Université Paul Sabatier – Toulouse III
IUT A - Toulouse Rangueil

Concepts de la programmation par objet

8 février 2012
Semestre 3

TD n° 1 — Au zoo

1 Le perroquet

```
class Animal {
  private String nom;
  private Int age;

  public Animal(String p_nom, Int p_age){
    this.nom = p_nom;
    this.age = p_age;
  }

  public void dormir(){
  }

  public void manger(){
  }

  public String getNom(){
    return this.nom;
  }

  public Int getAge(){
    return this.age;
  }

  public void setAge(Int a){
    this.age = a;
  }

}
```

Listing 1 – Classe Animal

```
class Oiseau extends Animal {
  public Oiseau(String p_nom, Int p_age){
    super(p_nom, p_age);
  }
  public void voler(){
  }
}
```

Listing 2 – Classe Oiseau

```java
class Perroquet extends Oiseau {
  private Proprietaire proprietaire;

  public Perroquet(String p_nom, int p_age, Proprietaire p_proprietaire){
    super(p_nom, p_age);
    this.proprietaire = p_proprietaire;
  }
  public void parler () {
  }

  public Proprietaire getProprietaire(){
    return (this.proprietaire);
  }

  public void setProprietaire(Proprietaire p_proprietaire){
    this.proprietaire = p_proprietaire;
  }
}
```

Listing 3 – Classe Perroquet

```java
class Proprietaire{
  private String nom;
  private String prenom;
  private String adresse;
  private Int age;

  public Proprietaire (String p_nom, String p_prenom, String p_adresse, Int p_age)
     {
    this.nom = p_nom;
    this.prenom= p_prenom;
    this.adresse= p_adresse;
    this.age= p_age;
  }
}
```

Listing 4 – Classe Propriétaire

1-4 En faisant hériter Perroquet de Oiseau, et faire hériter Oiseau de Animals

```java
class main {
  public static void main(String args){
    Proprietaire leProprio = new Proprietaire("Churchill", "Winston", "", "");
    Perroquet monPerroquet = new Perroquet(Charline, 10, leProprio);
    monPerroquet.manger();
    monPerroquet.parler();
  }
}
```

Listing 5 – Instanciation

2 Le koala

```
class Marsupial extends Animal {

  public Marsupial(String p_nom, Int p_age){
    super(p_nom, p_age);
  }

  public void manger(){
    this.mangerHerbe();
  }

  private void mangerHerbe(){
  }
}
```

Listing 6 – Classe Marsupial

```
class Koala extends Marsupial {

  public Koala(String p_nom, Int p_age){
    super(p_nom, p_age);
  }

  public void manger(){
    this.mangerEucalyptus();
  }

  private void mangerEucalyptus(){
  }
}
```

Listing 7 – Classe Koala

3 Les anguilles

```
class Ovipare extends Animal{
  public void pondre(){
  }

}
```

Listing 8 – Classe Ovipare

```
class Poisson extends Ovipare {
  public Poisson(String p_nom, Int p_age){
    super(p_nom, p_age);
  }
}
```

Listing 9 – Classe Poisson

```
class Anguille extends Ovipare, AnimalMarin, AnimalEauDouce {
  public Anguille(String p_nom, Int p_age){
    super(p_nom, p_age);
  }

  public void pondre(){
    this.allerDansMerDesSargasses();
    super.pondre();
  }

  private void allerDansMerDesSargasses() {
  }
}
```

Listing 10 – Classe Anguille

```
class AnimalEauDouce extends Animal

}
```

Listing 11 – Classe AnimalEauDouce

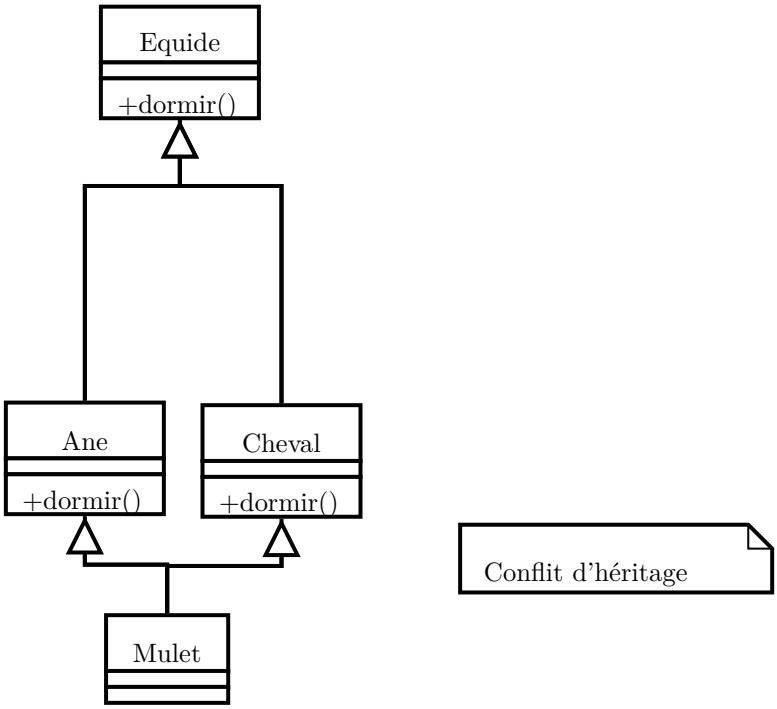
```
class animalMarin extends Animal {

}
```

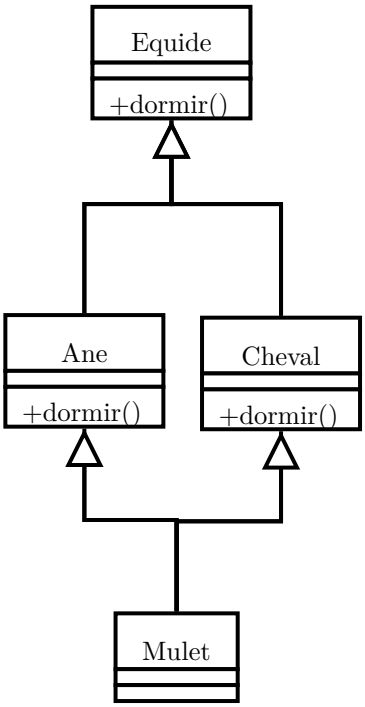
Listing 12 – Classe AnimalMarin

4 Le mulet – Conflit en héritage multiple

Deux méthodes pour résoudre le conflit :
- Parcours du graphe d'héritage (smalltalk par exemple)
- Annotations du programmeur (C++ par exemple)



4.1 Dormir spécifique à tout équidé



Méthode retardée

```
1 abstract class Equide extends Animal{
2   abstract public void dormir(){ }
3 }
```

Listing 13 – Classe Equide

```java
class Cheval extends Equide {
  public void dormir(){
    this.dormirDansHaras();
  }
  private void dormirDansHaras(){
  }
}
```

Listing 14 – Classe Cheval

```java
class Ane extends Equide{
  public void dormir(){
  }
}
```

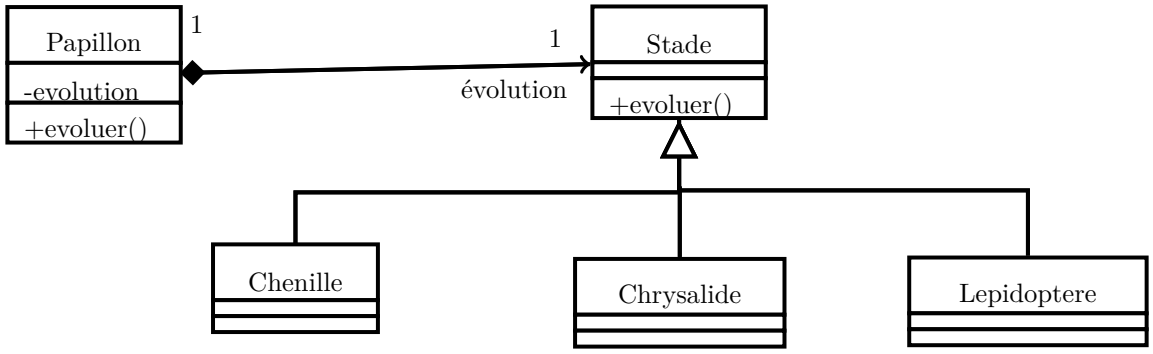
Listing 15 – Classe Ane

```java
class Mulet extends Anne, Cheval{


}
```

Listing 16 – Classe Mulet

5 Le papillon




```java
class Papillon extends Animal{
  private Stade evolution;

  public Papillon(){
    this.evolution = new Chenille();
  }
  public void evoluer(){
    this.setEvolution(this.evolution.evoluer());
  }
  private void setEvolution(Stade p_stade){
    this.evolution = p_stade;
  }
  public Stade getEvolution(){
    return (this.evolution);
  }
}
```

Listing 17 – Classe Papillon

```java
abstract class Stade {
  public abstract Stade evoluer(){
  }
}
```

Listing 18 – Classe Stade

```java
class Chenille extends Stade{
  public Stade evoluer(){
    return (new Chrysalide());
  }
}
```

Listing 19 – Classe Chenille

```
Class Chrysalide extends Stade {
  public Stade evoluer(){
    return (new Lepidoptere);
  }
}
```

Listing 20 – Classe Chrysalide

```
class Lepidoptere extends Stade {
  public Stade evoluer(){
    return (this);
  }
}
```

Listing 21 – Classe Lepidoptere

6 Le zoo

```
interface Iterateur <T> {
  // Constructeur : prend en compte les éléments présents dans la collection
  Iterateur<T> (Collection <T> c);

  // Se positionne sur le premier élément de la collection
  void premier();

  //passe à l'élément suivant de la collection
  void suivant ();

  //retourne l'élément courant de la collection
  T valeur ();

  // TRUE lorsque tous les élément de la collection ont été visités
  boolean termine();
}
```

Listing 22 – Interface Iterateur

```
public void endormirAnimaux(Ensemble <Animal> p_leZoo){
  // déclancher un objet Iterateur
  Iterateur <Animal> visite = new Iterateur <Animal>(p_leZoo);
  // parcours séquentiel de l'ensemble leZoo
  // et endormissement de chacun des animaux

  for(visite.premier() ; visite.termine() ; visite.suivant()) {
    visite.valeur().dormir();
  }
}
```

Listing 23 – Méthode endormirAnimaux

Remarque Importante

- Le code est indépendant du nombre d'animaux du zoo
- Le code est indépendant du nombre d'espèce du zoo
- Le code est indépendant d'ajout ou de retrait de nouvelle espèce
- Le code est indépendant de la représentation du zoo (ici un ensemble)

7 Le logis des animaux

```
class Animal {
  private Habitat logis;

  public Animal(Habitat p_habitat){
    this.logis = p_habitat;
  }
}
```

Listing 24 – Constructeur Animal

```
class Ours extends Animal {
  public Ours (Habitat p_habitat){
    super(p_habitat);
  }
}
```

Listing 25 – Classe Ours

```
class OursBrun extends Ours {
  public OursBrun (Taniere p_taniere){
    super(p_taniere);
  }
}
```

Listing 26 – Classe OursBrun

```
class OursBlanc extends Ours {
  public OursBlanc(Banquise p_banquise){
    super(p_banquise);
  }
}
```

Listing 27 – Classe OursBlanc

```
Habitat banquise;
banquise = new Banquise();
OursBlanc knut;
knut = new OursBlanc(banquise);
```

Listing 28 – Knut est un ours blanc du zoo