Algorithmie & Programmation - Cours 6

Conception orientée objet, Partie 2

Dr. Jérémie Sublime

LISITE Laboratory - RDI Team - ISEP

jeremie.sublime@isep.fr

Plan

1. Rappels
2. Diagrammes de classe
3. Notions avancées de programmation orientée objet
4. Exercices d'application

Plan

1. Rappels
2. Diagrammes de classe
3. Notions avancées de programmation orientée objet
4. Exercices d'application

Qu'est ce qu'un objet ?

Objet : Définition

Une objet est une **instance de classe** : C'est l'attribution de valeurs au différents attributs qui crée l'objet.

- En Java, l'instanciation d'un objet se fait avec le mot clé **new**.
- La classe étant un modèle, il peut y avoir plusieurs instances de la même classe, et donc plusieurs objets d'une même classe.
- Les méthodes non-statiques de chaque classe ne peuvent être utilisées que si l'objet est instancié (puisqu'elles portent sur les attributs).

Attributs et méthodes statiques

Le mot clé **static** peut être utilisé devant un attribut pour définir une variable qui soit une variable associée à l'ensemble de la classe plutôt qu'une variable liée à un objet instancié : Par exemple un compteur !

- Contraitement aux attributs "classiques", un attribut statique peut être appelé sans instancier l'objet : *maClasse.monAttribut*.
- La valeur d'un attribut statique est partagée entre toutes les instances de la classe : modifier sa valeur sur un objet de cette classe la modifiera pour tous les autres.

Comme pour les attributs, une méthode peut-être statique :

- Une méthode statique peut être utilisée sans instancier l'objet ou depuis un objet déjà instancié.
- **Une méthode statique ne peut utiliser que des attributs eux-mêmes statiques** ! Sinon le code ne compilera pas.

Comment choisir quelles méthodes définir ?

- Au delà des getters et setters, le choix des méthodes à coder dans une classe se fait selon les besoins du programme.
- Chaque classe ne doit contenir que des méthodes qui la concerne et qui utilise ses attributs : Si vous codez dans une classe une méthode ne manipulant aucun des attributs de la classe, votre méthode est probablement dans la mauvaise classe ou au mauvais endroit.
- Le choix de placer une méthode dans une classe doit se faire selon la logique de la programmation orientée objet : Est-il logique que cette classe possède cette méthode ?

Savoir reconnaitre un code pourri

Si vous mettez une fonction dans un fichier de classe avec lequel elle n'a rien avoir parce que votre programme ne compile pas lorsque vous mettez ce code ailleurs : Vous avez un problème de conception.

Plan

1 Rappels

2 Diagrammes de classe

3 Notions avancées de programmation orientée objet

4 Exercices d'application

Les étapes de création d'un algorithme

Rappels: Méthode pour résoudre un problème

- Lister les objets nécessaires : Il s'agit de **Modélisation**
- Lister les étapes nécessaires : Il s'agit de **Conception**

Les étapes de création d'un algorithme

Rappels: Méthode pour résoudre un problème

- Lister les objets nécessaires : Il s'agit de **Modélisation**
- Lister les étapes nécessaires : Il s'agit de **Conception**

On s'intéresse à l'étape de modélisation :

- Nous avons vu que la programmation orientée objet permettait de définir des structures et des objets complexes.
- Nous allons maintenant voir comment modéliser les relations entres les différents objets et structures.

Diagramme de classes

Un **diagramme de classes** est un schéma **UML** utilisé en génie logiciel pour présenter les classes ainsi que les différentes relations entre celles-ci lors de la **modélisation** d'un programme **orienté objet**.

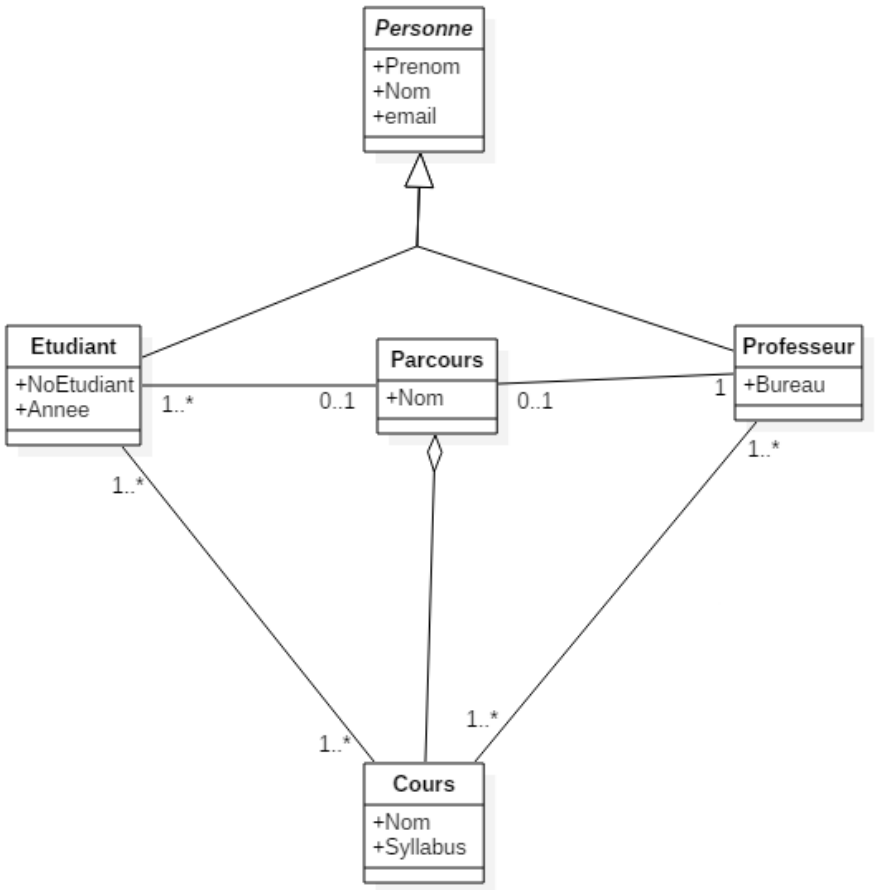
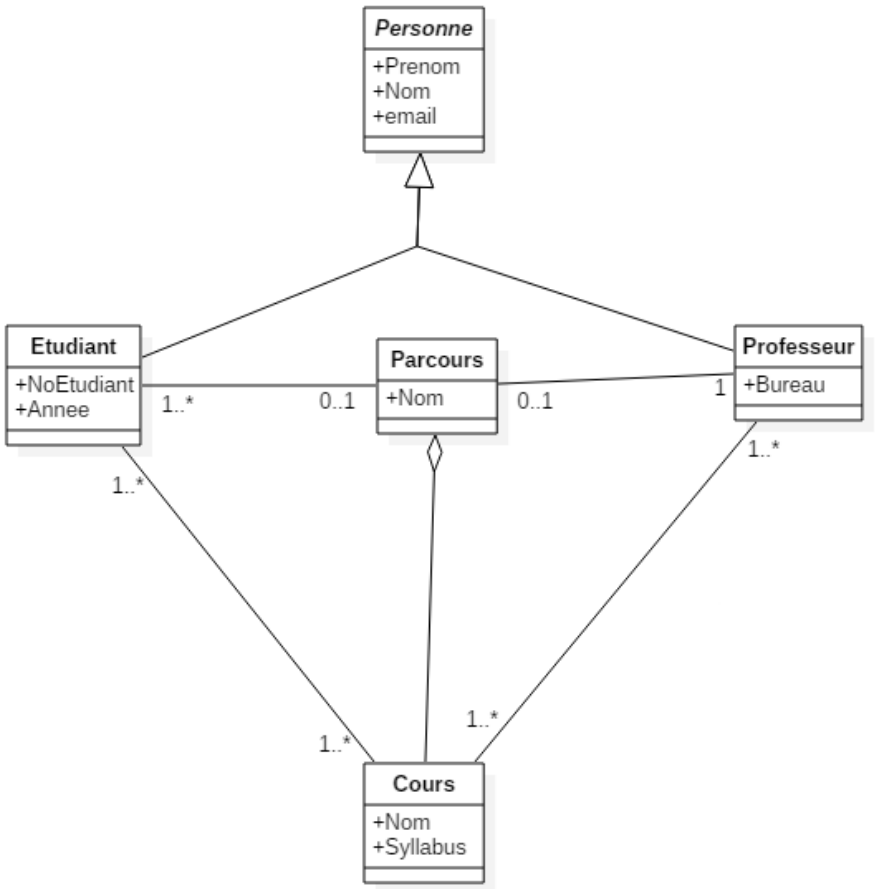


Diagramme de classes

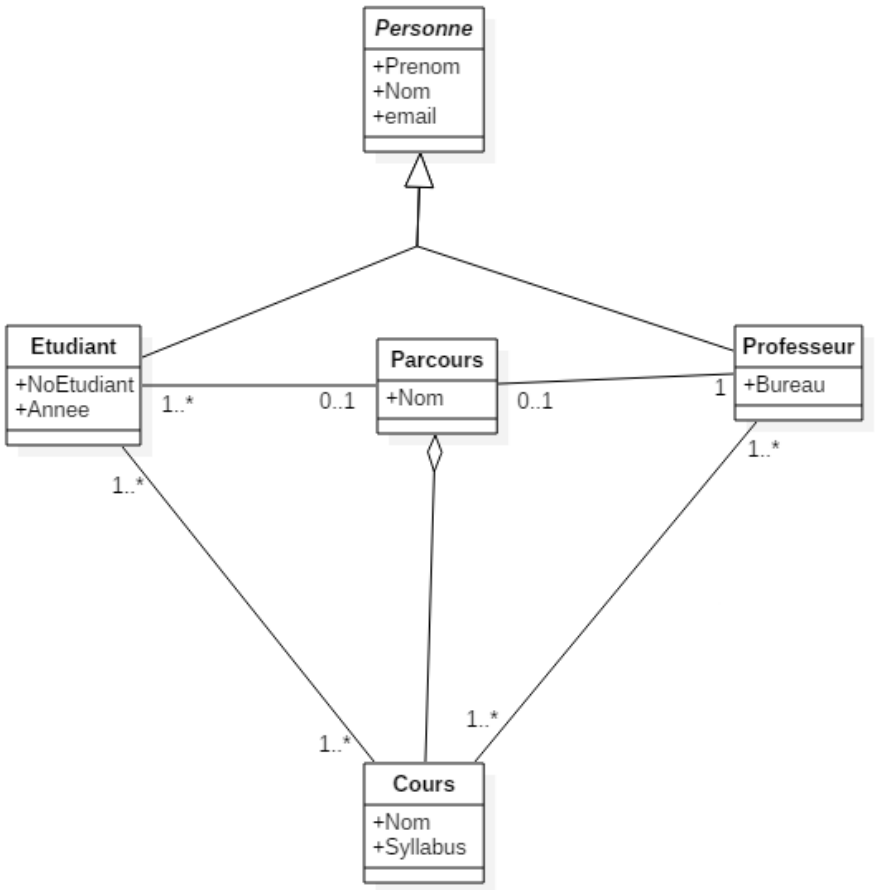
Un **diagramme de classes** est un schéma **UML** utilisé en génie logiciel pour présenter les classes ainsi que les différentes relations entre celles-ci lors de la **modélisation** d'un programme **orienté objet**.



Les diagrammes de classes sont des diagrammes de type "entité-association" très semblables à ceux utilisés en base de données.

Diagramme de classes

Un **diagramme de classes** est un schéma **UML** utilisé en génie logiciel pour présenter les classes ainsi que les différentes relations entre celles-ci lors de la **modélisation** d'un programme **orienté objet**.

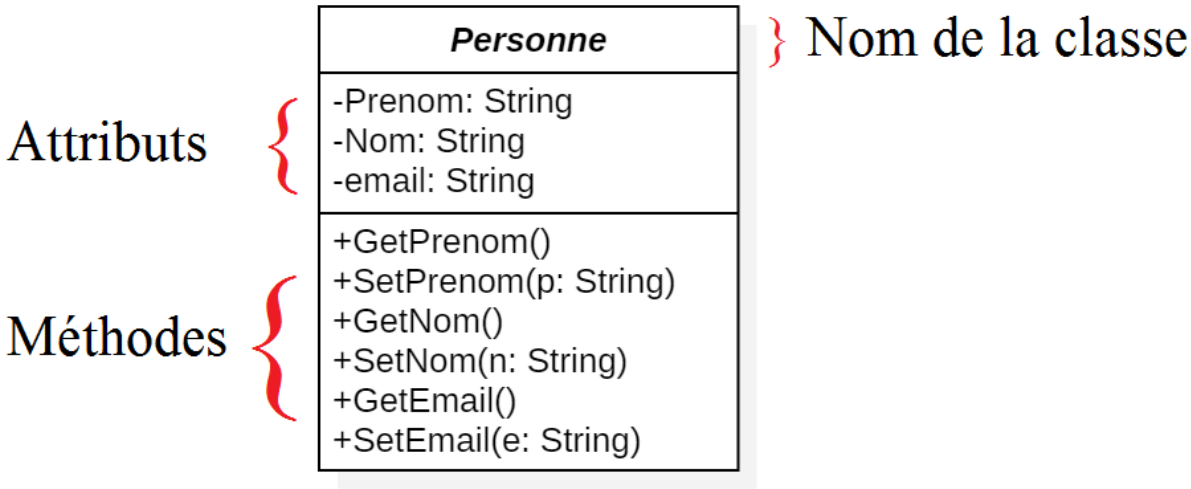


Les diagrammes de classes sont des diagrammes de type "entité-association" très semblables à ceux utilisés en base de données.

Logiciels pour faire de l'UML:
Umbrello, StarUML

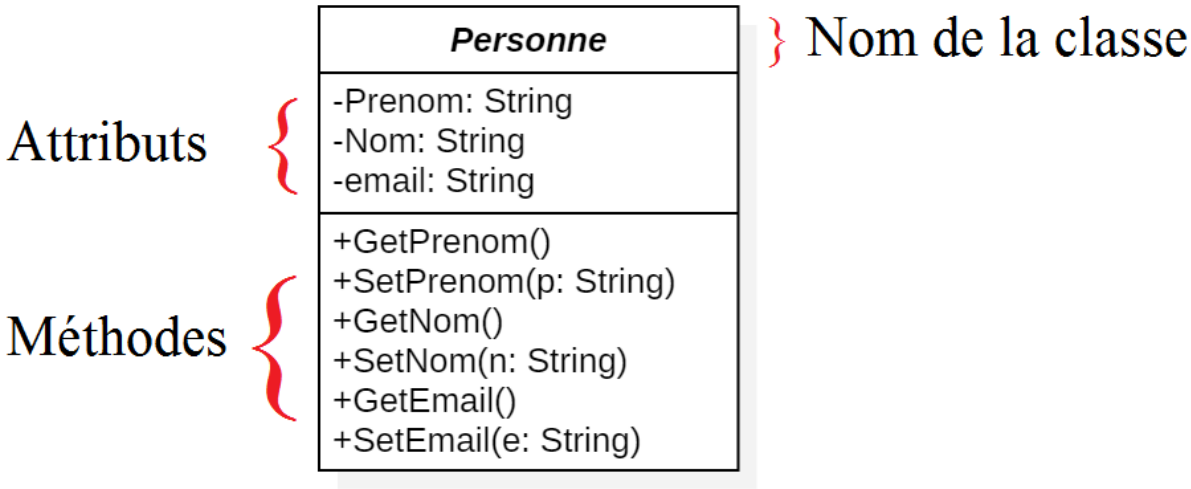
Représentation des classes

- Dans un diagramme de classes, le nom, les attributs et les méthodes de chaque classe sont indiqués.



Représentation des classes

- Dans un diagramme de classes, le nom, les attributs et les méthodes de chaque classe sont indiqués.



Remarque sur l'encapsulation

- Les méthodes et attributs publics sont précédés d'un "+", les protégés d'un "=" et les privés d'un "-".
- Les méthodes et attributs statiques sont soulignés.

Objets composites

En modélisation orientée objet, vous serez amenés à rencontrer des objets composés d'autres objets. Ces interactions peuvent être modélisées de 2 façons dans un diagramme de classe:

- Une **composition**
- Une **aggrégation**

Objets composites

En modélisation orientée objet, vous serez amenés à rencontrer des objets composés d'autres objets. Ces interactions peuvent être modélisées de 2 façons dans un diagramme de classe:

- Une **composition**
- Une **aggrégation**

Aggrégation vs Composition

- On utilise la composition (A est composé de B) dans le cas où les objets B n'ont pas de raison d'exister seuls sans A.
- Dans le cas contraire (A contient un aggrégat de B), les objets B peuvent exister indépendemment de A.

Composition et aggrégation

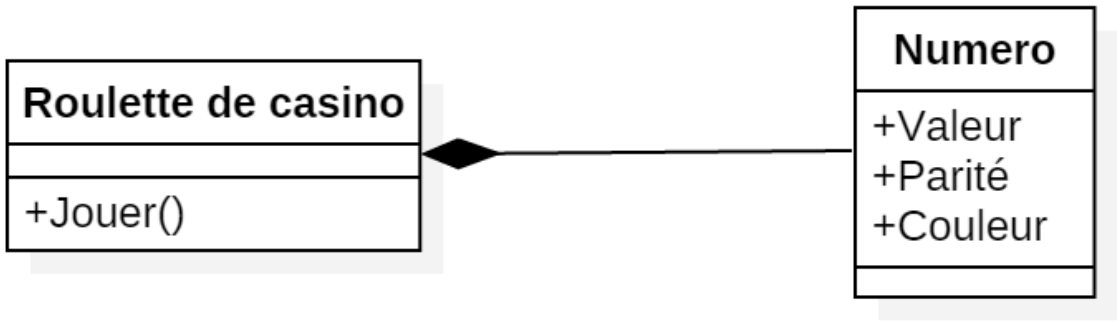


Figure: Exemple de Composition

Composition et aggrégation

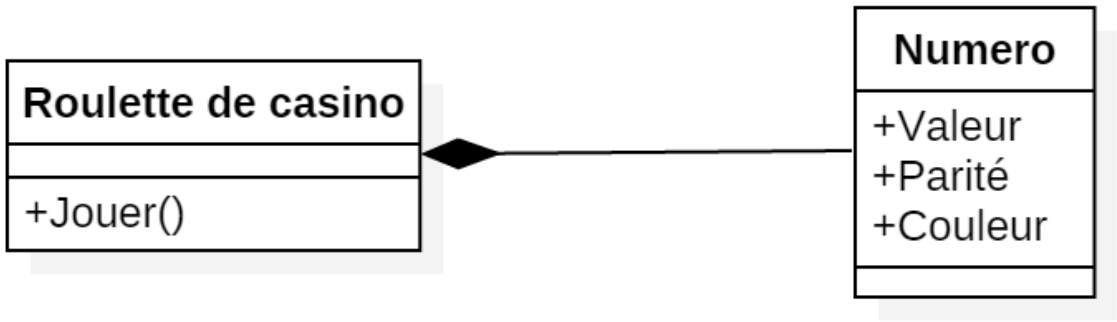


Figure: Exemple de Composition

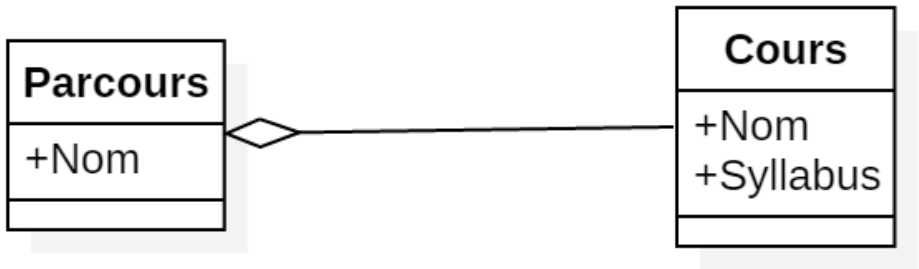


Figure: Exemple d'aggrégation

Composition et aggrégation en Java

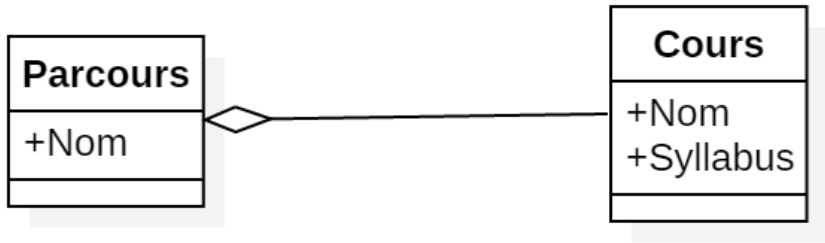
En Java, les aggrégation et les compositions seront par exemples modélisées par un attribut de type tableau ou liste.

```
class cours{
   String nom;
   String syllabus;

   /* */
}
```

```
class parcours{
   String nom;
   cours [] listeCours;

   /* */
}
```



Composition et aggrégation en Java

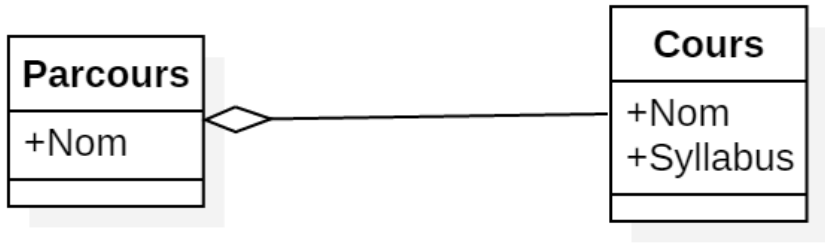
En Java, les aggrégation et les compositions seront par exemples modélisées par un attribut de type tableau ou liste.

```
class cours{
   String nom;
   String syllabus;

   /* */
}
```

```
class parcours{
   String nom;
   cours [] listeCours;

   /* */
}
```

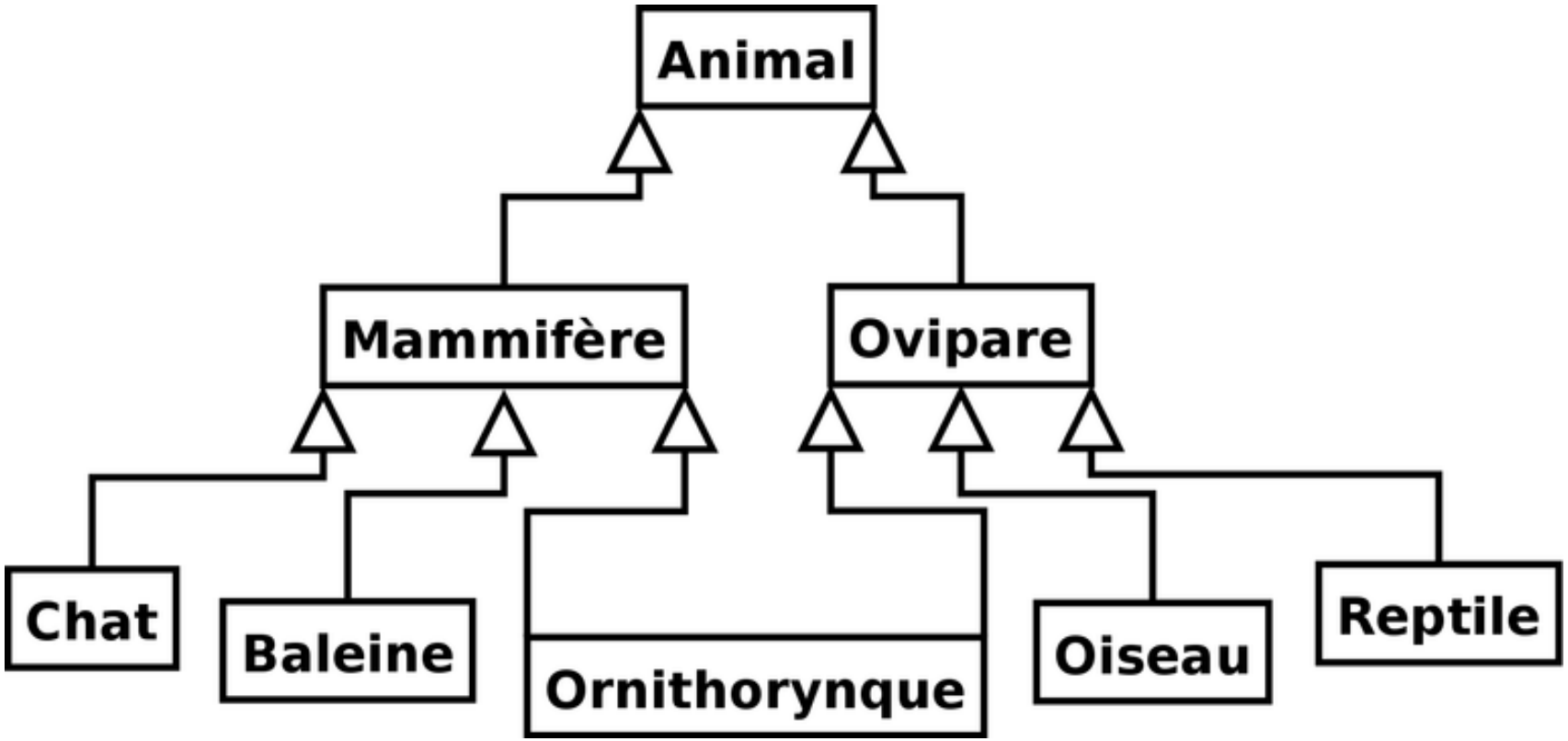


Remarque

Le choix d'un tableau simple, tableau à double entrée ou autre containter dépend de l'application.

La notion d'héritage

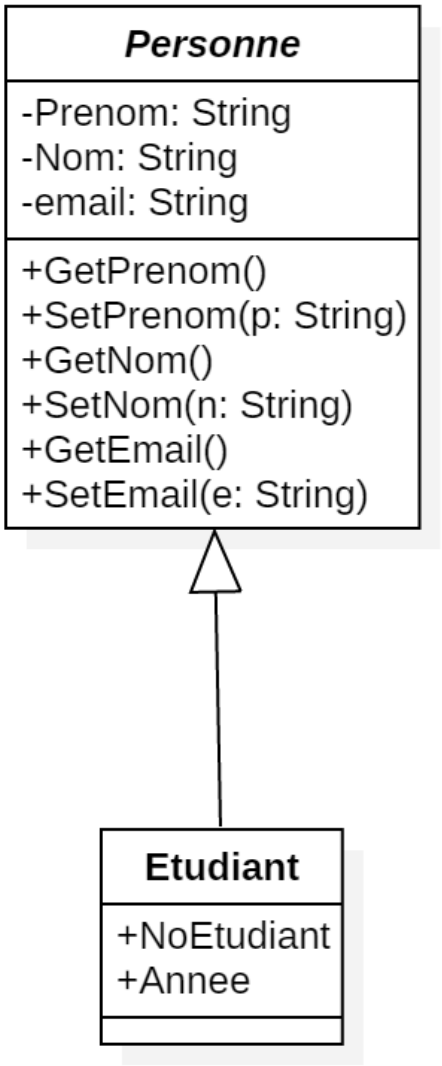
La notion d'**héritage** est utilisée en programmation orientée objet pour modéliser le fait que certaines classes sont des cas particuliers de classes déjà pré-existantes.



Les propriétés de l'héritage

- Lorsqu'une classe fille hérite d'une classe mère, la classe fille peut utiliser tous les attributs et méthodes de la classe mère et peut avoir en plus des attributs et méthodes qui lui sont propres.
- Cette technique permet de **factoriser** son code.
- Le mot clé **protected** est utilisé pour que la classe fille puisse avoir accès aux éléments de la classe mère tout en conservant l'encapsulation.
- Selon le language de programmation, les héritages multiples sont possibles (en C++ par exemple).
- Le mot clé Java pour l'héritage est "*extends*".

L'héritage en Java




```java
class personne{
   String nom;
   String prenom;
   String mail;

   /* */
}
```

```java
class etudiant extends personne{
   int noEtudiant;
   int annee;

   /* */
}
```

Autres liens entre classes

Il peut exister d'autres liens d'association entre classes : utilisation, dépendances, etc.

- Les dépendances sont représentées par des flèches en pointillées.
- Les associations autres que dépendances, aggrégation, ou composition (par exemple : "un étudiant suit des cours") sont représentés par de simples traits avec les cardinalités associées.

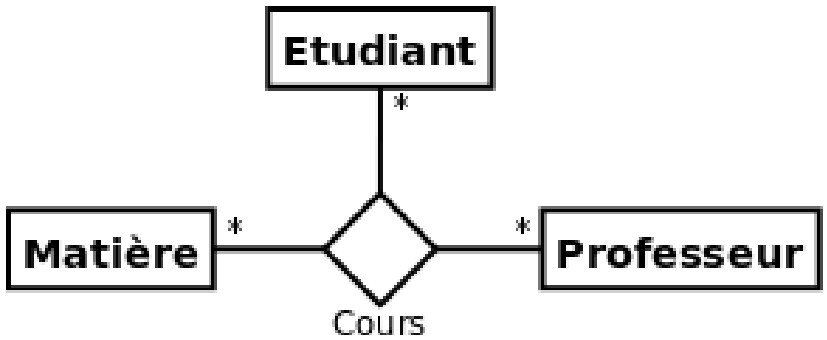
Remarque

En cas d'ambiguité, les associations -même simples- peuvent être fléchées pour qu'on sache dans quel sens elles vont.

Associations spécifiques entre classes : associations n-aire

Un association *n-aire* est une association qui permet de lier plus de deux classes ensembles :

- Des professeurs font des cours de plusieurs matières à plusieurs étudiants.
- Un même cours peut être enseigné par plusieurs professeurs.

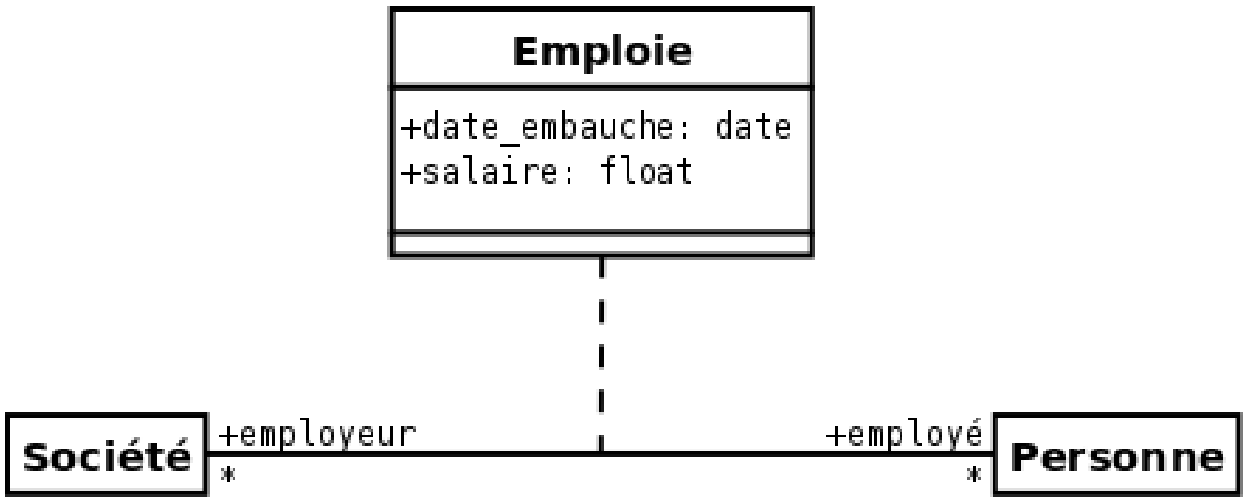


Les associations n-aire sont représentées par un losange reliant les différentes classes concernées.

Associations spécifiques entre classes : classes d'association

Parfois, une association doit posséder des propriétés :

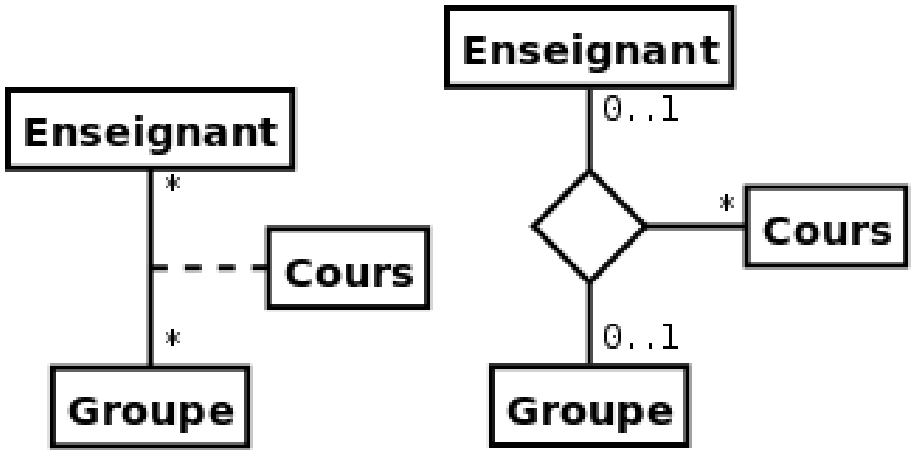
- L'association *Emploie* entre une société et une personne possède comme propriétés le *salaire* et la *date d'embauche*.
- Ces deux propriétés n'appartiennent ni à la société, qui peut employer plusieurs personnes, ni aux personnes, qui peuvent avoir plusieurs emplois. Il s'agit donc bien de propriétés de l'association. Il faut introduire un nouveau concept pour modéliser cette situation : celui de **classe-association**.



Classe d'association VS relation n-aire

Il est parfois difficile de trancher une classe d'association et relation n-aire.

- Il faut garder à l'esprit que la classe d'association est avant tout une association et est donc indisociable de l'association entre les 2 autres classes.



Dans l'exemple ci-dessus, si un cours doit pouvoir exister indépendamment du lien entre enseignant et groupe, c'est le modèle de droite qui est correct.

Plan

1 Rappels

2 Diagrammes de classe

3 Notions avancées de programmation orientée objet

4 Exercices d'application

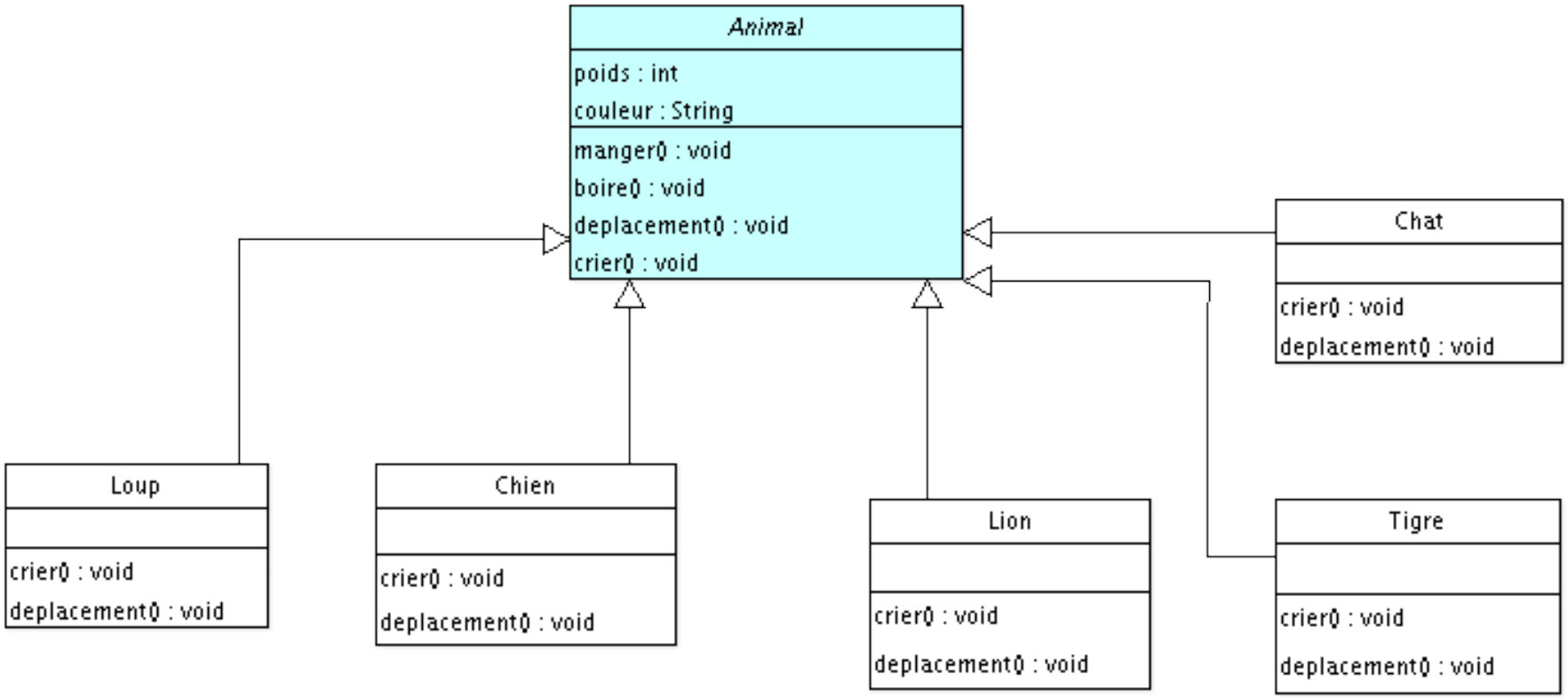
La notion de classe abstraite

Définition

Une **classe abstraite** est une classe **non-instanciable** soit parce qu'elle est **incomplète**, soit parce qu'elle est un simple **modèle conceptuel** servant de seule base à des classes filles.

- Les classes abstraites facilitent la conceptualisation de modèles complexes.
- Les classes abstraites permettent de mutualiser facilement les fonctionnalités et donc le code de classes et d'objets ayant de fortes similitudes.
- Dans un diagramme de classe, les classes abstraites sont indiquées en italiques.
- En Java, on écrira "*abstract class*" au lieu de "*class*".

Exemple de classe abstraite



Méthodes abstraites

Définition

Une **méthode abstraite** est une méthode dont seul le prototypage est indiqué dans le fichier de classe et qui devra être définie dans les classes filles.

- Les méthodes abstraites sont précédées du mot clé "*abstract*".
- Le prototypage est suivi par un point-virgule.

```
abstract class Animal{

  abstract void deplacement(); //Méthode abstraite

}
```

La notion d'interface

Définition

Une **interface** est un type particulier de classe abstraite qui **ne contient que des méthodes abstraites** et aucun attributs. Une interface peut être vue comme **un modèle** que certaines classes devront respecter.

- Les interfaces permettent de définir tous les prototypes de méthodes qui devront exister dans les classes "filles" qui implémenteront cette interface.
- L'héritage multiple n'est pas possible en Java, mais il est en revanche possible pour une classe d'implémenter plusieurs interfaces.
- En Java, on écrira "*public interface*" au lieu de "*public class*".

Implémenter une interface

- En Java, pour dire qu'une classe implémente une interface, on utilisera le mot clé "*implements*".
- Une classe peut implémenter plusieurs interfaces.
- Pour que le code compile, toutes les méthodes des interfaces implémentées doivent être définies (non-abstraites) dans les classes concernées.

Plan

1 Rappels

2 Diagrammes de classe

3 Notions avancées de programmation orientée objet

4 Exercices d'application

Diagramme de classe : Diagramme d'une école

- Les étudiants et les enseignants sont deux sortes de personnes.
- Un doctorant est un étudiant qui assure des enseignements.
- Les étudiants et les doctorants doivent pouvoir s'inscrire en début d'année dans un établissement, ou éventuellement modifier leur inscription.
- On connait les noms et prénoms de toutes les personnes.
- On doit pouvoir calculer les salaires des doctorants et des enseignants.

Diagramme de classe : Hôtel

- Un hôtel a les caractéristiques suivantes : une adresse, un nombre de pièces et une catégorie.
- Un hôtel a au moins 2 chambres.
- Chaque chambre a un numéro, un prix par nuit, et contient au moins un lit et une salle d'eau de type douche ou baignoire.
- L'hôtel héberge des personnes qui peuvent être des enfants ou des adultes, qui doivent donner leur nom et leur prénom en arrivant.
- L'hôtel a un certain nombre d'employés et est dirigé par un directeur.
- Pour chaque chambre on peut s'avoir qui l'occupe et à quelle date, et on peut donc calculer le loyer rapporté chaque chambre en fonction de son occupation.
- Il est possible de calculer le chiffre d'affaire de l'hôtel entre 2 dates.