ZOO en OOP

jgwacyk

March 2020

1 Introduction

Ce mini-projet va permettre d'apprendre le principe de classe et d'héritage. L'idée est de créer un Zoo en orienté objet à l'aide du langage Python 3. Attention ce document ne contient aucune correction, il permet uniquement de définir les spécifications nécessaires à la création de ce mini projet. Attention les digrammes de classes ne sont pas complets, les getters et les setters ne sont pas présents. De plus certaines fonctionnalités nécessitent peut être d'autre fonctions non affichées afin d'être implémentées.

Les principes du zoo est le suivant:

- Un zoo est défini par un nom et une liste d'enclos.
- Un enclos est défini par un nom et une liste d'animaux.
- Un animal est défini par un nom, des point de vies et des points d'attaques.
- Il y a différent type d'animaux ayant des comportements unique en fonction de leur type.

Voici les différentes évènements du zoo :

- Il peut arriver que tous les animaux d'un enclos "parlent". Chaque type d'animal a son propre cri.
- Il est possible de nourrir tous les animaux d'un enclos, Tous les animaux de cet enclos vont manger et gagner des points de vies.
- Il peut arriver que tous les animaux d'un enclos s'attaquent mutuellement. C'est à dire que chaque animal va attaquer aléatoirement un autre animal de l'enclos.
- Tous les événements peuvent également s'enclencher de manière global dans tous le zoo. Dans ce cas, tous les enclos vont faire l'évènement en même temps.

Cela se traduit par le diagramme de classe de la figure 1.

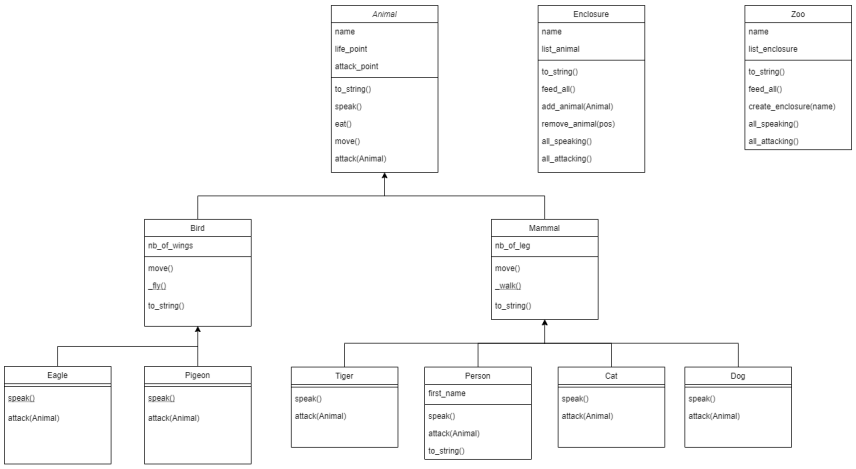

Figure 1: Le zoo

2 Animal

Voici la classe Animal :

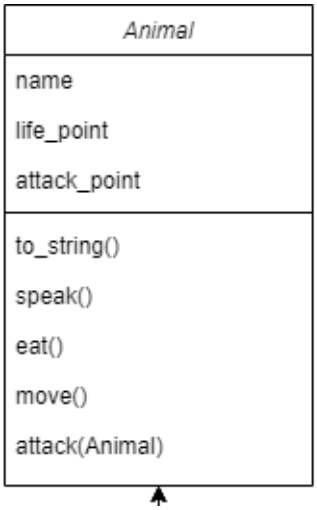

Figure 2: Animal

Cette classe est une classe abstraite permettant de définir un animal. Ses attributs sont :

- name : String, un nom.
- life_point : int, son nombre de point de vie.
- attack_point : int, son nombre de point d'attaque. définit le nombre de point de vie qu'il enlève à un autre animal.

Ses fonctions sont :

- to_string() : String, permet d'avoir une description de l'objet au format String. exemple : "Chat1 a 10 PV et 2 PA".
- speak(), permet d'écrire un "cri" dans le terminal.
- eat(), permet de redonner 5 PV à l'animal.
- move(), permet d'écrire un message signalant que l'animal se déplace dans le terminal.
- attack(Animal), permet d'attaquer un autre animal en lui enlevant un montant de point de vie égale au nombre de point d'attaque de l'animal attaquant.

3 Bird et Mammal

Voici les classe Bird et Mammal :

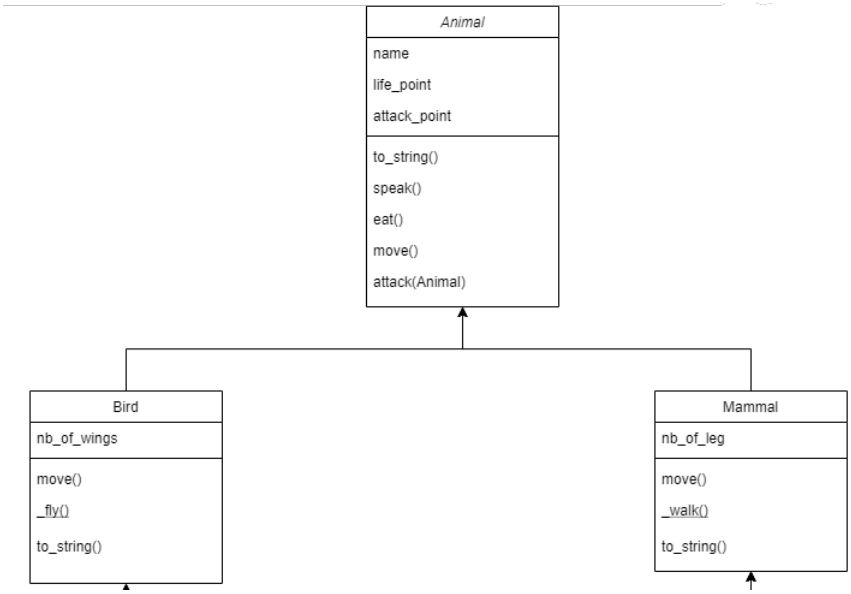

Figure 3: Bird et Mammal

Ces classes sont des classes abstraites héritant de la classe Animal permettant de définir un mammifère et un oiseau.

L'oiseau possède un nouvel attribut :

- nb_of_wings : int, permet de définir combien d'ailes a un oiseau. Il doit être fixé à deux par défaut mais devra être modifiable par son constructeur.

Le mammifère possède un nouvel attribut :

- nb_of_leg : int, qui permet de définir combien de pattes a un mammifère. Il doit être fixé à quatre par défaut mais devra être modifiable par son constructeur.

L'oiseau possède une nouvelle fonction :

- _fly() : String, cette fonction doit être utilisée par la fonction move() afin d'avoir un texte spécifique au vol.

Le mammifère possède une nouvelle fonction :

- _walk() : String, cette fonction doit être utilisée par la fonction move() afin d'avoir un texte spécifique a la marche.

Pour les deux Classes mammifère et oiseau, les fonctions suivantes doivent être redéfinies :

- move(), permet d'écrire un message signalant que l'animal se déplace dans le terminal. L'oiseau doit utiliser _fly() et le mammifère doit utiliser _walk().
- to_string(): String, permet d'avoir une description de l'objet au format String. Cependant, il faut ajouter le nombre d'aile pour l'oiseau et le nombre de patte pour les mammifères dans cette description.

4 Eagle, Pigeon, Tiger, Person, Cat et Dog

Voici les classes Eagle, Pigeon, Tiger, Person, Cat et Dog :

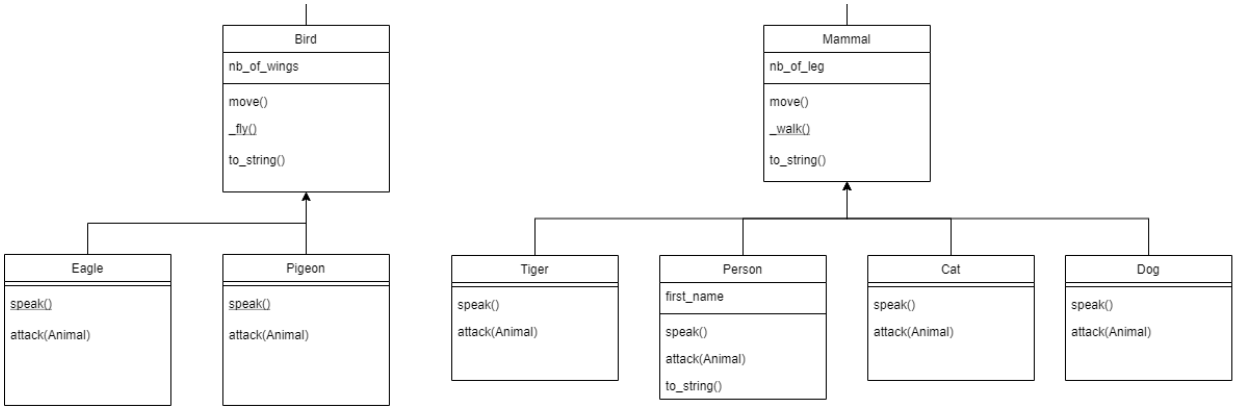

Figure 4: Eagle, Pigeon, Tiger, Person, Cat et Dog

Les classes Eagle et Pigeon hérite de la classe Bird. Les classes Tiger, Person, Cat et Dog héritent de la classes Mammal. Toutes les classes doivent redéfinir la fonction :

- speak(), afin de définir un message différent pour chaque classe.

La classe Person possède un nouvel attribut :

- first_name : String

La classe doit redéfinir la fonction :

- to_string() : String, afin d'ajouter l'attribut first_name à la description.

5 Enclosure

Voici la classe Enclosure :

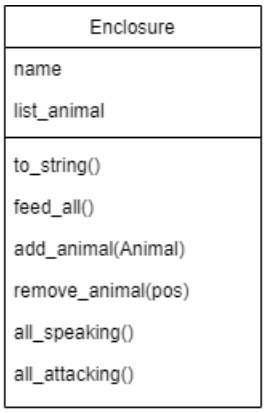

Figure 5: Enclosure

La classe Enclosure, définit un enclos. Ses attributs sont :

- name : String, un nom.
- list_animal : list(Animal), une liste d'animaux que contient l'enclos.

Ses fonctions sont :

- to_string() : String, permet d'avoir une description de l'objet au format String.
  Exemple : "L'enclos appelé prison1 contient :
  - Chat1 a 10 PV et 2 PA qui a 4 pattes
  - Aigle1 a 10 PV et 2 PA qui a 2 ailes"

- feed_all(), permet de nourrir tous les animaux de la liste.
- add_animal(Animal), permet d'ajouter un Animal dans la liste.
- remove_animal(pos), permet de retirer un Animal de la liste en fonction de sa position dans la liste.
- all_speaking(), permet de faire parler tous les animaux de l'enclos.
- all_attacking(), permet de faire attaquer tous les animaux un à un. Un animal va attaquer aléatoirement un autre animal de l'enclos. Si, un animal est à 0 PV ou moins il doit être retiré de la liste.

6 Zoo

Voici la classe Zoo:

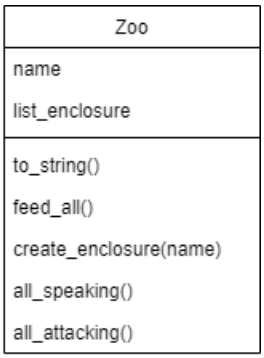

Figure 6: Zoo

La classe Zoo définit un zoo. Ses attributs sont :

- name : String, un nom.
- list_enclosure : list(Enclosure), une liste d'enclos que contient le zoo.

Ses fonctions sont :

- to_string() : String, permet d'avoir une description de l'objet au format String.
  Exemple : "Le zoo appelé Zoomba contient :
  L'enclos appelé prison1 contient :
  - Chat1 a 10 PV et 2 PA qui a 4 pattes
  - Aigle1 a 10 PV et 2 PA qui a 2 ailes
  L'enclos appelé prison2 contient :
  - Chat2 a 10 PV et 2 PA qui a 4 pattes
  - Aigle2 a 10 PV et 2 PA qui a 2 ailes"
- feed_all(), permet de donner de la nourriture dans tous les enclos de la liste.
- create_enclosure(name) : Enclosure, permet de créer un enclos avec le nom name, de l'ajouter dans la liste d'enclos et de le retourner. Si le nom est déjà pris par un autre enclos alors la fonction retourne None.
- all_speaking() : permet de faire parler tous les animaux de tous les enclos.
- all_attacking(), permet de faire attaquer tous les animaux de tous les enclos.

7 Exercice

Afin de faire cette exercice, il est important de faire les questions dans l'ordre. De plus, il vous est demandé de réaliser ce projet sur GitLab. Il est également obligatoire de respecter la Pep8 via pylint et de faire des tests unitaires via pytest pour chaque question.

7.1 Animal

1. Réalisez la classe Animal.
2. Rédigez un main.py dans le quelle vous instanciez deux Animal. Utilisez les fonctions to_string(), speak(), eat() et move() d'un Animal puis attack() avec l'autre Animal. Vérifiez bien les PV des animaux.
3. N'oubliez pas de rédiger les tests unitaires et de respecter la Pep8.
4. Faites un premier tag sur git avec la version V1.00.

7.2 Bird et Mammal

1. Réalisez les Classes Bird et Mammal (conseil : pour la fonction to_string(), vous pouvez utiliser la fonction to_string de la classe Animal dans votre fonction et la compléter des informations supplémentaires)
2. Modifiez la fonction move() de la classe Animal afin quelle soulève une erreur si la fonction move() n'est pas implémenté dans ces sous-classes (big help : NotImplementedError("Subclasses should implement this!")
3. Modifiez votre main.py dans le quelle vous instanciez un Animal, un Bird et un Mammal. Testez les différentes fonctions. Vérifiez que l'Animal retourne bien une exception quand la fonction move() est utilisé (help : try ... except ...).
4. N'oubliez pas de rédiger/modifier les tests unitaires et de respecter la Pep8.
5. Faites un second tag sur git avec la version V2.00.

7.3 Eagle, Pigeon, Tiger, Person, Cat et Dog

1. Réalisez les classes Eagle, Pigeon, Tiger, Person, Cat et Dog (conseil : pour la fonction to_string() de Person, vous pouvez utiliser la fonction to_string de la classe Mammal dans votre fonction et la compléter des informations supplémentaire)
2. Modifier la fonction speak() de la classe Animal afin quelle retourne une erreur si la fonction n'est pas implémenté dans ces sous-classes.
3. Modifiez votre main.py dans le quelle vous instanciez un objet de chaque classes. Testez les différentes fonctions. Vérifiez que l'Animal retourne bien une exception quand la fonction move() ou speak() est utilisé (help : try ... except ...). Vérifiez bien que Eagle, Pigeon, Tiger, Person, Cat et Dog ont bien un crie différent via la fonction speak()
4. N'oubliez pas de rédiger/modifier les tests unitaires et de respecter la Pep8.
5. Faites un troisième tag sur git avec la version V3.00.

7.4 Enclosure

1. Réalisez la Classe Enclosure (conseil : pour la fonction to_string() n'hésitez pas à utiliser la fonction to_string() des Animaux de votre liste)
2. Utilisez le module random pour la fonction all_attacking() pour s'assurer du choix aléatoire.
3. Modifiez votre main.py. Instanciez un Enclosure et des animaux de différentes classes. Testez votre enclos et vérifiez l'état de celui-ci avant et après all_attacking()
4. N'oubliez pas de rédiger/modifier les tests unitaires et de respecter la Pep8.
5. Faites un quatrième tag sur git avec la version V4.00.

7.5 Zoo

1. Réalisez la Classe Zoo (conseil : pour la fonction to_string() n'hésitez pas à utiliser la fonction to_string() des Enclosure de votre liste)
2. Modifiez votre main.py. Instanciez un Zoo, des enclos et des animaux de différentes classes. Testez votre zoo et vérifiez l'état de celui-ci avant et après all_attacking()
3. N'oubliez pas de rédiger/modifier les tests unitaires et de respecter la Pep8.
4. Faites un cinquième tag sur git avec la version V5.00.