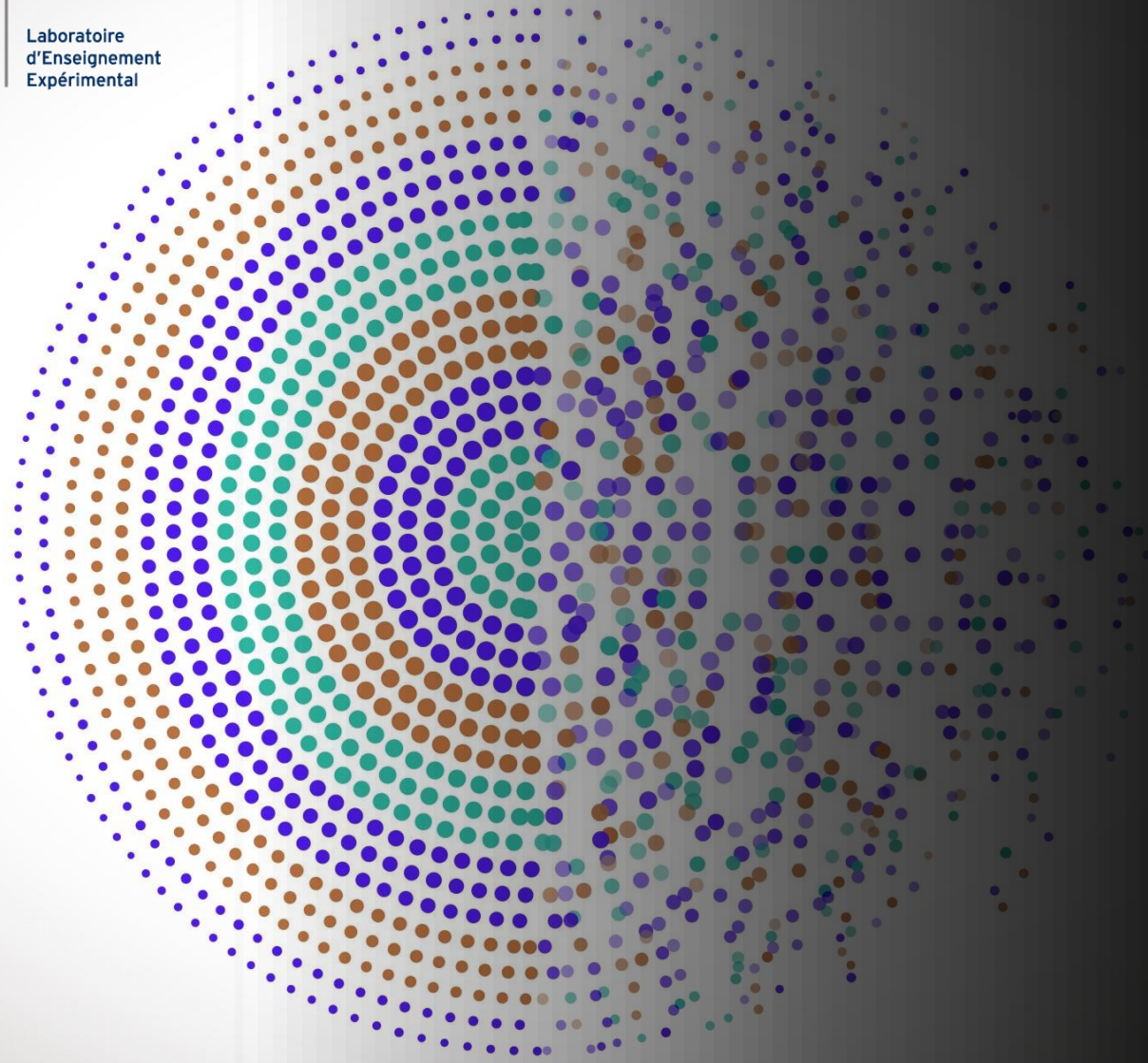
INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech | LEnsE Laboratoire d'Enseignement Expérimental



# Un monde d'objets

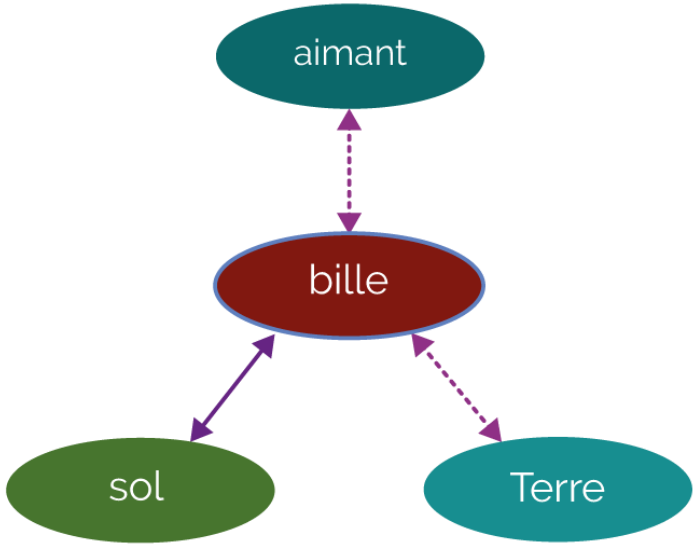
Outils Numériques / Semestre 5 / Institut d'Optique / B0_3

# Un monde d'objets

https://masevaux.fr/objets_trouves/

- Des objets qui interagissent

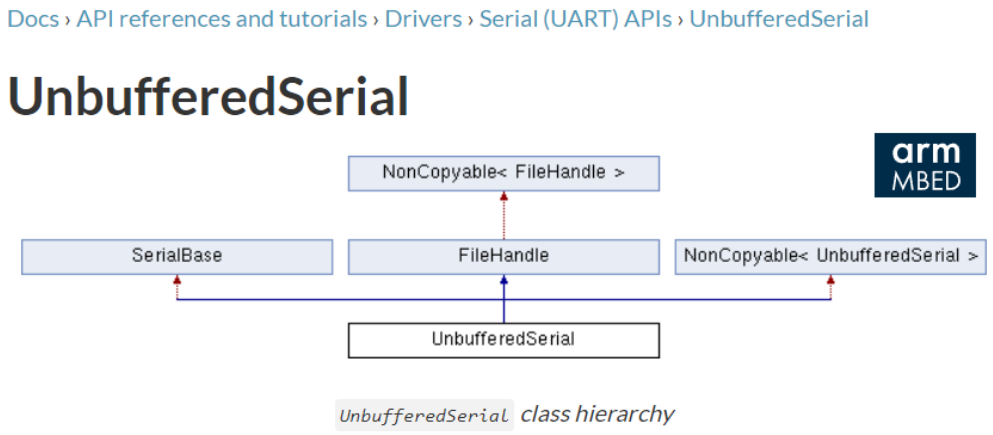
https://www.lepoint.fr/dossiers/societe/velo-libre-service-velib/



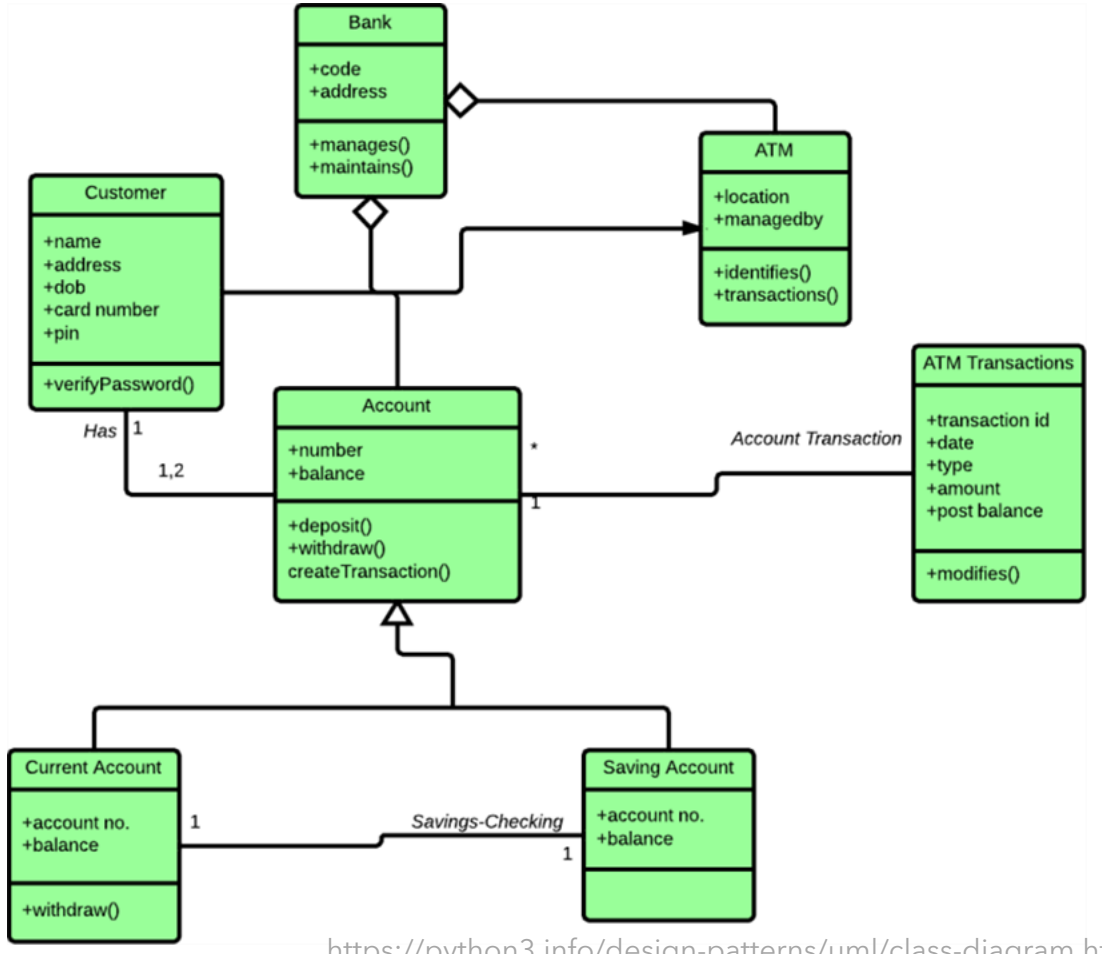

https://www.maxicours.com/se/cours/les-diagrammes-objet-interaction/

# Un monde d'objets informatiques

- Mise en œuvre informatique




`UnbufferedSerial` *class hierarchy*



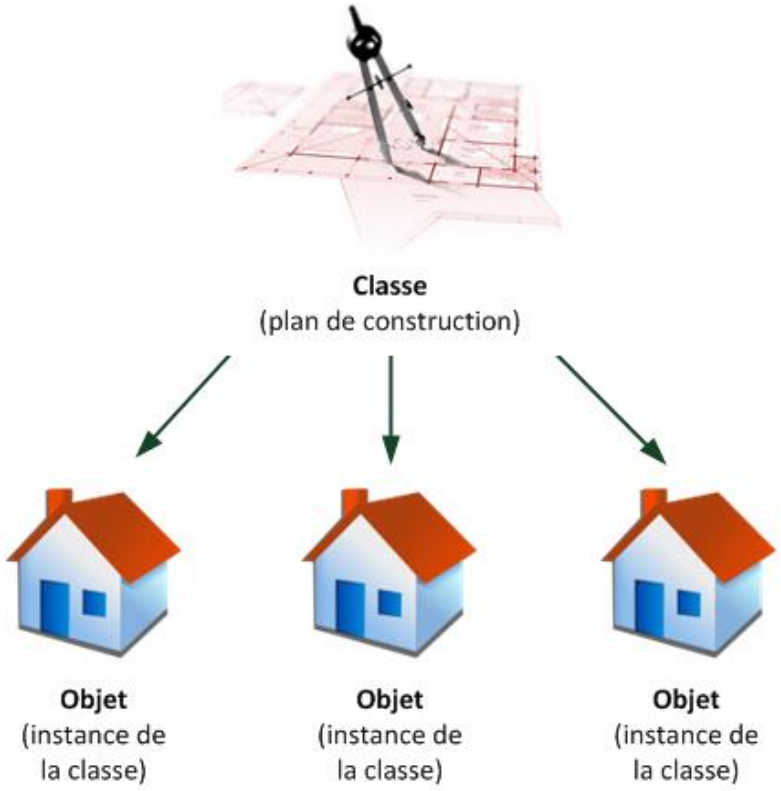

https://python3.info/design-patterns/uml/class-diagram.html

INSTITUT d'OPTIQUE GRADUATE SCHOOL ParisTech | LEnsE Laboratoire d'Enseignement Expérimental

# Programmation orientée objet

## Eléments de base

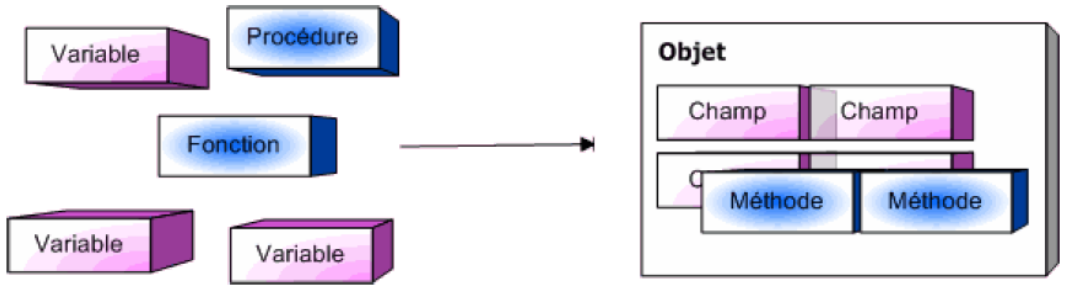
- **Classe** : rassemblement de différentes données et fonctions
- **Objet** : instance d'une classe



# Programmation orientée objet

## Concepts fondamentaux

- **Encapsulation** : regroupement de différentes données et fonctions sous une même entité
- **Héritage** : arborescence de classes permettant la spécialisation
- **Polymorphisme** : réaction différente de deux objets (de classe différente) à une même procédure

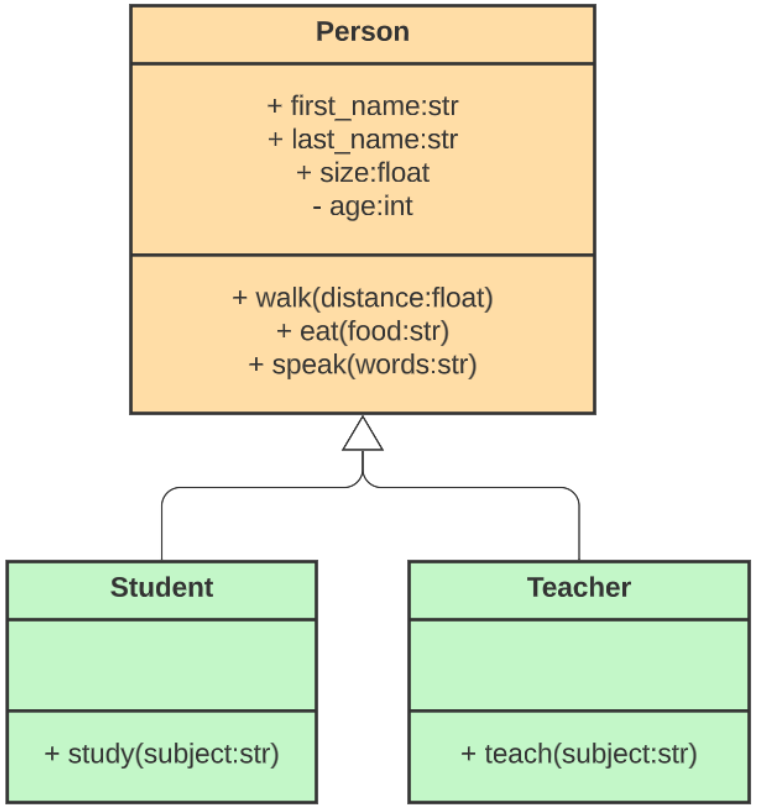


# Programmation orientée objet

## Concepts fondamentaux

- **Encapsulation** : regroupement de différentes données et fonctions sous une même entité
- **Héritage** : arborescence de classes permettant la spécialisation
- **Polymorphisme** : réaction différente de deux objets (de classe différente) à une même procédure

Classe mère

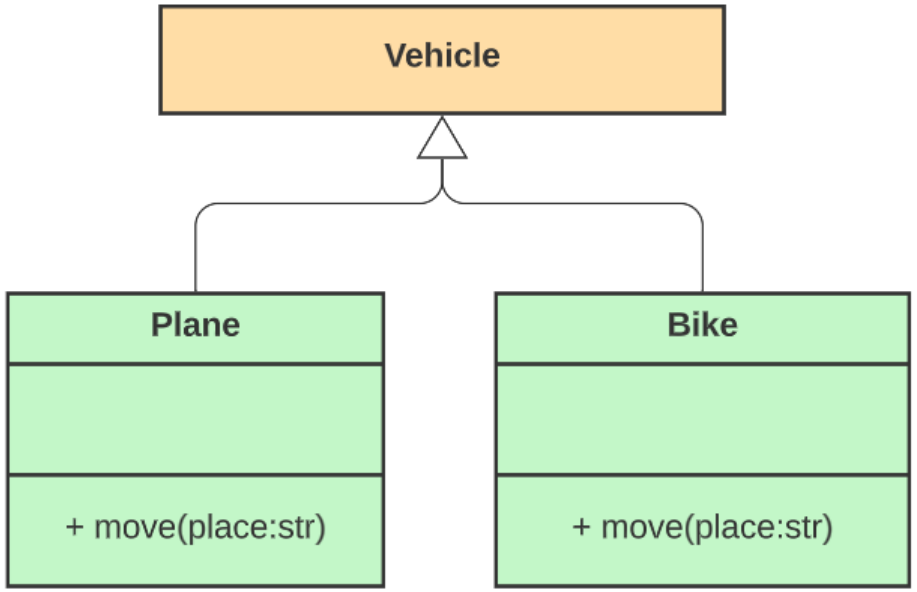



Classes filles

# Programmation orientée objet

## Concepts fondamentaux

- **Encapsulation** : regroupement de différentes données et fonctions sous une même entité
- **Héritage** : arborescence de classes permettant la spécialisation
- **Polymorphisme** : réaction différente de deux objets (de classe différente) à une même procédure



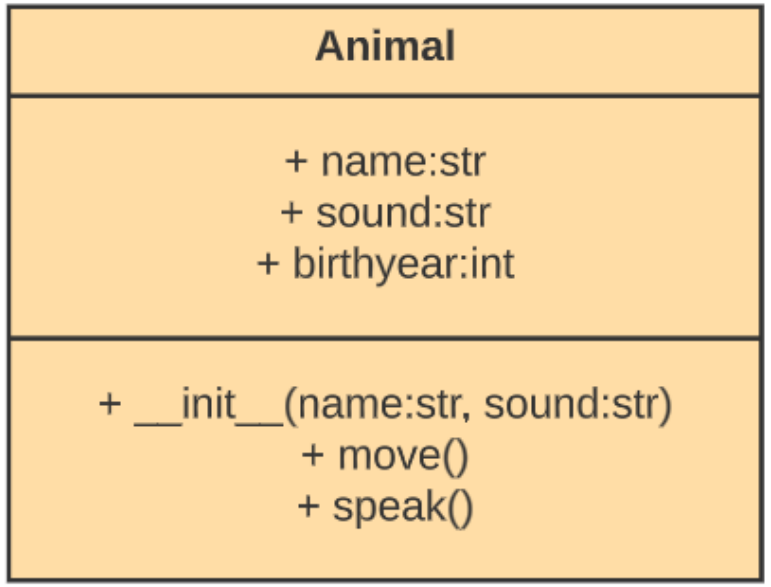
# Exemple d'une classe

**Encapsulation** : regroupement de différentes données et fonctions sous une même entité

```python
class Animal:
    """ object class Animal
    """
    def __init__(self, name="Hello", sound="..."):
        """ Animal class constructor
        :name: name of the animal
        """
        self.name = name
        self.sound = sound
        self.birthyear = 2000

    def move(self):
        print(f"\t[ {self.name} ] is moving")

    def speak(self):
        print(f"\t[ {self.name} ] is saying {self.sound}")
```

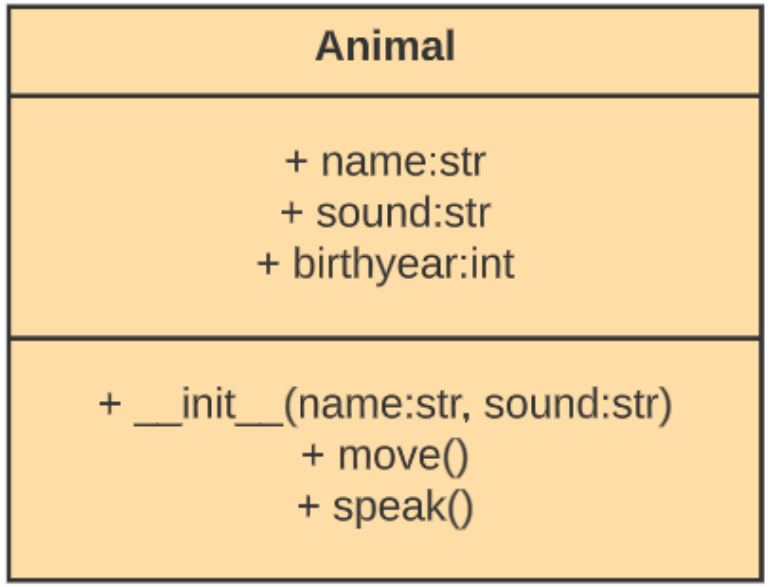


# Exemple d'une classe

**Encapsulation** : regroupement de différentes données et fonctions sous une même entité

```
# Test of the class Animal
if __name__ == '__main__':
    animal1 = Animal()
    print("Animal 1 Name = ", animal1.name)
    animal2 = Animal("Garfield")
    print("Animal 2 Name = ", animal2.name)

    print(animal1)
```




```
Animal 1 Name =  Hello
Animal 2 Name =  Garfield
<__main__.Animal object at 0x0000020C594D2F10>
```

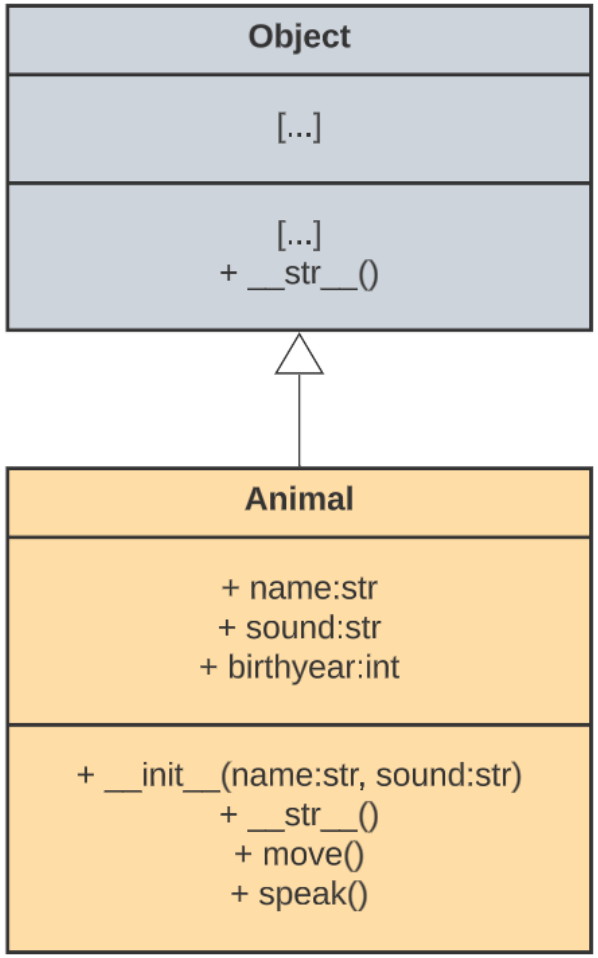
# Exemple d'une classe

**Redéfinition** : définir une méthode déjà existante dans une classe mère pour spécialiser cette nouvelle classe

```
class Animal:
    """ object class Animal
    """
    [...]

    def __str__(self):
        """ Animal class display
        """
        return f"Animal [ {self.name} ] born in {self.birthyear}"

    [...]
```



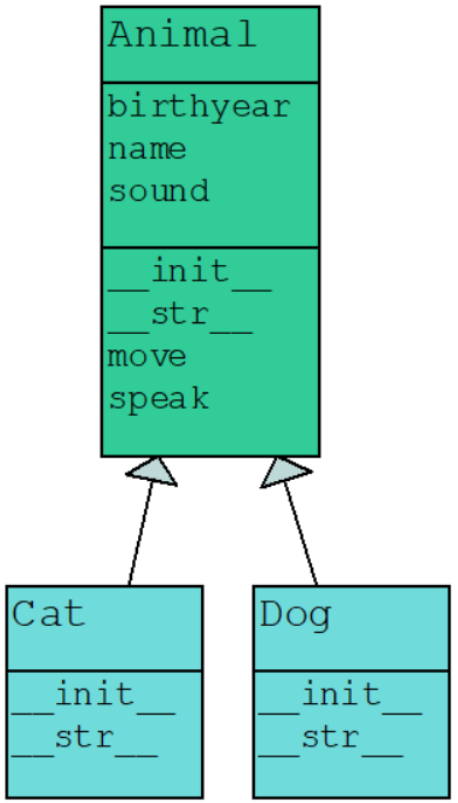

```
Animal 1 Name =  Hello
Animal 2 Name =  Garfield
Animal [ Hello ] born in 2000
```

# Exemple de classes héritées

**Héritage** : arborescence de classes permettant la spécialisation

```
class Cat(Animal):
    """ Object class Cat, inherit from Animal
    """
    def __init__(self, name="Hello", sound="Miaouh"):
        """ Cat class constructor
        :name: name of the animal
        """
        super().__init__(name, sound)

    def __str__(self):
        """ Cat class display
        """
        return f"Animal/CAT [ {self.name} ] born in {self.
        birthyear}"
```



# Exemple de classes héritées

**Héritage** : arborescence de classes permettant la spécialisation

```
dog1 = Dog("Ralph")
dog1.birthyear = 2012
```

```
Animal 1 Name =  Hello
Animal 2 Name =  Garfield
Animal [ Garfield ] born in 2000
      [ Garfield ] is moving
      [ Garfield ] is saying ...
Animal/CAT [ Tigrou ] born in 2000
      [ Tigrou ] is moving
      [ Tigrou ] is saying Miaouh
Animal/DOG [ Ralph ] born in 2012
      [ Ralph ] is moving
      [ Ralph ] is saying Wouaf
```