# ANL

Analyse : apprendre à réfléchir sainement pour trouver des solutions au problème.
Utiliser la capacité de réfléchir, comprendre.

Projet d'informatisation : par d'un existant, le modèle, … définit des points faire, dis pk se sont des points faibles et proposes des solutions.

La ligne du temps : la ligne du temps défile en fonction des étapes.
La 1ère étape, **analyse** : pointe les problèmes et les besoins, trouver les acteurs, leur rôles, UC (cas d'utilisation), vocabulaire qui permet de pointer sur les concepts.
La 2ème étape, **conception** : trouver des solutions.
La 3ème étapes, **réalisation** : /
Quand on rentre dans le domaine on doit connaître le vocabulaire de celui-ci.
Acteurs : enseignants, parents, tous, …
Rôle de l'analyste : commencer par du flou.
Conceptualiser : globaliser le débat, tout structuré, monter en abstraction.
Mettre en évidence les concepts.
Analyse : recherche d'un problème, trouve les besoins.
Il faut être capable de modéliser.

# INTRODUCTION

## Qu'est-ce que l'analyse

Développement informatique

- Le client va décrire son projet pour que le manager et/ou l'analyste puissent évaluer la faisabilité et le budget / échéance du projet
- L'analyste va collaborer avec le client
  - L'analyste va discuter avec le client pour extraire toutes les informations pour développer le bon produit
  - L'analyste va structurer ces informations sous forme de documents, modèles, schémas, diagrammes, descriptions
  - Le client peut valider toutes ces informations une fois structurées et affiner sa description
- Les développeurs reçoivent la documentation de l'analyste
  - Ils développent ce qui est défini
  - Le client peut valider ses choix sur des prototypes.

- Les testeurs vont s'assurer que les développeurs
    - N'ont pas laissé de bugs
    - Ont codé le bon produit
- Les opérations vont s'assurer de la mise en production du produit

Analyse est composée de :

- Analyse : La discussion entre le client et l'analyste pour définir les besoins du client, comprendre le problème le plus finement possible.
- Conception : la production de documents explicatifs pour les développeurs, produire une solution sur papier qui répond au problème.

Il y a toujours des problèmes de communications.

Outils pour l'analyste :

- Diagramme ULM
    - Notation conventionnelle et internationale.
    - Permet de décrire schématiquement tout un système informatique.
- Gabarits de documentations (texte préformaté, plus cas remplir)
    - Comme templates

## Analyse et conception en UML

Qu'est-ce qu'un modèle :
Une représentation de la réalité

- Simplifiée
    - Ne schématise qu'une partie de la réalité
- Conventionnelle
    - Respecte des conventions pour communiquer simplement
- Pertinente
    - Inclut un maximum d'informations nécessaires

Modélisation = UML+méthodologie
Méthodologie : méthode nous dis ce qu'on doit faire à chaque étape.

Unified Modeling Language (UML)
Ce langage unifie les notations et les concepts orientés objets
Depuis 2005, l'Object Management Group a adopté la norme UML 2.0
Actuellement, on est à la version 2.5 (mars 2015)
Il permet de construire les modèles nécessaires à la réalisation et la documentation d'un système informatique.

UML 2.x : 14 diagrammes classés en 3 catégories

- Diagrammes de structure (statiques)
- Diagrammes de comportement (dynamiques)
- Diagrammes d'interaction (dynamiques)

# ANL

## DIAGRAMME DE CAS D'UTILISATION

Diagrammes de classes, objets et packages

- Décrivent les données du système analysé

Diagramme de cas d'utilisation

- Décrivent les interactions avec le système analysé
- Définissent les limites en termes de fonctionnalités

Idée : différents acteurs


Diagramme de cas d'utilisation (exemple 1)

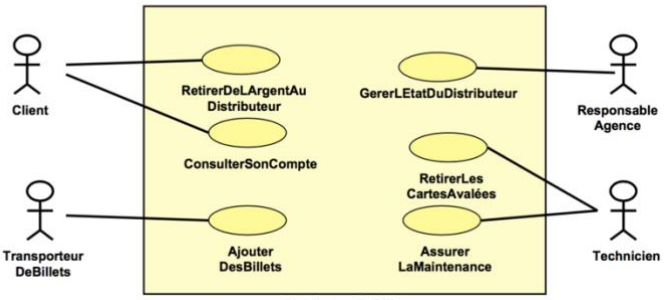



Diagramme de cas d'utilisation (exemple 2)

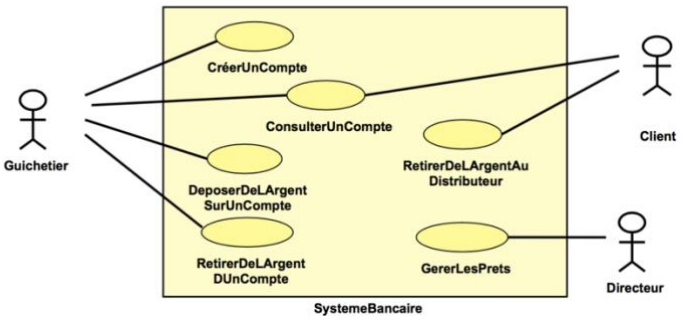
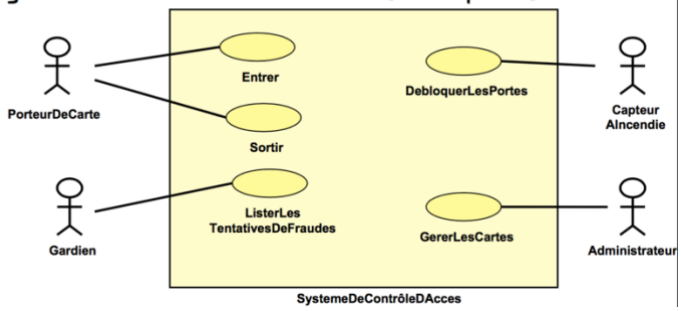
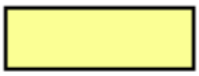



Diagramme de cas d'utilisation (exemple 3)





SystèmeN


CasDUtilisationN


ActeurN

Après, à côté on écrit le nom du UC ainsi qu'une description de ce que ça fait et détailler, exemple : construire bloc est une des description et le détail c'est miner bitcoin. Ça s'appelle une description fonctionnelle. Sur StarUML aussi.
Il faut mettre un texte qui montre que l'on a compris comme JavaDoc.
Sur StarUML, pour mettre un titre on commence par #

MCT = Modèle Conceptuel des Traitement

# ANL

## Élément de base

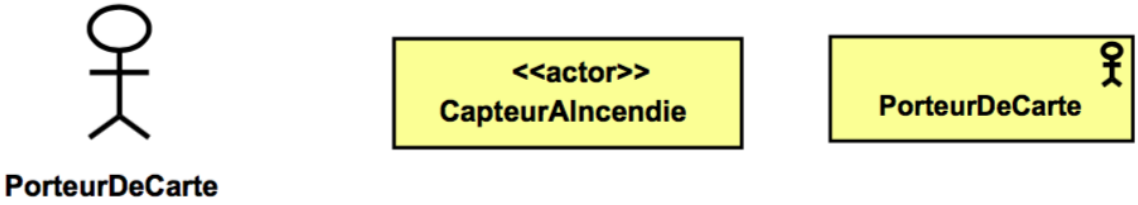
### Acteurs

- Élément externe qui interagit avec le système (prend des décisions, des initiatives. Il est "actif")
- Rôle qu'un "utilisateur" joue par rapport au système


### Acteurs vs utilisateurs

- Une même personne peut jouer plusieurs rôles（exemple : Maurice est directeur mais peut jouer le rôle de guichetier)
- Plusieurs personnes peuvent jouer un même rôle (exemple : Paul et Pierre sont deux clients)
- Un rôle par rapport au système plutôt que position dans l'organisation (exemple : PorteurDeCarte plutôt qu'Enseignant)
- Un acteur n'est pas forcément un être humain (exemple : un distributeur de billet peut être vu comme un acteur)

Il y a plusieurs notations pour un acteurs :




On utilise actor uniquement lorsque l'acteur n'est pas joué par un humain
Ces trois notations peuvent indiquer le même acteur.

Différents types d'acteurs

- Utilisateurs principaux (exemple : client, guichetier)
- Utilisateurs secondaires（exemple : contrôleur, directeur, ingénieur système, administrateur...)
- Périphériques externes（exemple : un capteur, une horloge externe, ...)
- Systèmes externes (exemple : système bancaires)

Règles de nommage :

- Forme nominale (un nom)
- Vocabulaire métier
- Style CamelCase

### Cas d'utilisation (CU ou UC)

- Représente
  - Une manière d'utiliser le système

# ANL

- Une suite d'interactions entre un acteur et le système
- Correspond à une fonction du système visible par l'acteur
- Permet à un acteur d'atteindre un but
- Doit être utile en soi
- Regroupe un ensemble de scenario correspondant à un même but
- Possède une unité de temps


EnterPendantLesHeuresDOuverture ➔ car ça fais partie d'entrer, n'y a pas tous les cas, et si on veut entrer hors des heures d'ouverture ?

Règles de nommage

- Vocabulaire métier
- Forme verbale décrivant une action
- L'acteur est généralement le sujet
- Éviter les connecteurs (et, ou, puis, ...)
- Style CamelCase ou littéral
- Terme générique comme "Gérer" en cas de besoin seulement
  - Gérer = Créer, Lire, Ajouter/Modifier & Supprimer (CRUD) Exemple : GérerLesDroits

## Le système

Le système est :

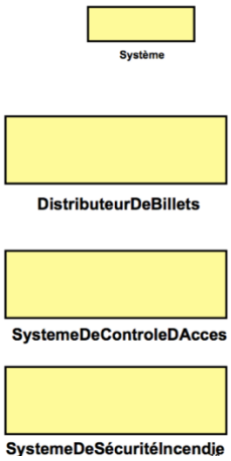
- Modélisé par un ensemble de cas d'utilisation
- Vu comme une boîte noire

Le système contient les cas d'utilisation :

- Mais pas les acteurs

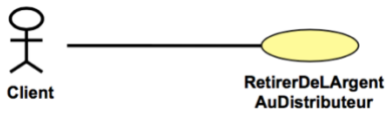
Permet de définir :

- Les limites du système

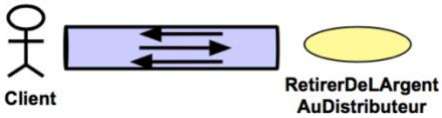



## Relations entre éléments

Vision objective : possibilités d'atteindre un but



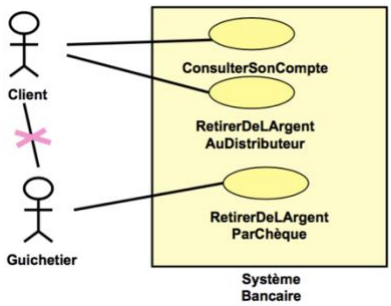

Vision technique : canal/protocole de communication

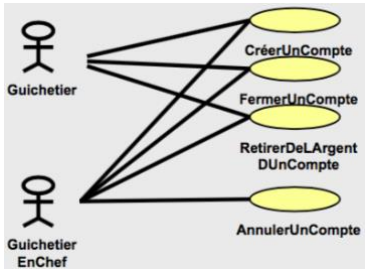


# ANL

## Acteur

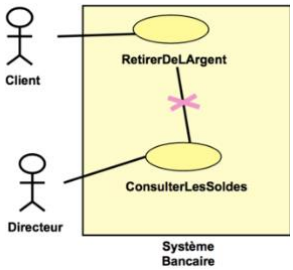
Communications externes non modélisées, UML se concentre sur la description du système et de ses interactions avec l'extérieur La seule relation entre acteurs est la relation de généralisation




## Cas d'utilisation

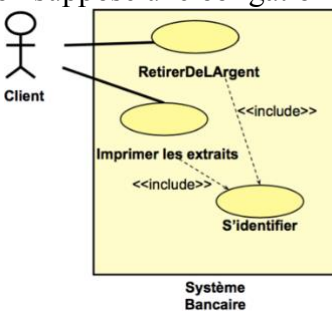
- Communications internes non modélisées
- Interactions système ↔ extérieur
- Formalisme bien trop pauvre pour décrire l'intérieur du système. Utiliser les autres modèles UML pour cela.
- Autres relations possibles entre CU : Héritage, inclusion et extension




## Dépendance entre CU

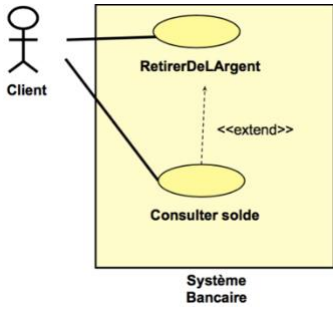
### Include

- Si plusieurs CU ont des sous-ensembles communs d'actions
- Permet la factorisation des CU
- Une instance de RetirerDeLArgent va engendrer une instance de S'identifier
- RetirerDeLArgent dépend de S'identifier
- S'identifier n'existe pas seul
- La relation d'inclusion suppose une obligation d'exécution des interactions



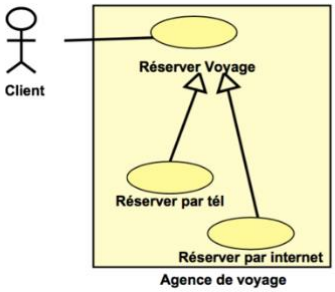
### extend

- Permet d'étendre les fonctionnalités d'un CU
- Ce CU peut fonctionner seul mais peut également être complété par un autre, sous certaines conditions et à certains moments précis
- La relation d'extension suppose une option d'exécution des interactions (pas d'obligation)




## Généralisation/Spécification des CU

Un CU peut hériter d'un autre CU.
À voir comme un polymorphisme de cas.
Le but est le même mais les interactions pour y arriver ne sont pas les mêmes.



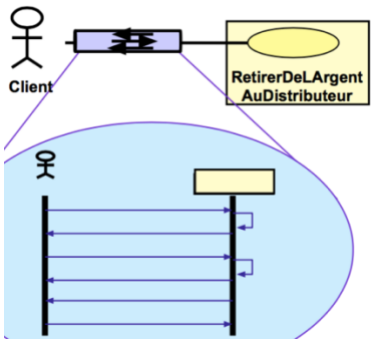

# Communication entre acteurs et système

## Retour sur la relation acteur - cas d'utilisation

Canal de communication
Décrit le comportement du système vu de l'extérieur
Échange de messages

### Description de l'interaction

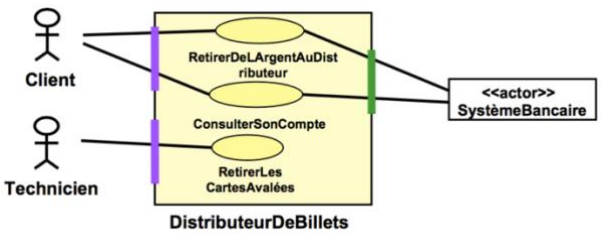
Diagrammes de séquences "système".



# ANL

## Limites du système et interface

Humain → IHM : Interface homme - système (machine)
Logiciel → API : Interface système - système

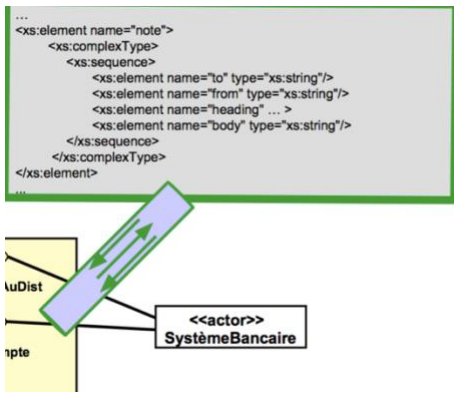



## Interfaces Homme-Système (IHM)

C'est le client qui va retirer de l'argent ou bien même le technicien qui retire la carte avalée.

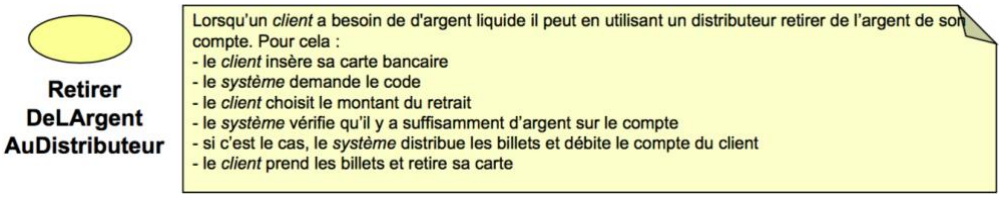
## Interfaces Système-Système (API)

C'est via du code.




## Description des cas d'utilisation

Les cas d'utilisation sont alors décrits de manière textuelle.




## Problème de granularité

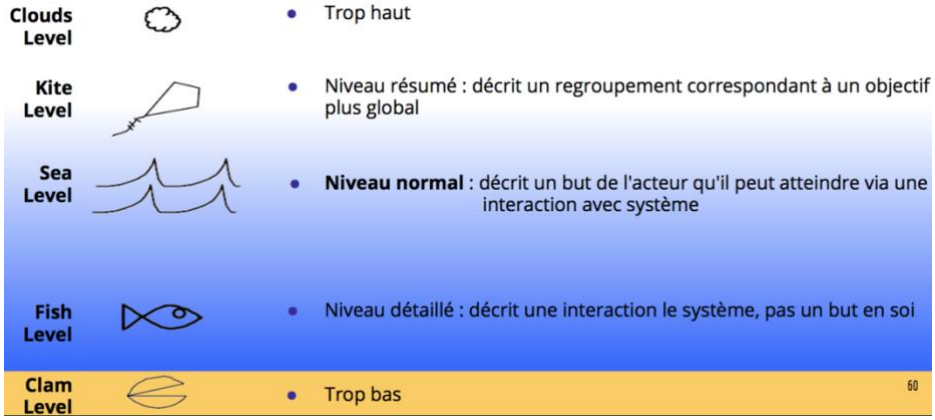
A quel niveau décrire les cas d'utilisation ? pas de réponse.

- Trop haut
  - Trop loin du système
  - Trop abstrait et "flou"
  - Trop complexe à décrire
- Trop bas
  - Trop de cas d'utilisation
  - Trop près de l'interface
  - Trop loin des besoins métiers

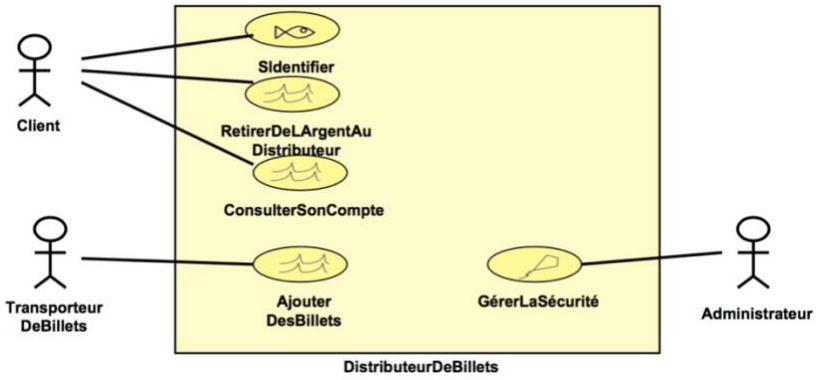
# ANL

## Granularité des cas d'utilisation/ niveaux de granularité

Tous les cas d'utilisations ne sont pas tous au même niveau, différents niveaux de détails, POURQUOI vs. COMMENT.
Décoration du niveau selon Cockburn, non standardisé mais intuitif et utile, niveau normal des cas d'utilisation : la mer




Exemple :




## Unicités des cas d'utilisation

Chaque CU doit respecter les 4 unicités suivantes :

- Unicité d'objectif Le CU doit fournir un résultat appréciable à l'acteur avec lequel il interagit.
- Unicité de responsabilité Pour un CU interactif, un seul acteur direct
- Unicité de temps
  - Unicité d'exécution Un enchaînement d'actions qui se suivent sans interruption
  - Unicité de périodicité Exécution à la même fréquence
- Unicité de mode manuel, interactif ou automatisé, et unitaire ou par lot

## Mode

| | Unitaire | Par lot |
|---|---|---|
| Manuel | Remplir un formulaire (Hors SIA) | Vérifier manuellement tous les formulaires reçus |
| Interactif | Enregistrer un passager sur un vol | Encoder les 1000 livres achetés par la bibliothèque |
| Automatisé | Traitement d'un signal d'alarme | Envoyer au siège central les opérations bancaires d'un distributeur de billets toutes les heures |

# DIAGRAMME D'OBJETS

## Objet

- Objet = élément du monde réel à propos duquel on souhaite conserver des informations. Ces informations seront soumises à des traitements.
- Les objets peuvent être abstraits ou concrets (par exemple, un compte en banque et une voiture).
- Ils possèdent : une identité, un état, un comportement.

## Caractéristique d'un objet

Identité

- Ce qui permet de différencier un objet d'un autre.
- Exemples pour les objets métier : numéro national d'une personne, numéro de châssis d'une voiture...

État

- Caractéristiques de l'objet à un moment donné, représentées par des valeurs d'attributs dans le SIA.
- Exemples pour les objets métier : taille d'une personne, nombre de kilomètres parcourus par une voiture...

Comportement

- Ensemble des opérations (méthodes) qu'un objet peut exécuter ou subir.
- Exemples pour les objets métier :
  - Être créé (Create)
  - Transmettre ses caractéristiques (Read)
  - Être modifié (Update)
  - Être effacé (Delete)

## Types d'objets

| **Métier** | Frontières | Contrôle | Données | Périphériques |
|---|---|---|---|---|
| **Objets qui correspondent au domaine étudié, qui seront décrits et compris par les utilisateurs.** | Objets qui permettent au SIA de communiquer avec son environnement (utilisateurs humains ou autres SIA) | Objets qui représentent les traitements subis par les objets. | Objets qui représentent la réalisation (implémentation) des objets métier dans le SI; typiquement sous la forme de base de données (Objets techniques) | Objets qui représentent le matériel sur lequel sera implémenté le SI. |

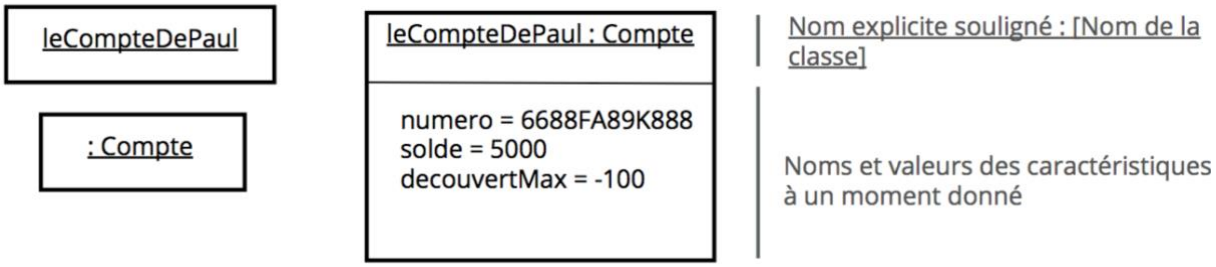
### Métier

| Signalétique | Gestion | Reporting |
|---|---|---|
| Objets dont les caractéristiques évoluent peu dans le temps.<br><br>*Ex: Un produit, Une personne, ...* | Objets qui exigent un suivi, dont l'état varie au fil du temps.<br><br>*Ex: Un dossier, un contrat, un document, ...* | Objets qui servent à gérer, piloter l'entreprise.<br><br>*Ex: Un bilan, l'état des stocks, le chiffre d'affaire, ...* |

# ANL

## Diagramme d'objets en UML

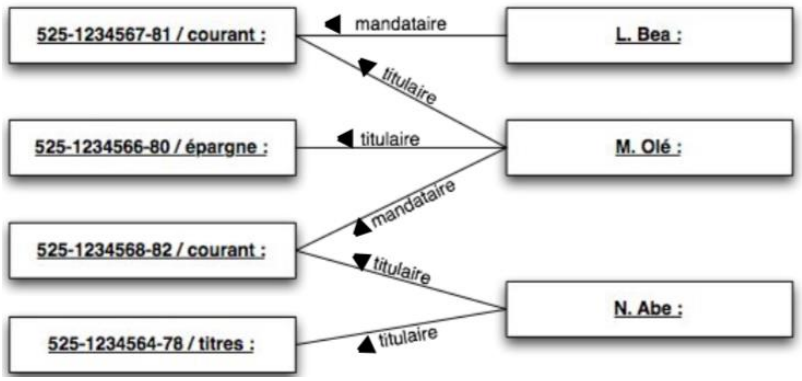
### Un objet




## Liens entre objets

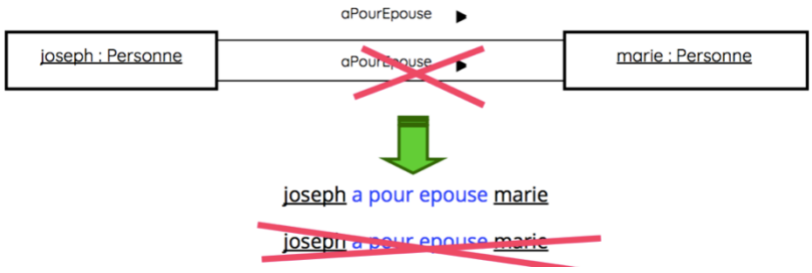
- Les objets sont liés entre eux par des lignes pleines dans un diagramme.
- Les noms des liens sont des formes verbales ou nominales et commencent par une minuscule.
- Indiquent le sens de la lecture (ex: « paul aPourCompte c1 »)

Ex: ici les objets du domaine *Clients* d'une banque. L.Bea est mandataire de …




### Contrainte sur les liens

Il peut y avoir au maximum 1 liens d'un type donné entre 2 objets donnés.




### Rôles sur les liens

Chacun des deux objets joue un rôle différent dans le lien.


(1) pierre a pour compte c1
(2) c1 **joue le role de** compte **pour** pierre
(3) pierre **joue le role de** titulaire **pour** c1

(2.b) **le [un des]** compte[s] **de** pierre **est** c1
(3.b) **le [un des]** titulaire[s] **de** c1 **est** pierre

Note de style :

- Choisir un groupe nominal pour désigner un rôle.
- Si un nom de rôle est omis, le nom de la classe fait office de nom.

ANL

# DIAGRAMME DES CLASSES

## De l'objet à la classe

Impossible de représenter tous les objets qui peuvent être manipulés par un SI. D'où la notion de classe.
Une classe représente des objets de même structure et de même comportement.
Un objet est une occurrence ou instance d'une classe.

## Caractéristiques d'une classe

- Une classe doit être définie soit :
  - Par extension : énumération, liste de tous ses éléments...
  - Sous forme de texte
- Déclaration de l'identifiant
- ○ Choix d'un attribut particulier, d'un ensemble d'attribut, de tous les attributs ou d'attributs de classes liées.
- Déclaration de l'état
  - Ensemble des attributs communs à toutes les instances de la classe.
  - Exemple : tous les comptes en banque ont un numéro, un solde… mais pas la même valeur !
  - Pour chaque attribut, on donne un nom et un type
- Définition du comportement
  - Ensemble des méthodes communes à toutes les instances de la classe.

Une **classe** spécifie la structure et le comportement d'un ensemble d'objets de même nature.

La structure d'une classe est constante

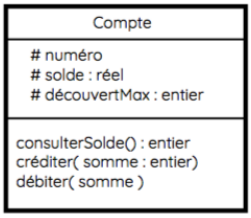



Diagramme de classes

Des **objets** peuvent être ajoutés ou détruits pendant l 'exécution

La valeur des attributs des objets peut changer

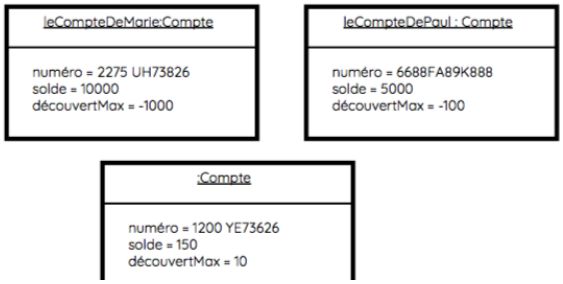
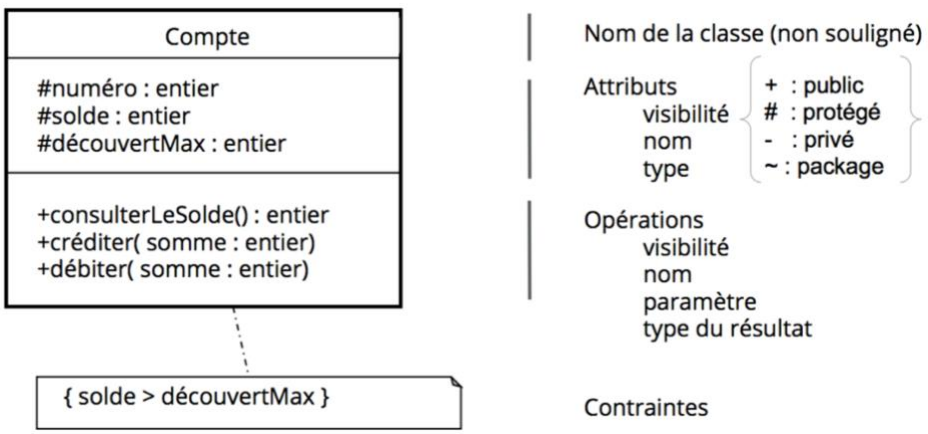


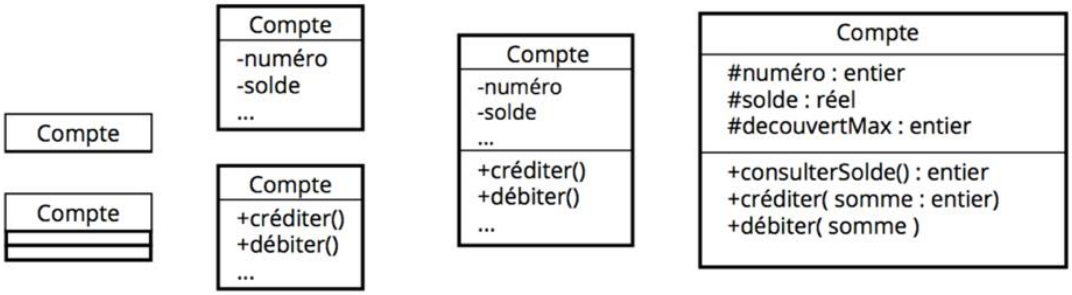

Diagramme d 'objets

## Une classe



# ANL

## Notations



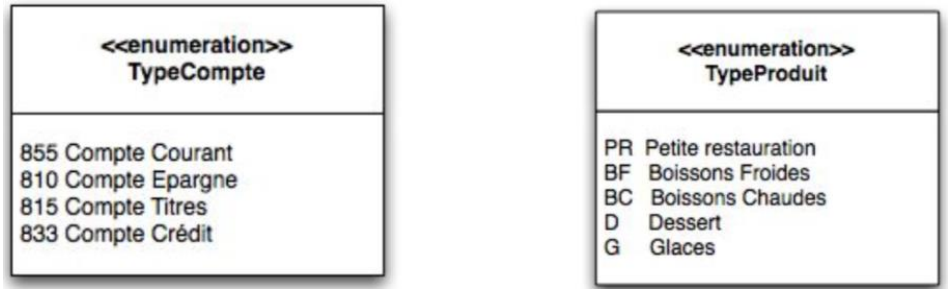

Notes de style :
• les noms de classes commencent par une majuscule
• les noms d 'attributs et de méthodes commencent par une minuscule

## Classes « types de données » généralise les objs de types donnés

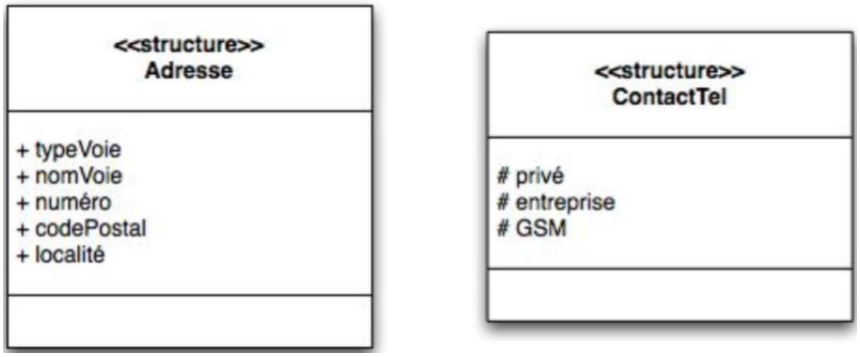
UML propose les types « standard » de données pour les attributs : entiers, booléens, date, ...

- On peut créer des données de type particulier qui seront reprises dans des classes.
- Notation : au-dessus du nom de la classe « particulière », on place le type de données entre << >>
- Classe <<énumération>>
  - Deux attributs : un code et un libellé (nom).
- Listes connues des utilisateurs.
- Un objet = une valeur de la liste.
- Classe <<structure>>
  - Modélisation d'un attribut décomposable (ou composé)
- Ces classes ne peuvent pas avoir d'associations

## Classe ≪énumérations≫ : exemple



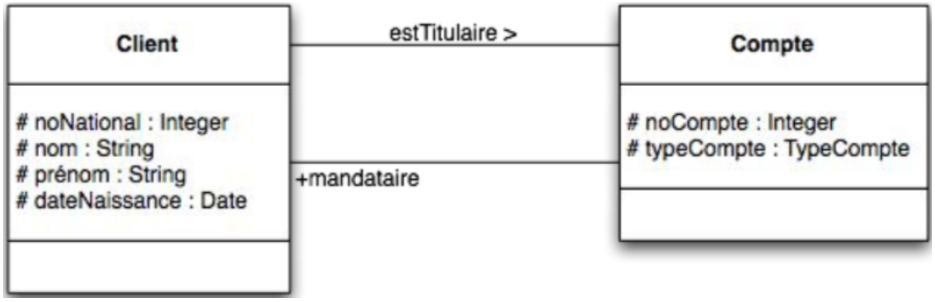

## Classe ≪structure≫ : exemple ce sont de nouveaux types de données



## Associations

Une association peut être définie par :

- Un verbe (accompagné d'une flèche qui indique le sens de lecture)
- Le rôle d'au moins une classe dans l'association.





Attention : le sens de lecture ne se met pas sur la flèche.
La signification d'une association fléchée (unidirectionnelle) est différente de celle d'une association bidirectionnelle : elle indique que l'accès est unidirectionnel.

Par exemple ici, un client peut accéder à ses comptes mais à partir d'un compte, on ne peut savoir qui sont les clients titulaires ou mandataires.

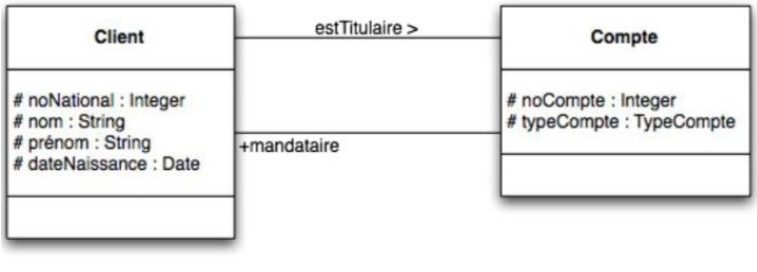
## Différence entre les diagrammes de classe et d'objets

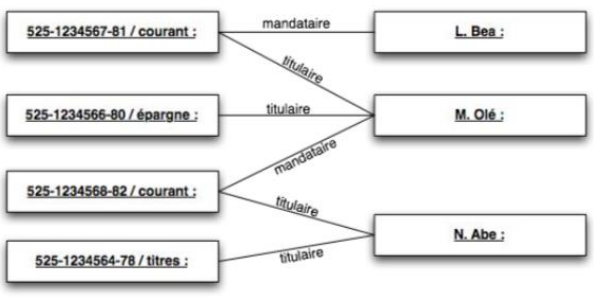
Un objet est une instance d'une classe et un lien est une instance d'une association.

| Diagramme d'objets | Diagramme de classes |
|---|---|
| *Instances* de classes | *Classes* |
| *Liens* entre les instances | *Associations* entre les classes |

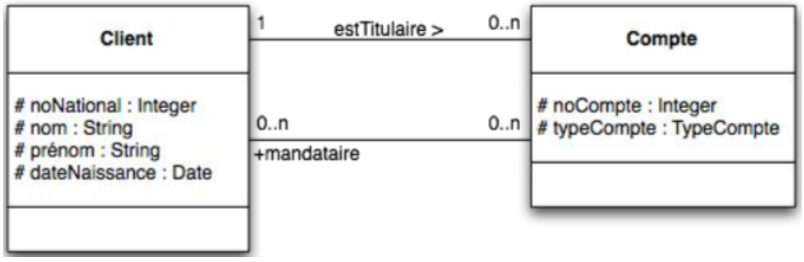
Association devient lien et classe devient objet.

## Relations entre les diagrammes de classe et d'objets



## Multiplicités




1 client est titulaire de 0 à n comptes …

1. Un compte à un et un seul titulaire.
2. Un client peut n'être titulaire d'aucun compte (0), de 1 ou de plusieurs (n ou *) comptes.
3. Un compte peut n'avoir aucun (0) mandataire ou en avoir 1 ou plusieurs (n ou *).
4. Un client peut n'être mandataire d'aucun compte (0), de 1 ou de plusieurs (n) comptes.

Permettent de représenter toutes les possibilités de nombres de liens d'un objet/d'une classe avec un objet de la même classe ou d'une autre classe.

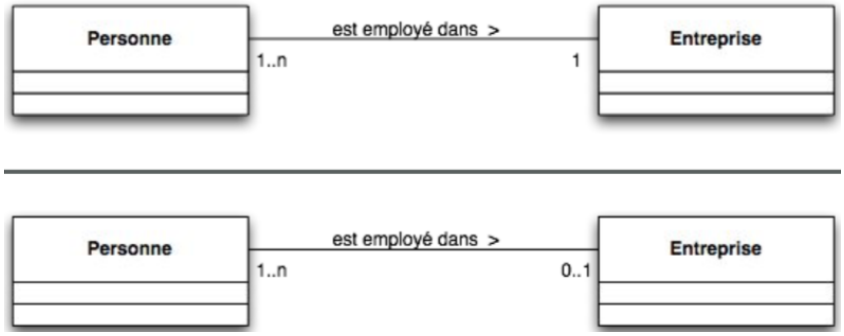
**Q° à se poser** : Pour un objet d'une classe, à combien d'objets sera-t-il lié au minimum et au maximum ? Attention : on ne lit qu'une seule multiplicité à la fois !

Syntaxe : {Minimum} .. {Maximum} !!! Si juste 2 ➔ 2..2 ; n = ∞

- Les plus courantes:
  - 0..1
  - 1..1 ou 1
  - 0..n ou n ou 0..* ou *
  - 1..n ou 1..*
- On peut avoir aussi:
  - 3..5
  - 0..18
  - 15..*
- Remarques
  - * équivalent à 0..*
  - 3 équivalent à 3..3

Attention :

- Ce sont des choix de gestion du SI.
- Ce sont des contraintes d'intégrité du SI.



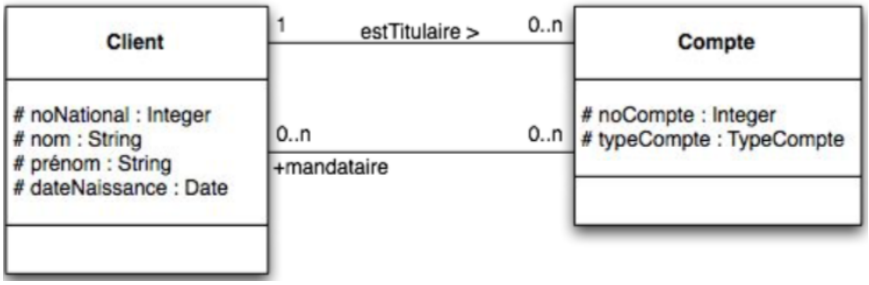




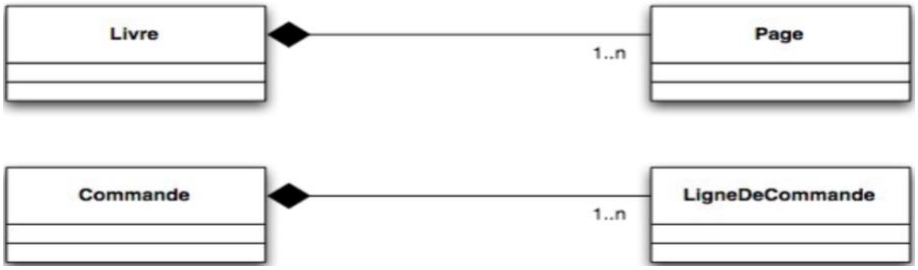
1 entreprise emploie 1 ou plusieurs personne.

# ANL

## Relations entre les multiplicités du diagramme de classes et du diagramme d'objets




## Composition

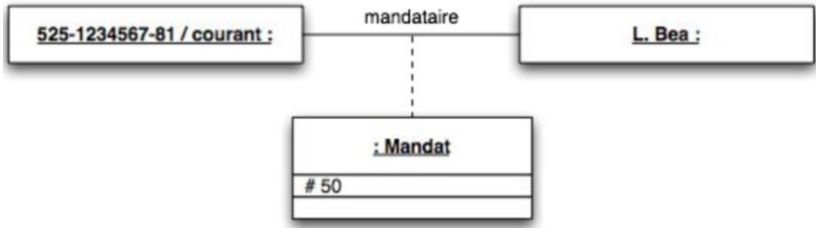



Un livre est composé de 1 à n pages. Une page est un élément d'un livre.

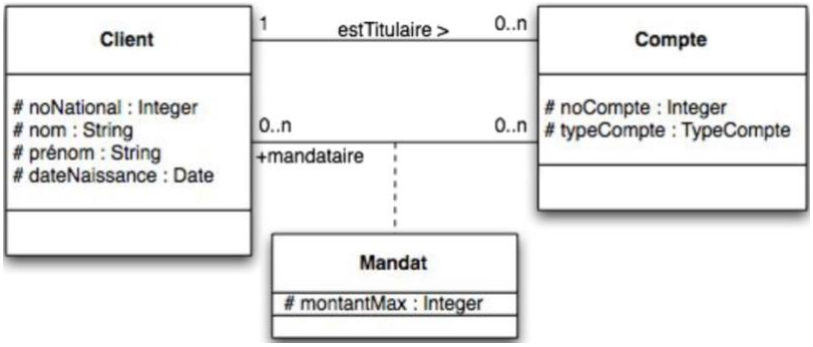
- Ajoute une sémantique : "est composé de" / "fait partie de"
- Contraintes liées à la composition :
  - Un objet composant ne peut être que dans 1 seul objet composite
  - Un objet composant n'existe pas sans son objet composite
  - Si un objet composite est détruit, ses composants aussi

## Classes d'association

On constate parfois qu'on veut conserver des données au sujet d'un lien entre objets. Ça donnera lieu à la création d'une classe d'association.




### Exemple

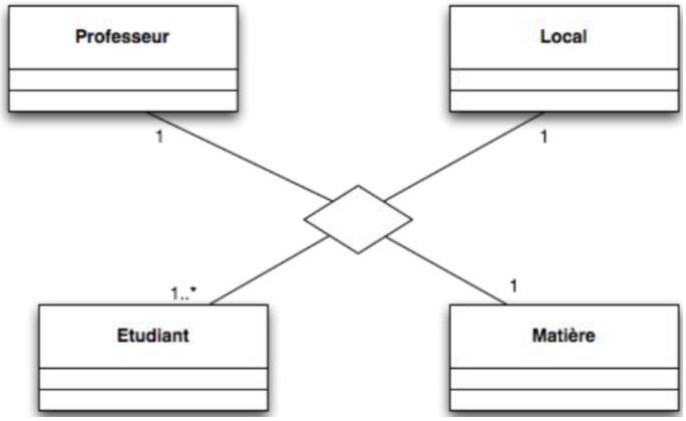


## Association n-aire

Parfois plus de 2 classes peuvent être liées. Une association ternaire, quaternaire, ... doit alors être créée. On appelle cela des associations n-aires.

### Exemple

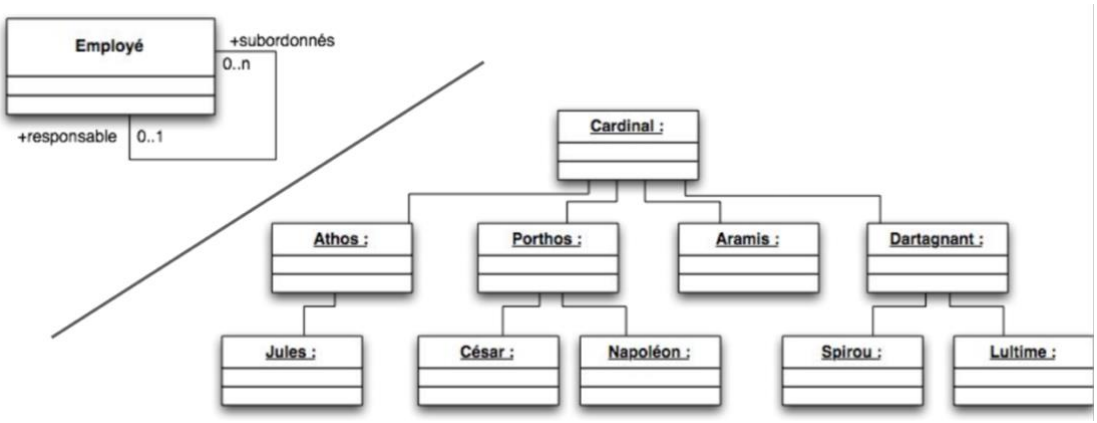
Cette association représente le lien entre un professeur qui donne un cours dans un local à un groupe d'au moins un étudiant.




## Association réflexive

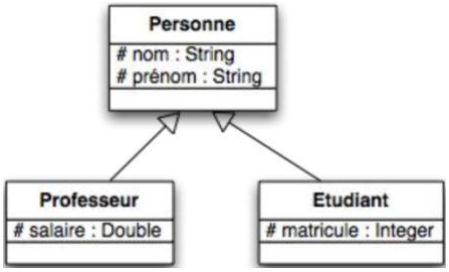
Association qui part et arrive sur une même classe.
Elle signifie donc qu'il existe des liens entre objets d'une classe.

### Exemple




## Héritage

- L'héritage permet de classifier les classes dans une hiérarchie.
- On peut ainsi définir dans une « super-classe » les attributs et les méthodes communs à plusieurs « sous-classes ». Cela permet donc la réutilisation.
- Toutes les classifications (botaniques, zoologie...) répondent à ce type de hiérarchie.
- On a une hiérarchie de classes quand on peut dire : « est un(e) »
    - Exemple : Un professeur (sous-classe) est une personne (super-classe).

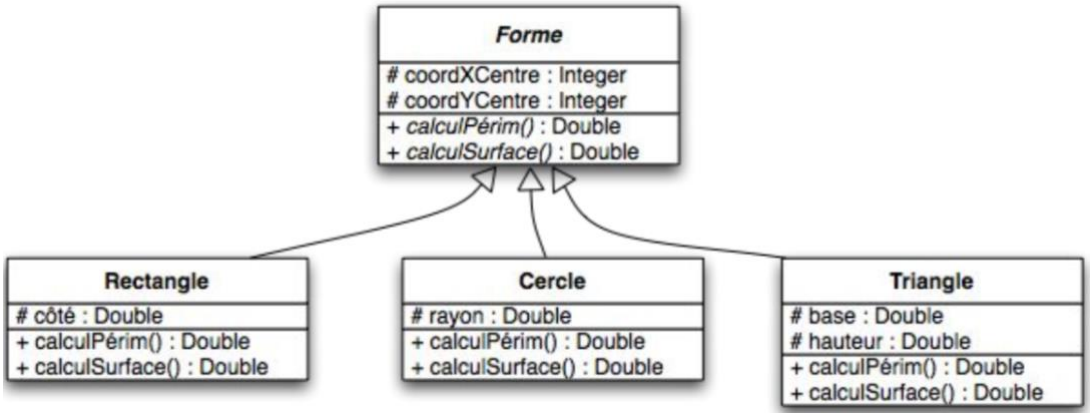


# ANL

## Héritage et polymorphisme

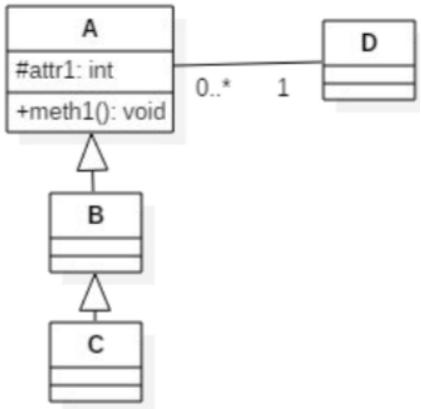
- Certaines opérations décrites au niveau d'une super-classe peuvent être redéfinies au niveau des sous-classes.
- On a donc plusieurs formes (poly-morphe) d'une même entité (ici une méthode).
- Il existe du polymorphisme de méthodes, de variables...

Exemple : calculPérim est différent en fonction de la forme.




## Héritage : propriétés

- Relation transitive
  - Pour les attributs
  - Pour les méthodes
  - Pour les associations
- A, B et C ont :
  - Un attribut entier protégé attr1
  - Une méthode meth1
  - Une association vers D
- Relation non réflexive
  - Une classe ne peut pas hériter d'elle-même.
- Relation non symétrique
  - Si F hérite de E alors E ne peut pas hériter de F.
- La hiérarchie ne peut pas former de cycle.



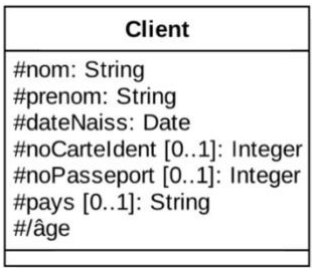

## Contraintes d'intégrité

- Définition : règles qui permettent de garantir la cohérence des données lors de leur mise à jour par rapport au monde réel.
- Ces règles sont soit :
  - Indiquées dans l'UML (technique à utiliser lorsque c'est possible)
  - Écrites en français dans une note UML ou dans un texte accompagnant le diagramme
- Sur les attributs
  - Types de données
  - Domaines de valeurs
  - Obligatoires ou facultatifs
  - Attributs liés
- Sur les associations
  - Multiplicités, inclusion et exclusion

# ANL

## Contraintes d'intégrité sur les attributs

- Domaine de valeurs d'un attribut :
  - Types de données : entier, réel, booléen, date (une classe connue), chaîne de caractères, ...
  - Limitation de ces valeurs :
    - Exemples : « La date de naissance doit être antérieure à la date du jour. », « Le numéro de client est un entier positif sur 5 positions en DCB. »
- Existence d'un attribut :
  - Obligatoire
  - Facultatif
    - Notation UML : [0..1]  ça veut dire facultatif.
- Lien entre attributs :
  - « noCarteIdent ou noPasseport »
  - « Pays n'est rempli que si on a un numéro de passeport. »

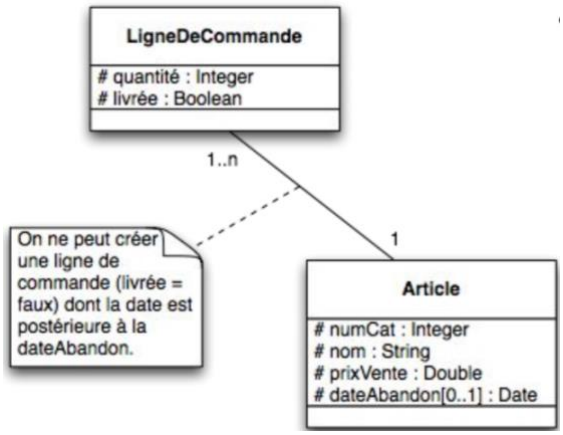



## Contraintes d'intégrité sur les associations

- Données la plupart du temps par les multiplicités :
  - Contrainte d'existence : multiplicité minimale = 1

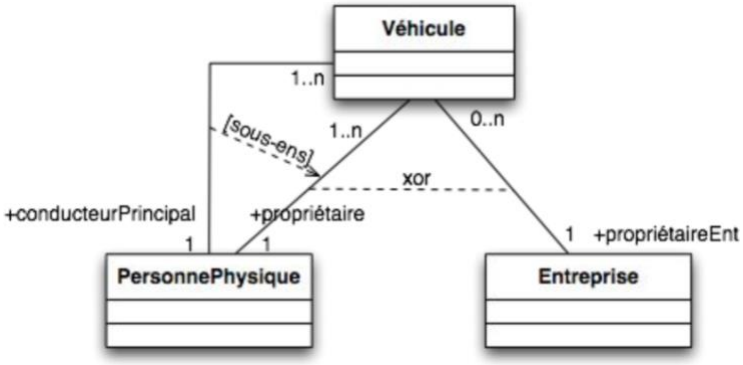

- Une ligne de commande ne peut exister que pour un article qui existe à la date de la commande.
  - date de LigneCommande doit être antérieure à dateAbandon.

C'est une note ➔




- Contrainte d'inclusion
  - Une association est incluse dans une autre quand un lien qui appartient à l'une appartient aussi à l'autre
    - Exemple : La personne directeur d'un département y est aussi employée.
- Contrainte d'exclusion
  - Deux associations sont exclusives si, quand un lien appartient à l'une, il ne peut pas appartenir à l'autre.
    - Exemple : Un enseignant dans une école ne peut pas y être étudiant.

Exemple : Un véhicule n'a qu'un propriétaire : une personne ou une entreprise (ou exclusif). Quand le propriétaire est une personne physique, il est aussi le chauffeur principal du véhicule.




## DIAGRAMME DE PACKAGES

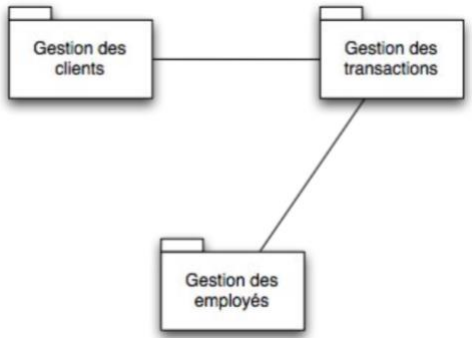
Découper un système en sous-système.

- Sur base des descriptions des utilisateurs, on divisera le problème en domaines.
- Les données du SI seront classées dans des packages correspondant à ces domaines.
- Comment identifier les domaines ?
  - Département de l'organisation
  - Regroupement par types d'objets/de classes :

| Signalétiques | Gestion | Reporting |
|---|---|---|
| Structure et état assez stables. Espèces de « catalogue » en général | Données dont on suit l'évolution. Structure relativement stable, état souvent modifié. | Structure modifiée en fonction des besoins. On conserve des « photos » de leur état à différents moments. |

### En UML : exemple

Un lien indique une "dépendance" : une modification d'un package peut entrainer une modification dans l'autre package.




(Itération = découpe d'un projet.)

---

Donner 2 ou 3 exemple de contrainte d'intégrité.