Patrick GIRARD
University of Poitiers

Object-Oriented Programming in Java™

Advanced Inheritance

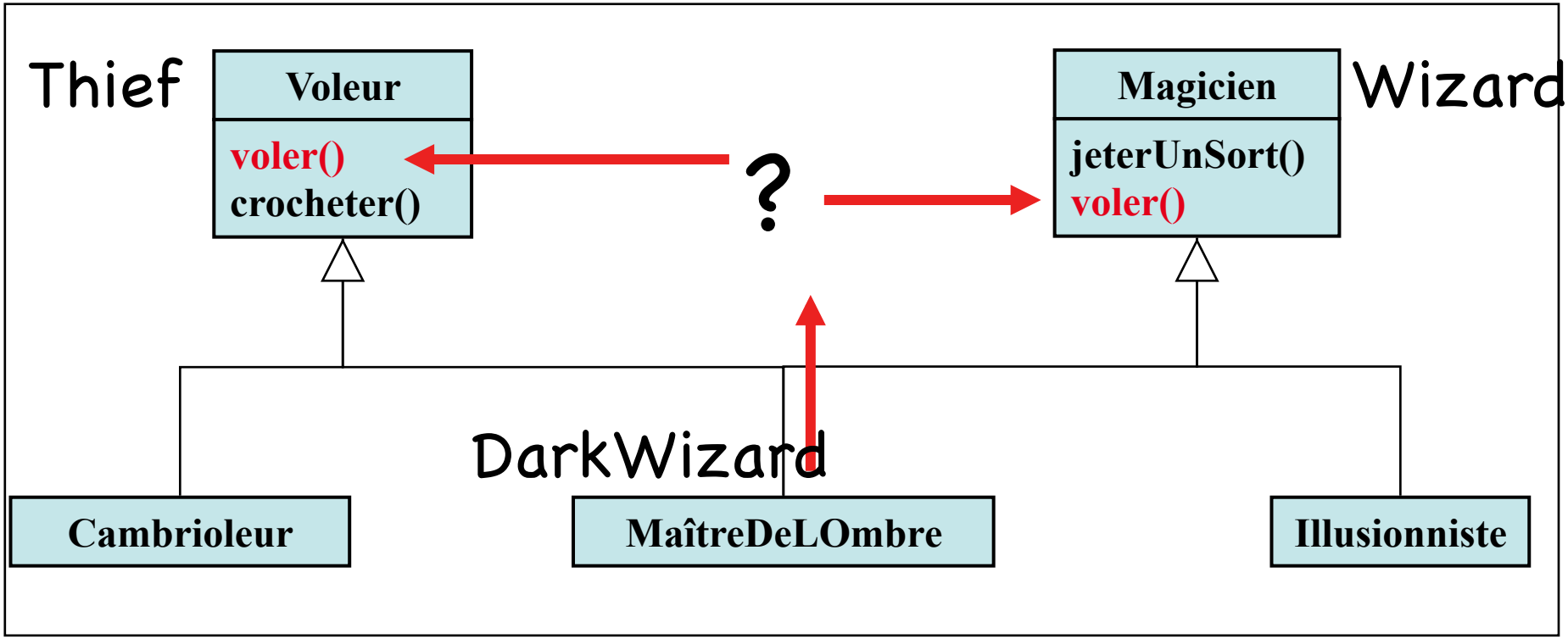
Chapter 6 - Section 4

Table of contents

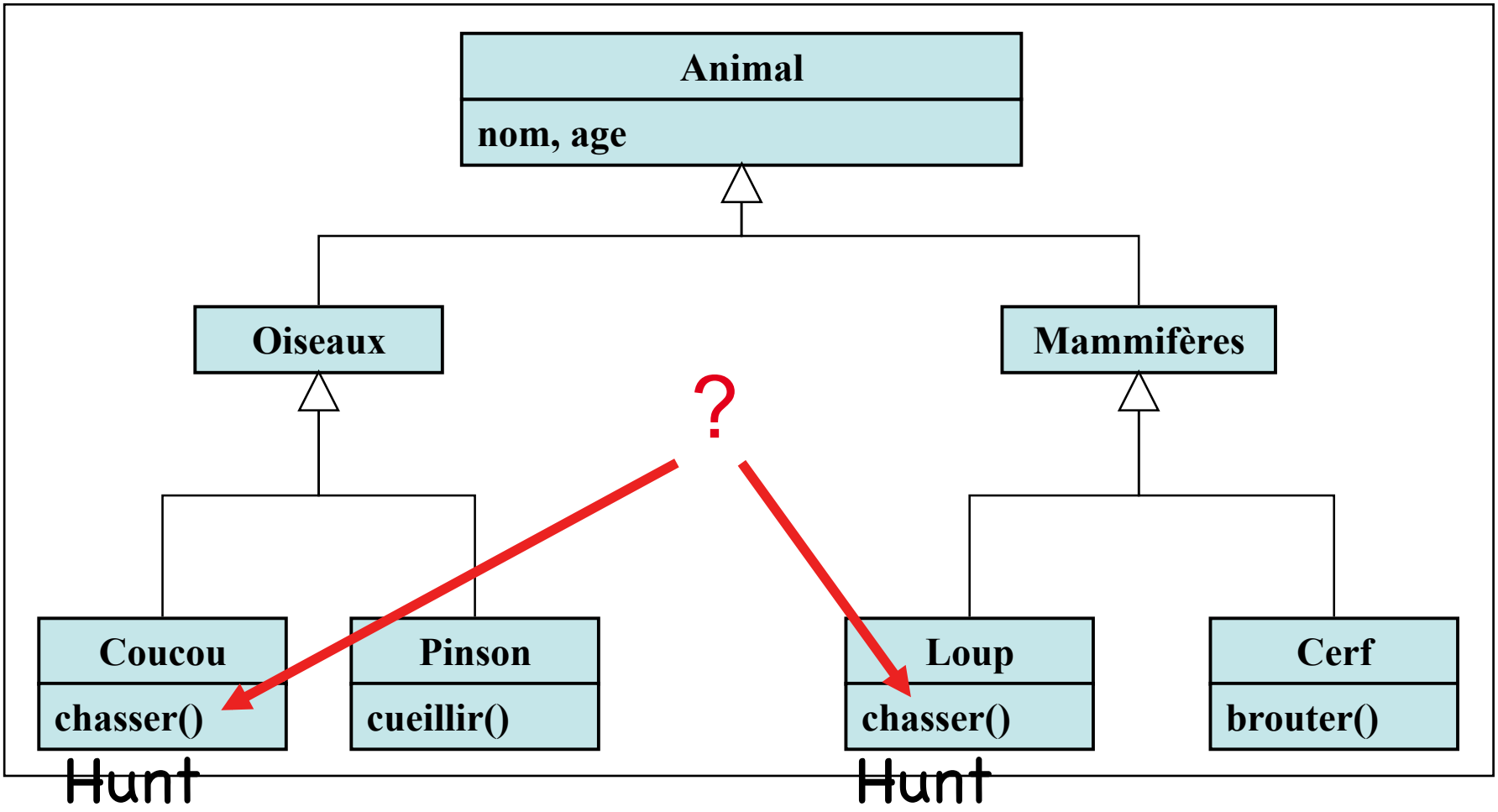
Multiple inheritance, the problem

- Vehicule
  - Ground Vehicle
    - Car
    - Motorcycle
  - Aquatic Vehicle
    - Barge
    - Yacht
- and the hydroplane?
- In Java, multiple inheritance is forbidden

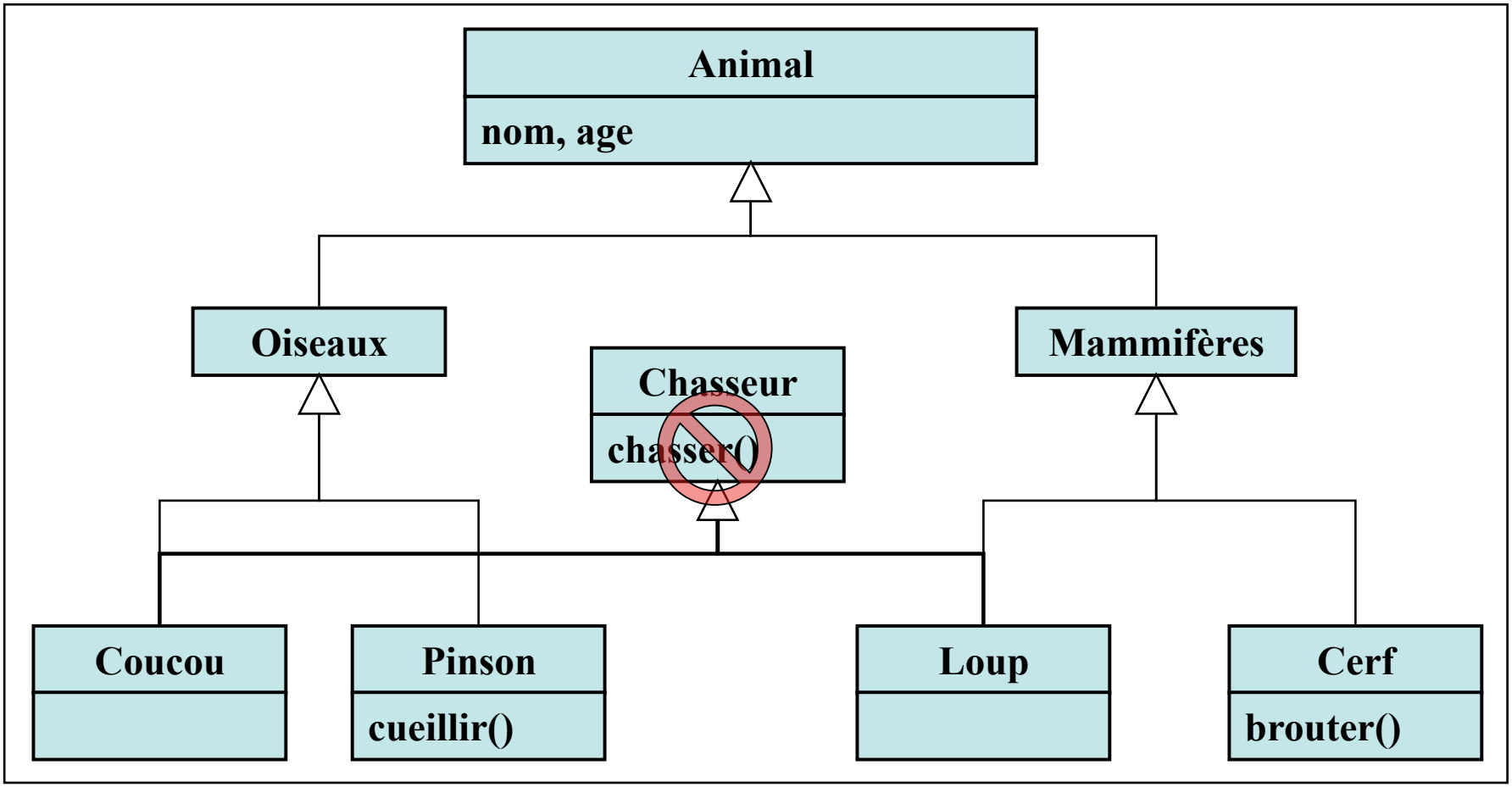
The problem of multiple inheritance



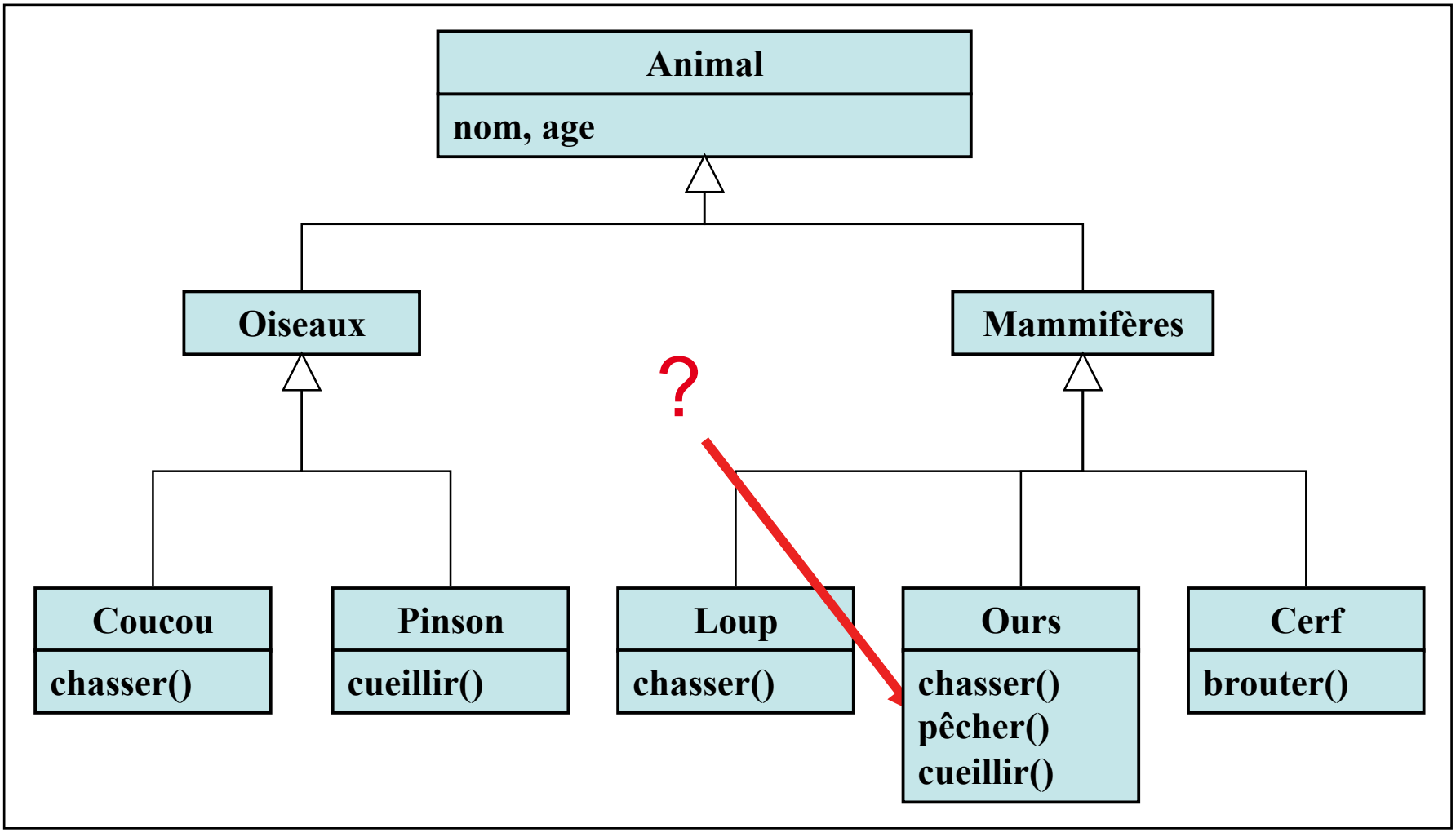
Another example: SimFaune



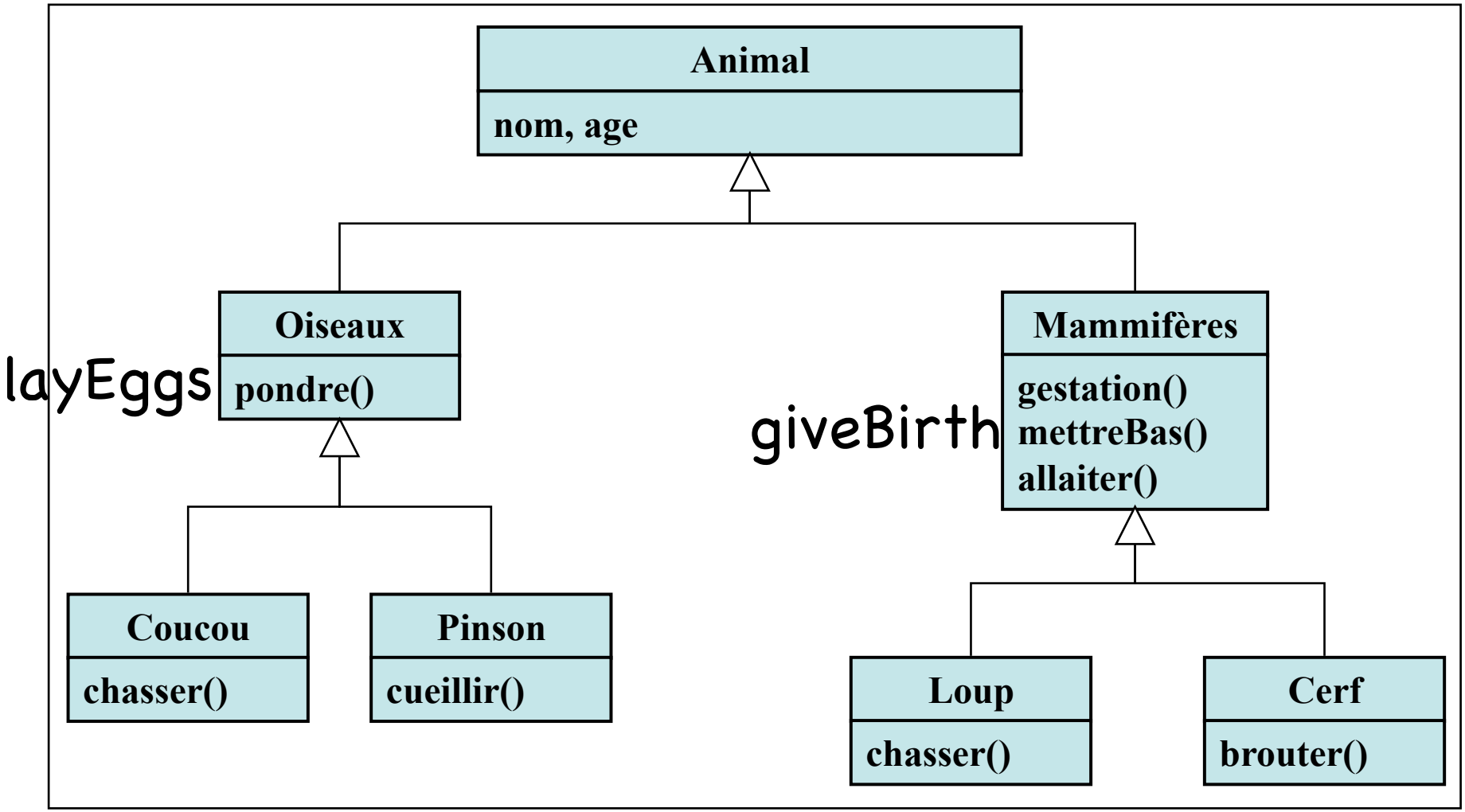
Another example: SimFaune



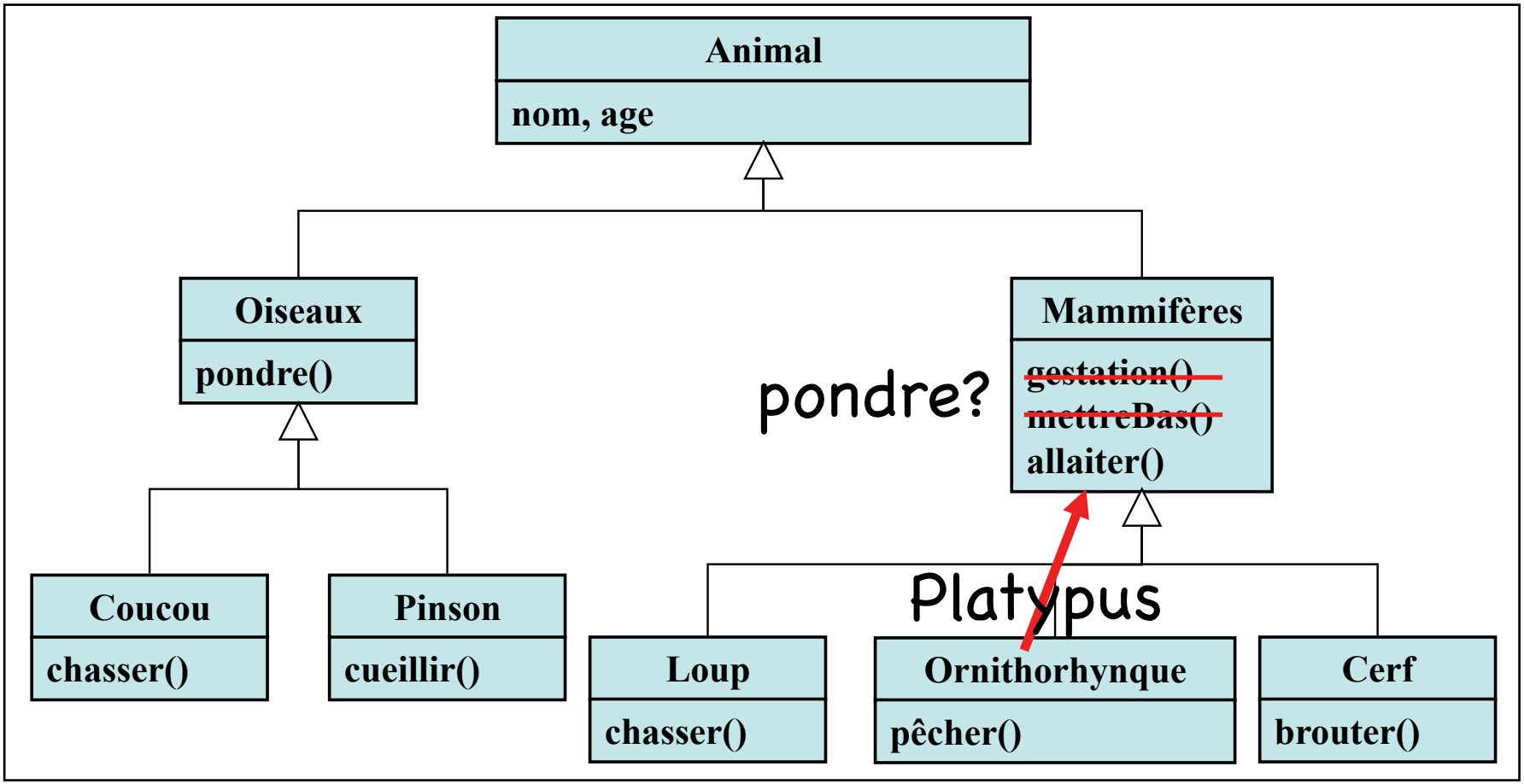
Another example: SimFaune

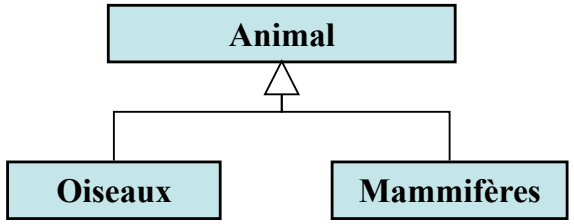


Another example: SimFaune



Another example: SimFaune





Interfaces in SimFaune

```
public class Animal {
   private int anneeNaissance;
   private String Nom;
   public Animal (String n, int annee) {
      Nom = n;
      anneeNaissance = annee;
   }
   public String nom () {return Nom; }
   public int age(int ac) {return ac - anneeNaissance; }
}
```

!

```
public class Mammifere extends Animal {
   public Mammifere (String n, int a) {
      super(n, a);
   };
   public void allaiter() {... };
}
```

!!

!!!

Interfaces: what do we need?

- Grouping function in a logical way
  - Oviparous, Viviparous,
  - Hunter, Fisher
- Allowing classes to provide these functions out of the inheritance tree
  - A Platypus (Ornithorhynque) is at the same time
    - Mammal
    - Oviparous
    - Fisher

Interface

Interface: definition

- Interfaces are pseudo-classes with only abstract methods
- Interfaces are posted out of the class inheritance tree
- The keyword is interface
- The implements mechanism

Interface: example

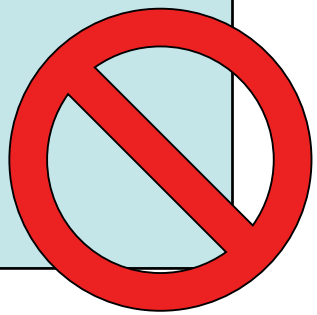
```
public interface Vivipare {
   public void gestation();
   public Portee mettreBas();
}
```

```
import java.util.Vector;
public class Portee {
   private Vector LaPortee = new Vector();

   public void add(Vivipare petit) {
      LaPortee.addElement(petit);}
   public int getSize() { return LaPortee.size(); }
   public Vivipare getPetit(int numero) {
      return (Vivipare) LaPortee.elementAt(numero - 1); }
}
```

Interface: usage

```
public class BadInterfaceUse {
   Vivipare v ;
   public BadInterfaceUse () {
      v = new Vivipare();
   }
}
```

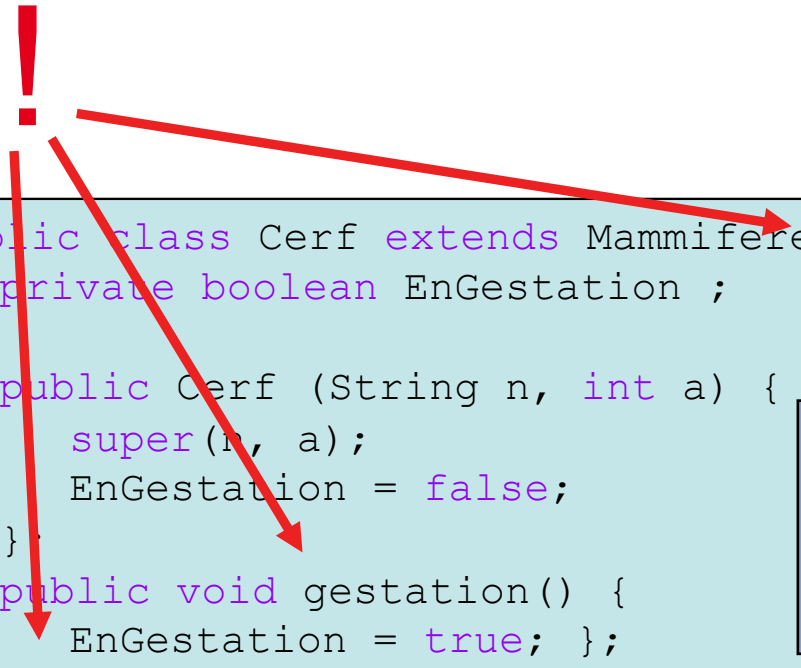


```
Vivipare is abstract; cannot be instantiated
            v = new Vivipare();
                ^
1 error
```

Interface: usage

- Interfaces are "models" of classes, more precisely "models of services"
- Class must explicitly declare the interfaces they implement
  - For example, Deers (Cerfs) are viviparous mammals
    - see next slide...

Interface: usage



```
public class Cerf extends Mammifere implements Vivipare {
   private boolean EnGestation ;

   public Cerf (String n, int a) {
      super(n, a);
      EnGestation = false;
   };
   public void gestation() {
      EnGestation = true; };
   public Portee mettreBas(int annee) {
      Portee portee = new Portee();
      EnGestation = false;
      for (int i=1; i<=4; i++)
         portee.add (
               new Cerf(this.nom() + "Jr" + (i), annee));
      return portee;
   };
```

```
public interface Vivipare {
   public void gestation();
   public Portee mettreBas();
}
```

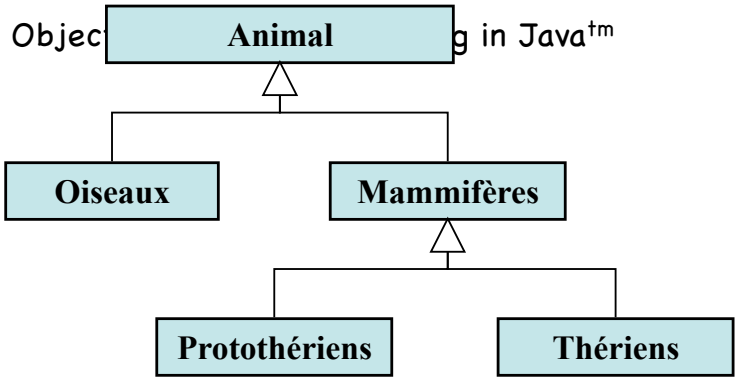
Interface: usage

```
    public String toString() {
        return nom() + " (" + age(2005) + "ans)";
    }
    public static void main(String args[]){
        Cerf c = new Cerf("Bambi", 1997);
        System.out.println ( c +
                " a eu les petits suivants : " +
                c.mettreBas(2003));
    }
}
```

```
Bambi (8ans) a eu les petits suivants : BambiJr1 (2ans),
BambiJr2 (2ans), BambiJr3 (2ans), BambiJr4 (2ans),
```

Interface: usage

- All services must be implemented...
- Example
  - a Therian is a viviparous mammal (opposed to Protoherians, which are oviparous)...



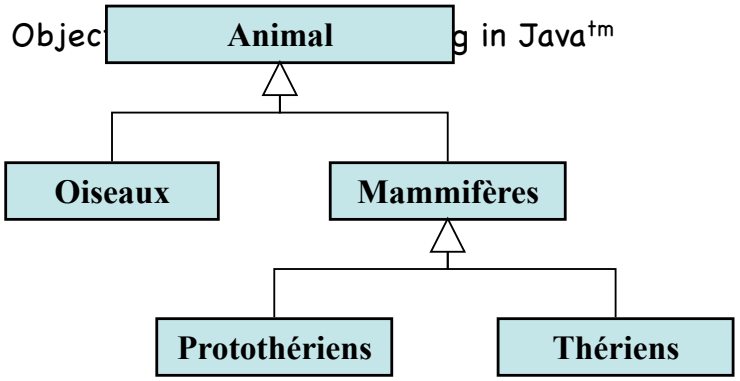
Interface: usage

```
public abstract class Therien
      extends Mammifere
      implements Vivipare {
   private boolean EnGestation ;
   public Therien (String n, int a) {
      super(n, a);
      EnGestation = false; };
   public void gestation() {
      EnGestation = true; };
   protected Portee mettreBas(){
      EnGestation = false;
      return new Portee();
};} }
```

!

!!

!!!



Interface: usage

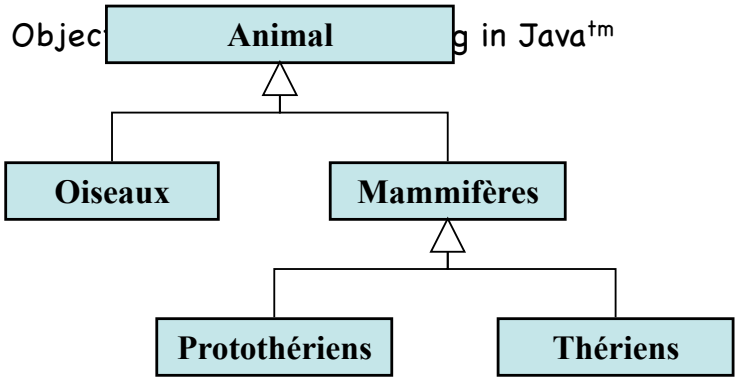
```
public class Loup extends Therien {
   public Loup (String n, int a) {
      super(n, a); };
   public Portee mettreBas(int annee) {
      Portee portee = super.mettreBas();
      for (int i=1; i<=4; i++)
         portee.add (
               new Loup(this.nom() + "Numero" + (i), annee));
         return portee; };
   public static void main(String args[]){
      Vivipare v = new Loup("Wolff", 2000);
      System.out.println (
               ((Animal)v).nom() +
                     " a eu " + v.mettreBas(2005));}
}
```

!

!!

!!!

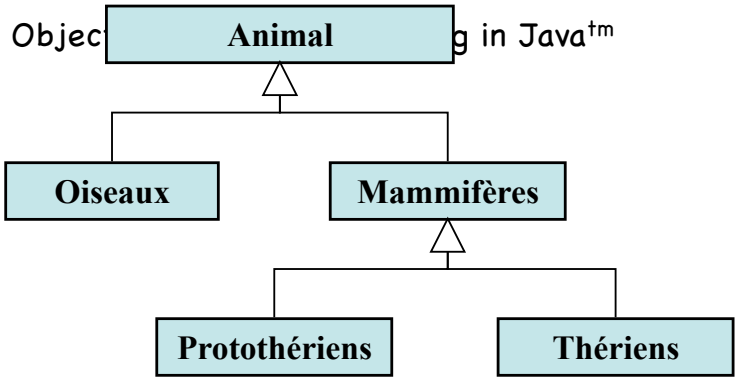
Wolff a eu Portee@be2358



Interface: usage

```
public class Ours extends Therien
   implements Chasseur, Pecheur, Cueilleur {

public Ours (String n, int a) {
   super(n, a);
};
public void Cueillir() {
   System.out.println (this.nom() + " cueille des baies");
}
public void Pecher() {
   System.out.println (this.nom() + " peche des poissons");
}
public void Chasser() {
   System.out.println (this.nom() + " chasse");
}
...
```



Interface: usage

```
...
   public Portee mettreBas(int annee) {
      Portee portee = super.mettreBas();
      for (int i=1; i<=2; i++)portee.add (new Ours(
              this.nom() + "Numero" + (i), annee));
      return portee;
   };
   public static void main(String args[]){
      Ours o = new Ours("Winnie", 2000);
      Vivipare v = o;
      System.out.println (((Animal)v).nom()
                          + " a eu " + v.mettreBas(2005));
      o.Cueillir();
      o.Pecher();
   }
}
```

!

!!

Interfaces: summary

- Interfaces cannot be instanciated
  - no constructor
- Interface give access to functions
- All functions of interfaces must be implemented
- Interfaces can be derived
  - interface toto extends titi...