que Population.java pour qu'à la fin de chaque tour de jeu, la population soit triée par âge croissant et ensuite affichée (NB : pour voir comment trier une collection, il existe des exemples à foison sur le Net)

2.2°/ Classement multi-critères.

- L'objectif est d'essayer de classer les humains selon leur âge mais pour deux hommes du même âge, de les classer selon leur salaire.
- L'idée est de redéfinir la méthode compareTo() pour ajouter ce cas.
- Pour cela, modifiez TP2.java pour tirer un salaire aléatoirement entre 1000 et 11000 lorsque l'on crée la population initiale.
- Modifiez également Homme.java :
 - ajoutez un attribut int salaire dans la classe Homme.
 - modifiez les constructeurs de Homme pour initialiser le salaire à 0 ou bien à une valeur donnée en paramètre
 - redéfinissez la méthode vieillir() pour qu'à 20 ans, on initialise le salaire d'un Homme avec une valeur tirée aléatoirement entre 1000 et 11000.
 - redéfinissez la méthode compareTo() :
 - si les ages de this et de h sont différents : on renvoie -1 ou 1, comme pour la méthode de Humain.
 - si les ages sont identiques, alors si h est une Femme, on renvoie 0 et si h est un Homme, on renvoie la différence entre le salaire de this et celui de h.

```

1 | class Homme extends Humain {
2 |     ...
3 |     public int compareTo(Humain h) {
4 |
5 |         // à compléter
6 |     }
7 |     ...
8 | }
```

- Lancez l'exécution de TP2 sur au moins 25 tours.
- On constate que le tri ne se fait pas comme on l'attend : le tri entre salaire n'est pas respecté.
- La solution est très simple à mettre en oeuvre puisqu'il suffit de changer une valeur. A vous de trouver laquelle.