TP de Programmation Orientée Objet : héritage

ZOO : Zoologie Orientée Objet

Équipe pédagogique Programmation Orientée Objet
Octobre MMXIV

Nous allons dans ce TP nous intéresser à nos amies les bêtes. Plus précisément, on veut créer une application permettant de simuler un parc zoologique, avec l'ensemble des animaux qui en font partie.

1 Un zoo dernier cri

Nous allons dans un premier temps créer une classe `Animal`, représentant, comme son nom l'indique, un animal. Tout animal a un nom, qui est une chaîne de caractère et un poids en kilogrammes (un entier). La classe fournit une méthode `toString()` qui retourne son nom, et une une méthode `crier` qui affiche simplement à l'écran la chaîne de caractères `"<nom> crie..."`.

Question 1 Écrivez le code de la classe `Animal`.

On s'intéresse maintenant à la classe `Zoo` suivante :

```
1  import java.util.LinkedList;
2
3  class Zoo {
4      private String nom;
5      private LinkedList<Animal> animaux;
6
7      public Zoo(String nom) {
8          this.nom = nom;
9          animaux = new LinkedList<Animal>();
10     }
11
12     public int getNbAnimaux() {
13         return animaux.size();
14     }
15
16     public void ajouteAnimal(Animal animal) {
17         animaux.add(animal);
18     }
19
20     public void crier() {
21         for (Animal a : animaux) {
22             a.crier();
23         }
24     }
25 }
```

Question 2 Observez cette classe, en particulier l'attribut `animaux`. Quel est l'action de la méthode `crier()` ?

Question 3 Créez une classe de test qui crée le zoo *Ensimag*, y ajoute les animaux *Catherine*, *Gwen*, *Mathias*, *Matthieu*, *Nicolas* et *Sylvain*, et fait crier toute cette ménagerie.

2 Un peu de biodiversité

Nous allons mettre un peu de diversité dans notre zoo et introduire plusieurs espèces d'animaux. Bien sûr, il s'agit maintenant de prendre en compte les cris spécifiques de chaque espèce.

Question 4 Créez quelques classes d'animaux spécifiques (`Lion`, `Tigre`, `Vache`, `Canard` ...). Vous pouvez ajouter des propriétés particulières : par exemple un canard a une couleur de plume (de type `String`) et une vache a un nombre donné de taches. Les cris des nouveaux animaux ressemblent maintenant à :

— Au lieu de `Sylvain crie...` on entend désormais `Sylvain, le canard aux belles plumes "noir tuxien", crie... il cancane`[1]

— `Matthieu, la vache sans taches a 17 taches, crie... il meugle`

Bien gérer les constructeurs, et redéfinir si nécessaire les méthodes `toString()` et `crier()`.

Question 5 Modifiez votre programme de test précédent pour créer non plus des animaux génériques mais des animaux particuliers (vaches, canards, lions). Avez-vous besoin de modifier votre classe `Zoo` pour que le programme fonctionne ? Les animaux crient-ils correctement ?

3 Jouons un peu avec l'héritage et la liaison tardive

Soit le programme défini par la classe de test suivante :

```
1  public class TestHeritage {
2      public static void main(String[] args) {
3          Canard donald = new Canard("Canardo", 5, 543); // 543 plumes
4          Animal tux = donald;
5
6          System.out.println("Faisons crier Donald deux fois... ");
7          donald.crier();
8          tux.crier();
9
10         System.out.println("un canard aux plumes... " +
11             donald.getCouleurDePlumes());
12         System.out.println("un canard aux plumes... " +
13             tux.getCouleurDePlumes());
14
15         /***************/
16         Lion  simba = new Lion("Simba", 243);
17         Animal unAnimal = simba;
18
19         unAnimal = new Lion("Leo", 332);
20         Lion leo = (Lion) unAnimal;
21
22         unAnimal = new Animal("Leo", 332);
23         leo = (Lion) unAnimal;
24     }
25 }
```

Question 6 Essayez de deviner, pour chacune des lignes 4, 7, 8, 11, 13, 17, 20, 23, si cette ligne compile et s'exécute correctement. Expliquez pourquoi. Testez ce programme.

4 À table...

Chaque espèce animale a un régime alimentaire particulier, commun à tous les individus de cette espèce. Par exemple, un `Canard` mange 1 kg de graines par jour et une `Vache` mange 100g d'herbe multiplié par son poids en kilos.

Le zoo souhaite calculer ce que cette nourriture lui coûte globalement chaque jour.

Question 7 Créer une classe `Regime`, ayant comme attributs un nom (exemple : `"fruits secs"`, `"viande de gnou "`...) et un prix au kilogramme. Ajoutez à cette classe les accesseurs dont vous avez besoin.

1. Culture générale : Tux serait un canard ?

Question 8 Modifiez votre classe `Animal` pour y ajouter un attribut `Regime`, un accesseur sur cet attribut, et les constructeurs de toutes les classes de la hiérarchie des animaux.

Attention : comme le régime est commun à tous les animaux d'une espèce, il serait malvenu de passer une instance de `Regime` aux constructeurs des sous-classes, par exemple au constructeur de `Canard`... Comment alors initialiser l'attribut de type `Régime` dans `Animal` ?

Question 9 Ajoutez une méthode permettant de calculer le coût de nourriture de l'animal. Prenez garde au fait que tous les canards mangent la même quantité de nourriture (1kg), alors qu'une vache mange une quantité de nourriture fonction de son propre poids...

Question 10 Ajoutez à votre classe `Zoo` une méthode permettant de calculer le coût global de la nourriture pour tous les animaux.

Bien que chaque espèce d'animal ait un régime alimentaire bien défini, ce n'est pas le cas pour un animal générique (instance de la classe `Animal` sans être instance d'une de ses sous-classes). En effet : si l'on ne connaît pas de quelle espèce exacte est l'animal, comment savoir ce qu'il va manger ? De même le zoo a besoin de pouvoir calculer le coût de nourriture d'un animal, mais comment répondre à cette question pour un animal quelconque ? Plus généralement, à votre avis, serait-il légitime d'instancier directement un `Animal` qui n'est pas un canard, un tigre ou une vache ?