جامعة غليزان
RELIZANE UNIVERSITY

République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université de Relizane
Faculté des Sciences et Technologies
Département d'Informatique

3ème année Informatique

Génie Logiciel
Diagramme de classe

Présenté par: Dr. Benotmane.Z

Introduction

- Comme nous l'avons mentionné dans le chapitre précédent, le diagramme de cas d'utilisation montre les grandes fonctionnalités du système ainsi que les acteurs déclenchant ces dernières. Le diagramme de classes nous montre la structure interne du système. Il permet de fournir une représentation abstraite des objets du système qui vont interagir pour réaliser les cas d'utilisation.

- Il s'agit d'une vue statique, car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application en présentant les classes du système et leurs relations.

- Les principaux éléments de cette vue statique sont les classes et leurs relations : association, généralisation et plusieurs types de dépendances.

Diagramme de Classe

Un diagramme de classes montre la structure statique d'un système. Il permet la visualisation des classes et des relations entre elles. Son but est d'expliquer ce qu'il faut réaliser plutôt que d'expliquer comment le réaliser.

Classe et Instance de Classe

- Le monde est composé d'entités qui collaborent.
- Ces entités sont une abstraction d'objets réels.
- Une classe est la description d'un ensemble d'objets ayant des caractéristiques communes.
- On peut représenter:

❑des éléments concrets (ex : des avions),

❑ des éléments abstraits (ex : des commandes de marchandises),

❑ des composants d'une application (ex : les boîtes de dialogue),

❑ des structures informatiques (ex : des listes chaînées),

❑ des éléments comportementaux (ex : des tâches), etc.

Donc…

Un objet est une instance d'une classe.

C'est une entité dotée d'une identité, d'un état et d'un comportement que l'on peut invoquer.

Les objets sont des éléments individuels d'un système en cours d'exécution.

Un jeu d’objets dont les caractéristiques sont communes → Une classe.

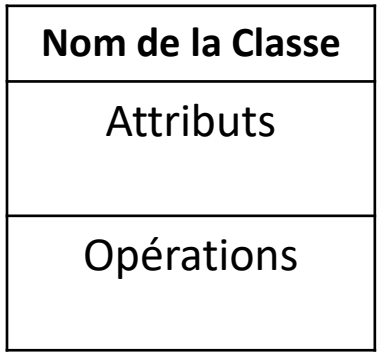
Classe

- Une classe est la description abstraite d'un ensemble d'objets faisant partie du même domaine (possédant les mêmes caractéristiques).
- Une classe est représentée par un classeur divisé en trois parties:

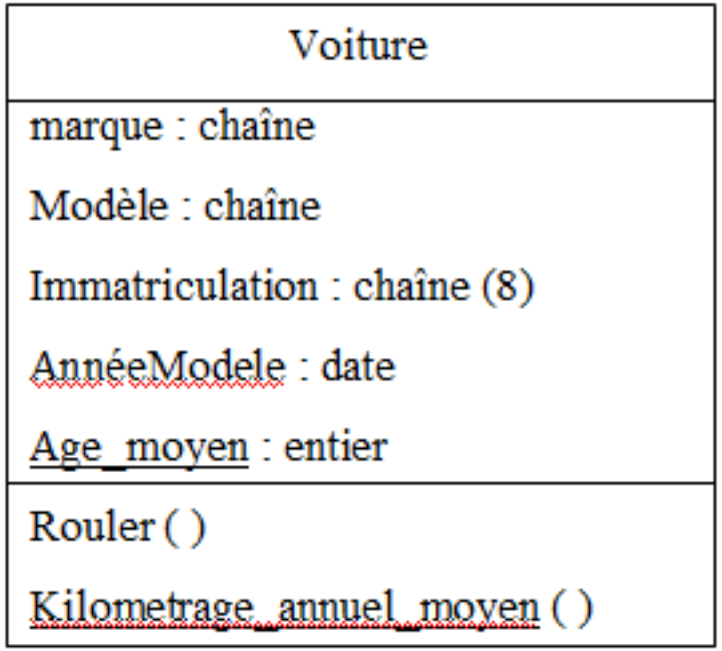
Le nom de la classe qui commence toujours par une majuscule.

Les attributs qui représentent les données de l'objet.

Les opérations qui représentent le comportement d'un objet



Exemple



L'*instanciation de la classe VOITURE est la création d'un objet de la classe VOITURE. Une instance de la classe VOITURE est une voiture donnée (objet « de type » VOITURE).*

Nom de la classe

- Le nom de la classe doit évoquer le concept décrit par la classe. Il commence par une majuscule.
- Pour indiquer qu'une classe est abstraite, il faut ajouter le mot-clef *abstract*.

Attributs

Les attributs définissent des informations qu'une classe ou un objet doivent connaître.

Chaque attribut est défini par un nom, un type de données, une visibilité et peut être initialisé.

Le nom de l'attribut est obligatoirement unique dans la classe.

Attribut de classe

- Par défaut, chaque instance d'une classe possède sa propre copie des attributs de la classe. Les valeurs des attributs peuvent donc différer d'un objet à un autre. Cependant, il est parfois nécessaire de définir un *attribut de classe* (*static*) qui garde une valeur unique et partagée par toutes les instances de la classe. Les instances ont accès à cet attribut mais n'en possèdent pas une copie.
- Un attribut de classe n'est donc pas une propriété d'une instance mais une propriété de la classe et l'accès à cet attribut ne nécessite pas l'existence d'une instance.

Attribut dérivé

- Les attributs dérivés peuvent être calculés à partir d'autres attributs et de formules de calcul.
- Les attributs dérivés sont symbolisés par l'ajout d'un « / » devant leur nom.

Méthodes

- Dans une classe, une opération (même nom et même types de paramètres) doit être unique.
- Il est possible que le nom d’une opération apparaisse plusieurs fois mais avec des paramètres différents, il s’agit alors d’une opération dite surchargée. En revanche, il est impossible que deux opérations ne se distinguent que par leur valeur retournée.

Signature d'une méthode

La Signature d'une méthode comporte la liste des arguments avec leur type et le type retourné par l'opération

- exemple : ***calculAge (dateNaiss : Date) : int***
- Le polymorphisme est la définition de plusieurs signatures pour la même opération. Exemple:
- ***calculAge (dateNaiss : Date) : int***
- ***calculAge () : void***

Méthode de classe

- Comme pour les attributs de classe, il est possible de déclarer des méthodes de classe. Une méthode de classe s'utilise au niveau de la classe, sans tenir compte des instances.
- C'est le cas par ex. de méthodes de tri des instances, ou d'un compteur du nombre d'instances de la classe. Elle ne peut manipuler que des attributs *de classe* et ses propres paramètres. Cette méthode n'a pas accès aux attributs *de la classe* (*i.e.* des instances de la classe) puisqu'elle est définie *en dehors* de toute instance.
- L'accès à une méthode de classe ne nécessite pas l'existence d'une instance de cette classe.

Classe Abstraite

- une classe abstraite est vouée à se spécialiser (notion d’héritage & de Généralisation).

Encapsulation

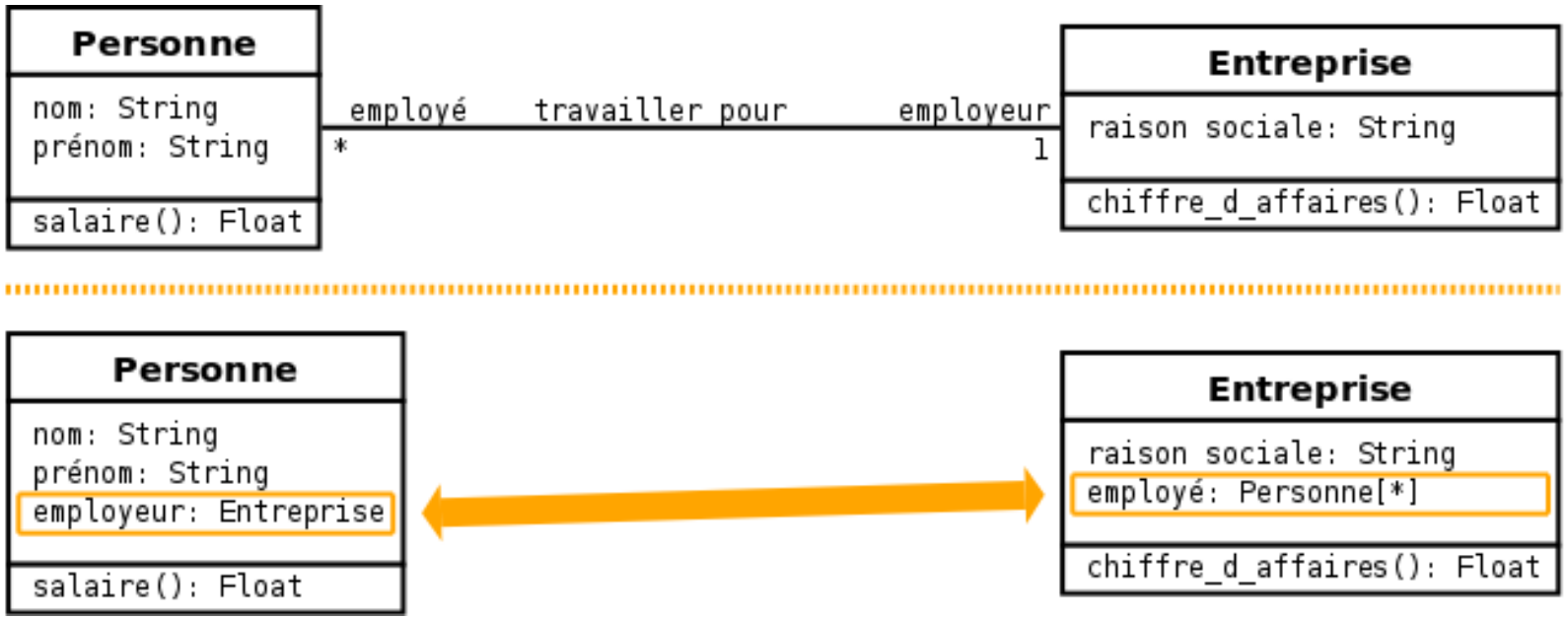
- L’encapsulation est un mécanisme consistant à incorporer les données et les méthodes dans une structure en cachant l’implémentation de l’objet, c’est-à-dire en empêchant l’accès aux données par un autre moyen que les services proposés. Ces services accessibles (offerts) aux utilisateurs de l’objet définissent ce que l’on appel l’interface de l’objet (sa vue externe).
- En d’autres termes, l’encapsulation assure l’intégrité des données contenues dans l’objet .
- Par défaut les valeurs des attributs d’un objet sont encapsulées dans l’objet et ne peuvent pas être manipulées directement par un autre objet.
- Des règles de visibilité précisent la notion d’encapsulation qui a pour but entre autre de réduire le temps d’accès aux attributs.

Visibilité

- ***Public* ou + :** qui indique que l'attribut est visible pour toutes les classes. En d'autres termes, tout élément qui peut voir le conteneur peut également voir l'élément indiqué.
- ***Protected* ou # :** seul un élément situé dans le conteneur ou un de ses descendants peut voir l'élément indiqué.
- ***Private* ou - :** seul un élément situé dans le conteneur peut voir l'élément.

Relations entre classes

Association binaire

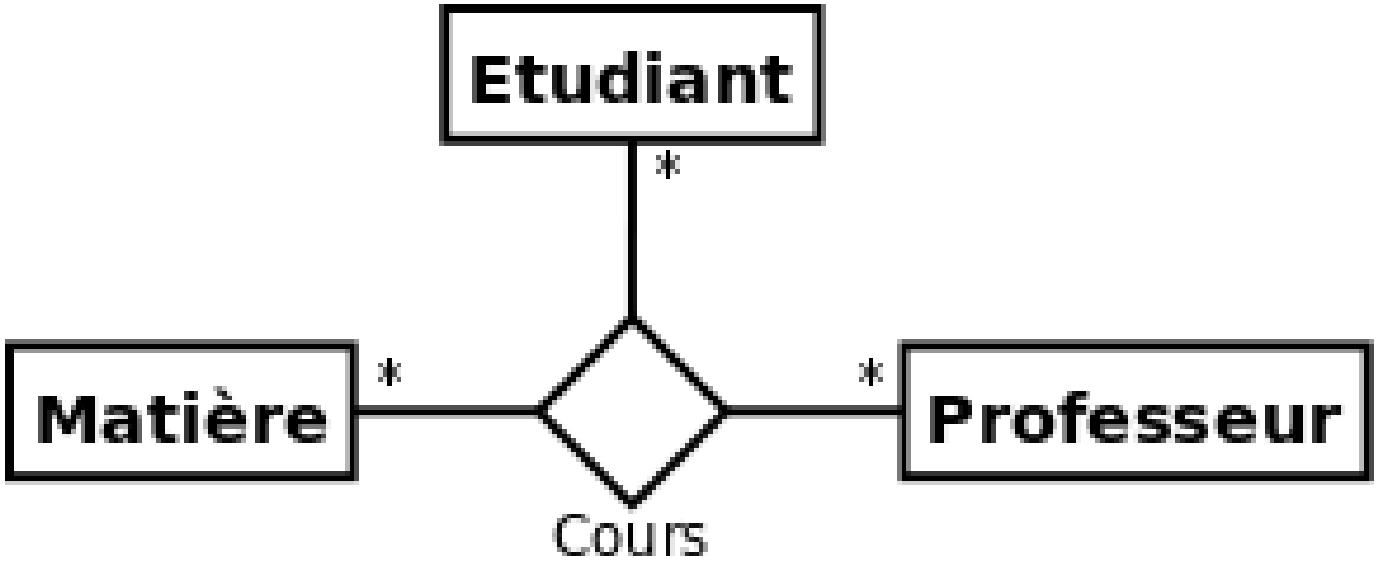


Association binaire

Chaque terminaison d'association peut définir les multiplicités suivantes :

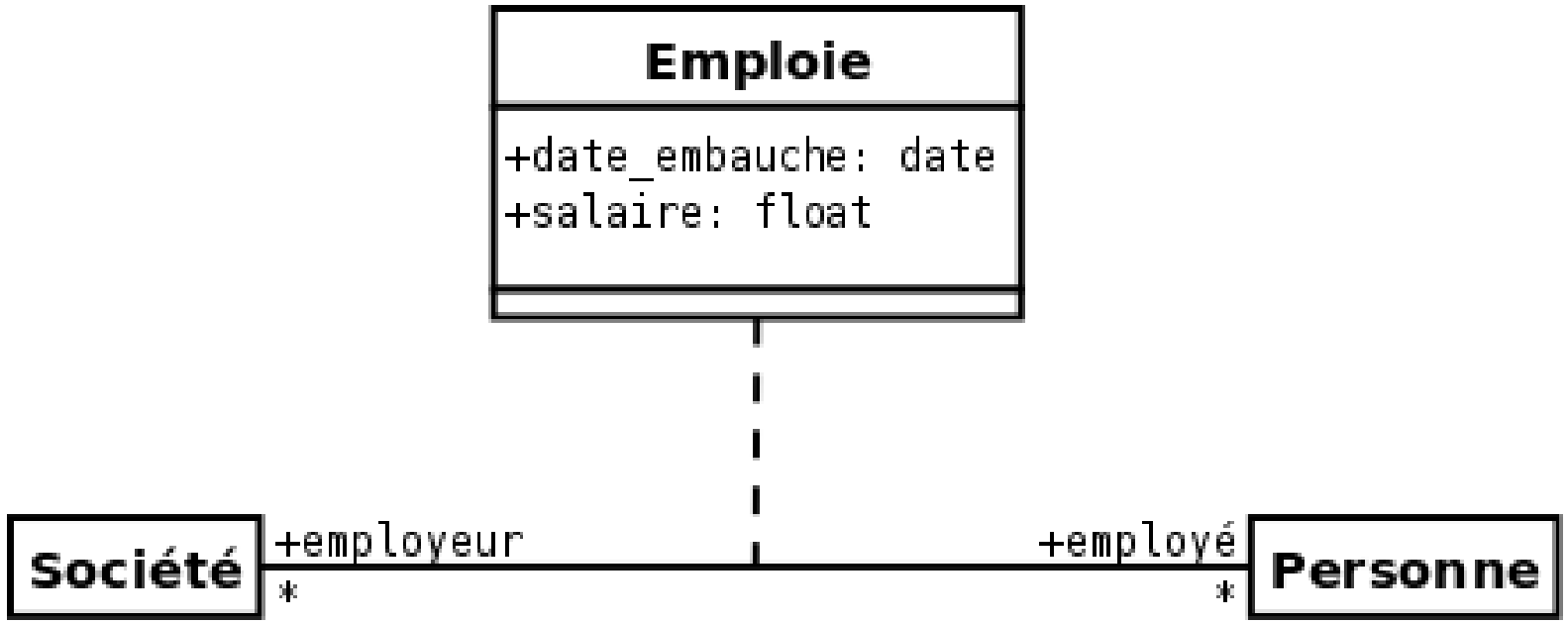
- 1 : C'est la valeur par défaut de toute terminaison qui précise qu'il y a un et un seul élément.
- 0..1 : Possibilité d'avoir 0 ou 1 élément.
- 1..* : Possibilité d'avoir 1 à n éléments.
- 0..* : Possibilité d'avoir 0 à n éléments.
- n : Définit un nombre précis d'éléments.
- n..m : Définit une fourchette précise d'éléments.
- * : équivalent de 0..*

Association n-aire

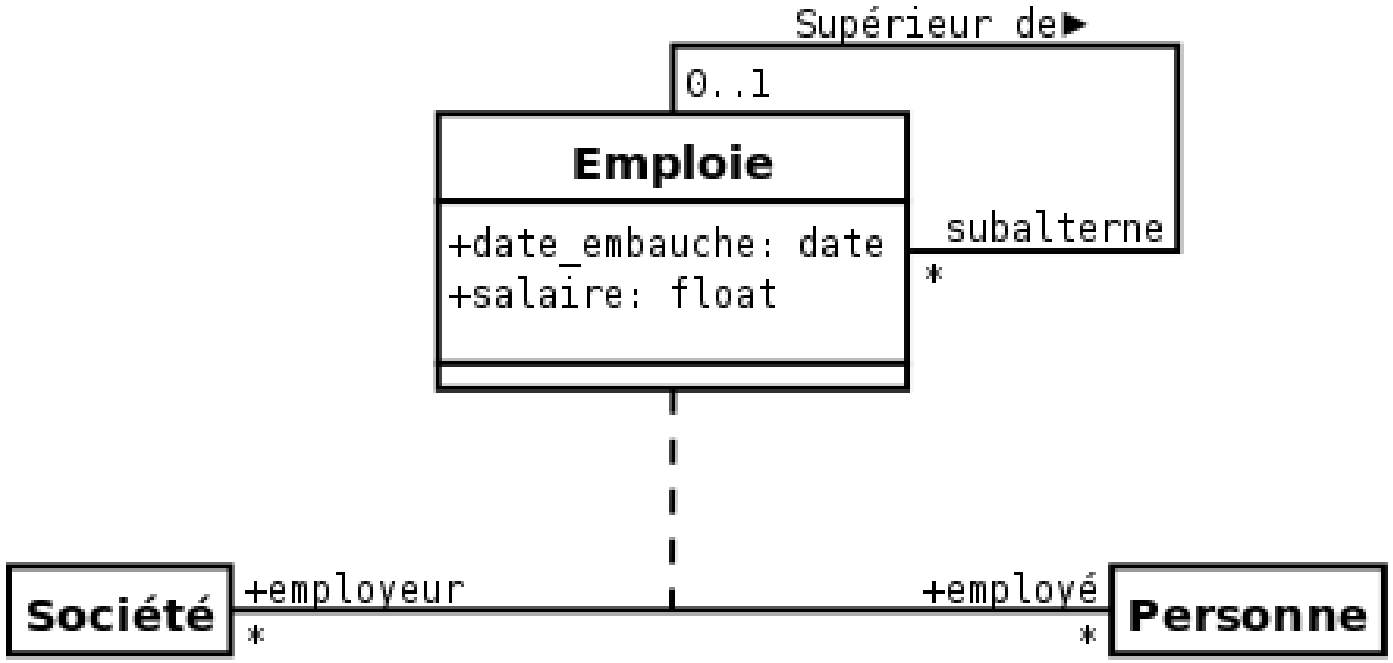


Navigabilité

Classe association

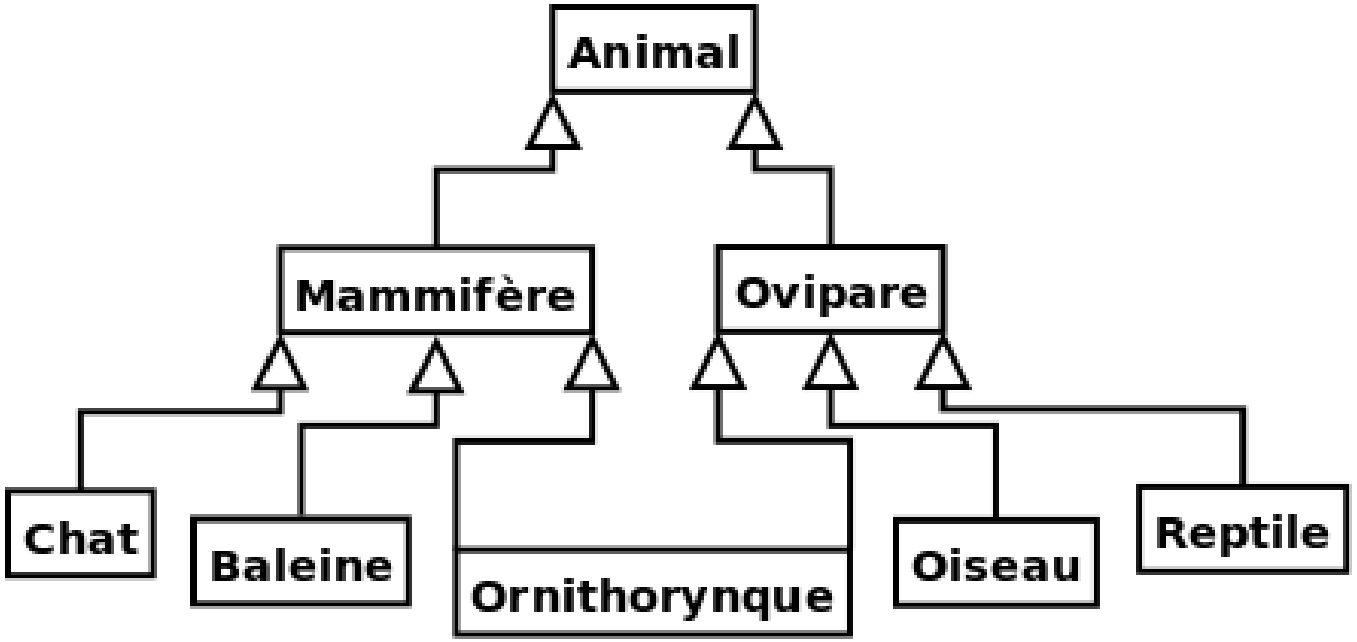


Association sur une classe association



Agrégation et composition

Héritage

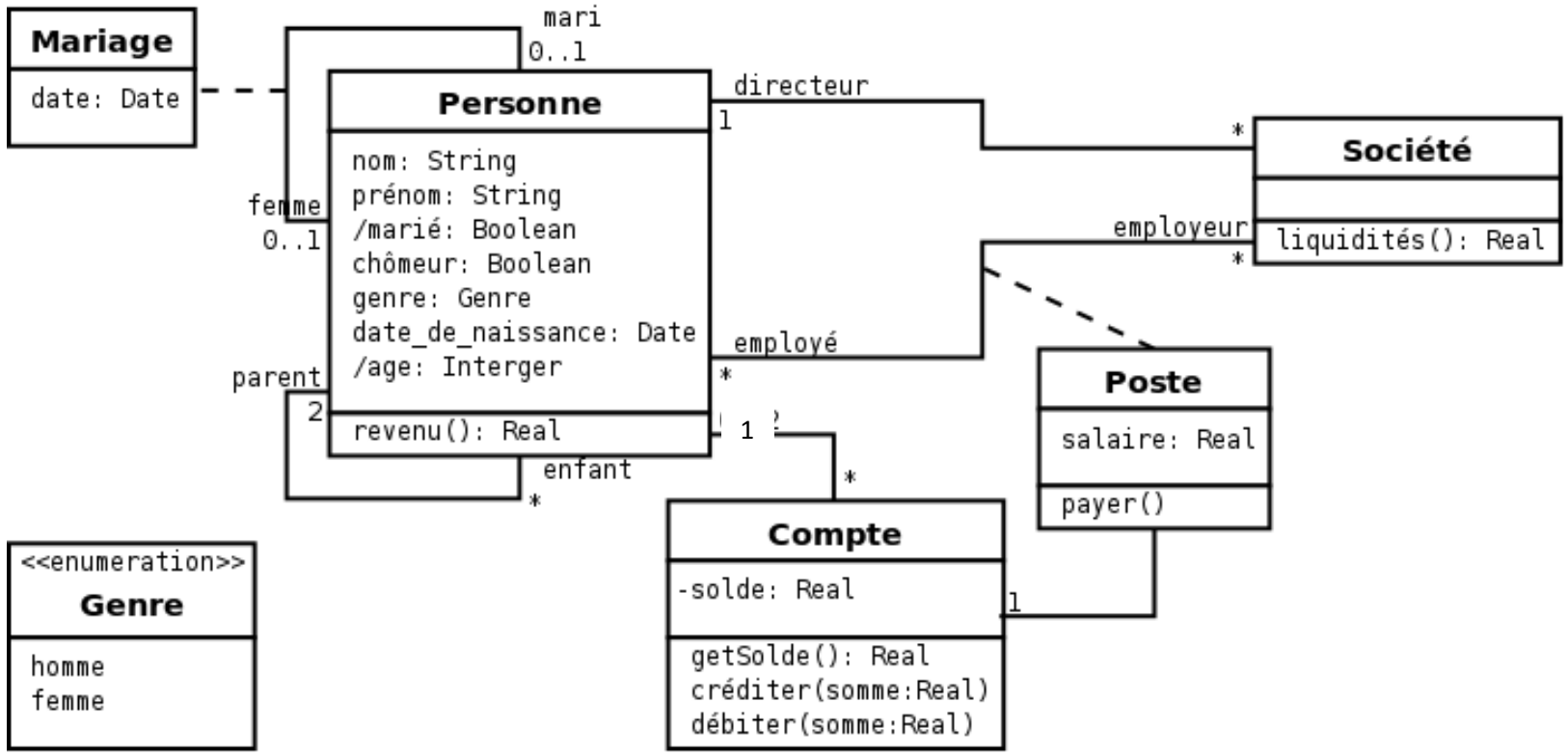

Animal
Mammifère
Ovipare
Chat
Baleine
Ornithorynque
Oiseau
Reptile

- la classe enfant possède toutes les caractéristiques de ses classes parents, mais elle ne peut accéder aux caractéristiques privées de cette dernière ;
- une classe enfant peut redéfinir (même signature) une ou plusieurs méthodes de la classe parent.
- toutes les associations de la classe parent s'appliquent aux classes dérivées ;
- une instance d'une classe peut être utilisée partout où une instance de sa classe parent est attendue. Exp. toute opération acceptant un objet d'une classe *Animal* doit accepter un objet de la classe *Chat* ;
- une classe peut avoir plusieurs parents, on parle alors d'héritage multiple .

Exemple

- Une personne est caractérisée par: Un nom, un prénom, un âge calculé à partir de sa date de naissance, un genre, il peut être marié ou pas, chômeur ou pas.
- Cette personne est marié avec une autre personne à une date précise, et peut être parent ou enfant d'une autre personne.
- Cette personne travaille dans une société, elle peut être « directeur » de cette société ou un simple employé.
- occupant un peste avec un salaire calculé par une opération.
- Cette personne peut (ou ne pas) posséder un compte, ce compte est caractérisé par un sold qu'on souhaite connaitre à tout moment, ainsi que le débiter ou le créditer (via le paiement)

Solution…


Mariage
date: Date
mari
0..1
Personne
nom: String
prénom: String
/marié: Boolean
chômeur: Boolean
genre: Genre
date_de_naissance: Date
/age: Interger
revenu(): Real
femme
0..1
parent
2
enfant
*
directeur
1
*
Société
liquidités(): Real
employeur
*
employé
*
Poste
salaire: Real
payer()
1
*
Compte
-solde: Real
getSolde(): Real
créditer(somme:Real)
débiter(somme:Real)
1
<<enumeration>>
Genre
homme
femme