Héritage

La notion d'héritage

Dans tous les langages orientés objets, l' héritage est un mécanisme qui permet à une classe d'acquérir les méthodes et les attributs d'une autre classe, ce qui signifie que vous pouvez définir une autre classe en indiquant uniquement en quoi elle diffère de cette classe.

Un exemple

Déclarons et définissons la classe `Personne` qui n'a que deux propriétés, le nom et le prénom de la personne, et une seule méthode publique en dehors des accesseurs en lecture, qui affiche dans une console un court message :

```
class Personne {
  private String prenom, nom;
  public Personne(String prenom,String nom) {
    this.prenom = prenom; this.nom = nom;}
  public String prenom() {return prenom;}
  public String nom() {return nom;}
  public void sePresenter() {
    System.out.format("Bonjour, je m'appelle %s %s"
    ,prenom,nom);
}
```

La notion d'héritage

Le mécanisme d'héritage est mis en place à l'aide du mot-clef `extends` en java.

Déclarons une classe `Etudiant` modélisant un étudiant qui hérite de la classe `Personne` :

```
class Etudiant extends Personne
{
  private String numero;
}
```

Le mécanisme d'héritage fait que les instances de la classe `Etudiant` possède trois attributs :

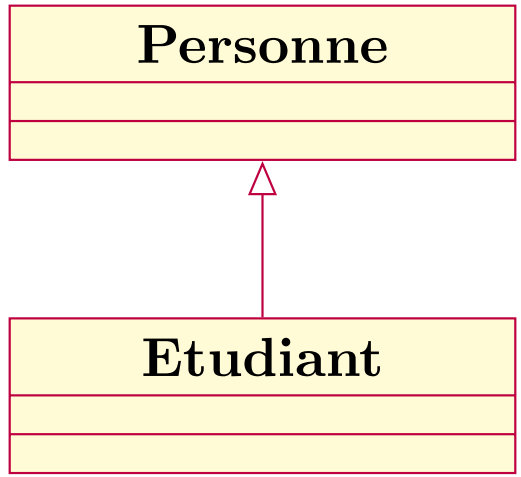
- la propriété `numero` définie dans la classe `Etudiant`
- les deux propriétés `prenom` et `nom` héritées de la classe `Personne`.

Relation d'héritage entre deux classes

Le mécanisme d'héritage induit une relation hiérarchique entre les classes :

- On dira que la classe `Etudiant` est **une** sous-classe de la classe `Personne`.
- On dira que la classe `Personne` est **la** super-classe de la classe `Etudiant`.

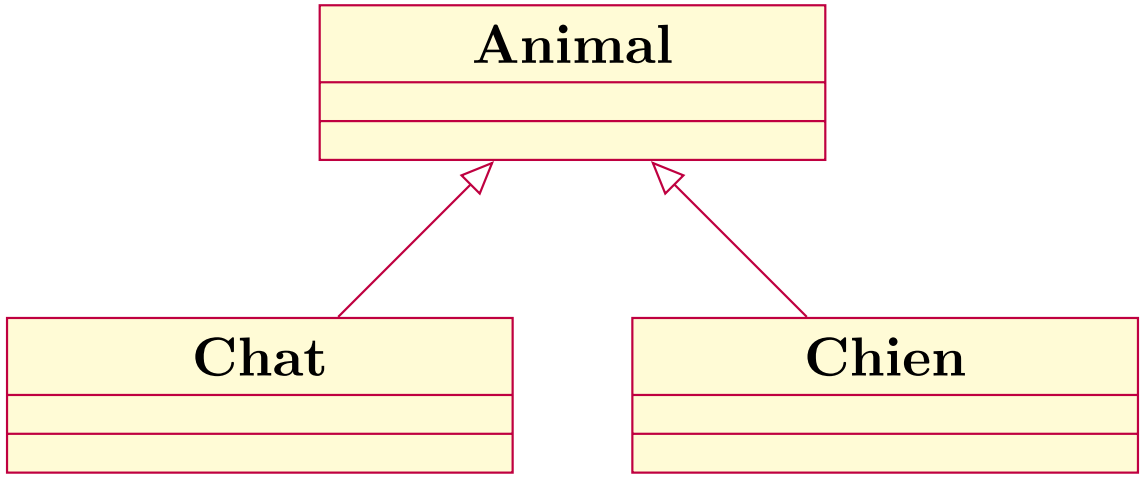
On symbolisera une relation d'héritage entre deux classes par le schéma suivant :



Relation d'héritage entre deux classes

Le langage java n'autorise que l'héritage simple entre deux classes, c'est-à-dire une classe ne peut avoir qu'une seule super-classe.

Il est en revanche possible de définir autant de sous-classes d'une classe que l'on veut.



Les classes `Chat` et `Chien` sont deux sous-classes de la classe `Animal`.

Instanciation et héritage

On décide que les trois propriétés d'une instance de la classe `Etudiant` seront initialisées à l'instanciation avec le prénom, le nom et le numéro de l'étudiant :

```
Etudiant e = new Etudiant("John","Doe","41241899");
```

Principe : le constructeur de la classe `Etudiant` utilisera le constructeur de la classe `Personne` pour initialiser les deux propriétés héritées, le nom et le prénom.

Instanciation et héritage

Pour tous les langages orientés objets, il existe un moyen syntaxique d'appeler un constructeur de la super-classe dans un constructeur d'une sous-classe :

```
class Etudiant extends Personne {
  private String numero;

  public Etudiant(String prenom,String nom,String
    numero) {
    // appel du constructeur Personne(prenom,nom)
    super(prenom,nom);
    this.numero = numero;
  }
}
```

Instanciation et héritage

Deux règles concernant l'appel d'un constructeur de la super-classe :

- On utilise le mot-clef **super**
- L'appel d'un constructeur de la super-classe est toujours la première instruction du constructeur.

```
class Etudiant extends Personne {
  private String numero;
  public Etudiant(String prenom,String nom,String
    numero) {
     this.numero = numero;
     // Interdit car l'appel du constructeur de la
    super-classe n'est pas la première instruction
     super(prenom,nom);
  }
}
```

Méthodes héritées

Une sous-classe hérite de toutes les méthodes de sa super-classe. EN PARTICULIER elle possède toutes les méthodes publiques de sa super-classe :

```
Etudiant e = new Etudiant("John","Doe","41241899
  ");

e.sePresenter(); /* affiche dans une console
"Bonjour, je m'appelle John Doe" */
```

La classe `Etudiant` hérite de la classe `Personne` et **possède** donc aussi la méthode publique `sePresenter`.

Attributs hérités

Une sous-classe hérite de tous les attributs de sa super-classe. MAIS ATTENTION les instances d'une sous-classe n'ont pas plus de droit d'accès aux attributs privés de leur super-classe que n'importe quelle instance d'une autre classe :

```
class Etudiant extends Personne {

  public Etudiant(String prenom,String nom,String
    numero) {
     this.prenom = prenom; // INTERDIT : prenom est
    un attribut privé de la classe Personne. Seules
     les instances de la classe Personne ont le
    droit d'y accéder directement
     ...
     }
}
```