COURS JAVA 2

CHARLES 'ARYS' YAICHE

« LE BUT DE L'INFORMATIQUE, C'EST D'AUTOMATISER LES TACHES POUR PAS AVOIR À LES FAIRE NOUS MÊMES. AU FOND, UN INFORMATICIEN C'EST UN PEU UN ARTISAN DU TEMPS »

--- MRMOPI5002

ENCAPSULATION

- En Java, comme dans beaucoup de langages orientés objet, les classes, les attributs et les méthodes bénéficient de niveaux d'accessibilité, qui indiquent dans quelles circonstances on peut accéder à ces éléments.

- Ces niveaux sont au nombre de 4, correspondant à 3 mots-clés utilisés comme modificateurs : private, protected et public. La quatrième possibilité est de ne pas spécifier de modificateur (comportement par défaut).

ENCAPSULATION

- Private :
  - Un attribut ou une méthode private (privée) n'est accessible que de puis l'interieur de la même classe
- Protected :
  - Un attribut ou une méthode protected (protégé) n'est accessible que de puis l'interieur d'une même famille de classe ou à un même package
- Public :
  - Un attribut ou une méthode public (publique) est accessible de partout lol
- Par défaut :
  - Quand l'accessibilité d'un attribut ou d'une méthode n'est pas spécifié, la visibilité est public (publique)

ENCAPSULATION

```java
public class Human {
    public int taille;
    private int moyenneDuCoursDeJava;
    public boolean calvici;

    void Humain(int taille){
        this.taille = taille;
    }

    void setMoyenne_du_cours_de_java(int m){
        this.moyenneDuCoursDeJava = m;
    }


}
```

ANCRAGE (ÉLÉMENTS STATIQUES)

- Lorsque que l'on déclare un membre statique (static), ***placé dans un espace mémoire commun à tous les objets de la classe***. Si un des objets modifie la valeur d'un champ statique (par exemple), tous les objets verront la valeur de ce champ modifiée.
- Dans la mesure où un élément statique ne dépend d'aucune instance de la classe à laquelle il appartient, il est possible de l'appeler avec une syntaxe particulière, sans passer par une instance particulière de la classe.

MOT CLEF FINAL

- Final est un mot clefs qui indique l'immutabilité d'un élément Java (méthodes, attributs, classe)
- Une méthode indiquée comme final ne peut être redéfinie dans une classe dérivée.
- Appliqué à un attribut de type primitif, ce dernier devient une constante.
- Une classe final ne peux pas être dérivée

HÉRITAGE

```java
public class Animal {
    public int nb_legs;
    public String name;

    public Animal(int nb_legs, String name){
        this.nb_legs = nb_legs;
        this.name = name;
    }
    public String getName() {
        return name;
    }
}
```



```java
class Cat extends Animal {
    public Cat(String name) {
        super(4, name);
    }
}

class Dog extends  Animal {
    public Dog(String name) {
        super(4, name);
    }
}

class Snake extends Animal {
    public Snake (String name) {
        super(0, name);
    }
}

class Duck extends Animal {
    public Duck(String name) {
        super(2, name);
    }
}
```

CLASSE ABSTRAITE

- Une classe abstraite se défini par le mot clefs abstract
- Une classe abstraite ne peu pas être instancié
- Une classe abstraite peu définir des méthode abstraite qui devront être réimplémenté dans les classe filles

CLASSE ABSTRAITE

```java
public abstract class Animal {
    public int nb_legs;
    public String name;

    public Animal(int nb_legs, String name){
        this.nb_legs = nb_legs;
        this.name = name;
    }

    public abstract void makeNoise();

    public String getName() {
        return name;
    }
}
```

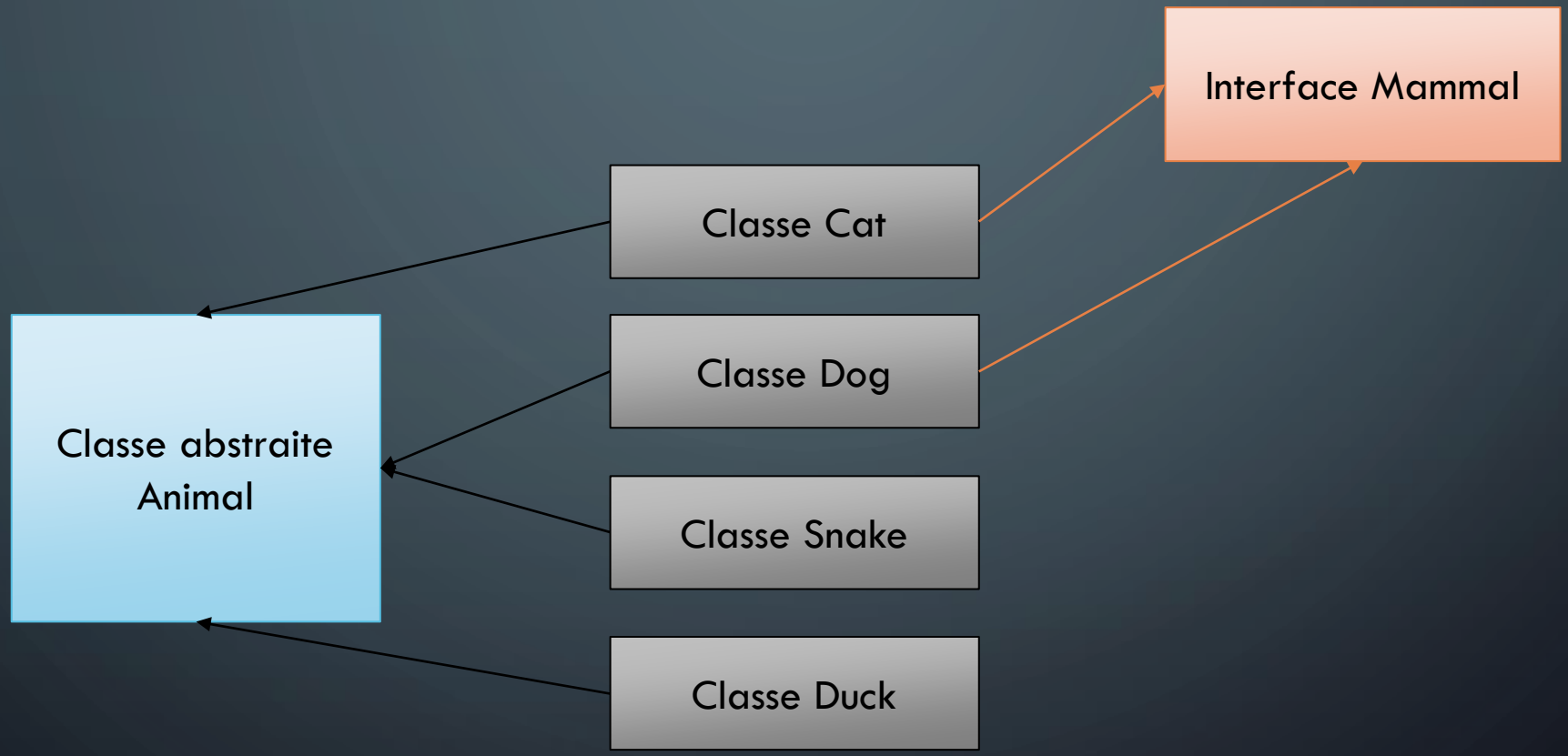
```java
class Cat extends Animal {
    public Cat(String name) {
        super(4, name);
    }

    public void makeNoise() {
        // Miaou Miaou

    }
}

class Dog extends  Animal {
    public Dog(String name) {
        super(4, name);
    }
    public void makeNoise() {
        // Ouaf Ouaf

    }
}

class Snake extends Animal {
    public Snake (String name) {
        super(0, name);
    }
    public void makeNoise() {
        // sssssssSSSSssssSSSssssSSS

    }
}

class Duck extends Animal {
    public Duck(String name) {
        super(2, name);
    }

    public void makeNoise() {
        //coin coin

    }
}
```

INTERFACE

- Une interface est un set de prototype de méthode qui peuvent être implémenté par plusieurs classes

```
interface Mammal {
    public void growFur();
}
```

INTERFACE


Interface Mammal
Classe Cat
Classe Dog
Classe Snake
Classe Duck
Classe abstraite
Animal

INTERFACE

```java
public abstract class Animal {
    public int nb_legs;
    public String name;

    public Animal(int nb_legs, String name){
        this.nb_legs = nb_legs;
        this.name = name;
    }

    public abstract void makeNoise();

    public String getName() {
        return name;
    }
}
```

```java
interface Mammal {
    public void growFur();
}
```

```java
class Cat extends Animal implements Mammal{
    public Cat(String name) {
        super(4, name);
    }

    public void makeNoise() {
        // Miaou Miaou
    }
    public void growFur() { }
}

class Dog extends  Animal implements Mammal{
    public Dog(String name) {
        super(4, name);
    }
    public void makeNoise() {
        // Ouaf Ouaf
    }
    public void growFur() { }
}

class Snake extends Animal {
    public Snake (String name) {
        super(0, name);
    }
    public void makeNoise() {
        // sssssssSSSSssssSSSssSSS

    }
}
```

SOURCE

- Ancrage :
- http://blog.paumard.org/cours/java/chap04-structure-classe-statique.html