# Programmation Orientée Objet – Java

## Cours 4 : Héritage

Viviane Pons

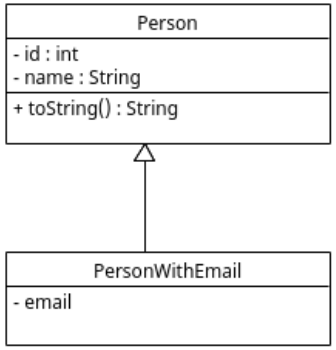
Master BIBS Université Paris-Saclay

# Qu'est-ce que l'héritage ?

Une notion essentielle de la programmation objet : permet de "récupérer" et d'augmenter la structure d'une autre classe

# Exemple

Voir l'exemple ici




```java
public class Person {

    private static int count = 0;

    private final int id;
    private final String name;

    ...
}
public class PersonWithEmail extends Pers

    private String email;

    ...

}
```

## Constructeurs

- S'il n'y a pas de constructeur sans paramètre, la classe fille doit implanter un constructeur explicite.
- On peut aussi **rajouter** un ou des constructeurs
- L'appel au constructeur de la classe mère se fait avec `super(...)`. Tout comme `this(...)`, il doit être appelé en première commande

```
public PersonWithEmail(String name) {
    super(name);
}

public PersonWithEmail(String name, String email) {
    this(name);
    setEmail(email);
}
```

## Rajouter et redéfinir des méthodes

La classe fille peut **rajouter** des méthodes supplémentaires et **redéfinir** (Override) les méthodes de la classe mère

```
public class PersonWithEmail extends Person {

    ...

    public String getEmail() {
        return email;
    }

    @Override
    public String toString() {
        return super.toString() + " -- email : " + getEmail
    }

}
```

**Accès à la classe mère avec `super`**

## Polymorphisme

```
Person p1 = new Person("Bob");

System.out.println(p1);

Person p2 = new PersonWithEmail("Alice", "alice@mail.com");

System.out.println(p2);
```

## Polymorphisme

```
Person p1 = new Person("Bob");

System.out.println(p1);

Person p2 = new PersonWithEmail("Alice", "alice@mail.com");

System.out.println(p2);
Person 1 : Bob
Person 2 : Alice -- email : alice@mail.com
```

La classe **déclarée** décide ce **qui se voit**. La classe instanciée décide **ce qui se fait**.

## Héritage unique / multiples

En Java une classe ne peut héritée que d'une seule autre classe. (Ce n'est pas le cas dans d'autres langages comme le python ou le C++)
**Avantage :** la superclasse est toujours bien définie
**Inconvénient :** moins de souplesse, potentielle duplication de code, compensé en partie par l'utilisation de multiples interfaces

## `final` pour les classes et méthodes

- `final` pour une classe (`public final class MyCLass`) : ne peut pas être étendue (pas de classes filles). Exemple, la classe `System` de l'API Java
- `final` pour une méthode : ne peut pas être redéfinie.

# Classes Abstraites et structure classique

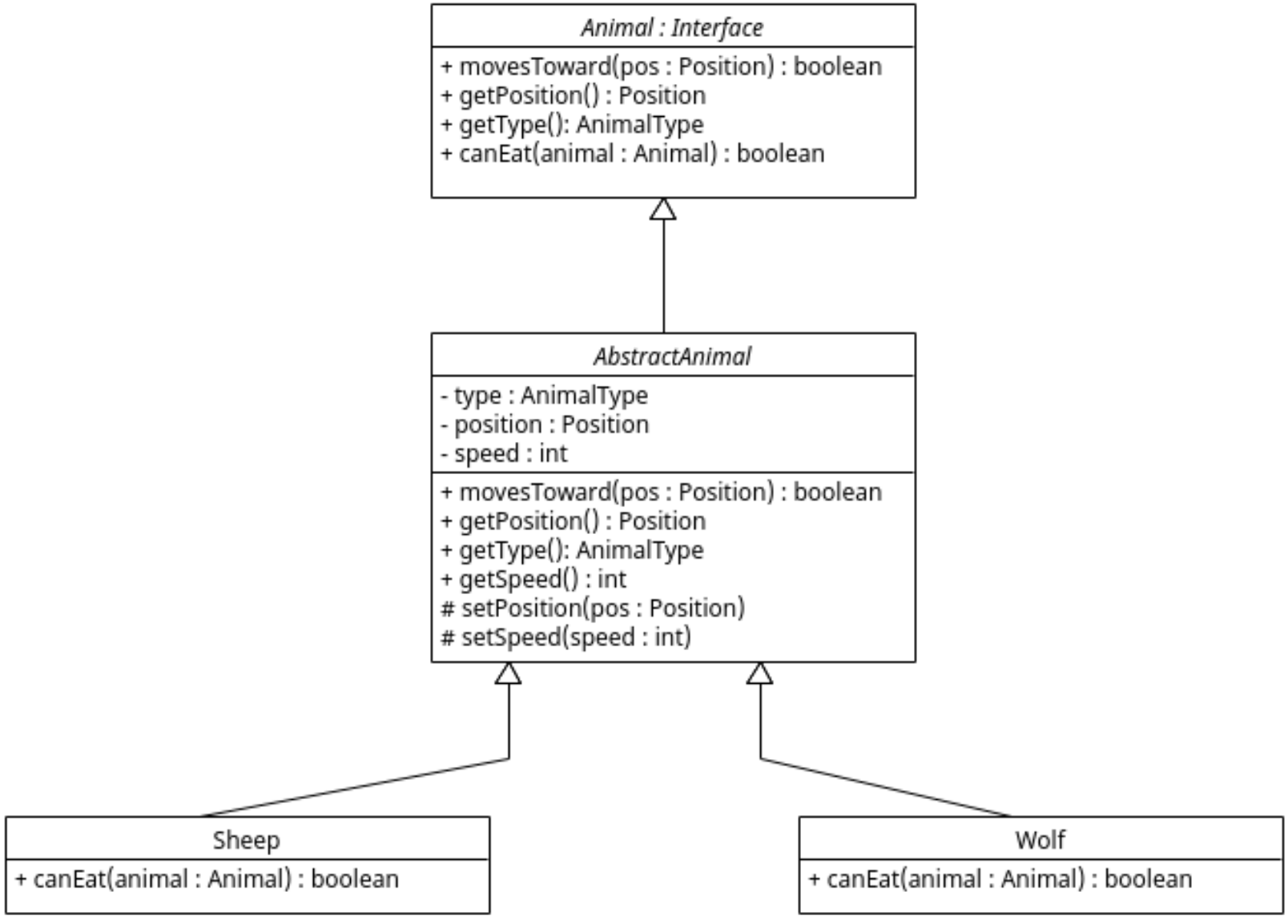
## Qu'est-ce qu'une classe abstraite ?

C'est une classe qui ne peut pas être instanciée ! Mais pourquoi ??

- Elle correspond à un socle commun partagé par plusieurs classes filles mais pas à un objet concret.
- Si elle implémente une interface, elle n'a pas besoin d'implémenter toutes les méthodes
- De la même façon, elle peut définir des *méthodes abstraites* (comme une interface) qui devront être implantées par les classes filles.

# Un schéma classique

Voir l'exemple ici



## Application

Réalisation d'un système dynamique avec une structure de classe complexe : Le TP