Programmation Orientée Objet

TP héritage – ZOO : Zoologie Orientée Objet

Proposé par : Samah Bouzidi

Dans ce TP, intéressons-nous à nos amies les bêtes (décidément...). Il s'agit de créer une application de gestion d'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` Tout animal a un nom (une chaîne de caractère), et un poids en kilogrammes (un entier). La classe fournit deux accesseurs sur le nom et le poids, et une méthode `crier()` qui affiche simplement à l'écran la chaîne de caractères `"<nom> crie..."`. Enfin, bien sûr, elle déclare un constructeur adapté et une méthode `toString()` qui retourne une représentation textuelle de l'état de l'objet.

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

On s'intéresse maintenant à la classe `Zoo` suivante (fichier disponible sur chamilo) :

```
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 ajouterAnimal(Animal animal) {
17         animaux.add(animal);
18     }
19
20     public void crier() {
21         for (Animal a  : animaux) {
22             a.crier();
23         }
24     }
25
26     @Override
27     public String toString() {
28         return "Le zoo " + nom + " contient : " + animaux;
29     }
```

1. Temps estimé, environ 5 minutes. Sinon, revoir les premières séances...

```
30 }
```

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 *EISTI*, y ajoute les animaux *Catherine*, *Julie*, *Sahar Benoit*, *Nicolas*, *Sebastien* 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 (1 ou 2) d'animaux spécifiques : `Lion`, `Tigre`, `Vache`, `Canard`, etc. En plus d'un nom et d'un poids, ajoutez des propriétés spécifiques à ces animaux : par exemple un canard a une couleur de plume (de type `String`, accesseur `getCouleurPlume()`) 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 crie... Ce canard de 3kg aux belles plumes noir tuxien cancane !` [2]

— Et à la place de de `Matthieu crie...` on a maintenant `Matthieu, la vache à 17 taches qui tachent, crie... il meugle`

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

Bien entendu, veillez à factoriser/réutiliser au maximum tout ce qui a déjà été écrit !

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 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 6 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.

Question 7 Modifiez votre classe `Animal` pour y ajouter un attribut `Regime`. Modifiez 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` déclaré dans la classe `Animal` ?

2. Culture générale : Tux serait un canard ?

Question 8 Ajoutez à la classe `Animal` 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 9 Ajoutez à votre classe `Zoo` une méthode permettant de calculer le coût global de la nourriture pour tous les animaux.

Question 10 *Ouverture :* 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, est-il légitime de pouvoir instancier directement un `Animal` qui n'est pas, en fait, un canard, un tigre ou une vache ?