Cours 4 : UML
diagramme de cas d'utilisation
diagramme de classes

1ière partie

UML

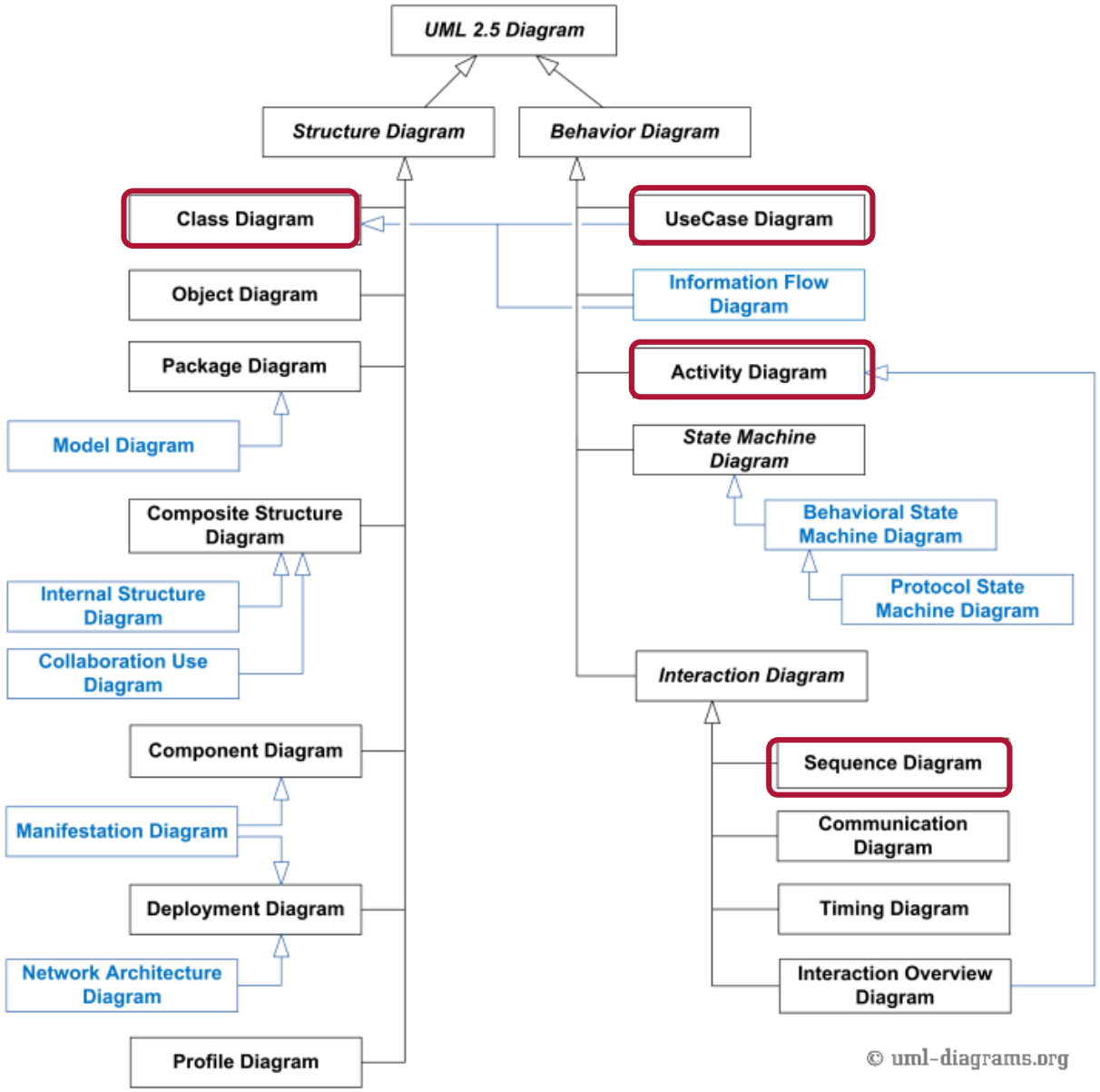
- **1994 : 50 méthodes objet**
- **Fusion des trois méthodes objet**
  - **OMT** (Rumbaugh 1990)
  - **Booch** (Booch)
  - **OOSE** (Jacobson)
- **UML est le résultat d'un large consensus**
  - très nombreux acteurs industriels (IBM, Oracle, HP, Microsoft, ... Rational Software, Softeam)
  - Normalisation ISO en cours
- **Tout le monde partage une même notation ("esperanto" informatique)**
  - ouvert
- **Modélisation en plusieurs phases**
  - enrichissement progressif

UML pourquoi faire ?

- **UML est un langage de modélisation**
  - vocabulaire et règles
  - centré sur la représentation d'un système
  - Adaptable
  - **pas de méthode de développement**
    - il ne fait que proposer un standard de communication et un certain nombre d’artefacts utiles
    - Besoin de le coupler avec un modèle de conception
    - Cycle de vie en V
    - Méthode Agile

UML conçu pour

- **Visualiser**
  - langage graphique et textuel
- **Spécifier**
  - précis sans ambiguïté
- **Construire**
  - possible de générer du code
- **Documenter**
  - spécifier l'architecture du système
  - exprimer les besoins
  - spécifier les tests
  - modéliser la planification
  - gérer des versions



- 12 diagrammes statiques (structures)
- 11 diagrammes dynamiques (comportements)

Diagrammes de structure ou diagrammes statiques

- **Diagramme de classes**
  - ensemble de classes et de relations du système (« **extension d'E/R** »)
- **Diagramme d'objets (statique)**
  - ensemble d'objets des classes (cas réel ou prototype)
- **Diagramme de composants (statique)**
  - organisation des ressources (bibliothèques, fichiers, base de données…)
- **Diagramme de déploiement (statique)**
  - architecture et répartition d 'un système
  - il sert à représenter les éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage...) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent entre eux.

Diagrammes de structure ou diagrammes statiques

- **Diagramme des paquetages (Package diagram)**
  - un paquetage étant un conteneur logique permettant de regrouper et d'organiser les éléments dans le modèle UML,
- **Diagramme de structure composite (Composite Structure Diagram)**
  - depuis UML 2.x, permet de décrire sous forme de boîte blanche les relations entre composants d'une classe.
- **Diagramme de profils (profile diagram) :**
  - spécialisation et personnalisation pour un domaine particulier (Java, C#,C++) d'un meta-modèle de référence d'UML (depuis UML 2.2).

Diagrammes de comportement

- **Diagramme de cas d'utilisation**
  - ensemble de cas d’utilisation du système
  - représente les acteurs et leur activités
  - Facilite l’expression des besoins des utilisateurs
- **Diagramme d'états-transitions**
  - Cycle de vie d’objets (dynamique des états)
  - basé sur le réseau de pétri
- **Diagramme d'activité**
  - décrit les règles d’enchaînement des activités
  - aide à la conception de méthodes
  - basé sur le réseau de pétri

Diagrammes d'interaction ou diagrammes dynamiques

- **Diagramme de séquence**
  - exemple de fonctionnement pour un scénario donné
- **Diagramme de collaboration**
  - décrit les collaboration entre les objets (proche diagramme de séquence)
- **Diagramme de communication**
  - représentation simplifiée d'un diagramme de séquence se concentrant sur les échanges de messages entre les objets.
- **Diagramme global d'interaction**
  - permet de décrire les enchaînements possibles entre les scénarios préalablement identifiés sous forme de diagrammes de séquences
- **Diagramme de temps**
  - permet de décrire les variations d'une donnée au cours du temps.

Outils UML

- **Gratuit**
  - **Modelio**
  - ArgoUML (générateur un peu faible)
  - Papyrus (add on d'Eclipse) compliqué
  - Pour une liste des outils à jour voir https://en.wikipedia.org/wiki/List_of_Unified_Modeling_Language_tools

- **En ligne**
  - Par exemple Draw io de Google drive

Diagramme de cas d’utilisation

(use case)

Diagramme cas d’utilisation (use case)

- **But**
  - regarder le système à construire de l'extérieur, du point de vue de l'utilisateur et des fonctionnalités qu'il en attend.
  - spécification de la fonctionnalité offerte par cette même entité
  - très utile en phase de spécification des besoins
  - décrit un ensemble cohérent de fonctions pour l'utilisateur

Diagramme de cas d'utilisation

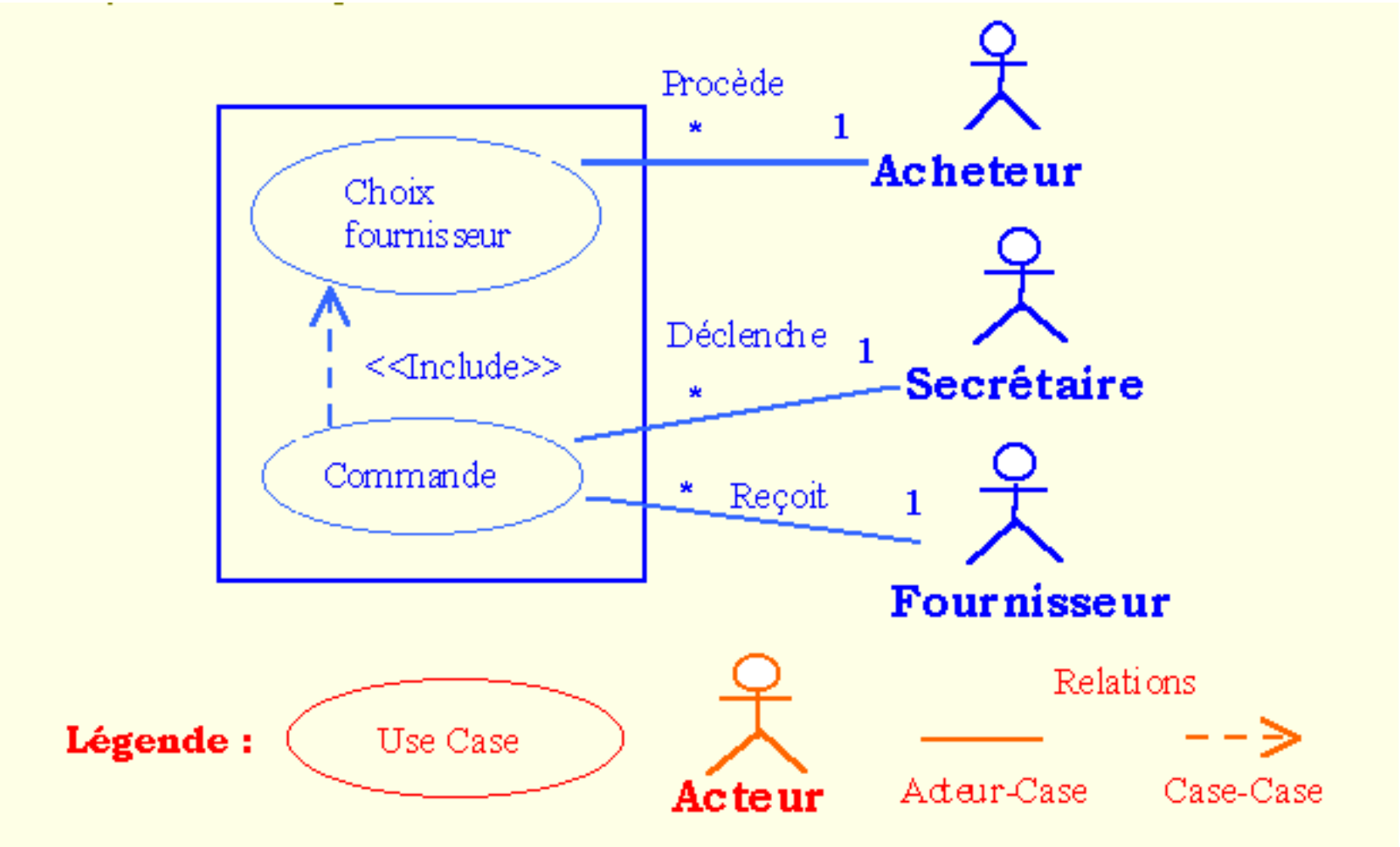
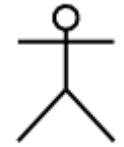


Diagramme de cas d'utilisation

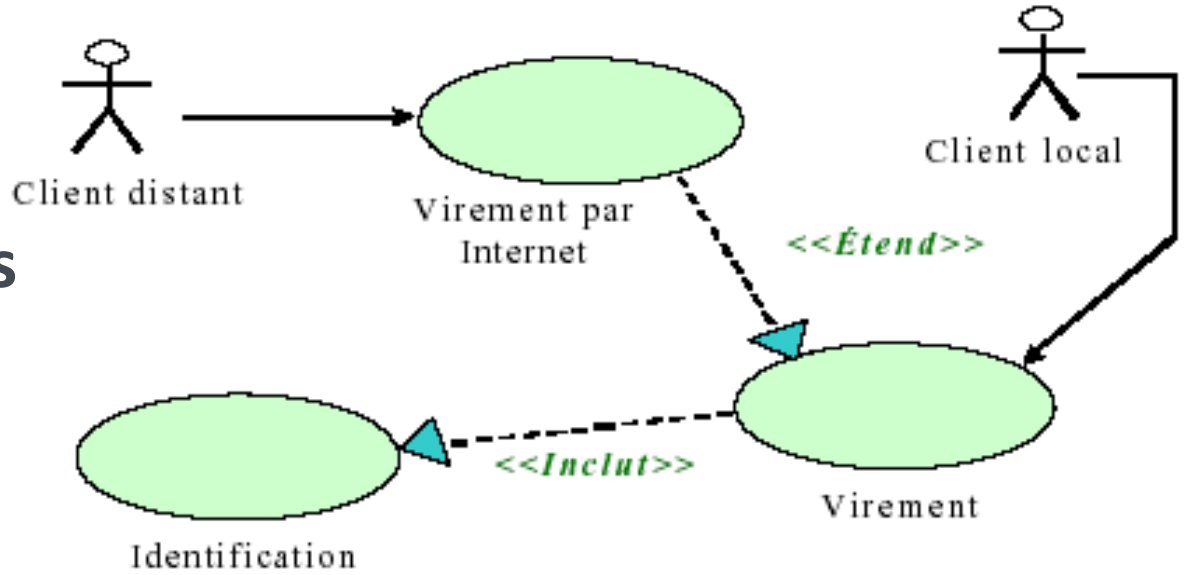
- **Le système**
  - Une boîte qui englobe les cas d'utilisations
- **Acteurs**
  - Le bonhomme (stick man)
  - Une boîte si système **extérieur**
- **Type d'acteurs**
  - *Principaux* pour un cas
    - ce cas rend service à cet acteur
    - Le stéréotype *<< primary >>* vient orner l'association reliant un cas d'utilisation à son acteur principal,
  - Secondaires pour un cas
    - Les autres acteurs, Ils sont sollicités pour des informations complémentaires.
    - le stéréotype *<< secondary >>* est utilisé pour les acteurs secondaires
- **Cardinalité (Multiplicité) d'associations**
  - * de 0 à n
  - 1…* de 1 à n
  - 0…1 0 ou 1
  - 1 1 et un seul



Client

Diagramme de cas d'utilisation

- **Les cas peuvent être interne**
  - Non relié à un acteur
- **Les cas peuvent être structurés**
  - <<inclut>> un cas inclut un autre
  - <<Etend>> variante d'un cas



Démarche d’utilisation des cas d’utilisation :

- **étapes**
  1. **Identifier les acteurs**
  2. **Identifier les cas**
  3. **Dessiner les diagrammes de cas**
  4. **Décrire les cas en écrivant des scénarii sous forme textuelle**
  5. **Récapituler les cas, les structurer et s’assurer de la cohérence d’ensemble**

Exercice : Guichet Automatique de Banque (G.A.B.)

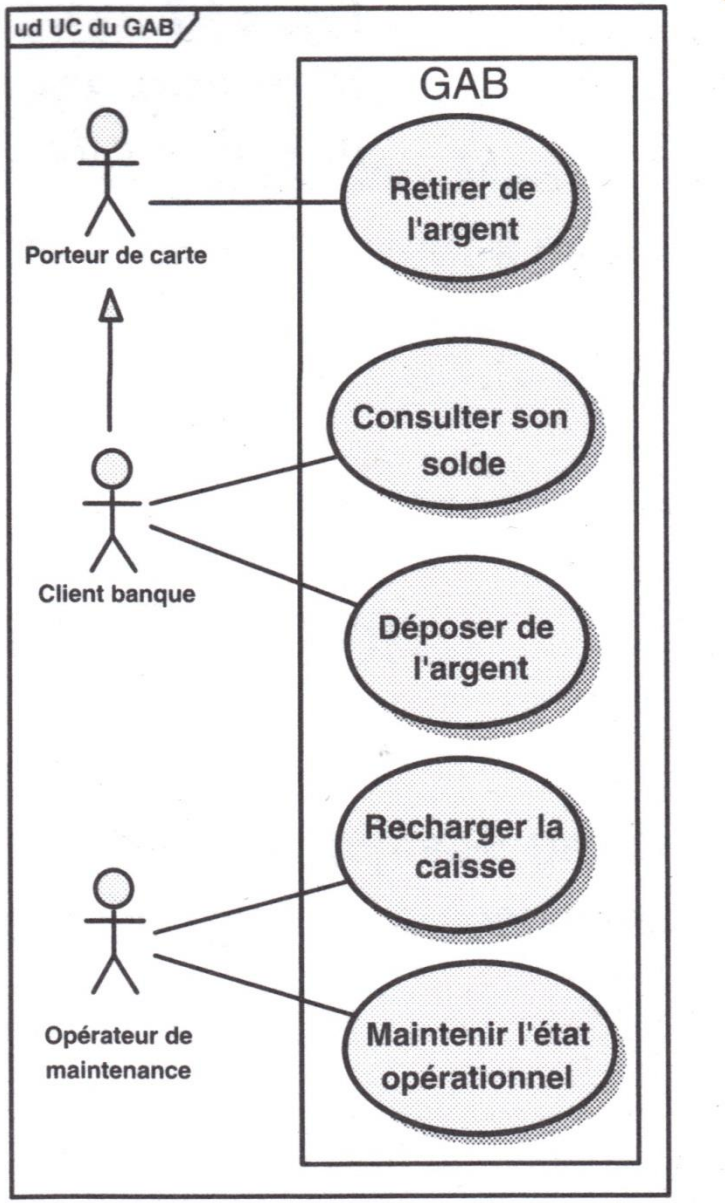
- **Le GAB offre les services suivants:**
  - Distribution d'argent à tout porteur de carte de crédit (carte Visa, ou de la banque), via un lecteur de carte et un distributeur de billets.
  - Consultation de solde de compte, dépôt en numéraire et dépôt de chèques pour les clients de la banque porteurs d'une carte de crédit de la banque.
- **Par ailleurs:**
  - Toutes les transactions sont sécurisées, via:
    - le Système d'Autorisation Visa, pour les transactions de retrait effectuées avec une carte Visa;
    - le Système d'Information de la banque, pour autoriser toutes les transactions effectuées par un client avec sa carte de la banque, mais également pour accéder au solde des comptes.
  - Le GAB nécessite des actions de maintenance, telles que le rechargement en billets du distributeur, la récupération des cartes avalées et des chèques déposés, etc.
- ***Remarque : L'énoncé précédent est sans doute incomplet et imprécis, comme il en est dans les projets réels.***

Exercice GAB

1. **Identification des acteurs humains et systèmes :**
2. **Identification des cas d'utilisations des humains (acteurs principaux)**
3. **Ajouter les systèmes extérieurs (acteurs secondaires)**

Correction GAB

- **Sans les acteurs secondaires**



Correction GAB

- Avec les acteurs secondaire
- Pour les clients

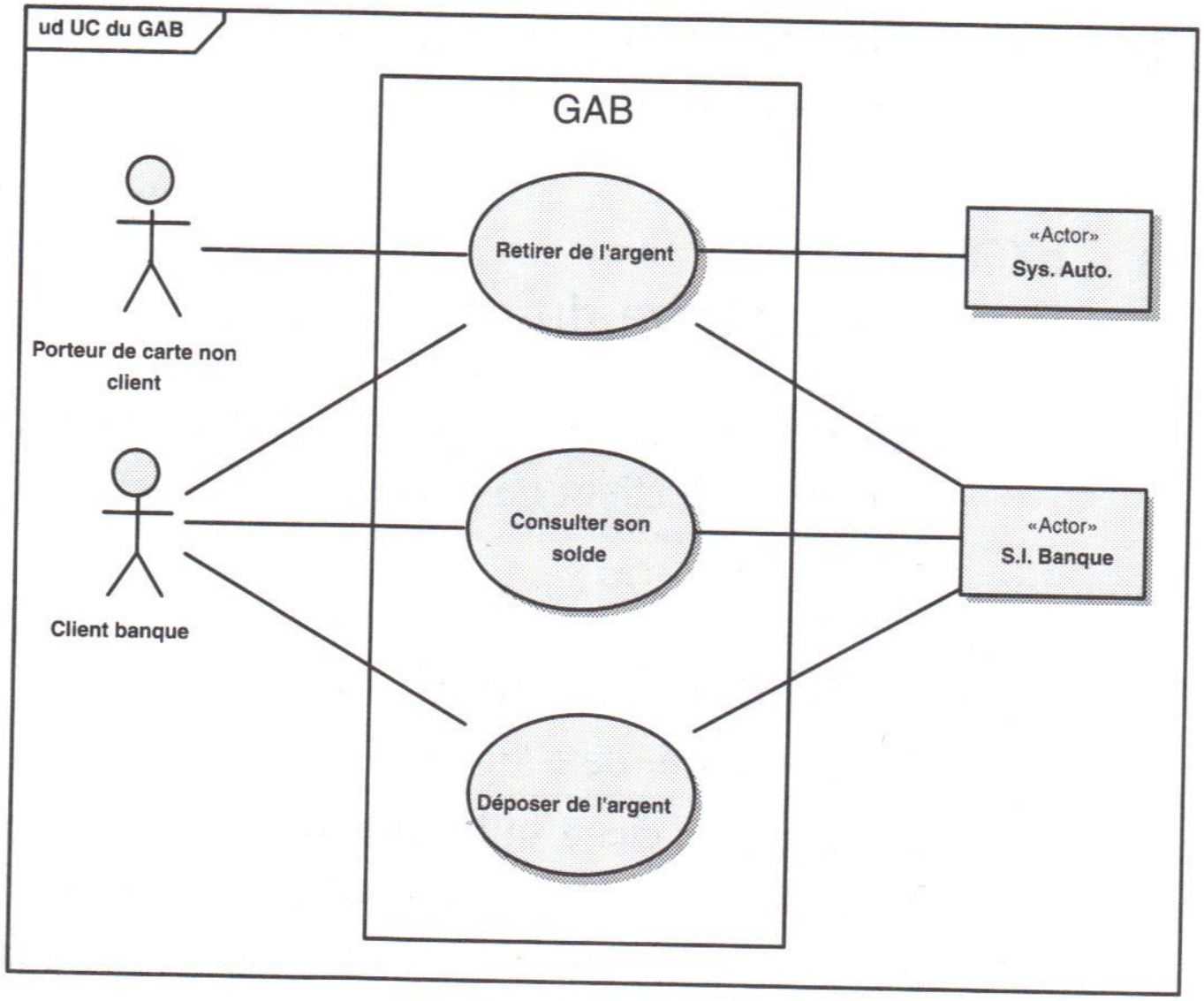


Diagramme de classes

statique

Diagramme de classes

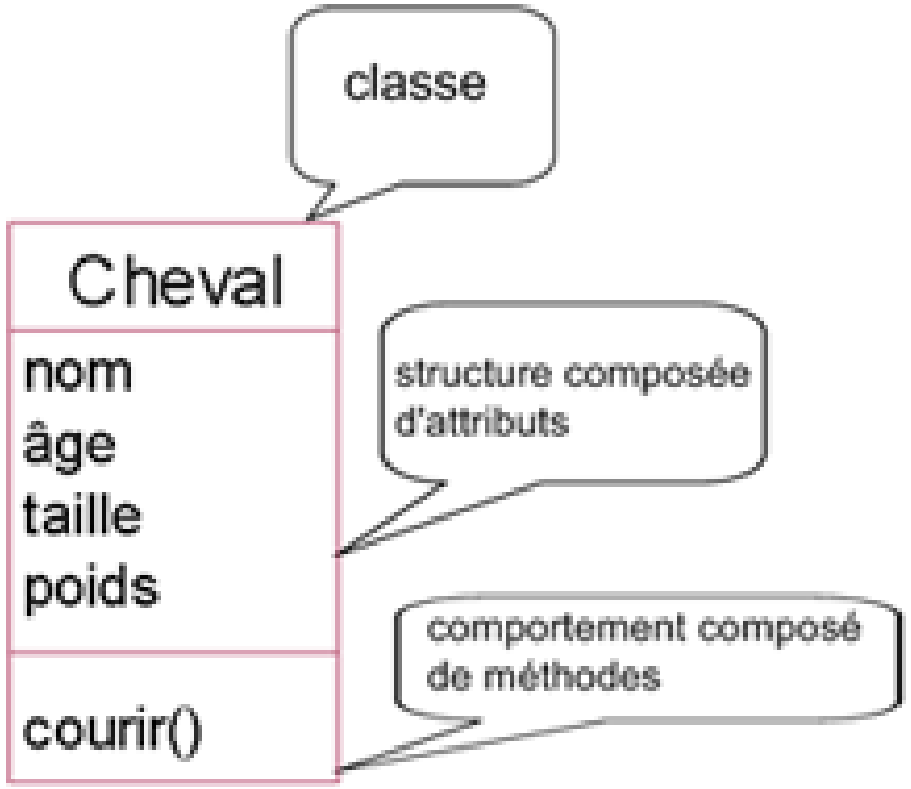
- **Reprise de E/R**
  - Entités→ classes
    - Identification gérer par le système (adresse)
  - Associations
    - Relie des classes
- **Ajout des notions objets et de notions de hauts niveaux**
  - Héritages
  - Encapsulation
  - Interfaces
  - Package
  - Associations d'agrégation de de compositions
  - Navigation d'une classe à une autres

Classe

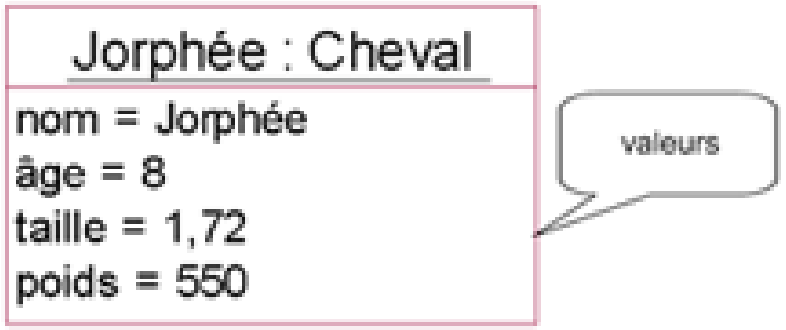
- **tout objet possède**
  - un ensemble d'attributs (sa structure)
    - variable destinée à recevoir une valeur
  - un ensemble de méthodes (son comportement).
    - ensemble d'instructions prenant des valeurs en entrée et modifiant les valeurs des attributs ou produisant un résultat.

- **Un ensemble d'objets similaires, (même structure et même comportement) forme une classe d'objets.**

Exemple de Classe et instance de classe



Objet



Les classes

- **Différents niveau de détail**
- **Brique de base**
  - Nom
  - attribut(s)
  - Opération(s) (méthode(s))

Nom de la classe

Représentation simplifiée

Nom de la classe :: Nom du paquetage
attribut attribut : type attribut : type = valeur par défaut
Opération Opération (paramètre mode_passage: type …) : type retour

mode_passage : in out inout

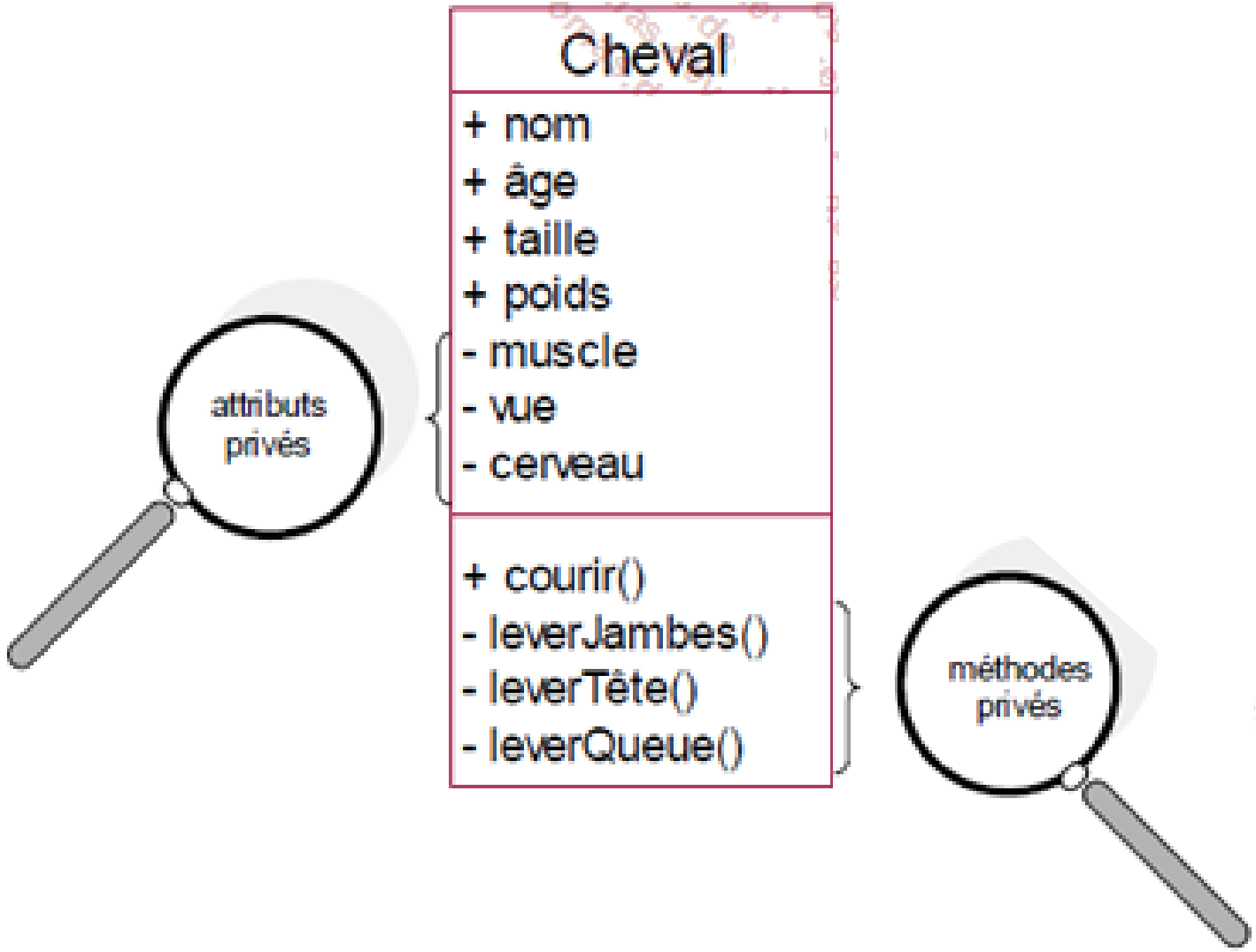
Représentation générale d'une classe

Encapsulation

Public = +

Protégé = #

Privé = -



Encapsulation

public	+	Élément non encapsulé visible par tous.
protégé	#	Élément encapsulé visible dans les sous-classes de la classe.
privé	-	Élément encapsulé visible seulement dans la classe.
paquetage	~	Élément encapsulé visible seulement dans les classes du même paquetage.

propriétés des variables

- **Valable pour attribuer des propriétés à une variable (attribut, paramètre et valeur de retour d'une méthode)**
  - {readOnly} : non modifiable. initialisée avec une valeur par défaut.
  - {ordered} : ensemble ordonnée de variable
  - {unique} : ensemble sans doublon.
  - {nonunique} : ensemble avec doublon possible

Propriété des paramètres d'une méthode

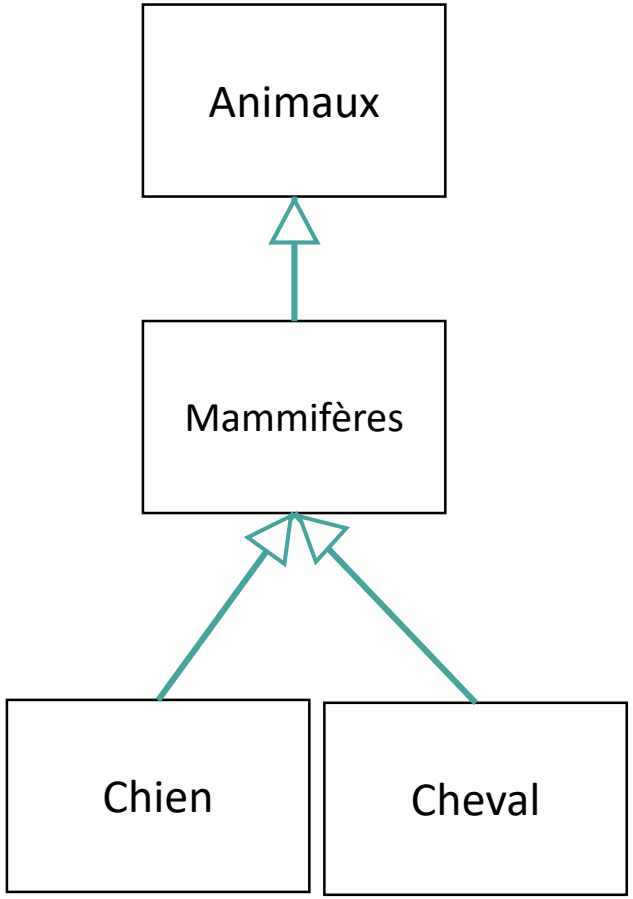
- **Entrée/ Sorite**
  - **in (par défaut)**
    - paramètre uniquement d'entrée passé par valeur.
    - Les modifications du paramètre ne sont pas disponibles pour l'appelant.
  - **out :**
    - paramètre de sortie uniquement.
    - la valeur finale est disponible pour l'appelant.
  - **inout :**
    - paramètre d'entrée/sortie. La valeur finale est disponible pour l'appelant.
- **Typage**
  - un nom de classe,
  - un type de donnée prédéfini

Héritage

- **sous-classe**
  - Une classe peut être un sous-ensemble d'une autre classe,
  - ce sous-ensemble devant toujours constituer un ensemble d'objets similaires.
  - C'est une spécialisation de cette autre classe.
- **Exemple**
  - **Spécialisation** : La classe des chevaux est une **sous-classe** de la classe des mammifères.
    - Un cheval est un mammifère
  - **Généralisation** (relation inverse) : La classe des mammifères est une **surclasse** de la classe des chevaux.

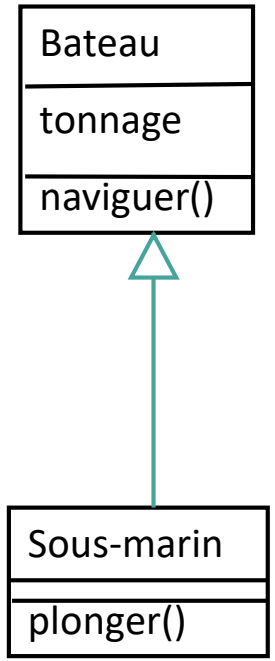
Hiérarchie de classes

- **La relation de spécialisation peut s'appliquer à plusieurs niveaux,**
  - Exemple
    - La classe des chevaux est une sous-classe de la classe des mammifères,
    - La classe mammifères est une sous-classe de la classe animaux.
    - La classe des chiens est une sous-classe de la classe des mammifères.

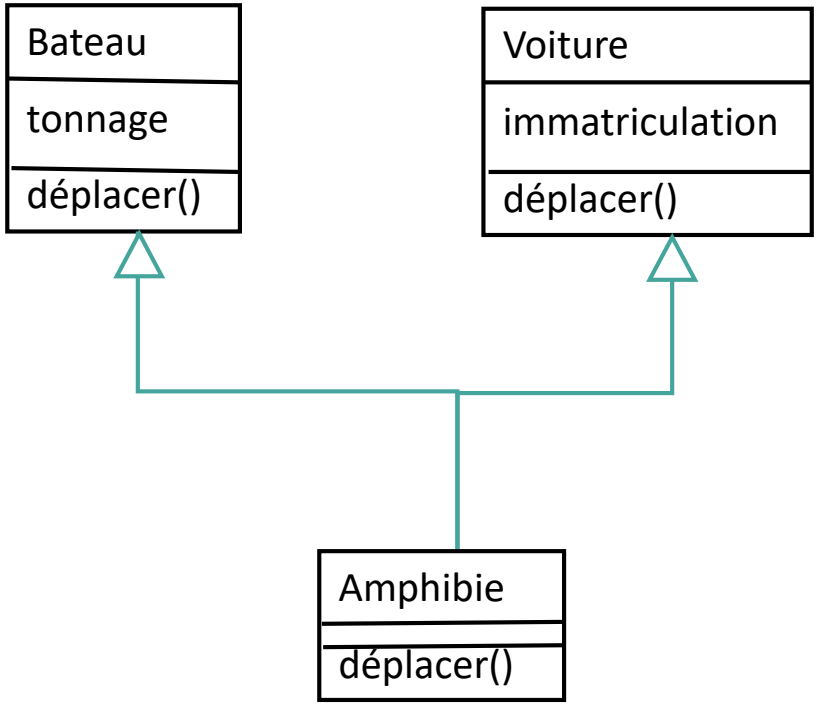


Généralisation

- **Simple**

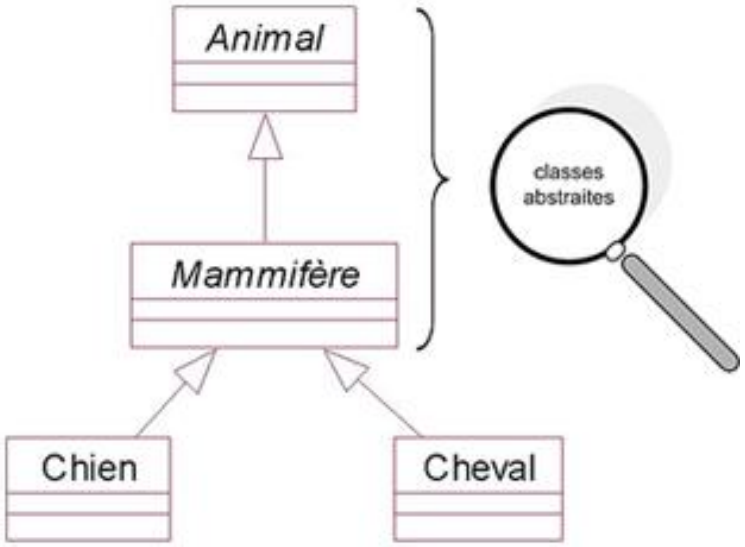


- **Multiple (pas en Java)**



Classe abstraite ou concrète

- **De type de classe**
  - Des classes qui possèdent des instances,
    - Ces classes sont appelées classes **concrètes**.
    - Exemple : les classes Cheval et Chien.
  - Des classes qui n'en possèdent pas directement,
    - Exemple : la classe Animal.
      - il existe des chevaux, des chiens,
      - le concept d'animal reste, quant à lui, abstrait.
    - Il ne suffit pas à définir complètement un animal.
    - La classe Animal est appelée une **classe abstraite**.

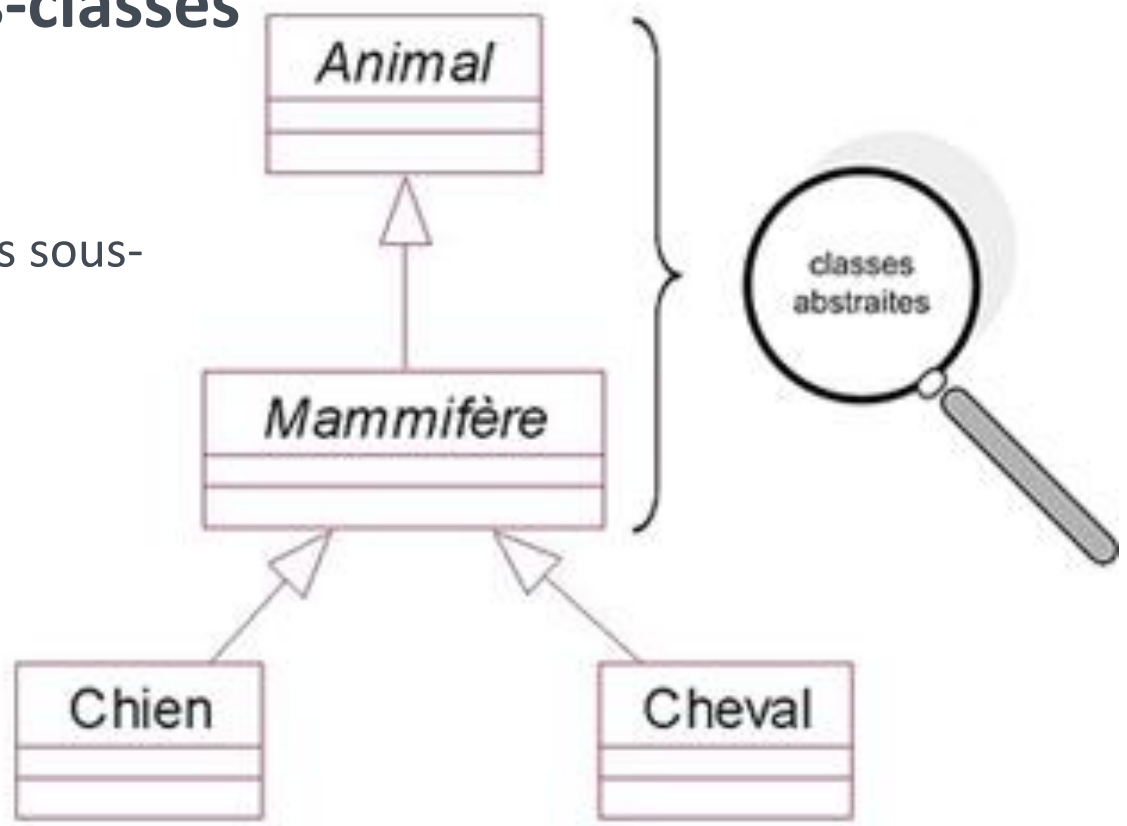


Exercice
diagramme UML matériel Informatique

- **Définir les hiérarchies de classes entre les classes suivantes**
  - Ordinateur, Imprimante, Encre, Consommable, Ramette de papier, Ordinateur portable, unité centrale, Ecran, clavier, périphérique, Matériel informatique, Souris,

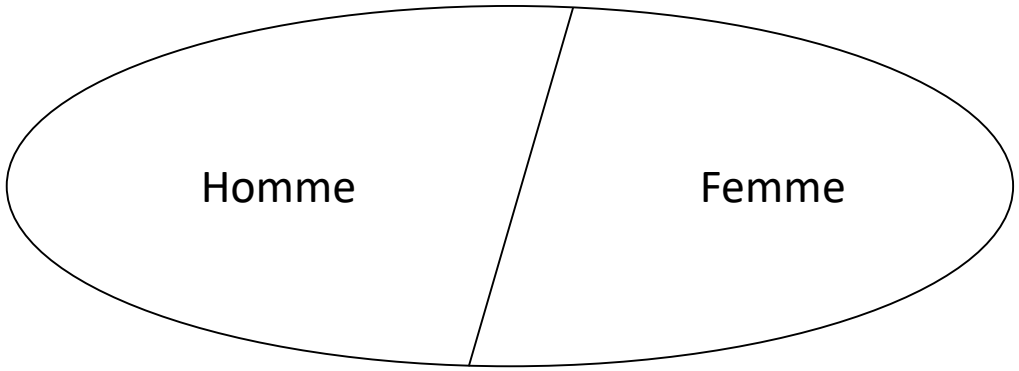
Classe abstraite

- **Une classe abstraite possède des sous-classes concrètes.**
  - Elle sert à factoriser
    - des attributs et des méthodes communs à ses sous-classes.
- **En UML, le nom des classes abstraites apparaît en caractères italiques ou avec le mot {abstract} post fixé**



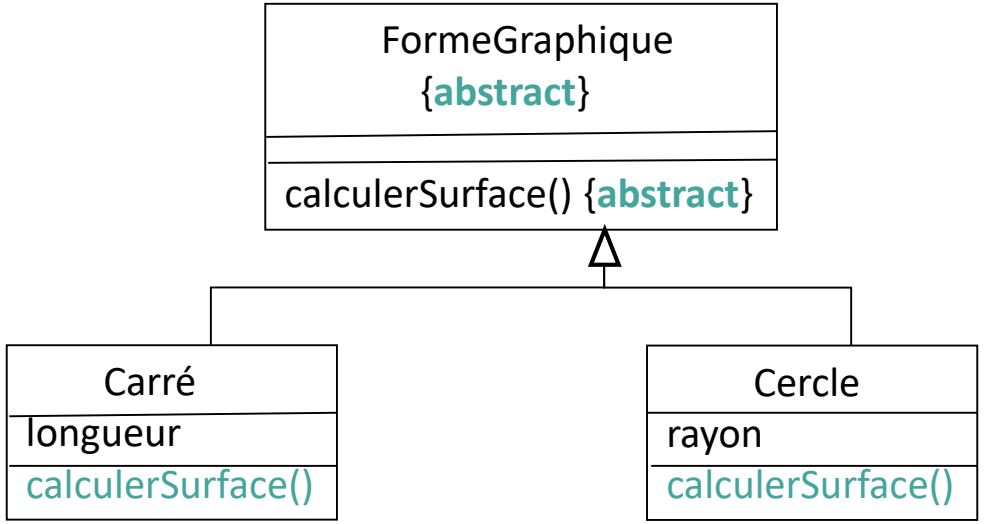
Classes abstraites

- **Une classe abstraite {abstract} est une classe dont les instances sont obligatoirement les instances d'une des classes filles**
  - exemple
    - Humain (abstrait)
      - homme
      - Femme
    - Homme et Femme forme une partition de la classe Humain



Opération abstraite : polymorphisme

- **Définition**
  - opération d'une classe abstraite redéfinie dans toutes les classes filles
- **Exemple**



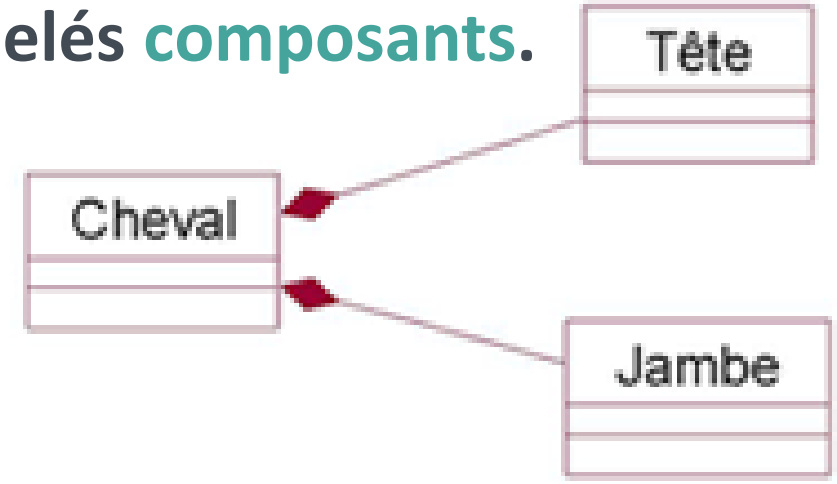
Les deux méthodes calculerSurface ont un comportement différent

Exercice (suite)
diagramme UML matériel Informatique

- **Pour cet exercice, quelles classes peuvent être abstraite ?**

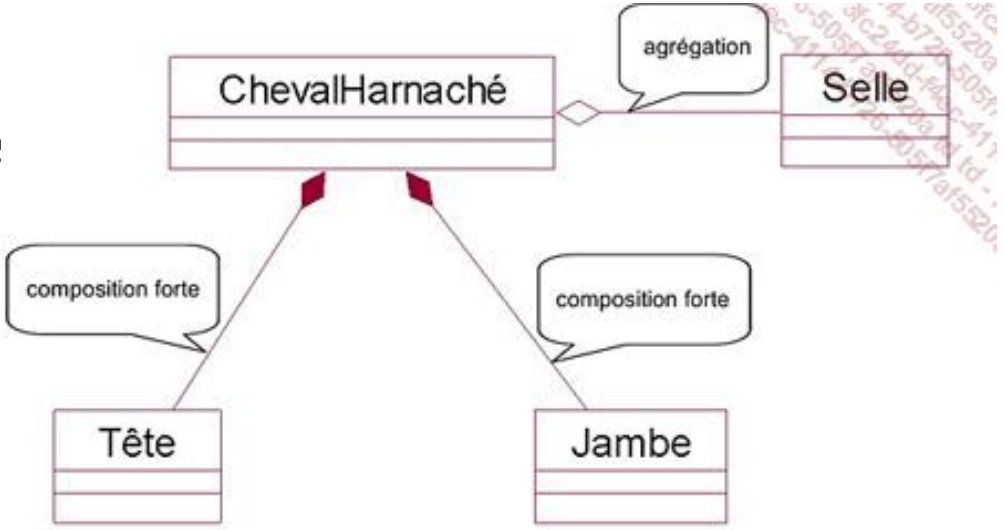
La composition

- **Un objet peut être complexe et composé d'autres objets. → composition.**
- **Elle se définit au niveau de leurs classes,**
- **Des liens sont bâtis entre les instances des classes.**
- **Les objets formant l'objet composé sont appelés composants.**
- **Exemple**
  - Un cheval est un exemple d'objet complexe. Il est constitué de ses différents organes (jambes, tête, etc.).



Deux type de compositions

- La **composition faible ou agrégation** ;
  - les composants peuvent être partagés entre plusieurs objets complexes. (**losange vide**)
- La **composition forte ou composition**,
  - les composants ne peuvent être partagés
  - la destruction de l'objet composé entraîne la destruction de ses composants (**losange plein**).
- **Exemple**



Exercices

- **Quel est le type de l'association (agrégation ou composition) entre ces classes ?**
  - Footballeur et équipe de football
  - Candidature et CV, lettre de motivations, lettres de recommandation
  - Classe et Etudiant
  - Bâtiment et pièce

Exercice (suite)
diagramme UML matériel Informatique

- **Quelle relation de composition et d'agrégation pouvons nous définir ?**

Les associations

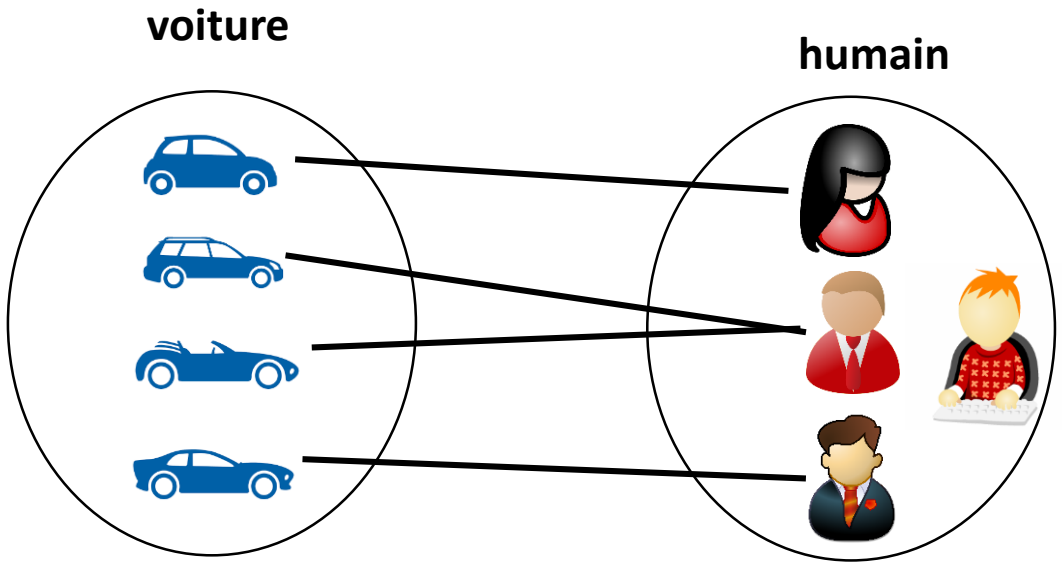
- **définition**
  - Une association est une relation entre des classes
  - Elle décrit les connexions structurelles entre leurs instances. Une association indique donc qu'il peut y avoir des liens entre des instances des classes associées.
  - Une association permet la description d’un concept à l’aide d’autres concepts

Les associations entre classes

Un humain a de 0 à N véhicules
Une voiture a un et un seul propriétaire

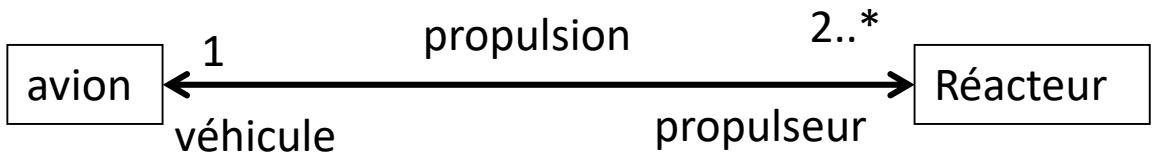
- **Notation**

- **Vue ensembliste**



Représentation des associations

- **Le nom**
- **les cardinalités**
- **Les rôles**
- **la navigabilité (← →)**
- **Remarques**
  - Pour lire une association il faut se situer du point de vue de chaque classe liée
    - L'avion possède au minimum 2 réacteurs et peut en avoir un nombre illimité
    - Les réacteurs jouent le rôle de propulseur pour l'avion
  - ici ont ne gère pas les avions en maintenance, pour lesquels les réacteurs peuvent ne pas être présents.



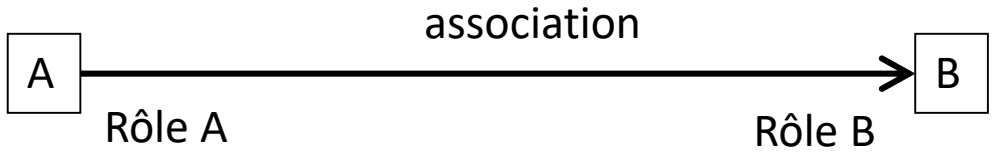
cardinalité

- Exactement un 1
- Exactement i
- Plusieurs (0 ou Plus) *
- Optionnelle 0..1
- 1 ou plus 1..*
- Cardinalité spécifique 1..2,4

- En général, uniquement les 4 cardinalités 1, *, 0..1 et 1..* sont gérées par les atelier de génie logiciel (AGL) UML

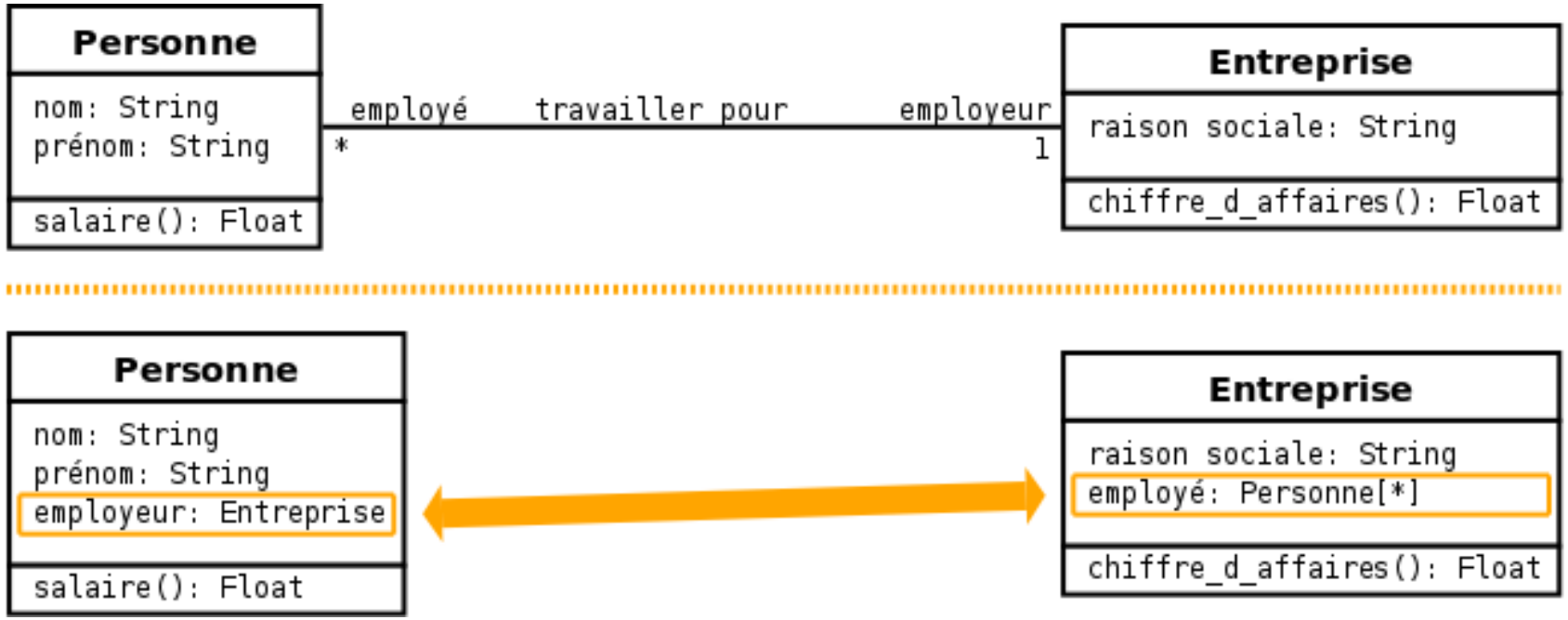
la navigabilité des associations

- **3 implications de navigabilité de A vers B**
  - Conceptuelle
    - Le concept de A s'appuie sur le concept de B
  - Logique
    - Les méthodes de A exploitent les méthodes de B
  - Physique
    - Les moyens d'accès de A vers B seront implémentés



Association versus couple d'attributs

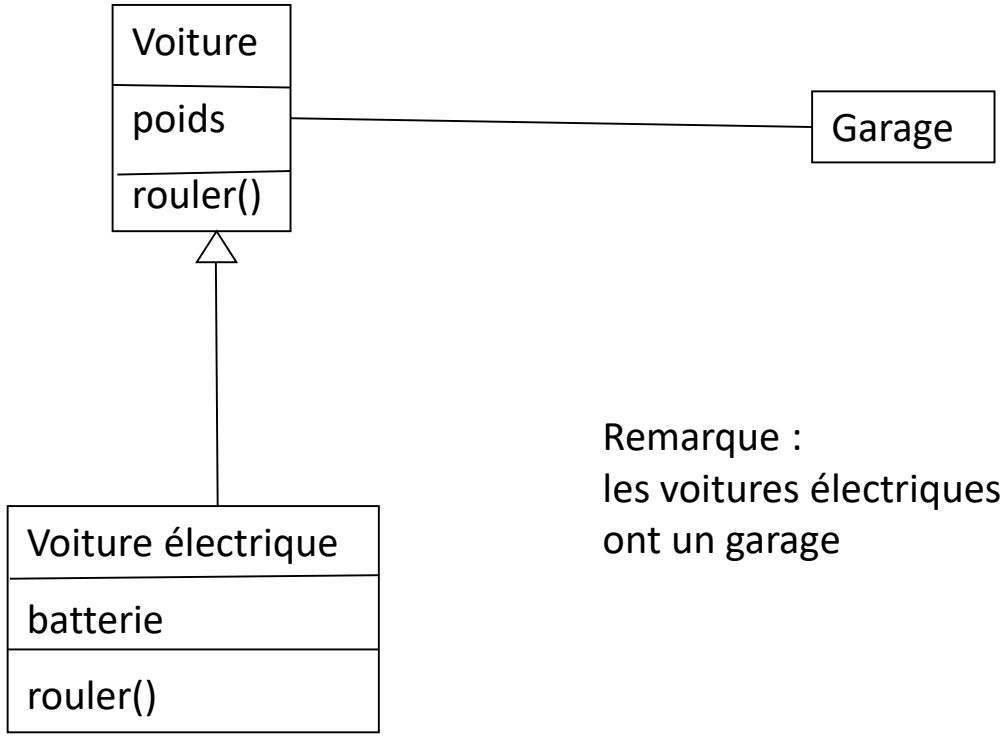
- **Deux manières de représenter une relation entre deux classes**



Association versus couple d'attributs

- **D'un point de vue implémentation (traduction en code source)**
  - C'est la même chose (sauf en SQL)
- **D'un point de vue conceptuel**
  - l'association est plus clair et plus simple
  - Elle constitue une entité distincte.
  - Il faut faire apparaître les associations en UML
  - Elle simplifie grandement la modélisation d'associations si elle sont complexes (n-aire ou avec des contraintes)
- **Dans les deux cas, c'est des propriétés des classes.**

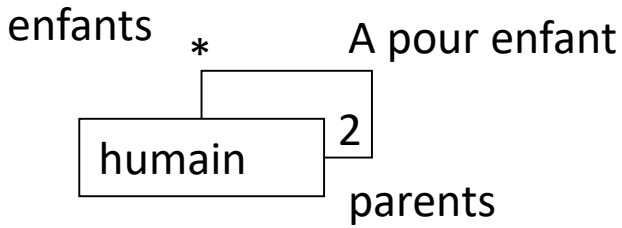
Généralisation des associations



Remarque :
les voitures électriques
ont un garage

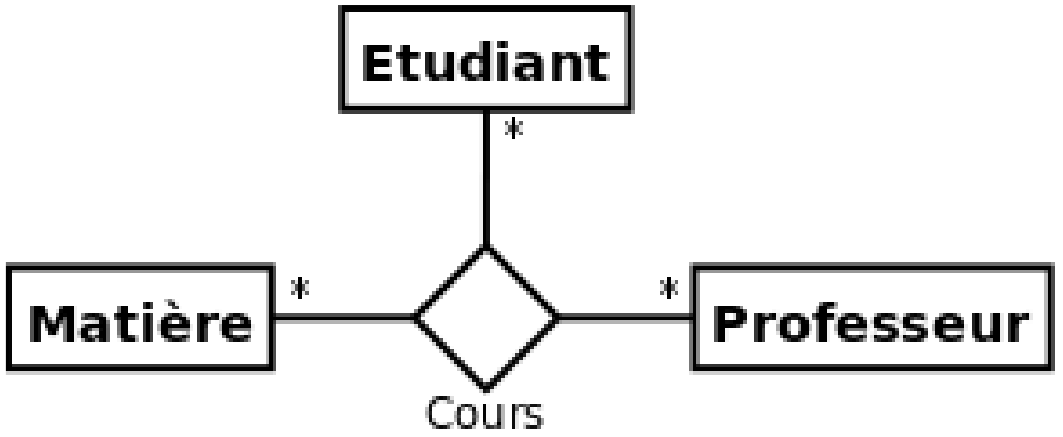
Associations spécifiques

- **Composition**
- **Agrégation**
- **Réflexives**

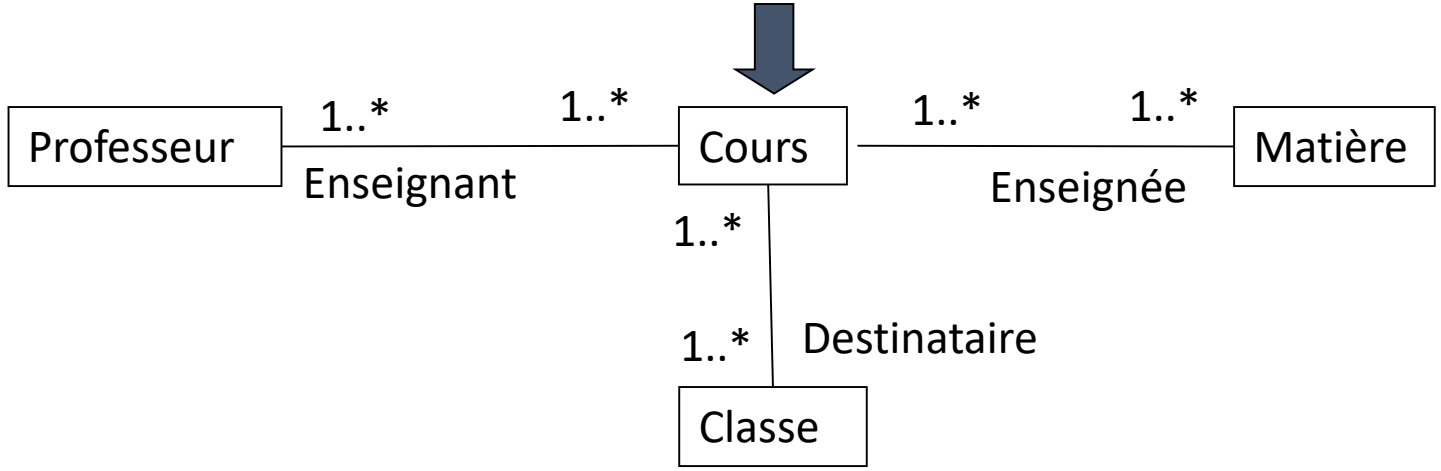


- **Association n-aire**
- **Association avec attributs**

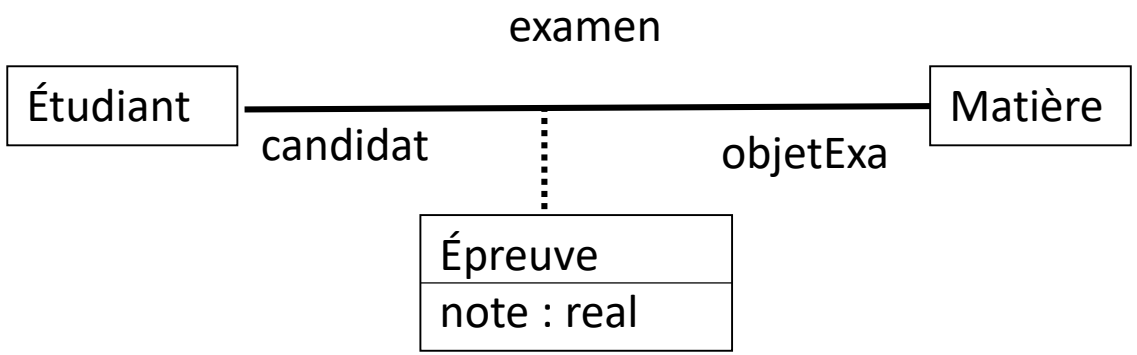
Association N-aires



Possibilité de transformer en classe

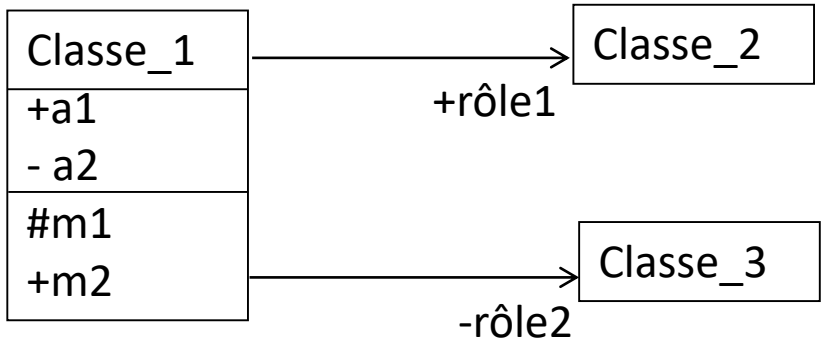


Association avec des attributs



Remarque :
les associations avec attributs sont souvent vues comme des classes

Visibilité et associations



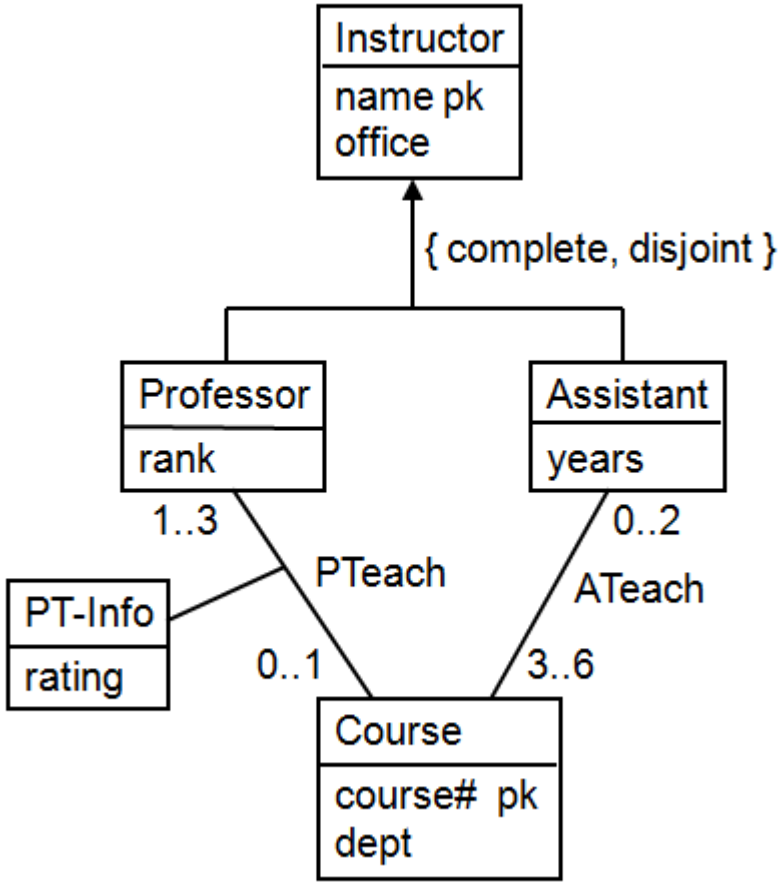
Remarques généralement :
attribut et rôle : -
méthode : +

Qualification des relations d'héritage

- **Différentes contraintes peuvent être ajoutées sur les relations d'héritage**
  - **Complétude**
    - {Complete} : Toutes les instances d'une classe sont des instances d'une de ses sous classes
    - {Incomplete} : Des instances d'une classe ne sont pas des instances
  - **Intersection**
    - {Disjoint }:
      - pas d'intersection possible
      - une instance d'une sous classe A ne peut pas être une instance d'une sous-classe B
    - {Overlapping} :
      - Intersection possible

Exemple et Exercices

- **Questions**
  - La classe instructor peut-elle être abstraite ?
  - Proposez une hiérarchie où les sous-classes ne sont pas disjointes ?
  - Proposez une hiérarchie où les sous-classes ne sont pas {complete}?



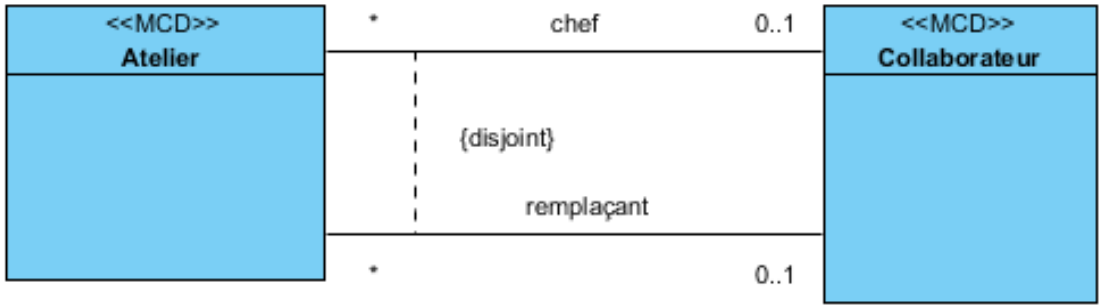
Qualification des associations

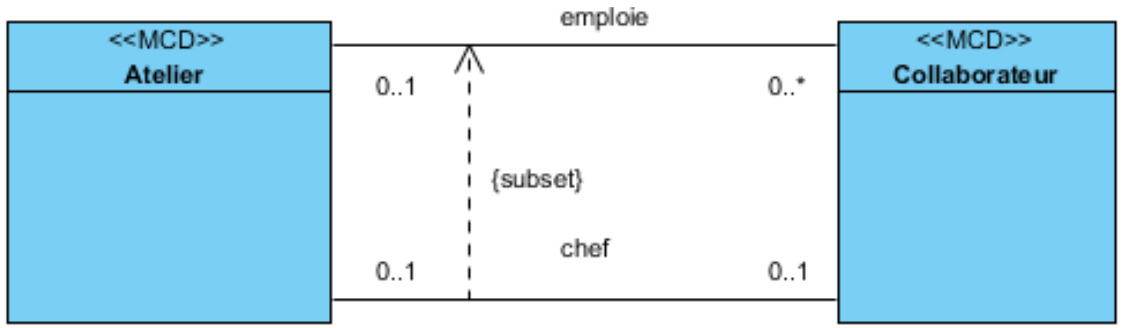
- **Différentes contraintes peuvent être ajoutées sur associations**
  - **ensembliste**
    - {Disjoint} ou {xor}:
      - pas d'intersection possible
      - Si x et y sont associées à l'aide de l'association A1, alors il ne peuvent pas être associé à l'aide de l'association A2
    - {subset} :
      - Si x et y sont associées à l'aide de l'association A2, alors ils sont associées à l'aide de l'association A1

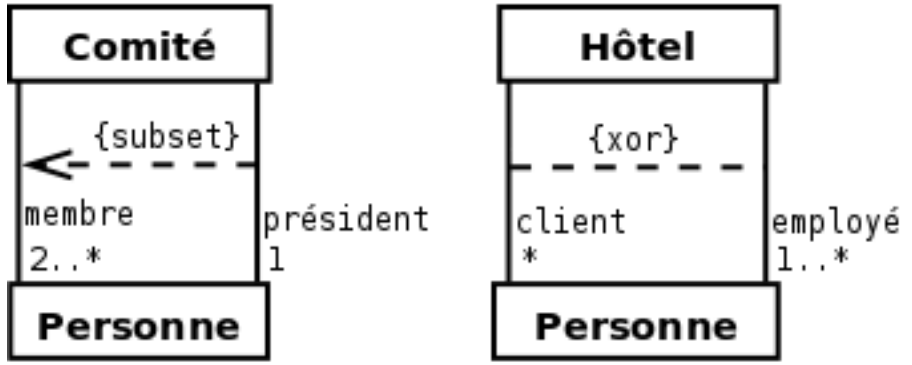
Exemple et Exercices

Questions

- Un chef peut il être un remplaçant ?
- Un chef est-il un employé ?
- Pouvons nous fusionner c'est deux premiers diagrammes ?
- Un président d'un comité est-il obligatoirement un membre du comité
- Pouvons nous être client de l'hôtel qui nous emploie ?

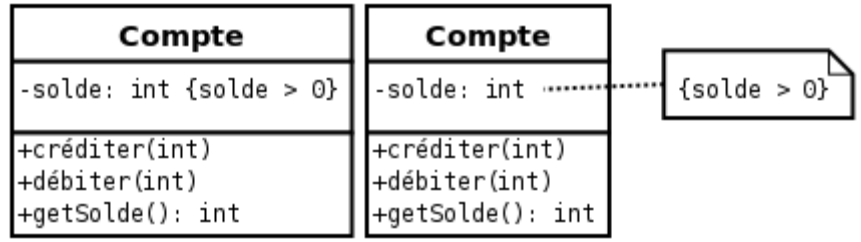






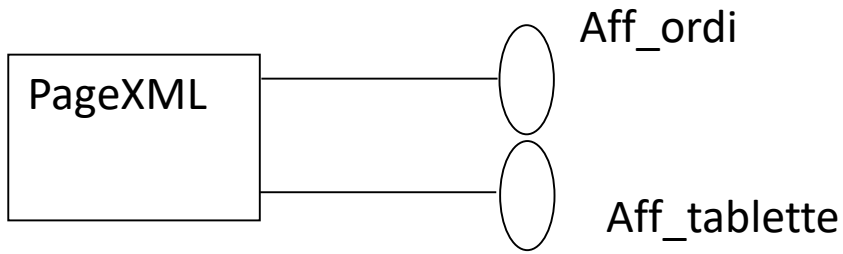
Autres contraintes sur les associations

- **Contraintes de type note**
  - Question : comment sont traduites ce type de contraintes ?
- **Ordonnancement {Ordered}**
  - L'ensemble des instances de ce rôle sont ordonnées
- **Frozen**
  - L'instance du rôle ne peut pas changer
- **addOnly**
  - Possibilité d'ajouter des instances à un rôle, mais pas dans supprimer



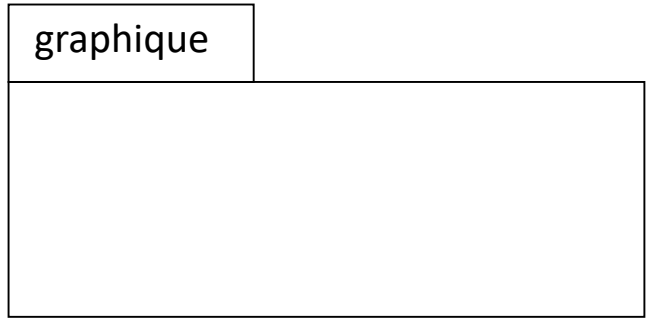
Les interfaces

- **Interface :**
  - abstraction d'un comportement
  - classe abstraite sans attribut
- **permet de définir plusieurs interfaces pour la même classe**



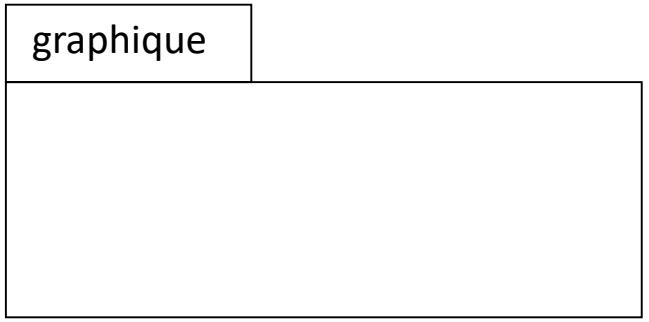
Les paquetages (packages)

- **Regroupement d'éléments de modélisation**
  - classes, paquetages, acteurs, diagrammes …
  - un élément (classe, …) appartient à un seul paquetage
  - un paquetage peut avoir des sous paquetages
  - les classes dans un paquetage ont des règles de visibilité (public, protégé, privé)



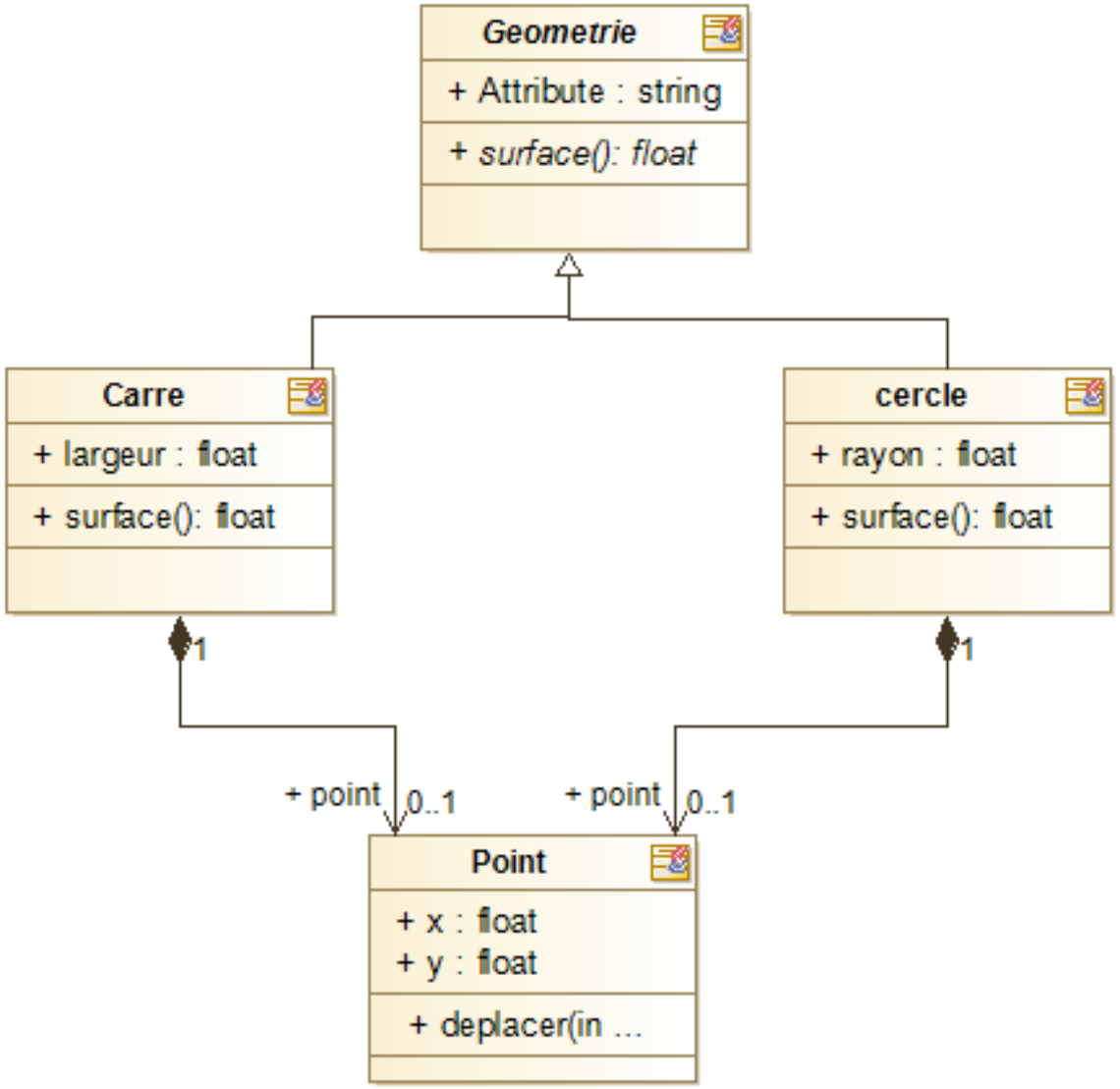
Les paquetages (packages)

- **Paquetages (packages)**
  - Les paquetages sont des **éléments d'organisation** des modèles.
  - Ils regroupent des éléments de modélisation, selon des critères purement logiques.
  - Ils permettent d'encapsuler des éléments de modélisation (ils possèdent une interface).
  - Ils permettent de structurer un système en catégories (vue logique) et sous-systèmes (vue des composants).
  - Ils servent de "briques" de base dans la construction d'une architecture.
  - Ils représentent le bon niveau de granularité pour la **réutilisation**.
  - Les paquetages sont aussi des **espaces de noms**.



Génération de code Java

- **Exemple (Modelio Java)**



Génération de code Java

- Dans le répertoire C:\Users\NomUtilisateur\modelio\workspace\NomProjet\src
- Code

```
public class Point {
    @objid ("3c84d139-077b-41ea-a165-48a7922fe8f6")
    public float x;
    @objid ("de6510ee-67aa-4364-b053-a4ad0b515548")
    public float y;
    @objid ("cfba5f2d-7b76-4bc2-bba8-b0c79c844ab4")
    public void deplacer(float deltaX, float deltaY) {
    }
}

public abstract class Geometrie {
    @objid ("8c9ebc66-4da6-42cb-bc33-f87f196d85e2")
    public String Attribute;
    @objid ("6b15723c-583f-4319-a7a5-ab0cdf6aefb9")
    public abstract float surface();
}
```

```
public class cercle extends Geometrie {
    @objid ("80b733d6-0383-44ab-a72f-cb0612f13f6e")
    public float rayon;
    @objid ("cb551914-3b49-4593-a4d1-78681a06eb90")
    public Point point;
    @objid ("bc9284af-6303-4e2a-a1fd-4042229d07af")
    public float surface() {
    }
}
public class Carre extends Geometrie {
    @objid ("656aac53-021d-4a4e-8b74-08623066fa15")
    public float largeur;
    @objid ("12f52c90-8d66-4819-8d65-627573c548ad")
    public Point point;
    @objid ("258f9e0b-0f97-4f15-b503-2547938dc19e")
    public float surface() {
    }
}
```

Exercice 1&2 : Diagramme de classes

- Dessiner les diagrammes de classes correspondant aux situations suivantes :
  - Un **polygone** est constitué de points. Un point possède une abscisse et une ordonnée.
  - Un **bateau** contient des cabines, occupées par des personnes qui effectuent des activités. Les personnes sont ou bien des guides, ou bien des animateurs, ou bien des passagers. Les guides expliquent des visites aux passagers et les animateurs animent des animations pour les passagers.

Exercice 3 : Diagramme de classes

- **Un ordinateur est composé d'un ou plusieurs moniteurs, d'un boîtier, d'une souris optionnelle et d'un clavier. Un boîtier a un châssis métallique, une carte mère, plusieurs barrettes de mémoire (RAM, ROM et cache), un ventilateur optionnel, des supports de stockage (SSD, disque dur, CD-ROM, DVD-ROM…) et des cartes périphériques (son, réseau, graphique). Un ordinateur possède toujours au moins un disque dur classique ou SSD**
- **Proposez un diagramme de classes qui représente l'architecture d'un ordinateur.**

Exercices 4 : Laboratoire de Recherche

- **Tiré du cours UML d'Huchard (Montpelier) http://www.lirmm.fr/~huchard/FOAD/metamodeleUML/metamodeleUML_HTML/Pages/rappeluml.html**
- **Nous nous proposons d'étudier quelques éléments relatifs au système d'information global d'un laboratoire de recherche. Un laboratoire de recherche accueille différents membres, qui peuvent être des chercheurs, des personnels administratifs ou des personnels techniques. Une personne peut être membre d'au plus deux laboratoires.**

Exercices 4 : Laboratoire de Recherche

Le laboratoire a un directeur qui doit être membre du laboratoire et ne peut diriger qu'un laboratoire. Tout membre est décrit par un nom. Un coût annuel de base est affecté à chaque catégorie de personnel (chercheur, administratif, technique, etc.) ; il est partagé par tous les membres d'une même catégorie ; un chercheur appartient à une et une seule catégorie. Les chercheurs ont un thème de recherche (par exemple bases de données, algorithmique, etc.) ; les administratifs une fonction (par exemple responsable de service, comptable, etc.) ; les personnels techniques une spécialité (par exemple réseau, téléphonie, etc.).
Le laboratoire offre un certain nombre de ressources matérielles. Nous nous limiterons dans ce sujet aux téléphones et aux stations de travail. Une ressource a un état ("bon", "moyen","mauvais") et un numéro affecté à sa création (et non modifiable par la suite). Elle peut à tout instant être affectée à un ou plusieurs utilisateurs (membres du laboratoire). Les téléphones ont de plus un numéro d'appel, un type et un tarif d'abonnement annuel. Les stations de travail disposent d'un système d'exploitation et ont un coût annuel de maintenance.
Le laboratoire regroupe plusieurs projets de recherche ou de veille technologique, qui lui sont propres et auxquels des membres du laboratoire de toutes les catégories sont affectés pendant une période de temps déterminée et selon un certain pourcentage. Par exemple, Marie peut être affectée au projet "Système d'aide à la synthèse de molécules chimiques" entre janvier 2004 et octobre 2007 à hauteur de 50% de son temps de travail.

Exercices 4 : Laboratoire de Recherche

- Deux opérations au moins sont possibles pour un laboratoire : une méthode calculeCoutAnnuel qui totalise les coûts annuels relatifs à l'ensemble de son personnel et de ses ressources,
- une méthode editeAnnuaire qui, à partir des téléphones existants, affiche le numéro de téléphone et les noms des membres qui y accèdent.
- **Questions** : Proposez un diagramme de classes permettant de représenter les éléments du texte ci-dessus. Prévoyez des méthodes d'accès (accesseurs) aux attributs, des constructeurs et des méthodes de toute nature (par exemple affectation d'une ressource à un utilisateur, inscription d'un membre du laboratoire, etc.).