TP2 - Héritage, polymorphisme et interfaces en Java

Septembre 2005

Objectif

L'objectif ce TP est de présenter l'héritage, le polymorphisme et l'utilisation des interfaces en Java. Vous devez écrire la javadoc (`http://java.sun.com/j2se/javadoc/writingdoccomments/`) au fur et à mesure de l'avancement et il est **fortement** conseillé d'avoir le documention de java sous les yeux : `http://java.sun.com/j2se/1.5.0/docs/api/index.html`

1 Appel d'une méthode la classe mère

Dans cet exercice nous allons modéliser des *plantes*, des *animaux*, des *mamifères* et des *oiseaux* mais aussi des *chiens*, des *aigles*, des *lapins* et des *hommes*.

1.1 Modélisation par héritage et Instanciation

- Modélisez cette hiérarchie (directement en Java) avec des classes. Ajoutez à chaque classe fille au moins une variable et une méthode nouvelle selon votre imagination (âge, nom, sexe, ...).
- Ajoutez une classe executable `Test` et créez des instances de chacunes des classes. Affichez avec la méthode `System.out.prinln()` chacun des objets.
- Redéfinissez la méthode `toString()` (héritée de `Object`, voir dans la documentation) de la super classe pour qu'elle affiche "Je suis un animal" (ou "Je suis une plante"). Exécutez la classe `Test`. Expliquer le résultat en étudiant la classe `java.lang.System`.
- Modifiez la méthode précédente pour qu'elle affiche "Je suis un animal et mon identifiant est X".

1.2 Polymorphisme simple

- Avec quels types de références pouvez-vous manipuler les classes précédentes ?
- Dans la méthode `main` la classe `Test`, créez des tableaux qui permettent de voir certains des animaux crées **dans la section précédente** comme des collections (de taille fixe) d'animaux ou de mamifères.
- Ajouter à chaque classe une méthode `String getInfo()` (qui appellera celle de la superclasse) pour que l'affichage du tableau suivant :
`{new Animal(12), new Animal(), new Chien(5,"Médor"), new Homme(), new Homme(25,"Robert")}`
soit de la forme :

```
Je suis un animal agé de 12 an(s).
Je suis un animal.
Je suis un animal agé de 5 an(s). Je suis un mamifère. Je suis un chien de nom Médor.
Je suis un animal. Je suis un mamifère. Je suis un homme.
Je suis un animal agé de 25 an(s). Je suis un mamifère. Je suis un homme de nom Robert.
```

1.3 Classes abstraites

- Modifiez les classes pour s'assurer que tous les Animaux possèdent la méthode `String moyenExpression()` qui retourne une phrase du type "Je parle", "J'aboie", ou "Je fais des bulles" (l'aigle trompette, glapit ou glatit ; le lapin clapit).
- Est-ce que le moyen d'expression peut être définit pour toutes les classes ? Peut-on alors créer une instance de la classe Animal, quelle est votre conclusion pour s'assurer de cela ? Faites de même pour toutes les classes pour les lesquelles cela vous semble nécessaire.
- Ecrivez une méthode `afficherAnimaux()` dans la classe `Animal` qui prend en paramètre un tableau d'animaux et qui affiche leur référence et leur cri.Testez la sur le tableau de la question précédente.

1.4 Administration de classe et polymorphisme

- Modifiez vos classes pour que l'on garde automatiquement des références (dans un tableau) vers tous les animaux créés (Vous vérifierez aussi que l'on ne crée pas plus de `NB_MAX_ANIMAUX`).

- Créez une méthode (de classe ou d'instance ?) qui affiche la description complète de tous les animaux crées.

1.5 Héritage multiple et Interface

- On souhaite maintenant considérer que certains animaux (pas forcément tous) sont des carnivores (ils comportent la méthode `void manger` avec comme paramètre un autre animal) ou des herbivores (ils comportent la méthode `void manger` avec comme paramètre une plante) ou les deux. Créez deux interfaces pour décrire ces comprotements
- Modifiez vos classes pour qu'elles implantent ces interfaces :
  - Les carnivores : quand ils mangent, les chiens affichent “Je mord X”, et les aigles affichent “Je déchire X”.
  - Les herbivores : les vaches affichent “Je broute X”, et les lapins “Je grignotte X”.
- Les hommes sont à les fois carnivores et herbivores, proposez deux solutions pour le représenter :
  - Dans un premier temps, sans créer de nouvelle interface.
  - Dans un second temps en créant une nouvelle interface `IOmnivore`.
- Le polymorphisme est aussi utilisable avec les interfaces :
  - Appliquez la méthode `manger(unePlante)` sur la Vache que avez déjà créee en la voyant à travers une référence de type `IHerbivore`. Peut-on appliquer la méthode `moyenExpression()` sur cette référence ?
  - Créez deux tableaux `lesCarnivores` et `lesHerbivores` pour manipuler les animaux existants. Créez un lapin, et faites le manger par tous les carnivores, créez une plante et faites la manger par les herbivores.

2 Si vous avez déjà fini...

- Vous pouvez construire depuis eclipse le diagramme UML correspondant à ce TP. Placez vous sur le répertoire du paquetage et et créer un nouveau diagramme de classe (bouton droit . . .). Vous pouvez alors insérer les classes Java existantes
- Vous pouvez alors ajouter des nouvelles classes ou modifier des classes existantes (le code Java est alors mis à jour). Ajoutez de nouveaux animaux (Poule, Tigre, . . .).