# Génie Logiciel TP3 : héritage

Nous allons ici illustrer l'utilisation de l'héritage en C++ sur un exemple simple : reproduire une forme de hiérarchie entre animaux en utilisant une hiérarchie de classe.

Pour bien comprendre les processus de création et destruction des objets dans ce contexte, il sera conseillé d'**afficher un message dans chaque constructeur et destructeur**, comme par exemple `Constructeur Oiseau` pour la classe `Oiseau`.

Ce TP sera utilisé dans un futur TP comme base pour comprendre le **polymorphisme**.

## 1 La classe Vertebre

Pour simplifier, on appellera ici *vertébré* tout animal qui possède des **pattes** et un **pelage**. Nous allons donc créer une classe qui reflète ces attributs : dans des fichiers en-tête (`vertebre.hpp`) et source (`vertebre.cpp`) séparés, créez une classe `Vertebre` contenant :

— un attribut `pattes` contenant le nombre de pattes du vertébré et un attribut `pelage` contenant la description du pelage (poils, plumes, ...) sous forme d'une chaîne de caractère (on pourra utiliser la classe `std::string` de la librairie `<string>`). Ces deux attributs **ne devront pas être publiques** (à vous de voir quelle visibilité est la plus adaptée).

— un constructeur permettant de spécifier ces deux attributs,

— un destructeur, potentiellement vide (sauf pour afficher le message conseillé),

— une méthode **publique** `affiche` qui décrit l'objet dans la sortie standard, du genre : `Vertebre avec 4 pattes et des poils`,

— deux méthodes **publiques** `get_pattes` et `get_pelage` qui retournent respectivement le nombre de pattes et le type de pelage du vertébré.

**Testez** ses fonctionnalités !

## 2 Premier héritage : la classe Oiseau

Nous allons maintenant considérer qu'un oiseau **est un** vertébré dont le seul attribut supplémentaire intéressant est son **nombre d'œufs**. Toujours dans des fichiers séparés (par exemple `oiseau.hpp` et `oiseau.cpp`), écrivez une classe `Oiseau` qui hérite de `Vertebre` (attention à la **visibilité** de cet héritage) et qui contient :

— un attribut **non publique** `oeufs` qui contient le nombre d'œufs de l'oiseau,

— un constructeur permettant d'indiquer le nombre d'œufs et qui initialise les attributs d'un vertébré,

— un destructeur,

— une méthode **publique** `affiche` qui décrit l'objet. Au lieu de réécrire toute la description de l'oiseau, on y utilisera la méthode `affiche` de la classe `Vertebre` pour obtenir un message du genre `Oiseau avec 3 oeufs, qui est un Vertebre avec 2 pattes et des plumes.`

— une méthode **publique** `get_oeufs` qui retourne le nombre d'œufs de l'oiseau.

**Testez** les fonctionnalités, en particulier vérifiez que vous pouvez toujours récupérer le nombre de pattes (ou le pelage) de l'oiseau.

**Observez** l'ordre d'appel aux constructeurs et destructeurs des deux classes.

# 3 Héritage multiple : la classe Volante

On considère maintenant que la description d'un oiseau n'est pas suffisante et qu'on souhaiterait caractériser le fait qu'un oiseau fait partie d'une classe particulière d'animaux, à savoir ceux qui savent voler (comme les insectes, les chauves-souris, ...). Cette classe `Volante` ne sera dotée que d'un seul attribut à savoir le **nombre d'ailes** de l'animal.

Dans un **premier temps**, écrivez une classe `Volante` qui contient :

— un attribut **non publique** `ailes` qui indique le nombre d'ailes de l'animal,

— un constructeur permettant de spécifier cet attribut,

— un destructeur,

— une méthode **publique** `affiche` qui décrit l'objet dans la sortie standard, par exemple `Volante avec 2 ailes`,

— une méthode **publique** `get_ailes` qui retourne le nombre d'ailes de l'animal.

Dans un **second temps**, modifiez la classe `Oiseau` pour qu'elle **hérite également** de la classe `Volante`. En particulier, pensez à :

— modifier son constructeur pour qu'il initialise le nombre d'ailes de l'oiseau,

— modifier sa méthode `affiche` pour décrire son statut d'animal volant en utilisant les méthodes `affiche` de ses deux classes mères `Vertebre` et `Volante`, avec par exemple un message du genre `Oiseau avec 3 oeufs, qui est un Vertebre avec 2 pattes et des plumes, et Volante avec 2 ailes`.

**Testez** les fonctionnalités, en particulier vérifiez que vous pouvez récupérer le nombre d'ailes de l'oiseau. **Observez** l'ordre d'appel aux constructeurs et destructeurs des trois classes impliquées.

# 4 Bonus

Remplacez toutes les méthodes `affiche` par des surcharges de l'opérateur `<<` de telle sorte à pouvoir lancer un code du genre :

```
1  Oiseau bidule(4);
2  Oiseau truc(1);
3  std::cout
4      << "Je possede deux animaux: Bidule qui est un " << bidule
5      << ", ainsi que Truc qui est un " << truc
6      << std::endl;
```