Université Paris Diderot **Programmation Orientée Objet**
L2/L3 **Année 2014-2015**

TD - Séance n°3

Héritage

> Les auteurs de ce sujet vous présentent par avance leurs excuses pour la fausseté grossière de la modélisation du règne animal qu'ils vont vous présenter ci-dessous.

Notes sur l'héritage :
Le but du TP est d'apprendre à se servir de l'héritage en Java. L'héritage permet, lorsque l'on définit une classe, de réutiliser une partie du code et des attributs de la classe parente à l'aide du mot clef `super`. Par exemple, si l'on souhaite définir des classes `Vehicule` et `Voiture`, on pourra écrire :

```
public class Vehicule {
  private int nombreDeRoues;
  public Vehicule(int n) {
    this.nombreDeRoues = n;
  }
  public String toString() {
    return "Ce vehicule a " + this.nombreDeRoues + " roues.";
  }
}

public class Voiture extends Vehicule {
  private String couleur;
  public Voiture(String couleur) {
    super(4); // Appelle le constructeur parent
    this.couleur = couleur;
  }
  public String toString() {
    String s = super.toString(); // Appelle la méthode parente
    return s + "\n" + "C'est une voiture " + this.couleur + " !";
  }
  public static void main(String[] args) {
    Vehicule v = new Voiture("rouge");
    System.out.println(v.toString());
  }
}
```

Attention : l'héritage demande de bien réfléchir à l'organisation des différentes classes. Par exemple, comment devrait-on s'y prendre pour définir une classe `Velo` ? Et une classe `VoitureVolante` ?

D'autre part, grâce à l'héritage, un objet peut sembler appartenir à plusieurs classes différentes. Par exemple, une `Voiture` pourra également être considérée comme un simple `Véhicule` si on le souhaite, alors que la réciproque est fausse : lorsque l'on crée un `Vehicule`, on ne pourra pas le manipuler en tant que `Voiture`. Concrètement, cela signifie que toute voiture est un véhicule, mais que tout véhicule n'est pas nécesasirement une voiture. Or, puisque certains véhicules sont quand même des voitures, il faut bien qu'on puisse tester si, oui ou non, un véhicule donné est une voiture ou pas !

C'est le rôle de `instanceof`. Par exemple, le code

```
public class Test {
  public static void main(String[] args) {
    Voiture v1 = new Voiture("rouge");
    Vehicule v2 = new Voiture("bleu");
    Vehicule v3 = new Vehicule("vert");
    System.out.println(v1 instanceof Voiture);
    System.out.println(v2 instanceof Voiture);
    System.out.println(v3 instanceof Voiture);
    System.out.println(v1 instanceof Vehicule);
    System.out.println(v2 instanceof Vehicule);
    System.out.println(v3 instanceof Vehicule);
  }
}
```

aura pour résultat l'affichage des lignes suivantes :

```
true
true
false
true
true
true
```

Exercice 1 (Jardin Zoologique)

Nous allons travailler à modélisation d'un jardin zoologique et des animaux qui l'occupent. Les animaux seront tout d'abord représentés par la classe suivante, qui sera amenée à évoluer par la suite :

```
public class Animal {
  private final String nom;
  private final String espece;
```

```
  public Animal(String nom, String espece) {
    this.nom = nom;
    this.espece = espece;
  }
  public String getNom() {
    return nom;
  }
  public String getEspece() {
    return espece;
  }
}
```

Créer une arborescence de classes incluant la classe `Animal` et de nouvelles classes `Mammifere`, `Fauve`, `Oiseau`, `Lion` et `Aigle`.

Exercice 2 (Conversion de type)

Parmi les commandes suivantes, lesquelles causeront une erreur au moment de la compilation ? Au moment de l'exécution ?

```
Animal a = new Mammifere("Baloo", "ours");
Mammifere b = new Animal("Scar", "lion");
Fauve c = new Lion("Simba");
Fauve d = new Fauve("Shere Khan", "tigre");
Fauve e = new Fauve("Willy", "orque");
Lion f = c;
Lion g = (Lion) c;
Animal h = new Animal("Mufasa", "lion");
Lion i = (Lion) h;
```

Exercice 3 (Nourriture)

On souhaite maintenant nourrir les animaux. Par défaut, on supposera que les animaux peuvent manger de la viande et des végétaux ; cependant, les aigles et les fauves se nourrissent exclusivement de viande.

1. On va donc ajouter une méthode `public boolean carnivore()` qui répond `true` si l'animal considéré se nourrit exclusivement de viande, et `false` sinon.
   Dans quelle(s) classe(s) ajouter cette méthode ? Quel doit être le code correspondant ?
2. On veut ensuite ajouter une méthode `public boolean nourrirAvecViande(boolean viande)`, qui simule le fait que l'on donne à manger à l'animal : on exécute `nourrirAvecViande(true)` si on lui donne de la viande, et `nourrirAvecViande(false)` si on lui donne des végétaux ; la méthode doit renvoyer `true` si l'animal accepte de manger sa nourriture, et `false` sinon.
   Dans quelle(s) classe(s) ajouter cette méthode ? Quel doit être le code correspondant ?

Exercice 4 (Publicité)

Désireux de faire de la publicité, le zoo souhaite acquérir des panthères noires. On souhaite, pour cette question seulement, ajouter aux classes déjà définies les classes `Panthere` et `PanthereNoire`.

1. Comment intégrer ces classes à la hiérarchie de classes préexistante ; en d'autres termes, quelles classes doivent hériter de quelles autres ?
2. Ajouter une troisième classe additionnelle `AnimalNoir` serait-il une bonne idée ? Pourquoi ?
3. Écrire une méthode `public static int comptePantheresNoires(Animal[] zoo)` qui prend un tableau d'animaux en argument et retourne le nombre de panthères noires contenues dans le tableau.

Exercice 5 (Âge)

On modélise le vieillissement des animaux comme suit :

— Les aigles sont *enfants* de 0 à 6 ans, puis *adultes* de 7 à 25 ans, et meurent le jour de leurs 26 ans.
— Les oiseaux autres que les aigles sont *enfants* de 0 à 6 ans, puis *adultes* de 7 à 18 ans, et meurent le jour de leurs 19 ans.
— Les lions sont *enfants* de 0 à 3 ans, puis *adultes* de 4 à 14 ans, et meurent le jour de leurs 15 ans.
— Les fauves autres que les lions sont *enfants* de 0 à 3 ans, puis *adultes* de 4 à 18 ans, et meurent le jour de leurs 19 ans.
— Les mammifères autres que les fauves sont *enfants* de 0 à 5 ans, puis *adultes* de 6 à 25 ans, et meurent le jour de leurs 26 ans.
— Les animaux autres que les mammifères et les oiseaux sont *enfants* de 0 à 8 ans, puis *adultes* de 9 à 40 ans, et meurent le jour de leurs 41 ans.

1. Pour implémenter la modélisation ci-dessus, on souhaite ajouter des attributs `int age` (qui indique l'âge de l'animal), `int ageAdulte` (qui indique l'âge auquel l'animal devient adulte) et `int ageMort` (qui indique l'âge auquel l'animal meurt). Comment modifier les constructeurs de manière adéquate ?
2. On souhaite ajouter plusieurs méthodes : `public boolean estEnfant()`, `public boolean estAdulte()`, `public boolean estMort()` et `public void anniversaire()` : dans quelles classes les ajouter, et quel doit être le code correspondant ?
3. Un zoo pouvant être vu comme un `Animal[]`, on souhaite fêter l'anniversaire de tous les animaux du zoo : créer une nouvelle méthode `public static void anniversaires(Animal[] zoo)` qui simule le vieillissement de tous les animaux d'un an et, le cas échéant, supprime de `zoo` les animaux morts de vieillesse.

Exercice 6 (Sexe et reproduction)

Chez la plupart des animaux, on garde son sexe (mâle ou femelle) toute sa vie :

1. Quels atrributs et/ou quelles méthodes ajouter ? Comment modifier les constructeurs de manière adéquate ?

En outre, deux adultes de la même espèce et de sexes opposés peuvent se reproduire. L'enfant obtenu sera bien évidemment de la même espèce, et vous pourrez choisir son nom comme bon vous semble.

2. Écrire une méthode `public Animal reproduction(Animal partenaire)` qui, si l'animal courant et son `partenaire` peuvent procréer, renvoie leur rejeton, dont le sexe aura été choisi aléatoirement. Cette méthode renverra `null` si les deux animaux ne peuvent pas procréer.
3. Si les partenaires considérés appartiennent à une sous-classe de `Animal`, comment pourrait-on modifier la méthode ci-dessus afin que le bébé obtenu appartienne à la classe la plus précise possible (c'est-à-dire que, si un `Mammifere` et un `Fauve` procréent, alors tous deux représentaient des fauves, donc leur enfant sera un `Fauve`) ?