Secteur Tertiaire Informatique

Filière « Etude et développement »

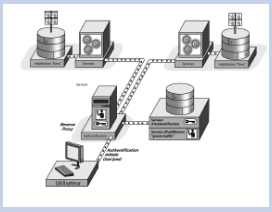
Héritage

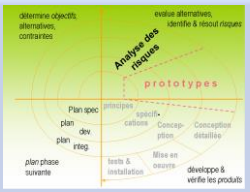
Programmation Orientée Objet

Apprentissage

Mise en situation

Evaluation





1. INTRODUCTION

L'objectif de ce projet est de concevoir un programme en console basé sur une approche objet et permettant de représenter les **classes biologiques** d'animaux.

Ceci vous permettra de travailler les concepts d'héritage.

Vous allez créer plusieurs classes :

- « **Animal** » : représente des animaux
- « **Mammal** » : représente des mammifères terrestres
- « **Bird** » : représente des oiseaux
- « **Fish** » : représente des poissons

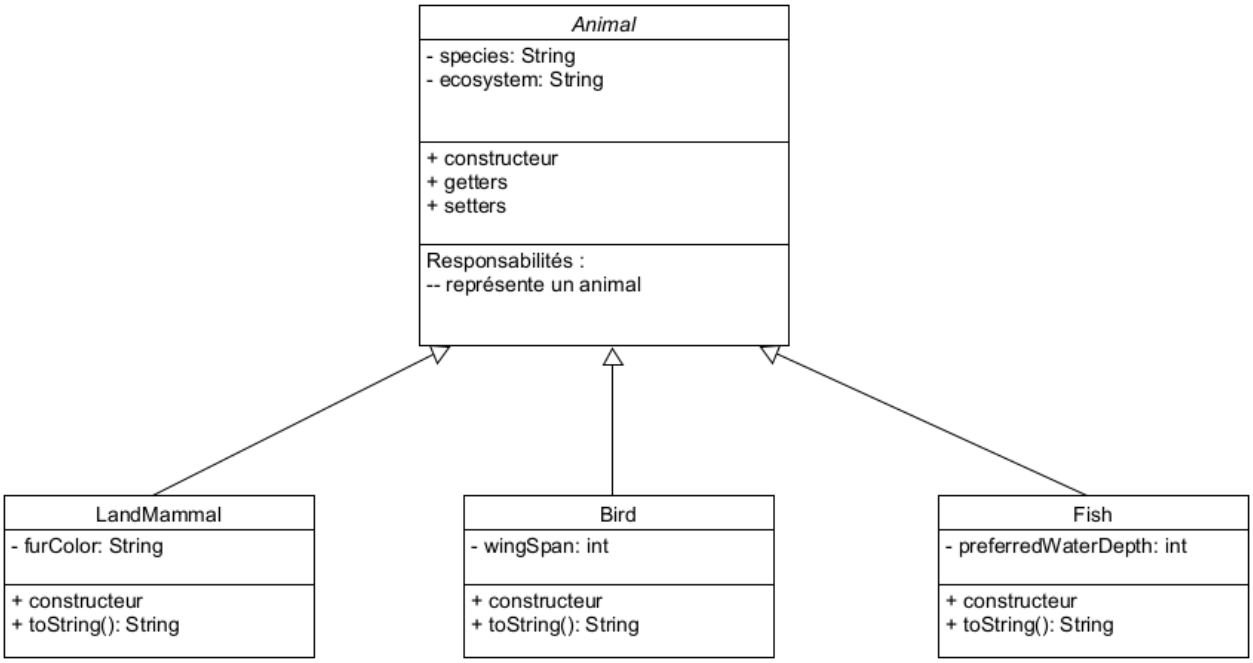
Le code attendu doit suivre les règles syntaxiques et les conventions de nommage du langage Java.

Il vous est demandé de créer un nouveau projet basé sur un archétype Maven existant.

2. IMPLEMENTATION DES CLASSES

2.1 REPRESENTATION UML

L'objectif est d'implémenter toutes les classes mises en avant par le diagramme UML suivant :



Ci-dessous le détail de chaque classe.

2.1.1 Classe « Animal »

Attributs de la classe :

- « **species** », l'espèce en chaîne de caractères (exemple : « Baleine », « Pangolin »…)
- « **ecosystem** », l'écosystème dans lequel l'animal vit (exemple : « Savane », « Fonds marins », « Jungle »)

Méthodes de la classe :

- Constructeur
- getters pour les attributs privés
- setters pour les attributs privés
- toString() qui permet de représenter sous forme de chaîne de caractère un objet de la classe "Animal"

2.1.2 Classe « LandMammal »

Classe représentant un mammifère terrestre, **hérite de la classe « Animal ».**

Nous considèrerons que tout mammifère terrestre est caractérisé par sa fourrure.

Attributs en plus par rapport à la classe mère :

- « **furColor** » : chaîne de caractère qui permet de stocker la couleur de la fourrure en toutes lettres (exemple : « Brun »)

Méthode de la classe :

- Constructeur (qui fera appel au constructeur de la classe mère)
- Getter sur le nouvel attribut privé
- Setter sur le nouvel attribut privé

Ré-implémentation de méthode :

- toString()

2.1.3 Classe « Bird »

Classe représentant un oiseau, **hérite de la classe « Animal ».**

Attributs en plus par rapport à la classe mère :

- « **wingSpan** » : entier stockant **l'envergure de l'oiseau en cm**

Méthode de la classe :

- Constructeur (qui fera appel au constructeur de la classe mère)
- Getter sur le nouvel attribut privé

- Setter sur le nouvel attribut privé

Ré-implémentation de méthode :

- toString()

2.1.4 Classe « Fish »

Classe représentant un poisson, hérite de la classe « Animal ».

Nous considèrerons que tout poisson est caractérisé par une profondeur d'eau dans laquelle il vit en milieu naturel)

Attributs en plus par rapport à la classe mère :

- « **preferredWaterDepth** » : entier qui permet de stocker la profondeur à laquelle le poisson vit en milieu naturel (valeur moyenne en **cm**). Exemple : un poisson rouge pourrait avoir un valeur de 30 cm.

Méthode de la classe :

- Constructeur (qui fera appel au constructeur de la classe mère)
- Getter sur le nouvel attribut privé
- Setter sur le nouvel attribut privé

Ré-implémentation de méthode :

- toString()

Attention

Dans les constructeurs des classes filles il est préconisé d'appeler le constructeur de la classe mère afin d'initialiser les attributs.

Ceci peut être fait en utilisant le mot clef « **super** ».

Lisez l'article suivant pour apprendre à vous servir de « **super** »:
https://codegym.cc/fr/groups/posts/fr.655.super-mot-cle-en-java

A faire

Implémentez toutes les classes détaillées ci-dessus.

Testez votre code en instanciant plusieurs animaux de types différents.

2.2 CLASSE ABSTRAITE

Nous pouvons nous poser des questions de conception lors du développement :

Est-ce qu'il est possible d'avoir un animal qui n'a pas de type particulier (qui n'est ni un mammifère, ni un poisson, ni un oiseau) ?

Autrement dit : est-il pertinent de pouvoir instancier un objet de la classe « Animal » ?

Ne vaudrait-il pas mieux pouvoir instancier uniquement des objets des classes filles ?

C'est pour répondre à ce questionnement qu'il existe le **mécanisme d'abstraction** en programmation Objet.

Info

Un classe déclarée **abstraite** (« abstract ») **ne peut pas être instanciée.**

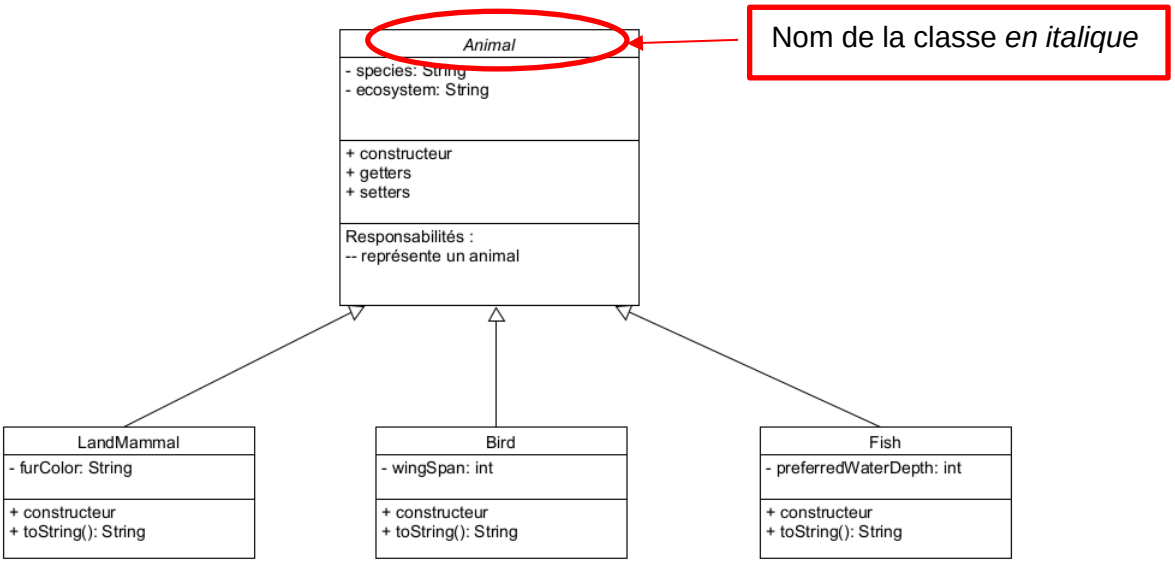
Elle peut, cependant, être **utilisée en tant que classe mère.**

Pour plus d'informations sur les classes abstraites, lisez l'article suivant : https://gayerie.dev/epsi-b3-java/langage_java/classe_abstraite.html

A faire

Déclarez la classe « Animal » en tant que classe abstraite.

En UML la classe abstraite est représentée en mettant son nom en italique :



2.3 AJOUT DE METHODE ABSTRAITE DANS LA CLASSE MERE

Il est possible d'ajouter des méthodes dans la classe mère qui seront héritées dans la classe fille pour être **ré-implémentées.**

Une **méthode abstraite** contenue dans une classe mère peut **simplement être déclarée** et ne pas avoir d'implémentation.

Dans ce cas c'est à la **classe fille d'implémenter la méthode.**

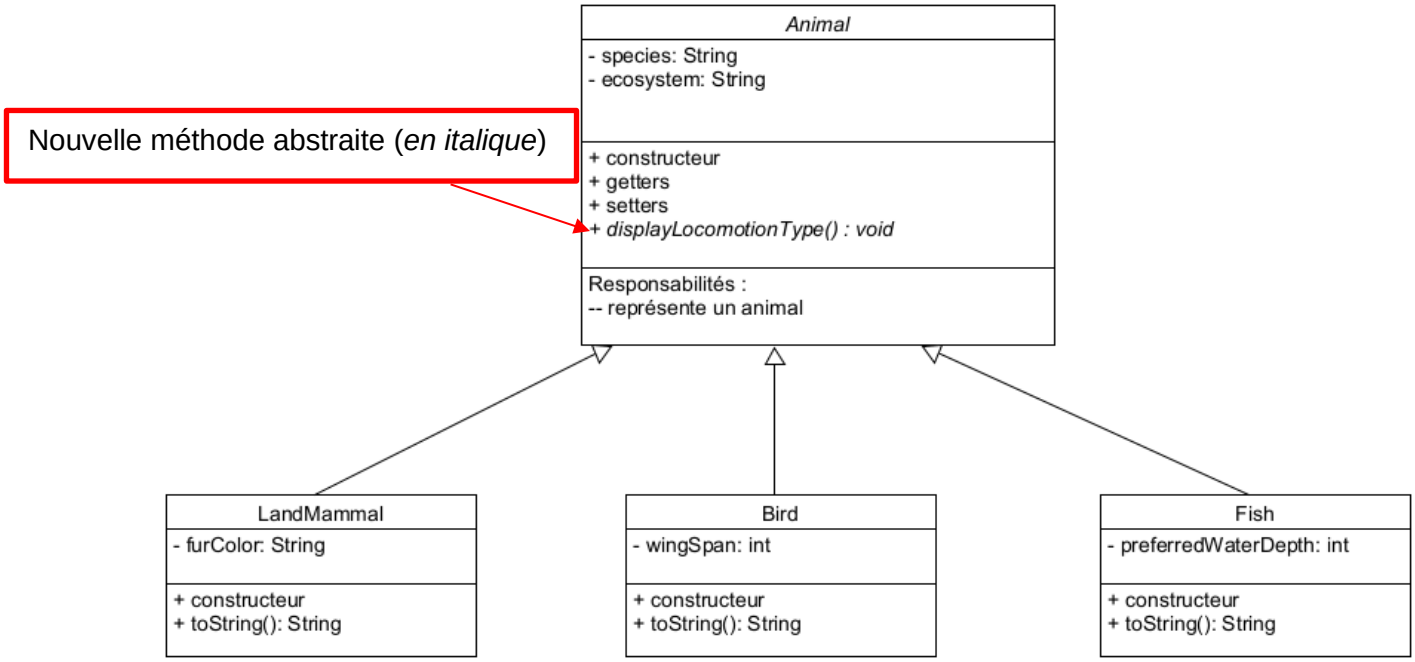
Pour plus d'informations sur les méthodes abstraites vous pouvez lire l'article suivant : https://www.geeksforgeeks.org/abstract-methods-in-java-with-examples/

Il vous est proposé d'ajouter la **méthode abstraite** « **void displayLocomotionType()** » qui a pour rôle d'afficher le type de déplacement le plus commun pour un catégorie d'animal.

Par exemple :

- Pour la classe « LandMammal » la fonction devra afficher « Marche/course/saut »
- Pour la classe « Bird » la fonction devra afficher « Vol »
- Pour la classe « Fish » la fonction devra afficher « Nage »

Le nouveau diagramme UML représentatif de ce qui est demandé est :



A faire

Pour ajouter la méthode abstraite dans la classe mère vous pouvez utiliser la **déclaration suivante** :

```
public abstract void displayLocomotionType();
```

Que se passe-t-il lorsque vous ajoutez cette méthode abstraite ?

Une erreur devrait apparaître, quelle est sa signification ?

Corrigez l'erreur en ré-implémentant la méthode dans les classes filles.

Testez cette nouvelle méthode en l'appelant à partir de votre fonction « main ».

2.4 POLYMORPHISME ET GESTION DE LISTE

2.4.1 Création d'ArrayList

A faire

Dans votre méthode « **main** », ajoutez une « **ArrayList** » qui permet de stocker un ensemble d'animaux de **plusieurs types**.

Commencez par instancier un objet de la classe « **ArrayList** » (information ci-dessous).

Pour rappel, la classe « ArrayList » est une **classe générique**.

Une classe générique est définie comme suit : **une classe dont la définition est paramétrée avec un ou plusieurs types « variables ».**

Voici un exemple d'instanciation d'un « **ArrayList** » qui permettra de stocker des objets de la classe « Animal » :

```
ArrayList<Animal> animals = new ArrayList<Animal>();
```

Il vous est également possible de ne pas mettre le **type générique** (celui situé entre les chevrons du constructeur) :

```
ArrayList<Animal> animals = new ArrayList<>();
```

La classe "**ArrayList**" propose un ensemble de méthodes permettant d'effectuer des opérations sur une liste de données :

- Ajout d'éléments
- Suppression d'éléments
- Accès à des éléments
- Récupération de la taille

Info

La classe « **ArrayList** » est ce que l'on appelle une **collection**. Les classe « collection » permettent de gérer des ensembles d'objets.

Pour une présentation en vidéo détaillée du fonctionnement de la classe vous pouvez consulter le lien suivant : https://www.youtube.com/watch?v=b9fg-2fAVMQ

Une fois une telle liste créée avec il est tout à fait possible d'y ajouter des objets de **toute classe héritant d'animal**.

A faire

Ajoutez les différentes instances d'animaux à une « ArrayList ».

2.4.2 Implémentation de boucle For..each

Pour afficher les différentes informations de la classe vous pouvez utiliser une boucle, par exemple la boucle « **for..each** » qui va permettre de passer en revue chacun des éléments d'une collection.

Pour plus d'informations concernant le boucle « for..each », consultez l'article suivant : https://www.ionos.fr/digitalguide/sites-internet/developpement-web/boucle-for-each-en-java/

A faire

Utilisez une boucle « for..each » pour afficher toutes les informations des différents objets instanciés.

CREDITS

ŒUVRE COLLECTIVE DE l'AFPA
Sous le pilotage de la DIIP et du centre d'ingénierie sectoriel Tertiaire-Services

Equipe de conception (IF, formateur, mediatiseur)
Michel Coulard – Formateur Evry
Chantal Perrachon – IF Neuilly sur Marne

Date de mise à jour : 26/06/2024