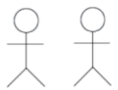
Unified **M**odeling **L**anguage

Langage unifié pour la modélisation objet

Sommaire

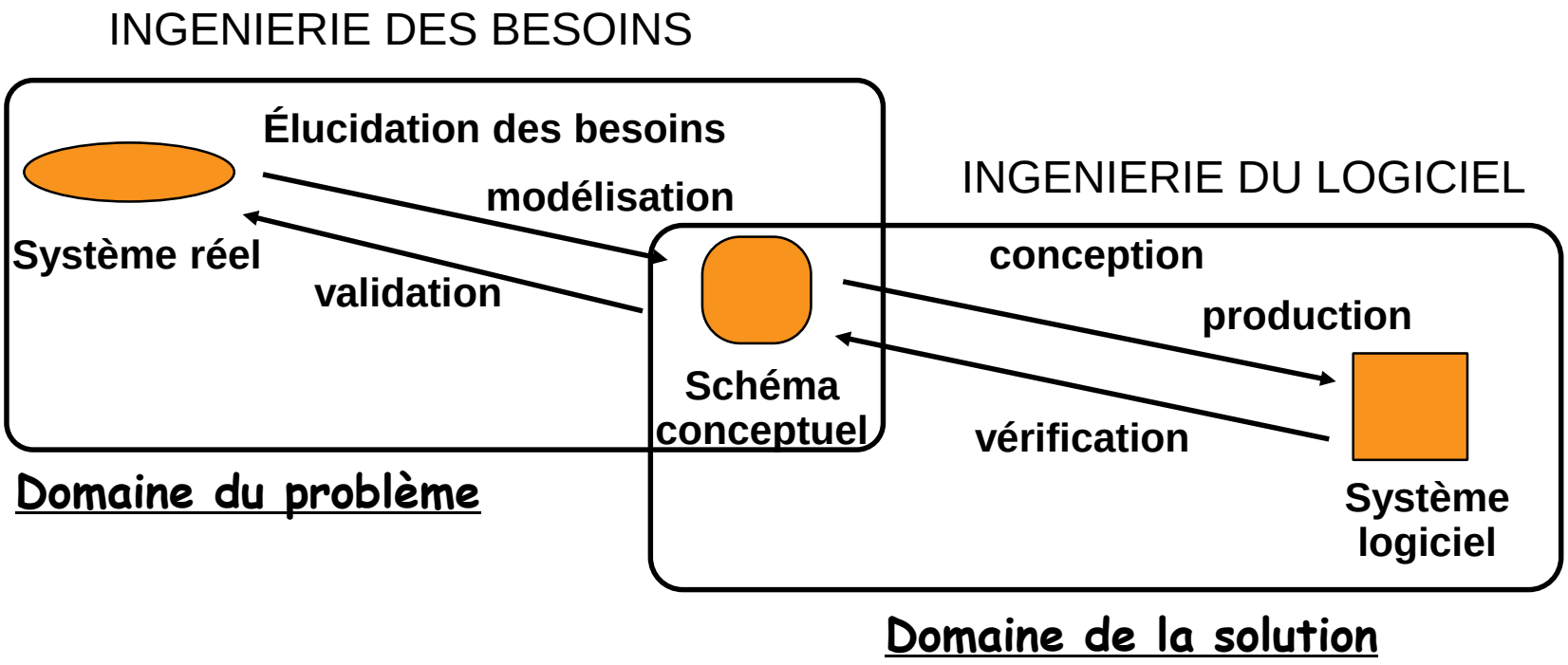
1. Généralités : Modélisation et approche P.O.O.

2. UML : Langage de modélisation
   a. Introduction
   b. le diagramme de cas d'utilisation (TP)
   c. le diagramme de classe (TP)
   d. le diagramme de sequence (TP)
   e. les autres diagrammes



INTRODUCTION

L'ingénierie des systèmes d'information



INTRODUCTION

POINT DE VUE HISTORIQUE

- **"Crise du logiciel" :**

L'ingénierie du logiciel comme une discipline

(milieu des années 60)

- **Facteurs d'évolution**

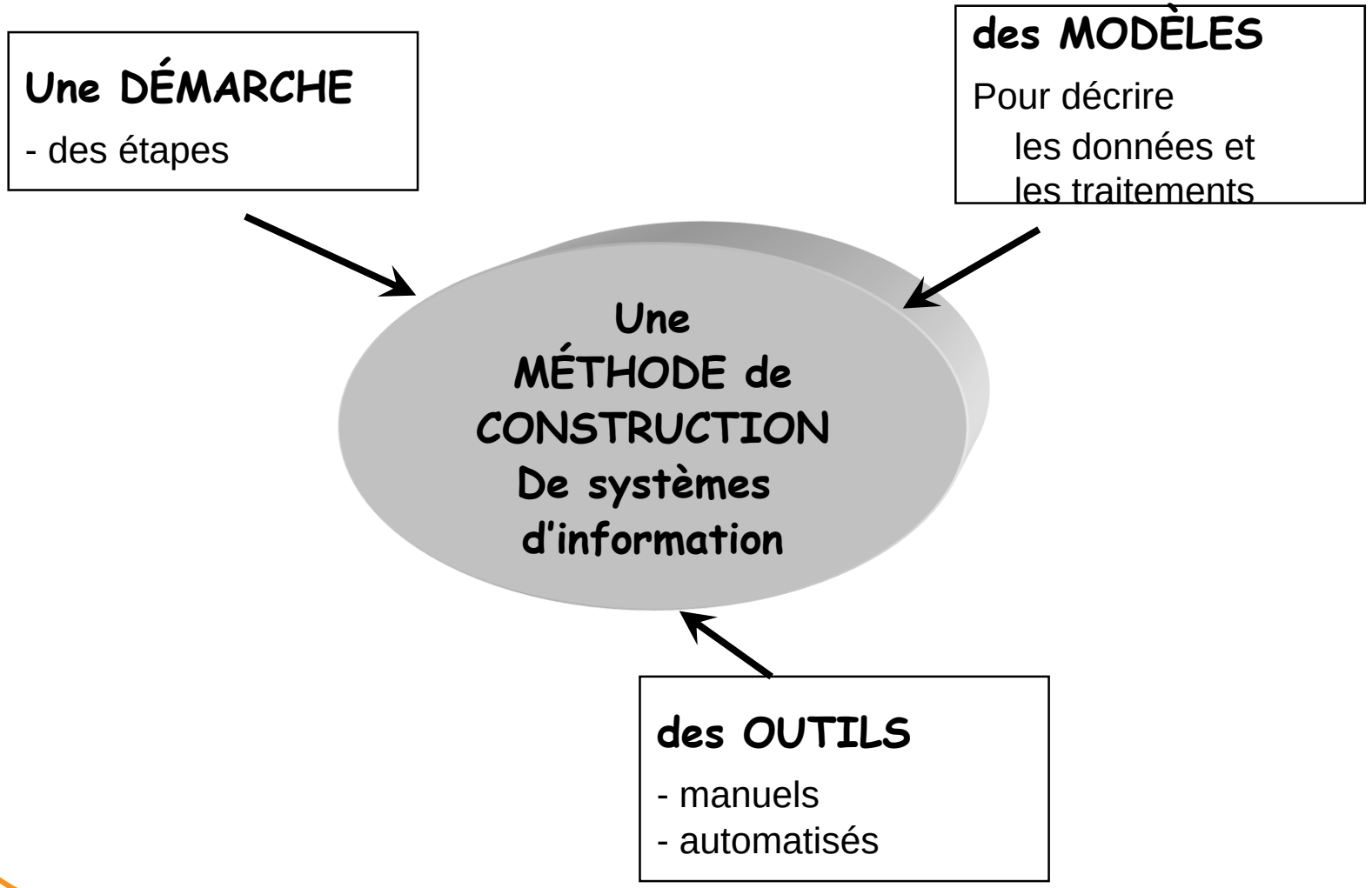
Coût du logiciel / coût du matériel

Le logiciel considéré comme un produit ayant son cycle de vie

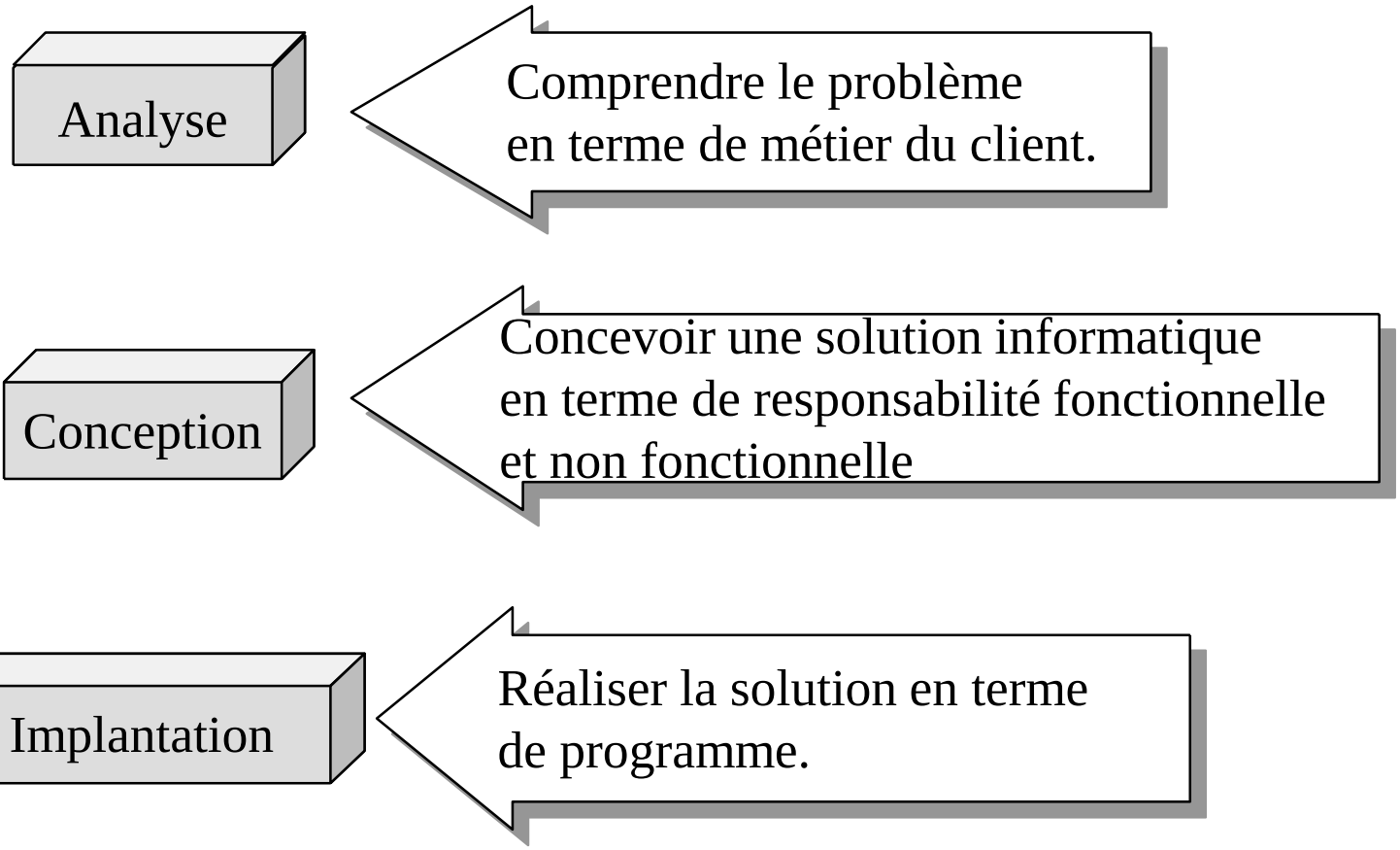
Nécessité d'avoir des Méthodes

INTRODUCTION

Composants d'une méthode de construction de SI



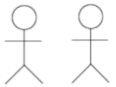
Processus de développement



INTRODUCTION

Des méthodes fonctionnelles aux méthodes Objet

- **L'orientation fonctionnelle (années 1970)**
  - Programmation structurée, méthodes d'analyse structurées.
- **L'orientation conceptuelle (années 1980)**
  - méthodes de conception systémique (Données et traitement)
- **L'orientation Objet (les années 1990)**
  - Programmation Objet, SGBD Objet, méthodes d'analyse et de conception orientée Objet



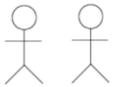
INTRODUCTION

La prolifération des méthodes objets

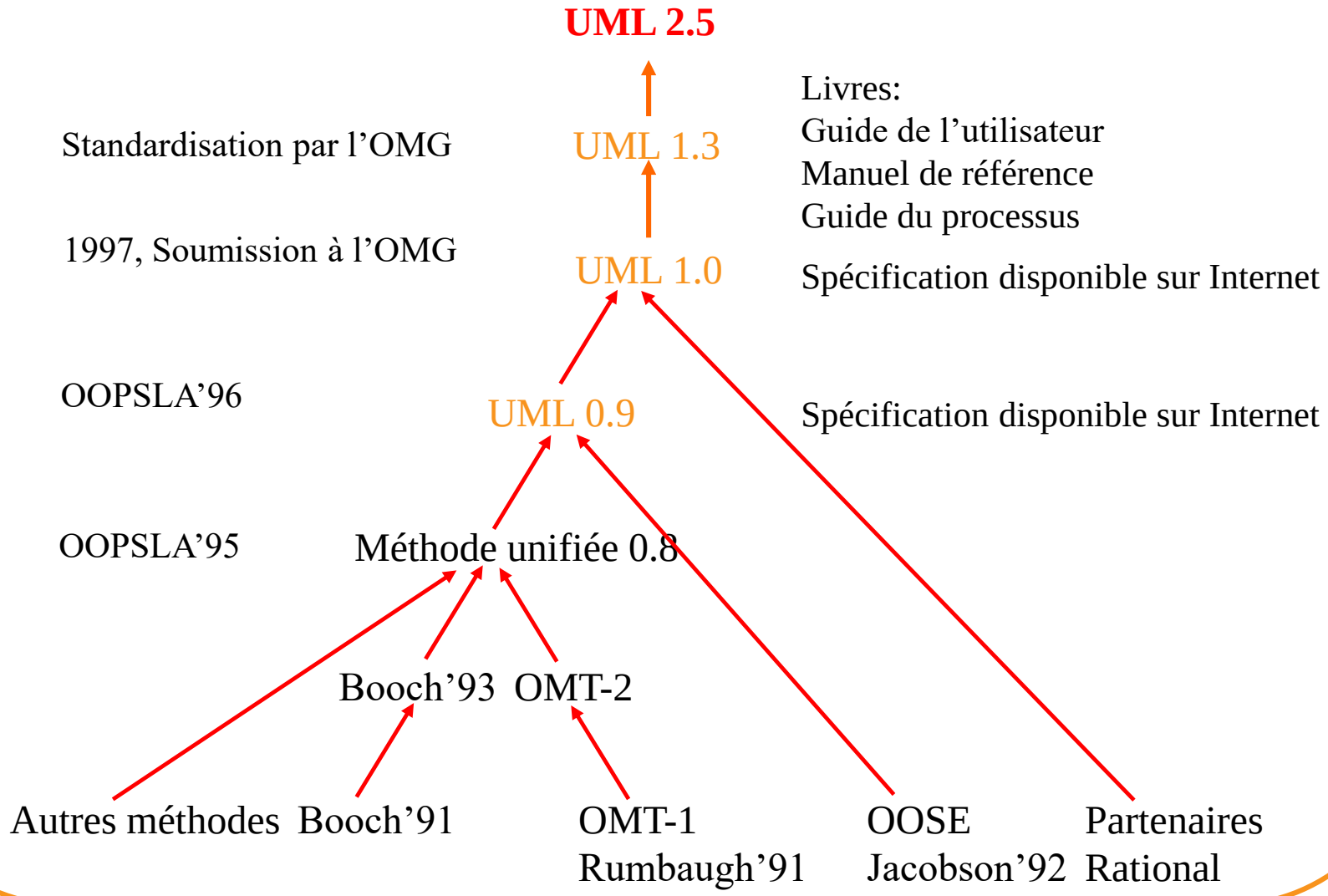
- **1990: Naissance d'une cinquantaine de méthodes orienté objet.**
  - Confusion autour de l'analyse et de la conception.

- **Idée: Examen des méthodes dominantes pour permettre de dégager un consensus autour d'idées communes.**

Rapprochement de 2 méthodes : OOAD de Grady Booch et OMT de Rumbaugh & Al



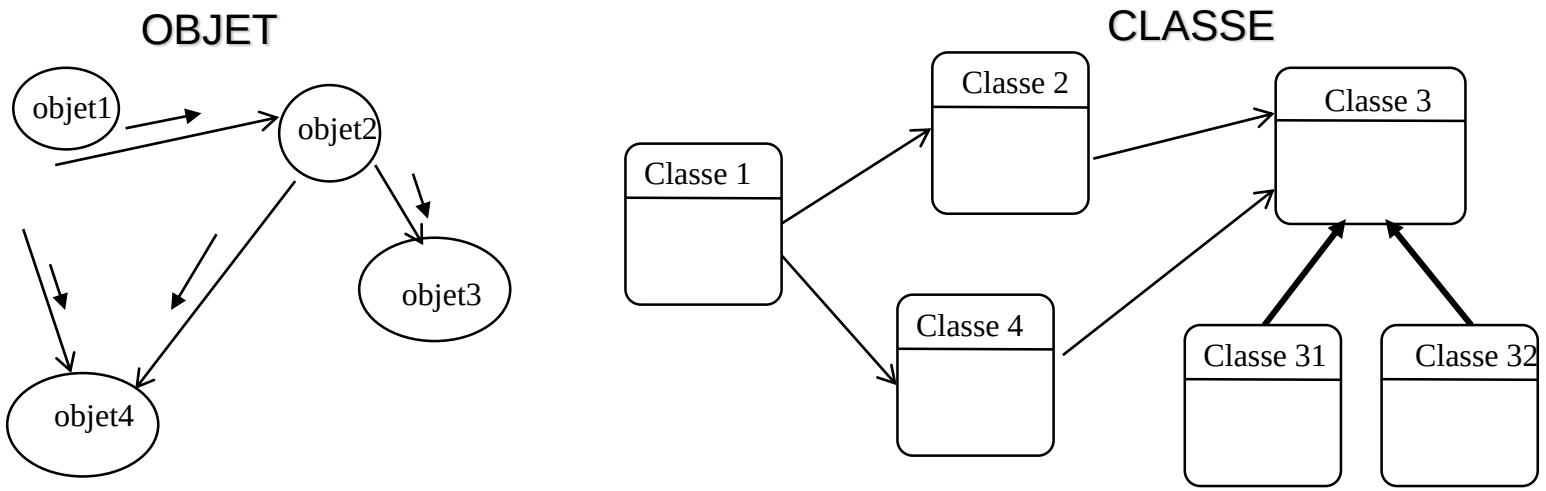
L'unification des méthodes :
Principales étapes de la définition d'UML



L'orientation Objet

- **Les principes de base de la Programmation Orientation Objet :**
  - Les objets, les classes, envoi de messages, l'encapsulation
  - l'héritage, le polymorphisme

- **Organisation d'un système en une collection d'objets disssociés mais collaborant entre eux**



Une architecture logicielle

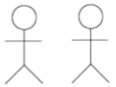
Pourquoi l'approche objet ?

But

- modélisation des propriétés statiques et dynamiques de l'environnement dans lequel sont définis les besoins (domaine du problème),
- formalisation de la perception du monde et des phénomènes qui s'y déroulent. L'abstractrion (tout est objet)

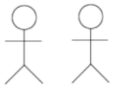
Avantages

- capacité à regrouper ce qui a été séparé,
- à construire le complexe à partir de l'élémentaire,
- à intégrer statiquement et dynamiquement les constituants d 'un système.



Qu 'est-ce qu 'un Objet ?

- **Définition Générale :**
  - « ce sur quoi porte notre connaissance »
- **Pour les technologies objet :**
  - « c’est une abstraction du monde réel »
- **Pour l’analyse du domaine :**
  - **«** c’est une entité pertinente du domaine »
- **Dans un langage de la POO :**
  - «c’est un ensemble de fonctions associées à une structure de données »



Les Objets

Définition

- Concept d'**abstraction** ou entité ayant des liens et un sens pour une application donnée.

Exemples:

- Objets matériels (table, chaise, crayon, avion…)
- Objets immatériels, concepts (compte en banque, équation, match de boxe…)
- Objets virtuels (groupe de travail, division…)

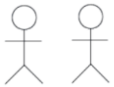
Tout est objet !

Tout objet possède les caractéristiques suivantes:

Objet = Identité + État + Comportement

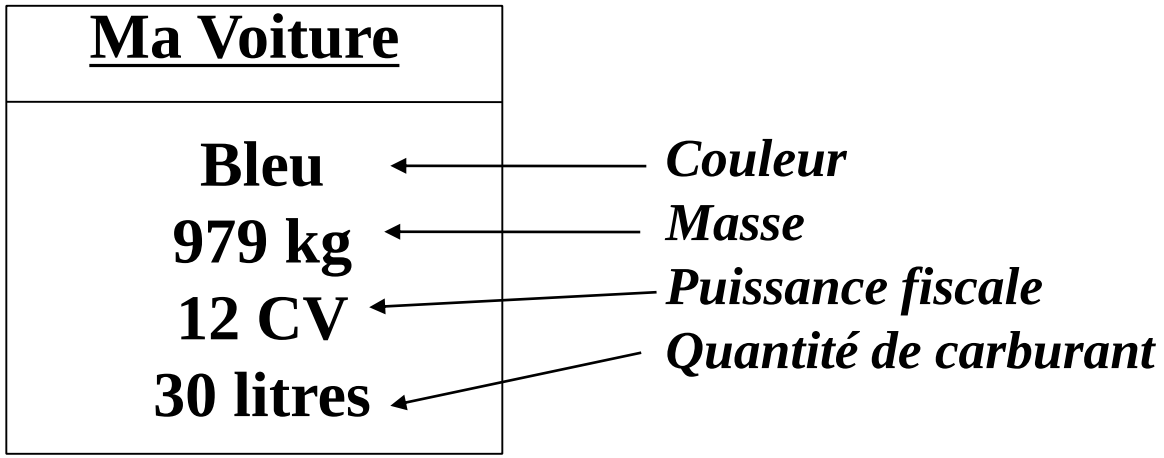
Les Objets: *L'identité*

- **L'identité caractérise l'existence propre de l'objet:**
  - immuable
  - permanente (permet de distinguer tout objet de façon non ambiguë indépendamment de son état)
  - concept qui ne se représente pas de manière spécifique mais qui peut être rajoutée dans l'état des objets comme un artifice de réalisation.



Les Objets : *L'état*

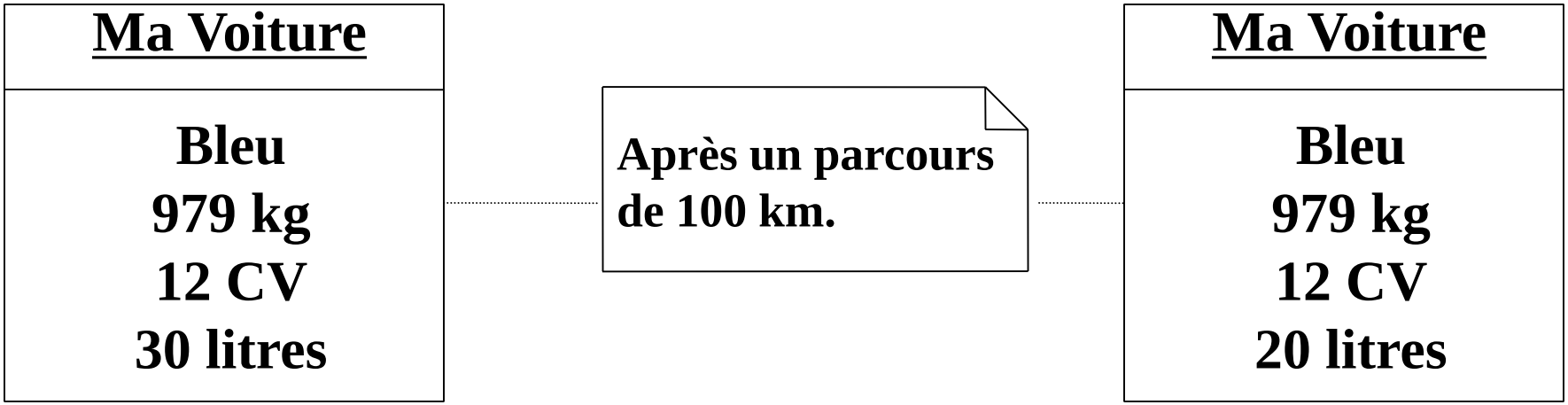
- Regroupe des valeurs instantanées de tous les attributs d'un objet.
- L'état d'un objet est décrit par l 'ensemble de ses attributs et liens et est exprimé au travers des opérations.
- Exemple



Les Objets: *L'état*

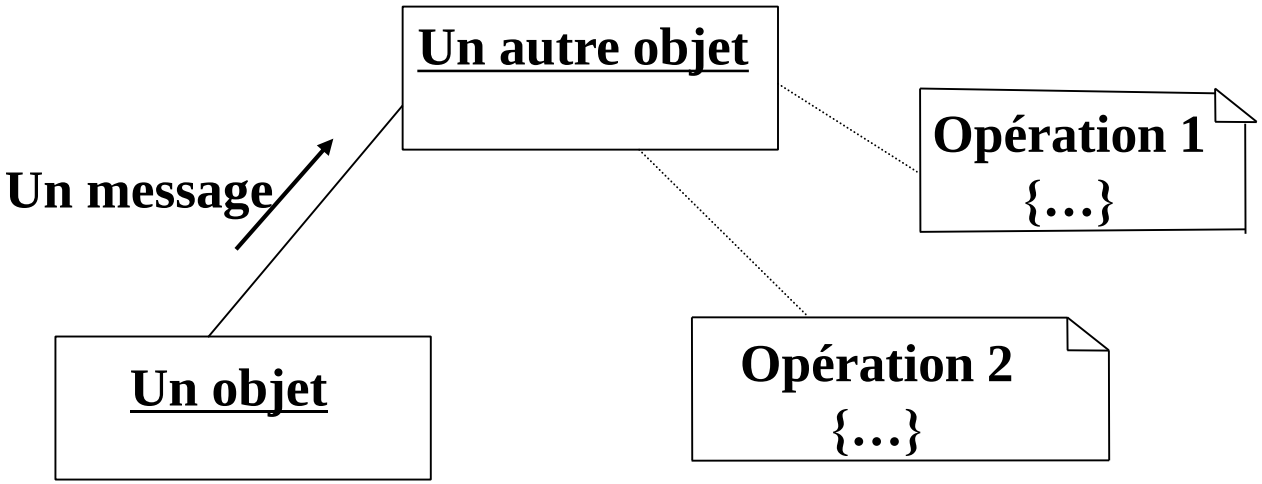
- **L'état évolue au cours du temps :**
  - Certaines valeurs d'attribut vont évoluer ;
  - D'autres vont êtres constants.

- **Exemple**



Les Objets: *Le Comportement*

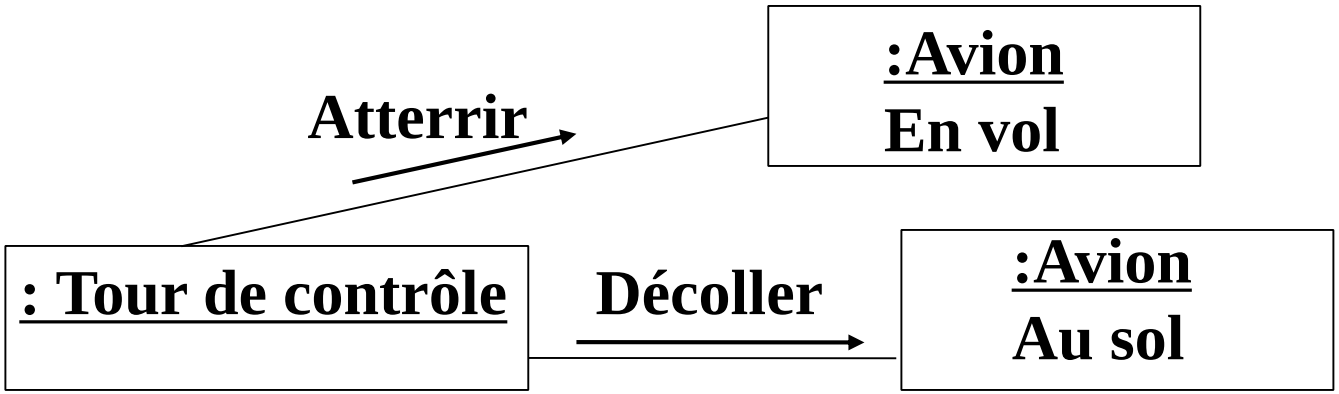
- **Regroupe toutes les compétences d'un objet, l'ensemble des opérations applicables à cet objet**
- **Décrit les actions et les réactions de l'objet.**
  - Concept d'opérations (méthodes) ; une opération a une signature et s 'applique à un objet
  - Une opération est déclenchée suite à une stimulation externe : message envoyé par un autre objet.



Les Objets: Caractéristiques

- **L'état et le comportement sont liés :**
  - Le comportement, à un instant donné, dépend de l'état courant.
  - L'état peut être modifié par le comportement.

- **Exemple: Un avion ne peut atterrir que s'il est en vol.**
  - *atterrir* représente un comportement de l'avion.
  - *en vol* correspond à un état de l'avion.



Les classes d'objet

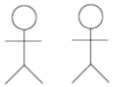
❑ Définitions

Une classe est la description d'un ensemble d 'objets qui ont les mêmes attributs, les mêmes opérations, les mêmes relations et des sémantiques communes

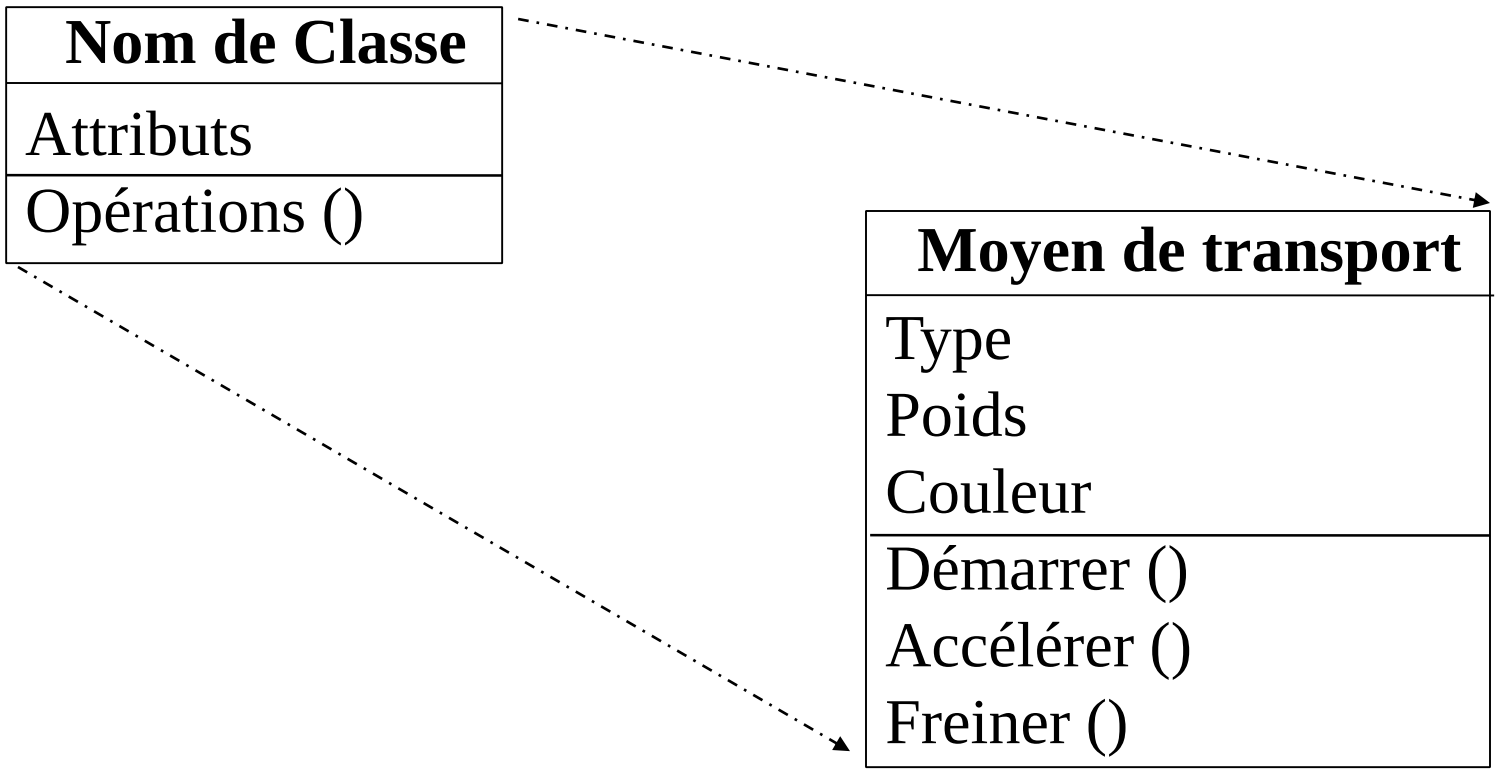
ex : Personne, Etudiant, Compagnie

Un attribut est une propriété nommée d'une classe qui décrit un ensemble de valeurs, il représente une propriété de l'élément modélisé

Une opération est l 'implantation d'un service qui peut être demandé aux objets d 'une même classe dans le but de déclencher un comportement

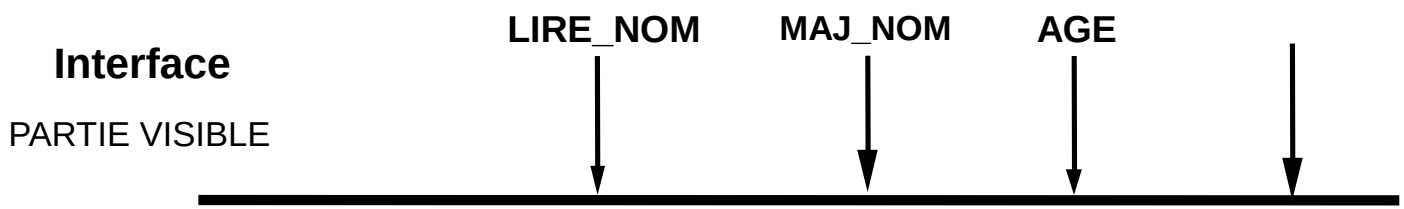


Représentation graphique des classes



L'encapsulation

Interface / Implémentation



age peut être implémenté par un attribut stocké ou par un calcul utilisant la date de naissance)

- **Abstraction : les objets clients ne peuvent accéder à un objet que via son interface**

  Ils ne connaissant pas tout le détail de l'objet

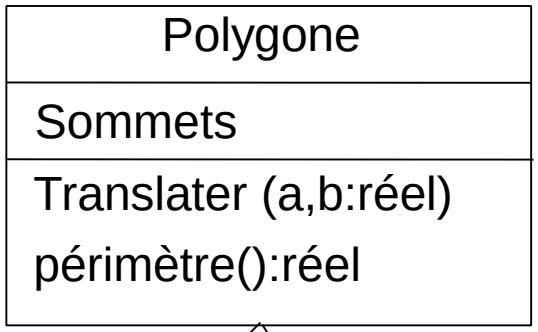
- **Masquage d'information :**

  Indépendance des clients vis à vis des structures de données et des algorithmes

Les relations entre classes : L'héritage

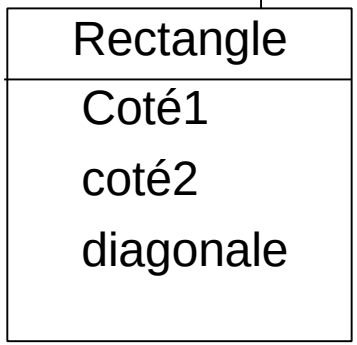
Le lien d'héritage entre classes est une relation entre un élément général et un élément dérivé .

Super classe



La classe Rectangle hérite de la classe Polygone

Sous-classe



Tous les attributs et toutes les opérations de la classe Polygone sont applicables de la même façon à la classe Rectangle

On n 'indique dans la sous classe que les Propriétés spécifiques

Les relations entre classes : L'héritage

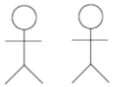
Principe de la redéfinition et Polymorphisme

La règle de calcul du périmètre d'un rectangle peut être optimisée selon la formule p = 2*(coté1+coté2))

> La méthode périmètre peut être redéfinie dans la classe Rectangle afin d'avoir un corps différent (ici la redéfinition est utilisée pour une meilleure efficacité)

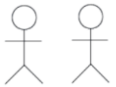
Le même nom "périmètre" désigne deux corps différents, une opération peut avoir plusieurs formes (Polymorphisme)

La redéfinition est un mécanisme sémantique qui assure que le même nom fait référence à des codes différents selon le type de l'objet auquel il s'applique



Les relations entre classes : l'association

- Connexion sémantique bidirectionnelle entre classes
- Abstraction des liens qui existent entre les objets instances des classes associées (en général, un objet ne peut pas agir seul, il a besoin de l 'aide d 'autres objets).
- Dés lors qu'un objet a un lien vers un autre objet, il peut lui envoyer un message pour déclencher une de ses opérations



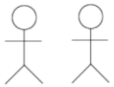
L'agrégation ou la composition : relations entre les classes:

- **Connexion bidirectionnelle dissymétrique**

Un train est composé de wagons, un wagon contient des sièges...

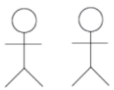
Une voiture possède quatre roues, un châssis, un moteur…

- **Les relations exprimées ici sont des relations de composition (ou d 'agrégation)**

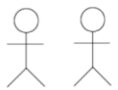


Chapitre 1
EXERCICE

Faire un diagramme d'objet d'un smartphone

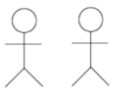


Chapitre 2
UML : Unified Modelling Language

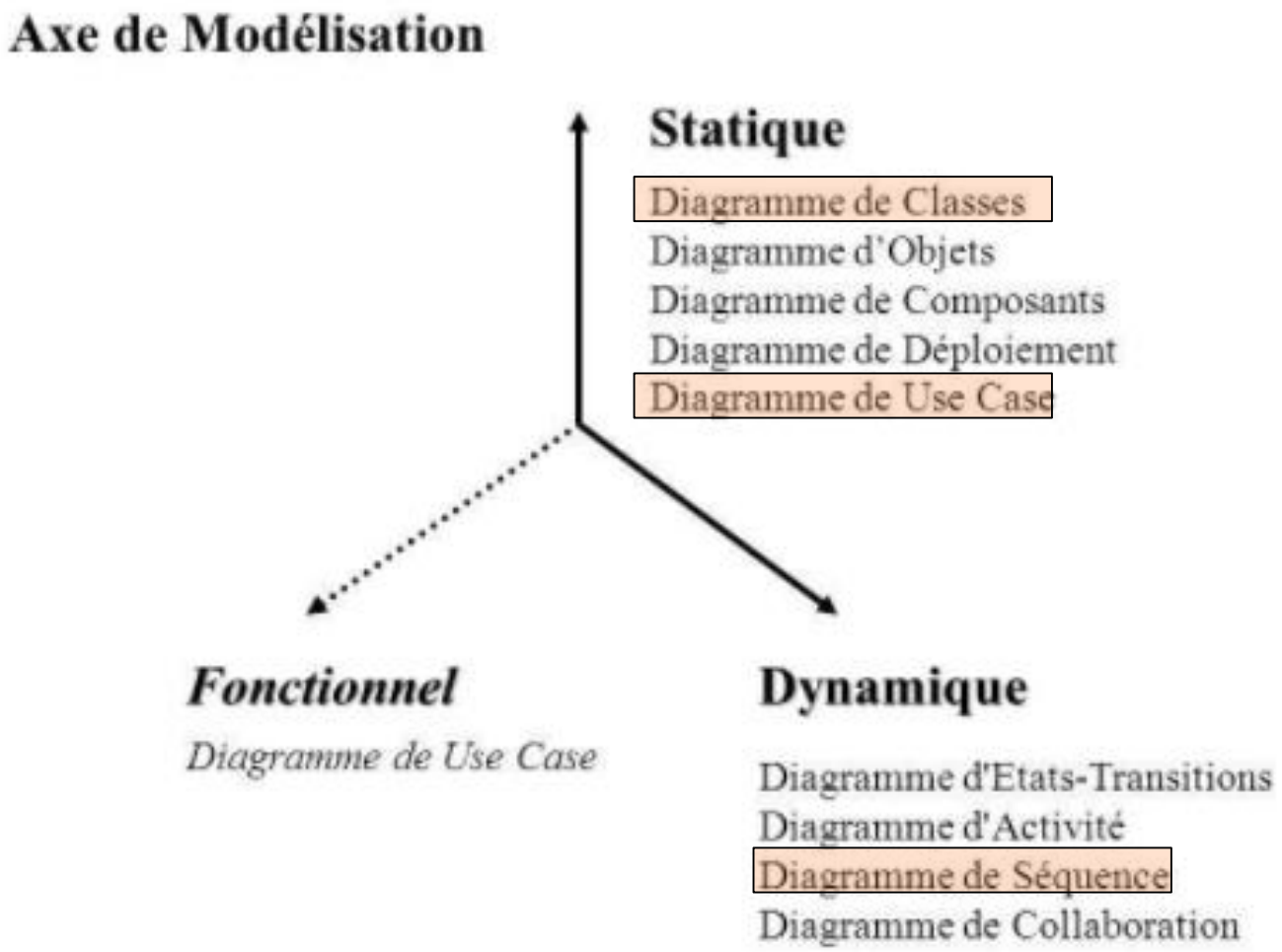


Objectifs d'UML

- **Proposer un langage visuel de modélisation**
  - Utilisable par toutes les méthodes
  - Adapté à toutes les phases du développement
  - Compatible avec toutes les techniques de réalisation
- **Proposer des mécanismes d'extension et de spécialisation pour pouvoir étendre les concepts de base**
- **Être indépendant des langages particuliers de programmation**
- **Proposer une base formelle pour comprendre le langage de modélisation**

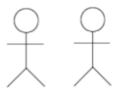


Les diagrammes d'UML



Les Briques de Base

- **Les briques de base de UML sont :**
  - Les éléments de modèle (classes, interfaces, composant, etc.)
  - Les relations (associations, généralisation, dépendences, etc.)
  - Les diagrammes (diagramme de classe, diagramme de cas d'utilisation, diagramme d'interaction, etc.)
- **Les briques de base simples sont utilisés pour construire des structures plus complexes et plus grandes**



Les modèles UML

- **Modèle d'approche : décrit le comportement du système à un haut niveau**

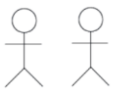
  les cas d'utilisation, les scénarios

- **Modèle de structure : décrit l'aspect statique du système**

  les objets, les classes et leurs relations

- **Modèle de la dynamique : décrit les échanges entre objets**

  interaction, communication, message

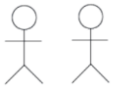


Modèles d'approche: les cas d'utilisation

- Le modèle des UC : une vue du système qui met l'accent sur le comportement du système tel qu'il apparaît aux utilisateurs externes. Il permet la représentation des fonctionnalités du système.

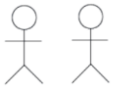
- Les diagrammes de cas d'utilisation sont élaborés pour visualiser les relations entre les acteurs et les cas d'utilisation

- Les diagrammes de cas d'utilisation présentent une vue extérieure du système



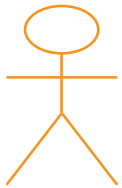
Les cas d'utilisation

- Formalisés par Ivar Jacobson.
- Décrivent sous la forme d'actions et de réactions, le comportement d'un système du point de vue de l'utilisateur.
- Manière spécifique d'utiliser un système. C'est l'image d'une fonctionnalité du système, déclenchée en réponse à la stimulation d'un acteur externe.
- Définissent les limites du système et les relations entre le système et l'environnement.



Acteurs et cas d'utilisation

- **Acteurs et cas d'utilisation permettent de décrire le système:**
  - Les acteurs interagissent directement avec le système



Un acteur

<<acteur>>
Un autre acteur

  - Les cas d'utilisation représentent l'utilisation du système par les acteurs

Un cas d'utilisation

Les Acteurs

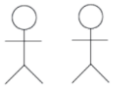
Un Acteur représente un rôle joué par une personne ou une chose qui interagit avec le système

guichetier

Conseiller financier

Un acteur a besoin d'échanger des informations avec le système.

Même si on les utilise dans les modèles, les acteurs ne font pas partie du système puisqu'ils résident en dehors de celui ci.



Les Acteurs

Trois types d ’acteurs :

- **les personnes, ce sont des utilisateurs du système**

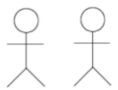
(acteurs principaux, acteurs secondaires)

- **le matériel externe, dispositif utilisé par le système**

ex: l’imprimante d’un distributeur de billet

- **les autres systèmes, qui communiquent avec le système**

ex: Le groupement bancaire dans un système de distributeur de billets



Les Cas d'utilisation

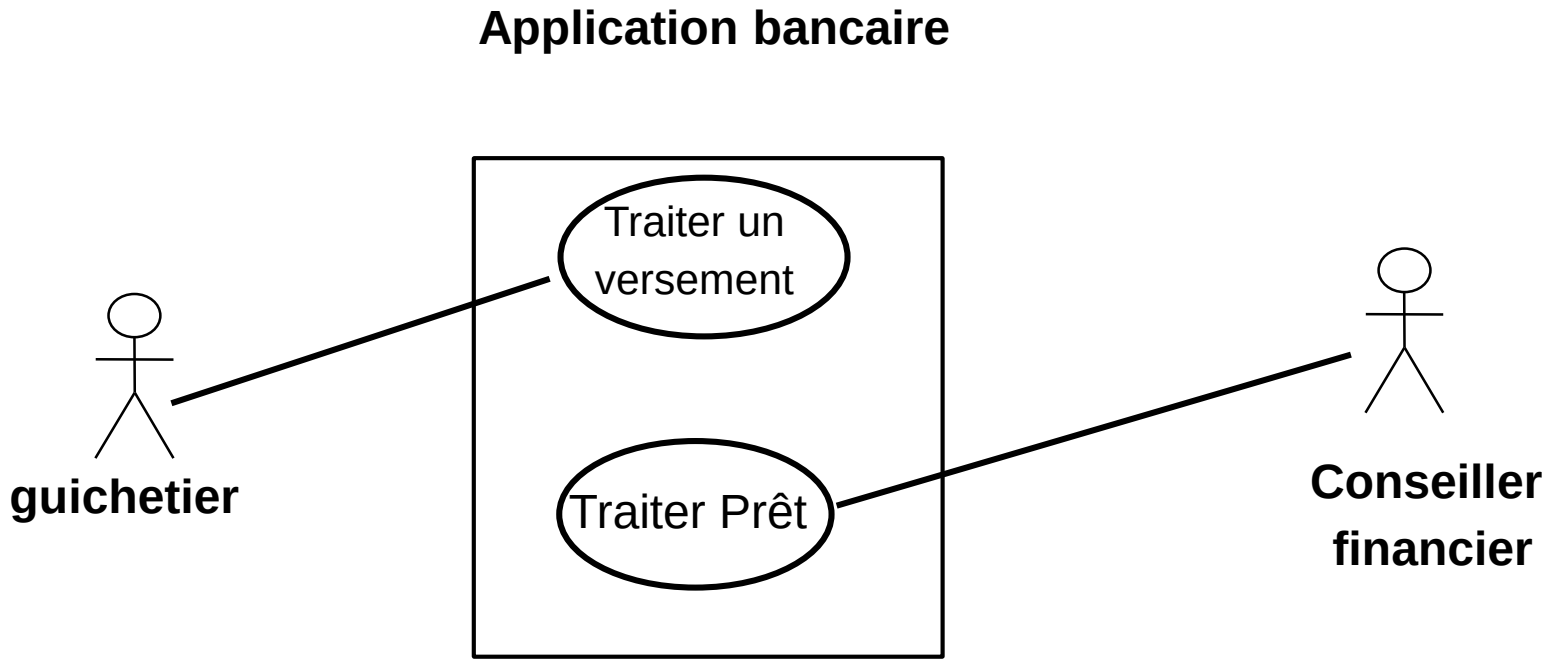
Un cas d 'utilisation modélise une fonctionnalité du système

Ex :

Un cas d 'utilisation décrit ce que fait un système mais ne précise pas comment il le fait

La réalisation d'un cas d'utilisation se traduit par un échange de messages entre le système et ses acteurs

Diagramme de cas d'utilisation



Un cas d'utilisation définit un comportement du système sans révéler sa structure interne; il spécifie les services que le système fournit à ses utilisateurs et les interactions acteurs/système

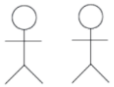
cas d'utilisation et scénarios

- Un cas d'utilisation se détermine en observant acteur par acteur les séquences d'interactions – scénarios – du point de vue de l'utilisateur

- Scénario = « instance » d'un cas d'utilisation ou sa « réalisation »

- Un scénario est un accomplissement d'un cas d'utilisation
  - Il est initié par un message venant d'une instance d'acteur
  - il accomplit une séquence d'actions telle que spécifiée par le cas d'utilisation

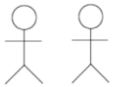
- La réalisation d'un cas d'utilisation est accomplie comme une transaction atomique
  - elle ne peut être interrompue par une autre instance de cas d'utilisation



cas d'utilisation et scénarios

- **Décrire le cas d'utilisation = décrire l'ensemble de scénarios potentiels**

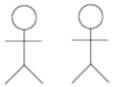
- **Un scénario est composé de plusieurs chemins**
  - un chemin de base ou nominal
    - l'ensemble le plus commun ou plus général d'interactions
  - un ou plusieurs chemins alternatifs
    - Variantes : itérations et alternatives
  - un ou plusieurs chemins d'exceptions
    - Anomalies : l'ensemble des interactions traitant les cas d'erreur



Description d'un cas d'utilisation

- **Spécifier une séquence initiée par l'utilisateur**
  - interactions entre
    - les utilisateurs et le système
    - les réponses effectuées par le système du point de vue extérieur (au système)

- **Intégrer les variantes à cette séquence**
  - alternatives
  - comportements exceptionnels
  - interception d'erreur



Description: patron textuel

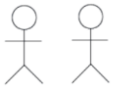
Relations entre cas d'utilisation

❑ Relation « include »

- inclusion d'un cas d'utilisation dans un autre
- à utiliser quand on répète plusieurs fois la même séquence dans différents cas d'utilisation

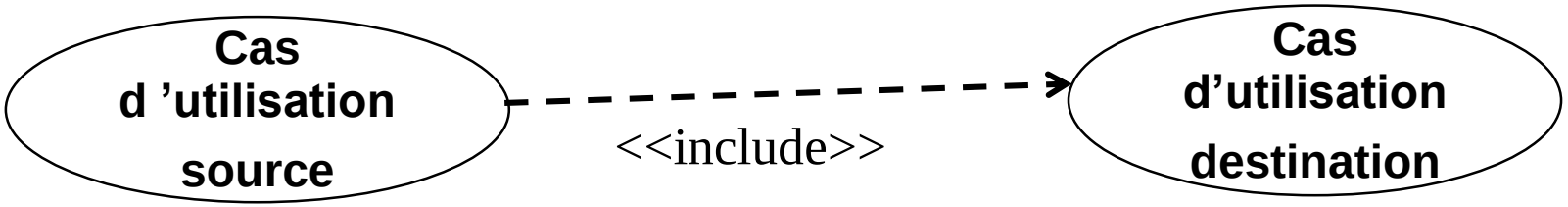
❑ Relation « extends »

- ajout optionnel de comportement dans un cas d'utilisation
  - à définir
    - condition d'extension
    - point d'extension dans le cas d'utilisation étendu
- à utiliser quand on décrit une variation sur un comportement normal

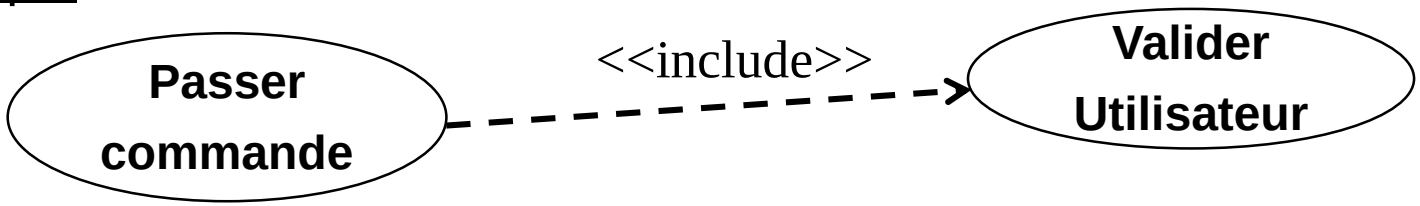


Relation d'inclusion : définition

La relation d ’inclusion signifie que le cas d'utilisation source comprend le comportement décrit par le cas d'utilisation destination en un point d’insertion bien déterminé



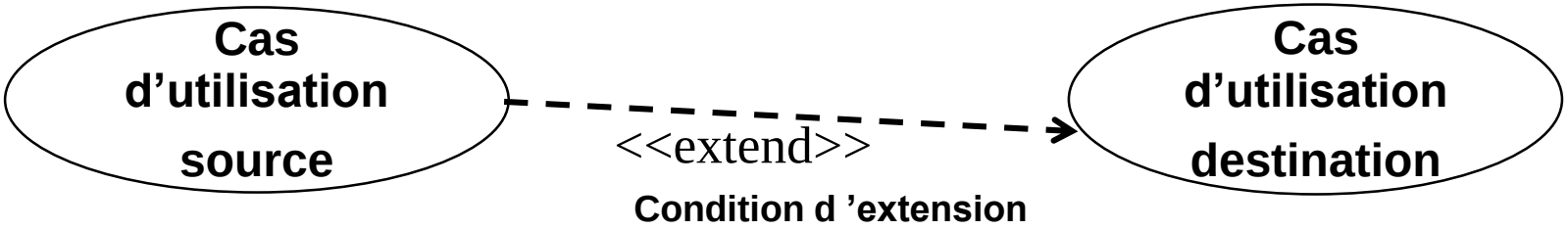
exemple



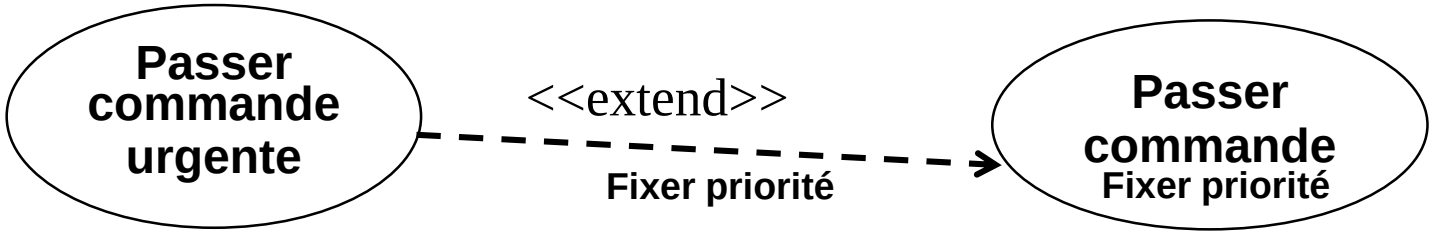
La relation d ’inclusion est un exemple de délégation. Un cas d’utilisation est partagée par plusieurs cas d ’utilisation

Relation d'extension : définition

Une relation d'extension entre cas d'utilisation signifie que le cas d'utilisation source ajoute son comportement au cas d'utilisation destination.

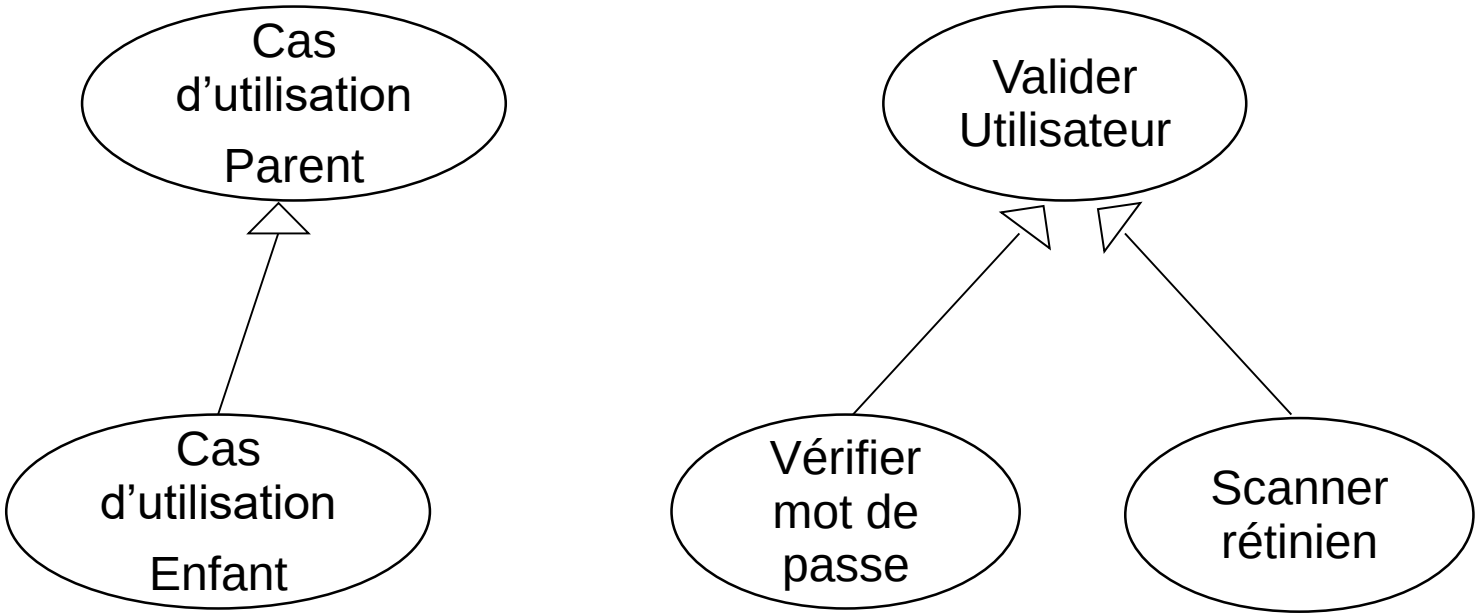


exemple

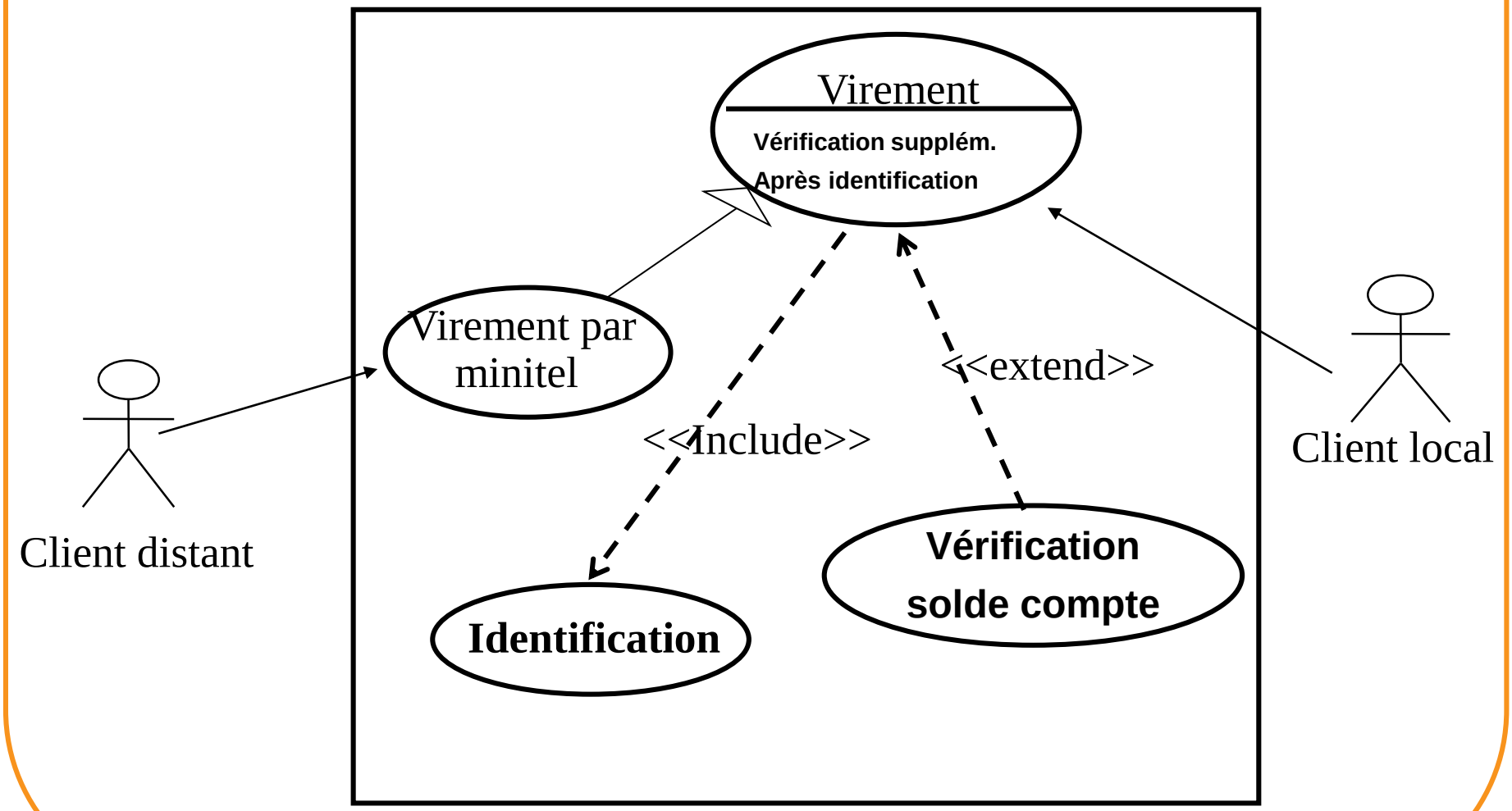


Relation de généralisation : définition

Une relation de généralisation entre cas d'utilisation signifie que le cas d'utilisation enfant est une spécialsation du cas d'utilisation parent. Le cas d'utilisation parent peut être abstrait.



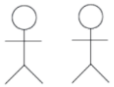
Un diagramme de cas d'utilisation



Construction des cas d'utilisation

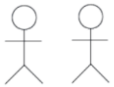
- Un cas d'utilisation doit avant tout être simple, intelligible, décrit de manière claire et concise.

- Lors de la construction, il faut se demander:
  - Quelles sont les tâches de l'acteur ?
  - Quelles informations l'acteur doit-il créer, sauvegarder, modifier, détruire ou simplement lire ?
  - L'acteur devra-t-il informer le système des changements externes ?
  - Le système devra- t-il informer l'acteur des conditions internes ?



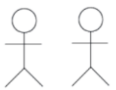
Règles de mise en œuvre des cas d'utilisation

- Un cas d'utilisation décrit une fonctionnalité, une interaction entre un acteur et un système sous la forme d'un flot d'évènements.
- La description de l'interaction se concentre sur ce qui doit être fait.
- Un cas d'utilisation doit être simple.
- Un cas d'utilisation doit éviter d'employer des expressions floues et imprécises.



Construction des cas d'utilisation

Chapitre 2
UML : Unified Modelling
EXERCICE

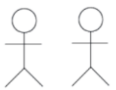


Les modèles de structure

- Montre la structure statique d'un modèle

  (e.g., classes, interfaces, composants, noeuds)

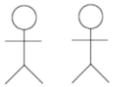
- Types de diagrammes
  - Diagrammes structurels statiques :
    - Diagramme de classes ;
    - Diagramme d'objets ;
  - Diagrammes d'implantation :
    - Diagramme de composant ;
    - Diagramme de déploiement.



Le Modèles de structure

- **Représentation de la structure statique du système**
  - Les composants du système ;
  - Les liens sémantiques entre ces composants ;

- **Contenu**
  - Objets et classes ;
  - Liens et associations ;
  - Généralisation ;
  - Modules ou paquetages.



Objet et Classe : définitions

Objet

- Une abstraction
- Une entité qui a un sens propre dans le domaine ou dans l'application
- Un objet est une instance d'une classe

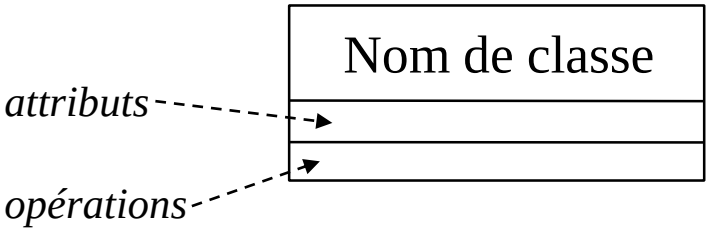
Classe

- Représente un groupe d'objets qui ont une même sémantique et des caractéristiques communes :
  - **attributs, opérations, relations avec d'autres objets**

Classe **Objet de classe**

Les Classes d'objet

Exemples:



Nom de classe

Personne
nom prénom dateNaissance
age()

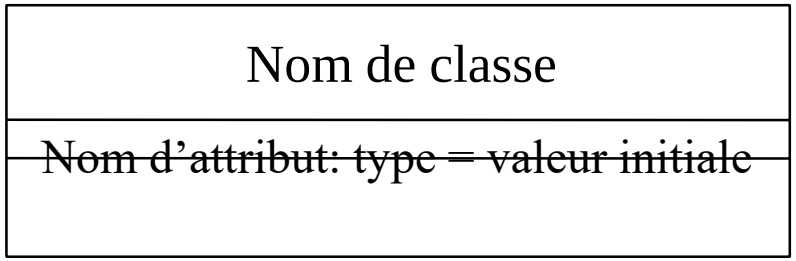
Personne

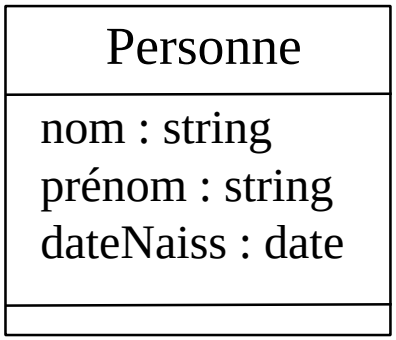
Poste de travail

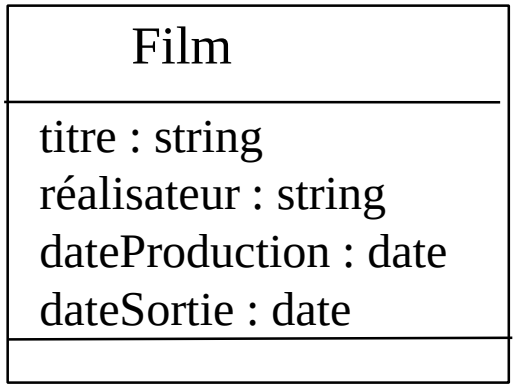
Département

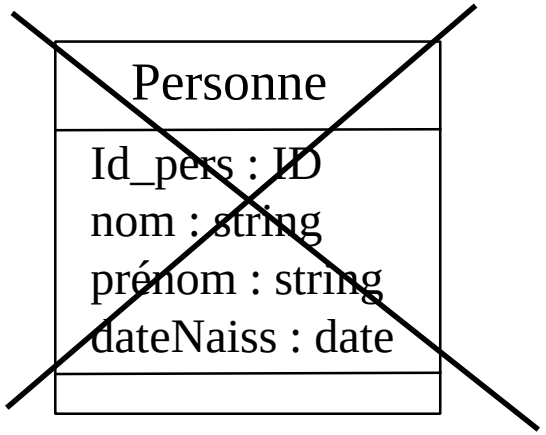
Objet et Classe : les *Attributs*

- **Définissent les propriétés des objets d'une classe**









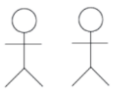
Chaque nom d'attribut est unique dans une classe.

L' identification est fournie par le système, elle n'est pas à considérer dans le modèle objet.

Objet et Classe : les *Attributs*

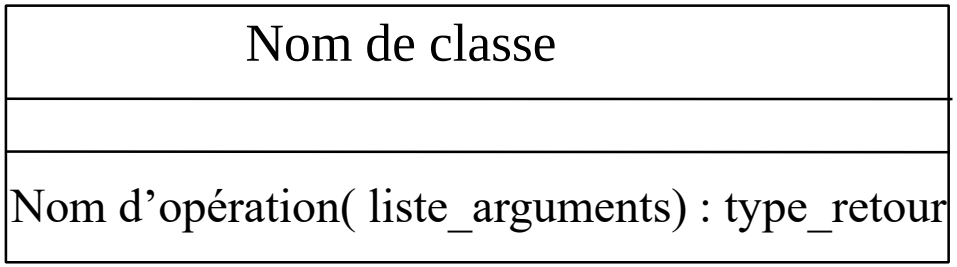
❑ Attribut

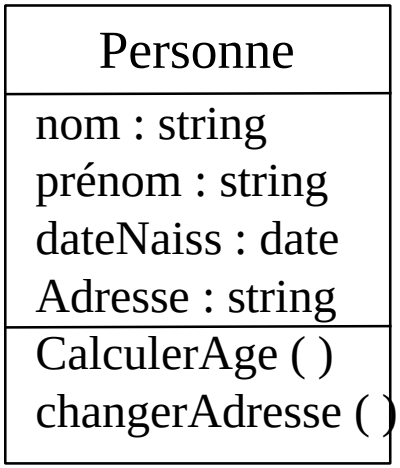
- Une donnée dont la valeur caractérise un objet
- Un attribut est une propriété de la classe
- Un attribut peut être structuré
  - **adresse = (numéro, rue, ville, code postal)**
- Attribut identifiant de l'objet
  - **Un objet n'a pas d'identifiant, sauf si un attribut l'identifie dans le monde réel**

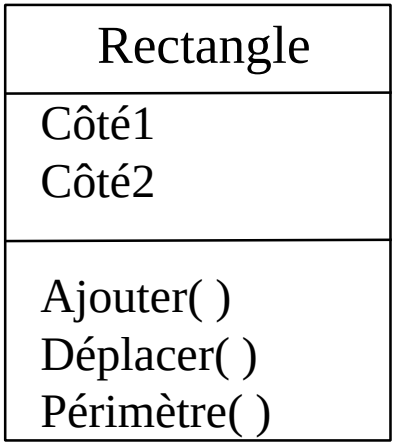


Objet et Classe : les opérations

Une opération définit une fonction applicable aux objets de la classe





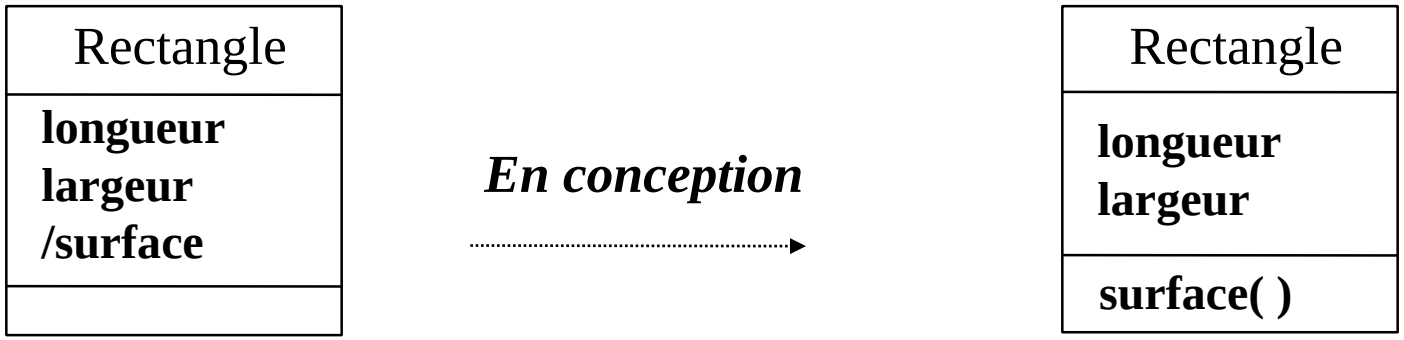


Une méthode est la mise en oeuvre d'une opération pour une Classe (plusieurs méthodes pour le même service)

Objet et Classe : Attribut dérivé

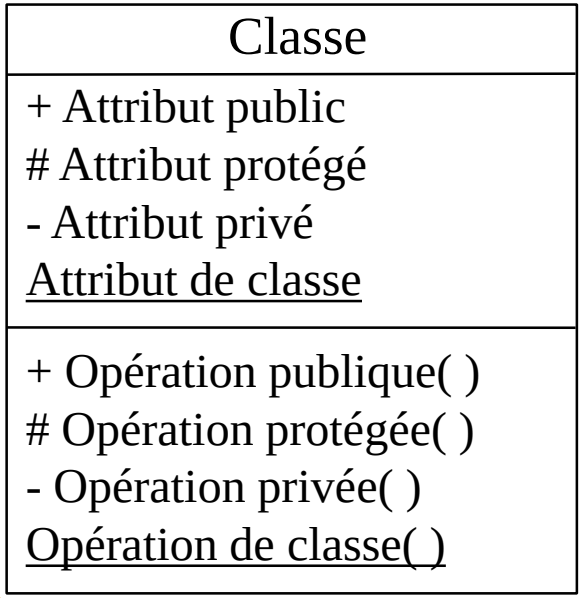
Les attributs dérivés offrent une solution pour allouer des propriétés à des classes, tout en indiquant clairement que ces propriétés sont dérivées d'autres propriétés déjà allouées.

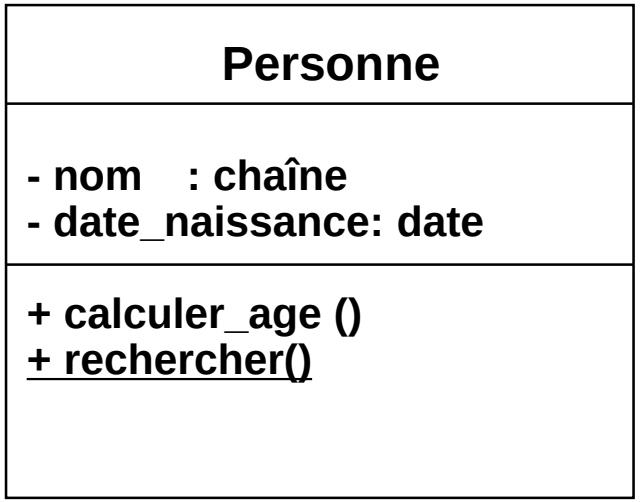
❑ Exemple:



Objet et classes : Visibilité des attributs et des opérations

- **3 niveaux de visibilité pour les attributs et les operations :**
  - *public* **(+)** qui rend l'élément visible à tous les clients de la classe
  - *protégé* **(#)** qui rend l'élément visible aux sous-classes de la classe
  - *privé* **(-)** qui rend l'élément visible à la classe seul

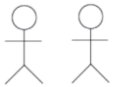




Les Relations entre classes

Lien et Association

- **Association**
  - Une relation sémantique entre classes
  - Représente l'ensemble des liens entre les objets des classes qui participent à l'association
- **Degré d'une association**
  - Le nombre de classes qui participent à l'association
  - Une association peut être réflexive
- **Lien**
  - Une connexion physique entre objets
  - Une instance d'une association
  - Similarité : classe-objet et association-lien



Les relations entre classes

Lien et Association

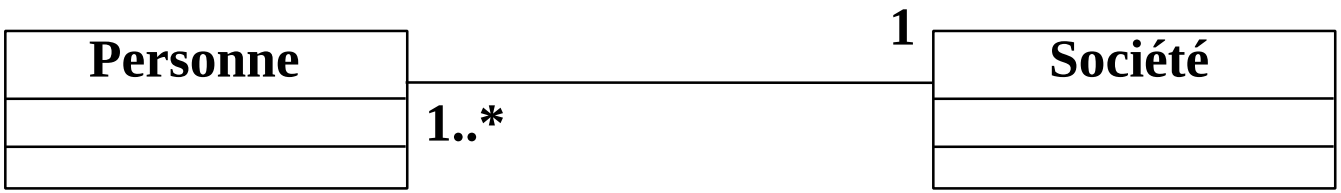
- Une association est une relation structurelle qui précise que les objets d ’une classe sont reliés aux objets d’une autre classe

Lien et Association : modélisation et mise en oeuvre

- **Une association ne doit pas être modélisée par un attribut**
- **Une association est mise en oeuvre (programmée) par**
  - un attribut identifiant un pointeur d'objet,
  - une classe
  - une structure d'accueil (container) telle que liste, tableau, etc.

Lien et Association : multiplicité

- **La multiplicité est une information qui définit combien d'objets de la classe considérée peuvent être liés à un objet de l'autre classe**



Chaque personne travaille pour une société,
chaque société emploie de une à plusieurs personnes.

Multiplicité = cardinalité

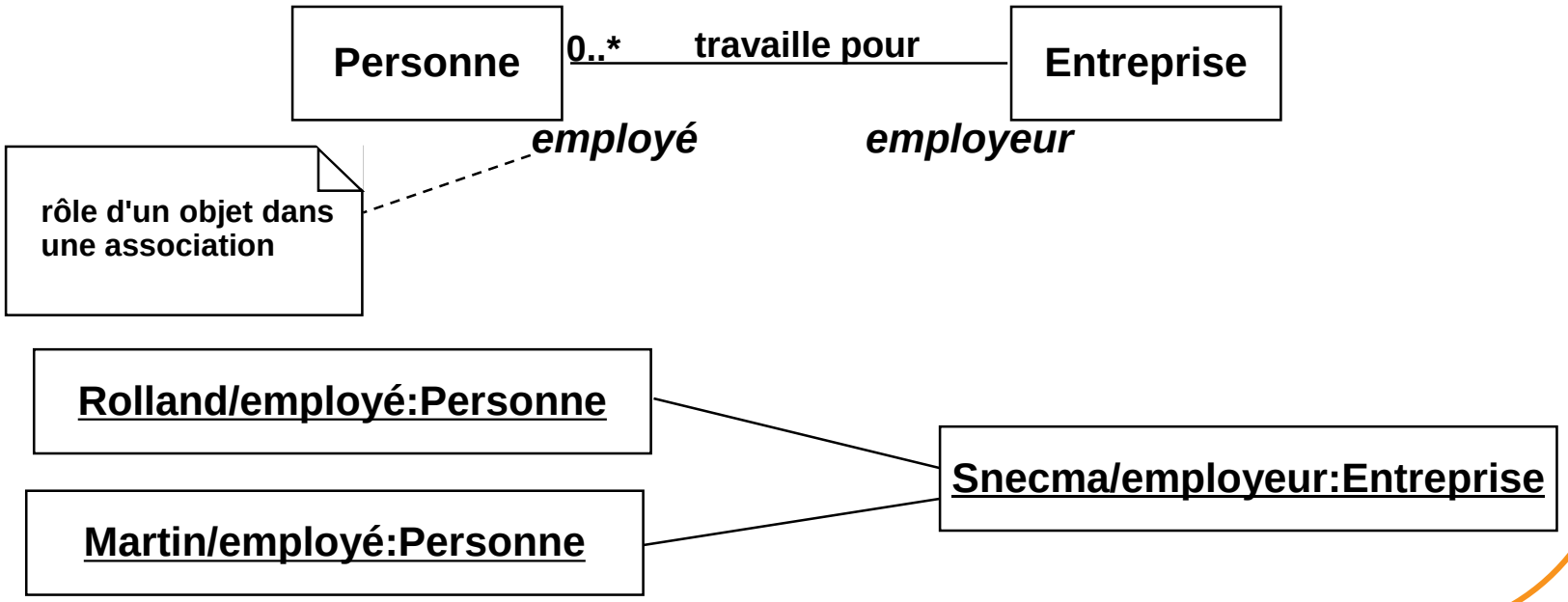
Lien et Association : multiplicité

1		1 (un et un seul)
*	0..*	plusieurs (zéro ou plus)
	0..1	facultatif (au plus un)
	1..*	obligatoire (au moins 1)
	2..4	deux, trois ou quatre

- Les valeurs de multiplicité expriment des contraintes liées au domaine de l'application, valables durant l'existence des objets.

Lien et Association : rôle

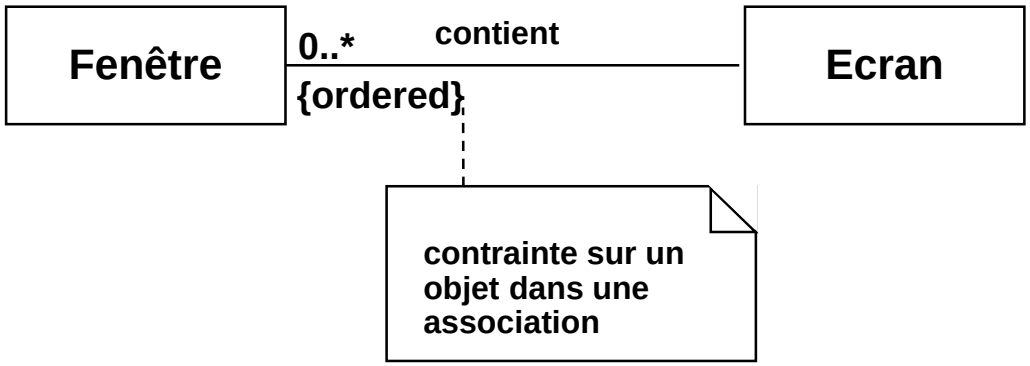
- **"Rôle" des participants dans une association**
  - Un "rôle" peut être spécifié pour une extrémité de l'association.
  - Il exprime le rôle d'une classe dans l'association.
  - il facilite la lecture et la compréhension du modèle objet.



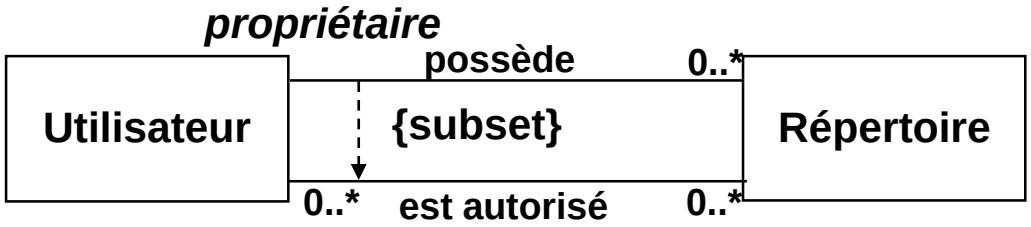
Lien et Association : contraintes

❑ Contrainte sur les participants dans une association

- ➢ **Ordre implicite sur les objets de la classe qui participe à l'association, avec une cardinalité supérieure à 1**



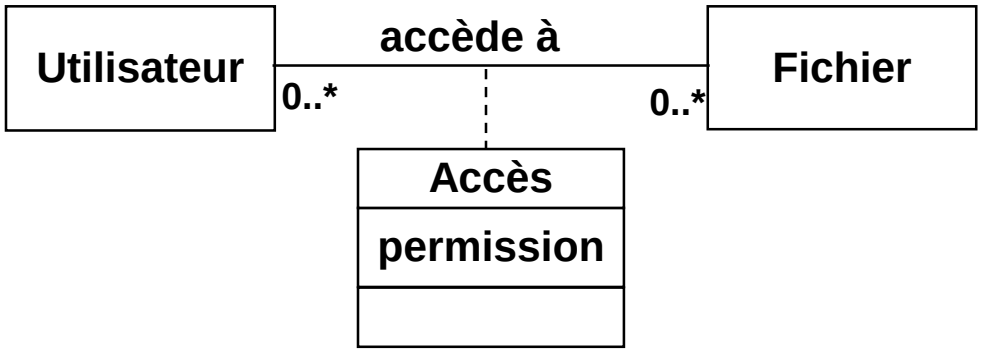
❑ Contrainte de sous-ensemble de liens entre associations



Lien et Association : attribut de lien

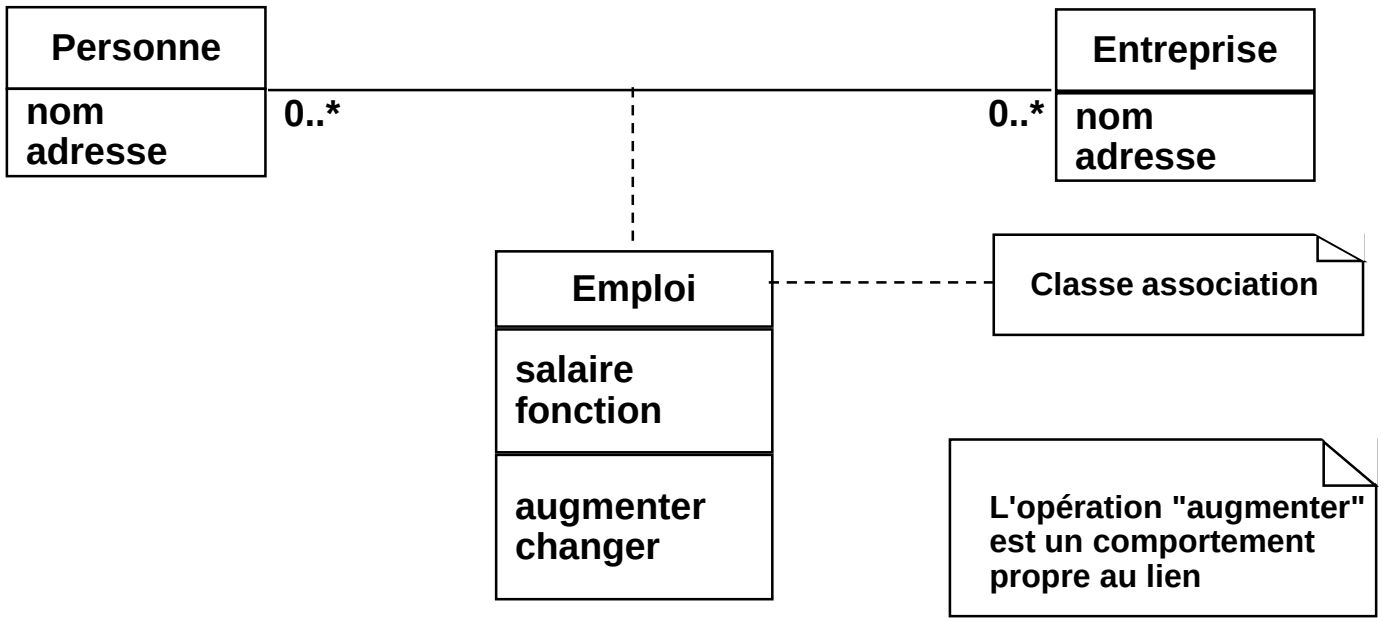
- Une valeur portée par un lien
- Une propriété de l'association

- Ce n'est pas une propriété d'un des objets qui identifie le lien
- Ce n'est pas une propriété d'une des classes qui identifie l'association



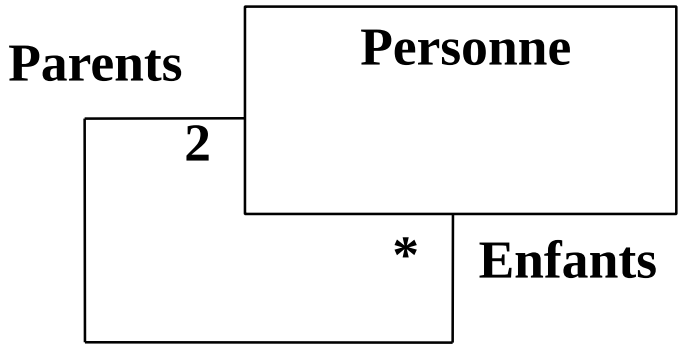
Lien et Association : association modélisée en classe

- Chaque lien est une instance de la classe qui modélise l'association
- A utiliser quand les liens ont des opérations



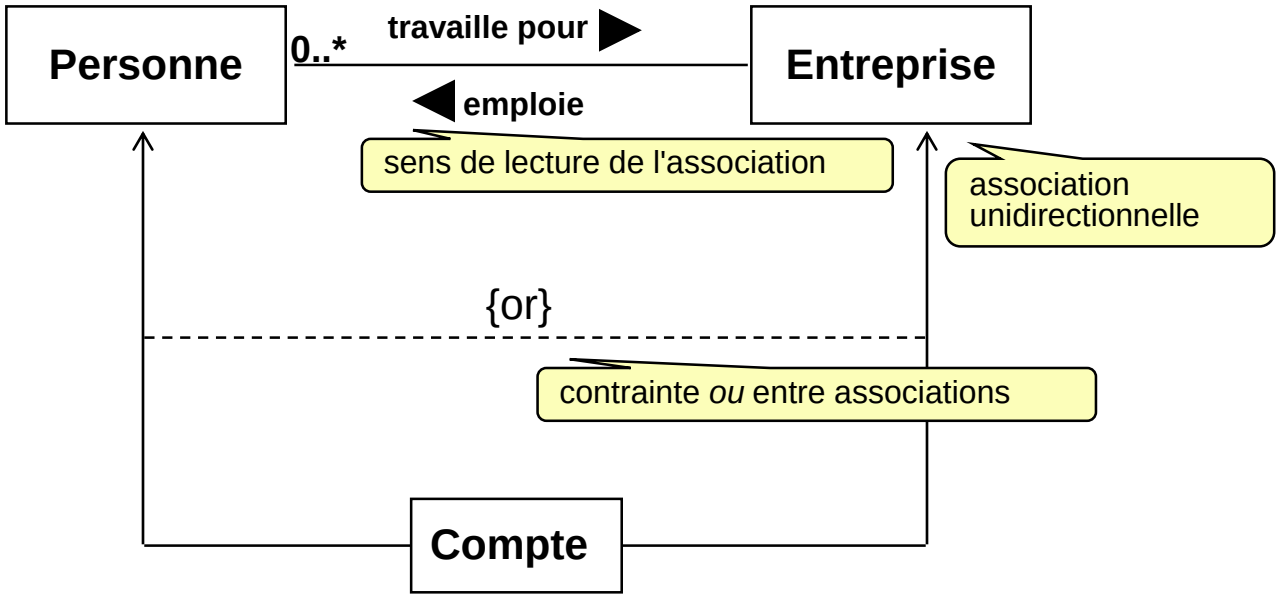
Les Associations, Contraintes sur les associations

- Association réflexive



Le nommage des rôles est essentiel à la clarté du diagramme.

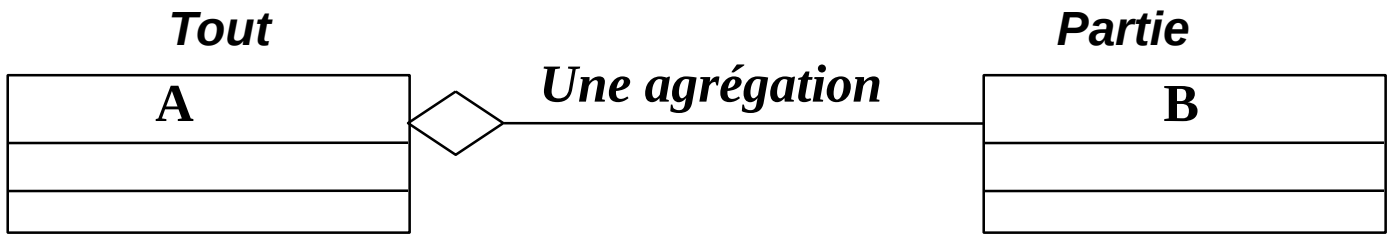
Lien et Association : sens, direction, contrainte



Les relations entre classes : Les agrégations

- Représente une association non symétrique dans laquelle une classe joue un rôle prédominant par rapport à l'autre classe.

- Les critères suivants impliquent une agrégation:
- une classe fait partie d'une autre classe;
- les objets d'une classe sont subordonnés aux objets d'une autre classe;



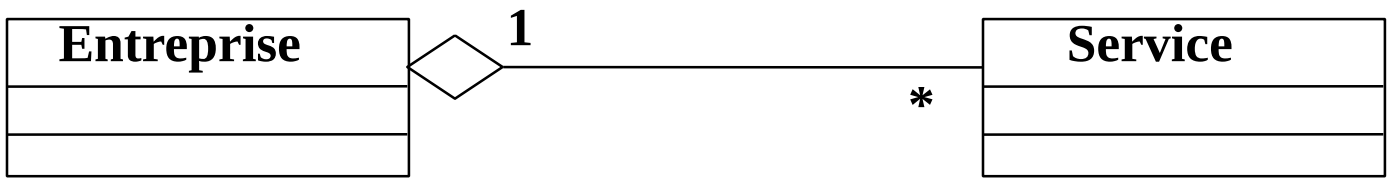
La relation d'agrégation représente la relation 'Possède'

Agrégat : caractéristiques

- Transitivité
- Antisymétrie
- Propagation d'opérations
  - Certaines opérations du "tout" se propagent (avec modification) aux "parties"
  - Exemples : corriger document, corriger paragraphe, corriger mot
- Quand utiliser l'agrégat
  - Une "partie" ne peut pas exister indépendamment du "tout"

Les relations entre classes : Les agrégations

- L'agrégation peut être multiple, comme l'association

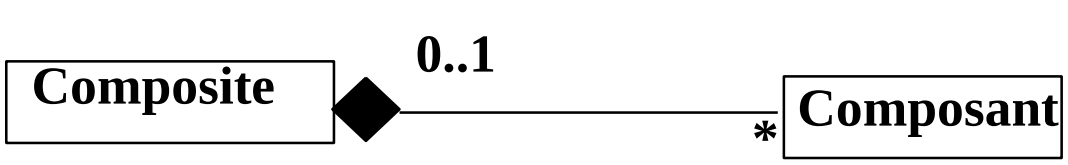


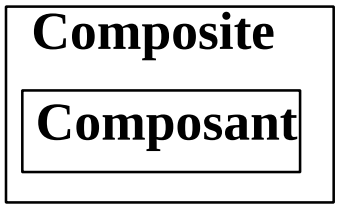
Cette forme d 'association a pour but de distinguer le tout de la partie.

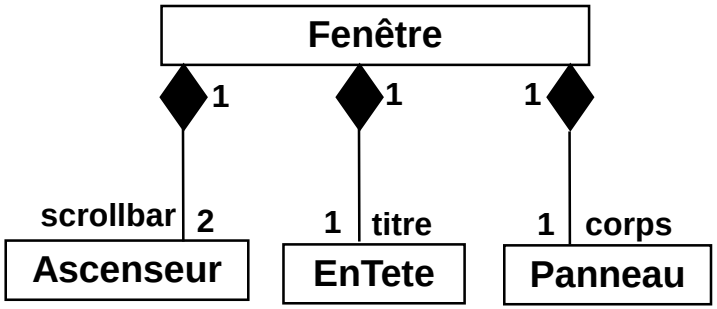
Elle est uniquement conceptuelle

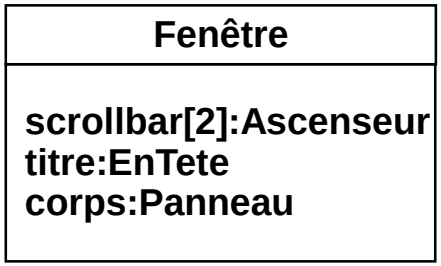
Les relations entre classes : La composition

- La composition est un cas particulier de l'agréation
- Agrégation réalisée par valeur sur des attributs: composition.
- Ils sont physiquement contenus dans l'agrégat.





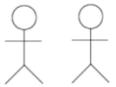




Les relations entre classes : La composition

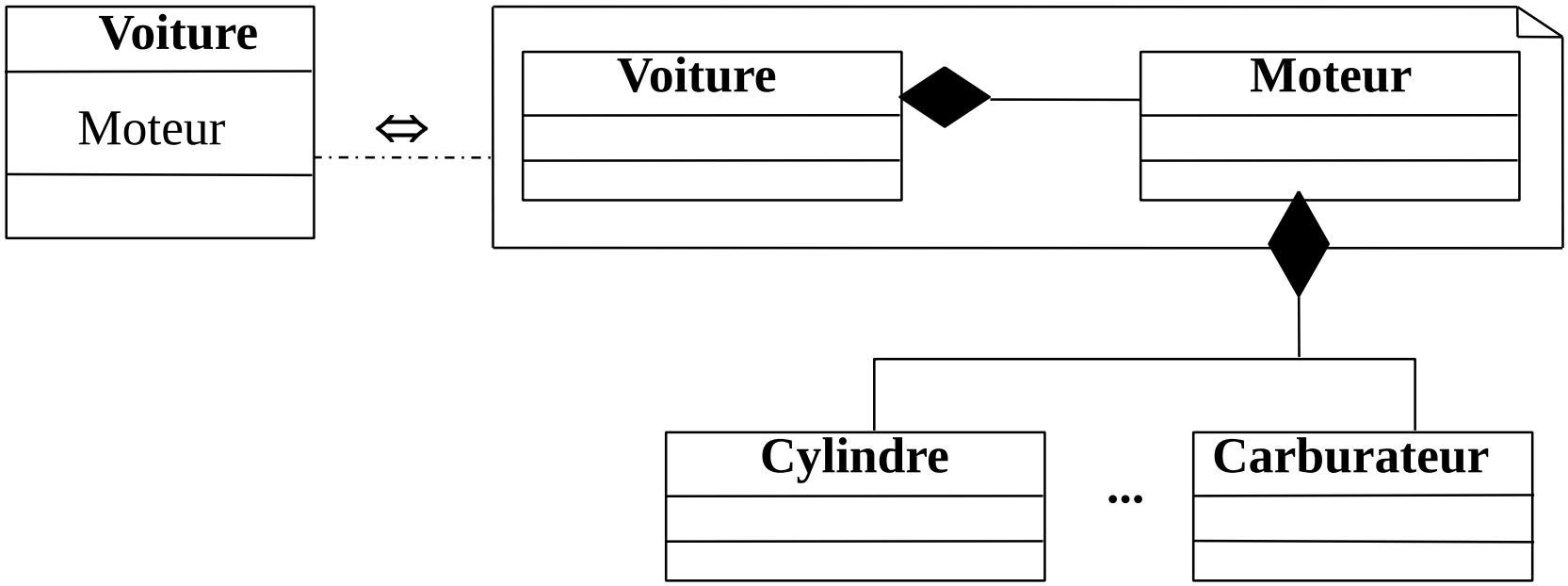
- Forme d'agrégat avec appartenance forte et vies coïncidentes des parties au tout:
  - La destruction du composite implique la destruction de tous ses composants,
  - La création, la modification et la destruction des composants sont de la responsabilité du composite

- La multiplicité du côté du tout ne peut dépasser « 1 »
  - Un composant est non partagée



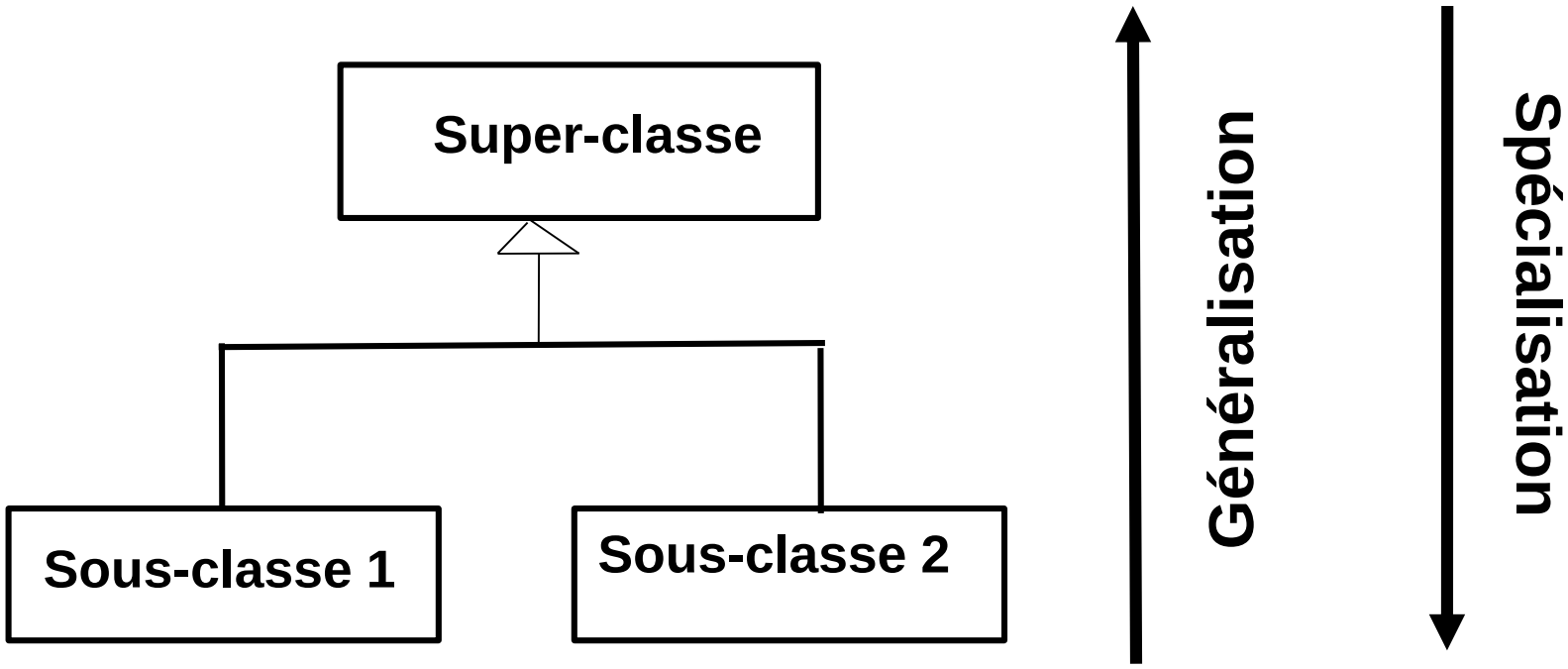
Les relations entre classes : La composition

Un autre exemple



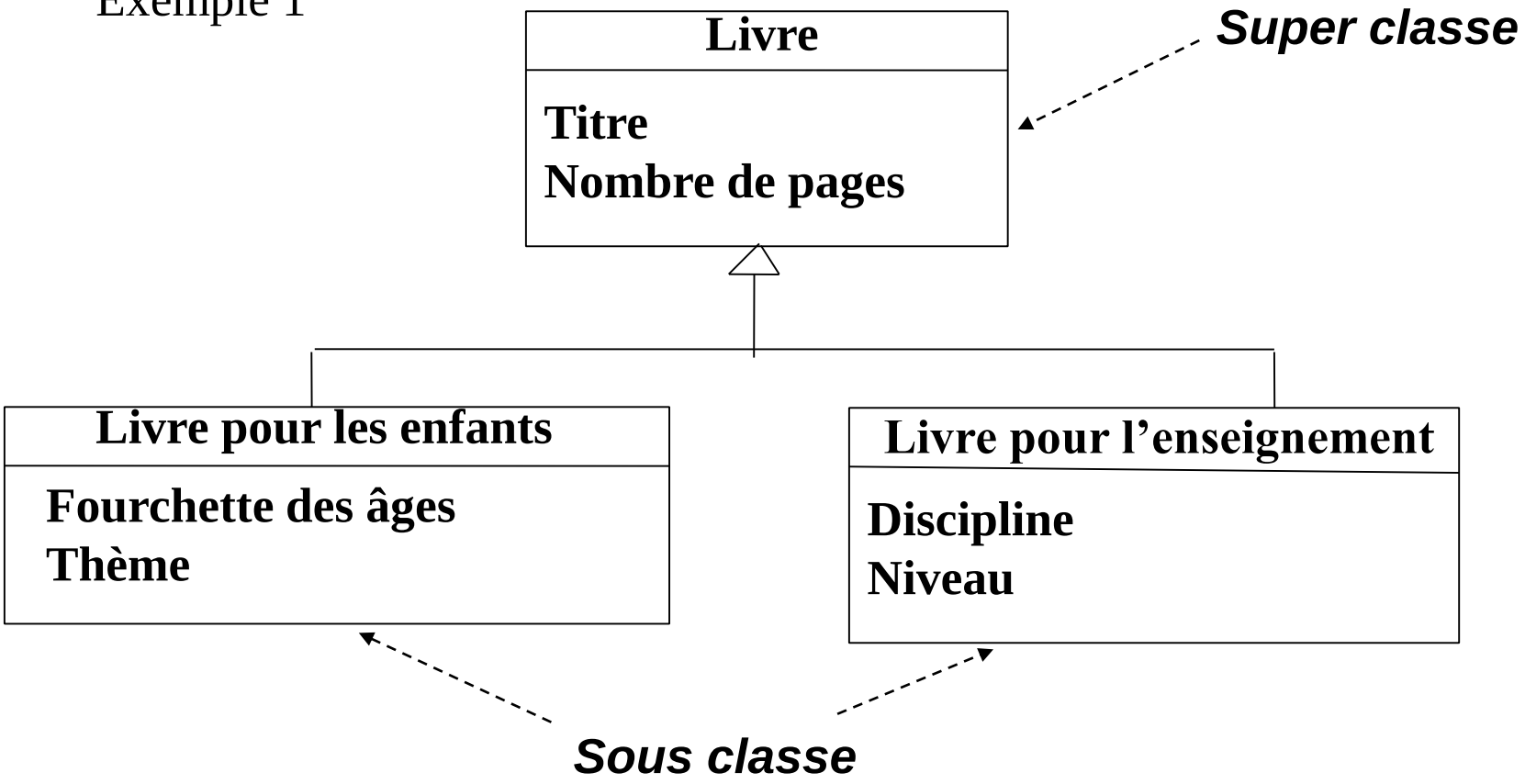
Les relations entre classes : généralisation/Spécialisation

- La généralisation est une relation entre classes qui exprime le lien sémantique “est-un”



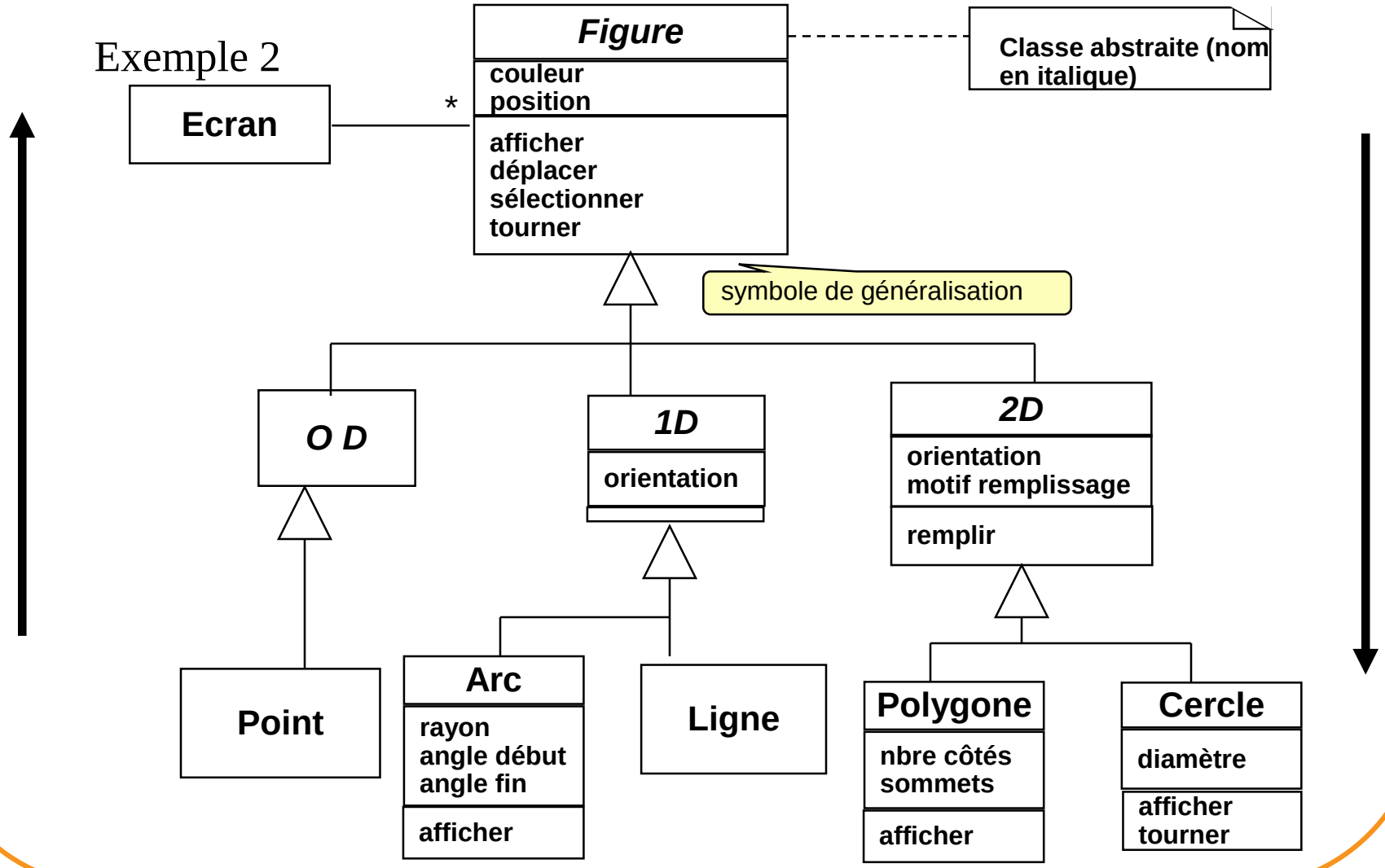
Les relations entre classes : généralisation/Spécialisation

Exemple 1



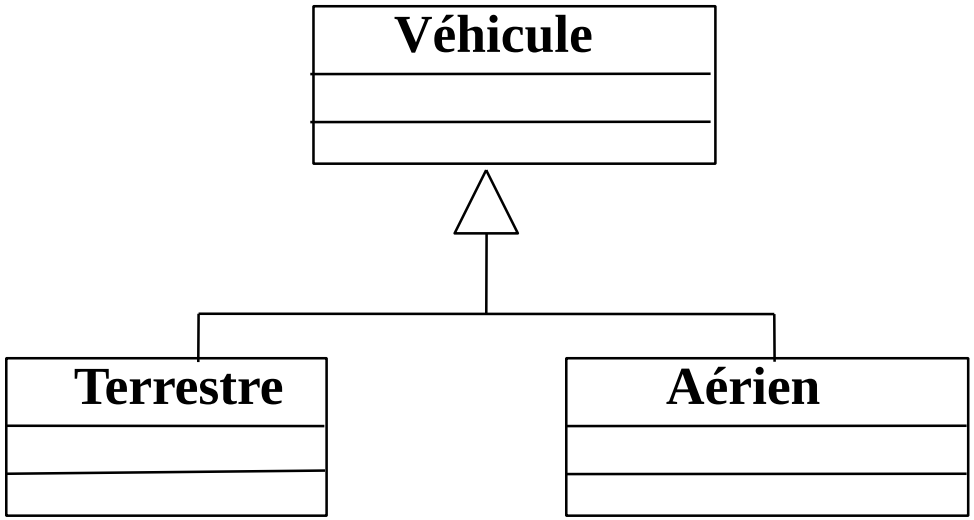
Les relations entre classes : généralisation/Spécialisation

Exemple 2



Les relations entre classes : généralisation/Spécialisation

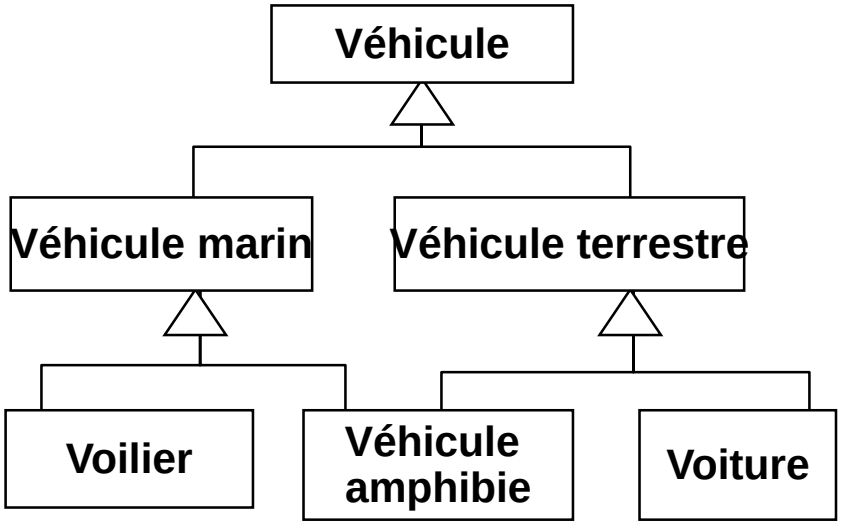
Les classes sont ordonnées selon une hiérarchie : une superclasse est une abstraction de ses sous-classes



- La généralisation est une relation non réflexive.
- La généralisation est une relation non symétrique.
- La généralisation est une relation transitive.

Généralisation : héritage multiple

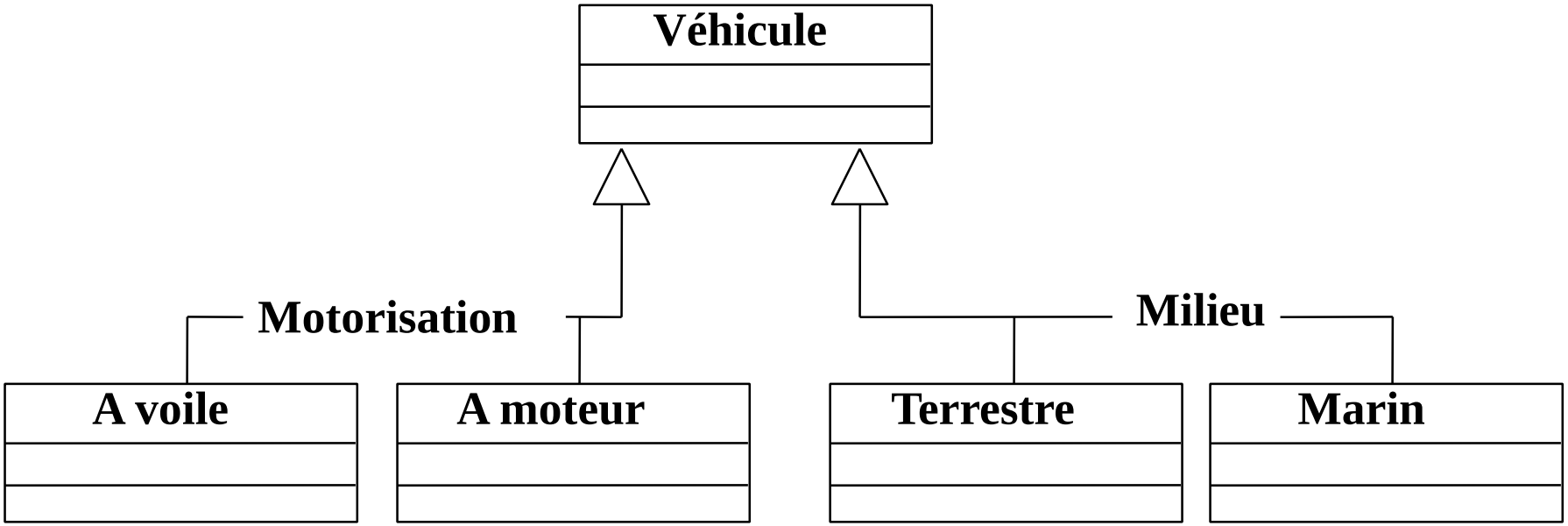
- **Possibilité d'hériter de plusieurs classes**



Problème : Conflit de nom

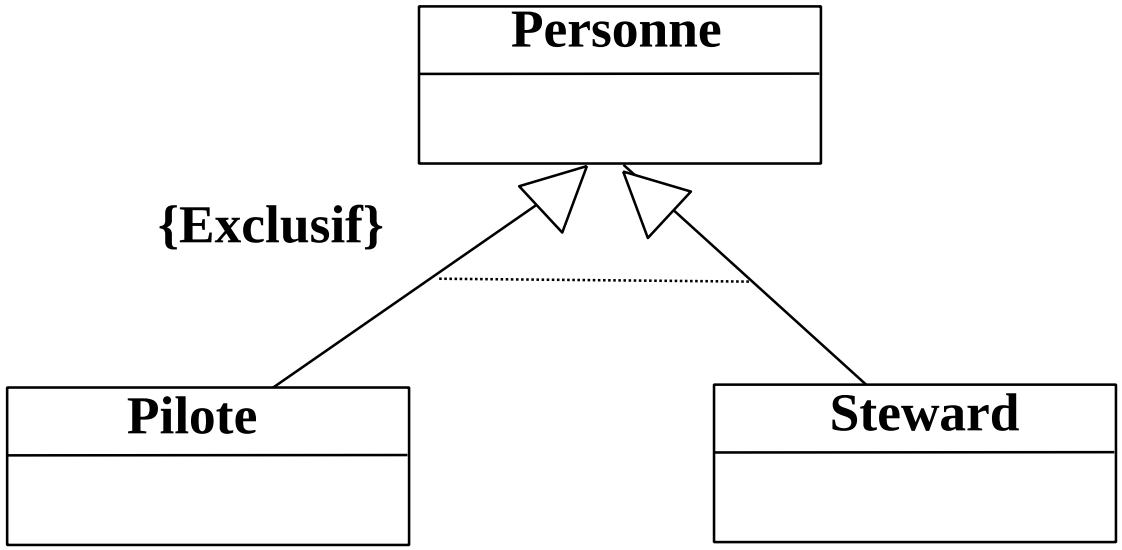
La généralisation

- Une classe peut être spécialisée selon plusieurs critères simultanément.



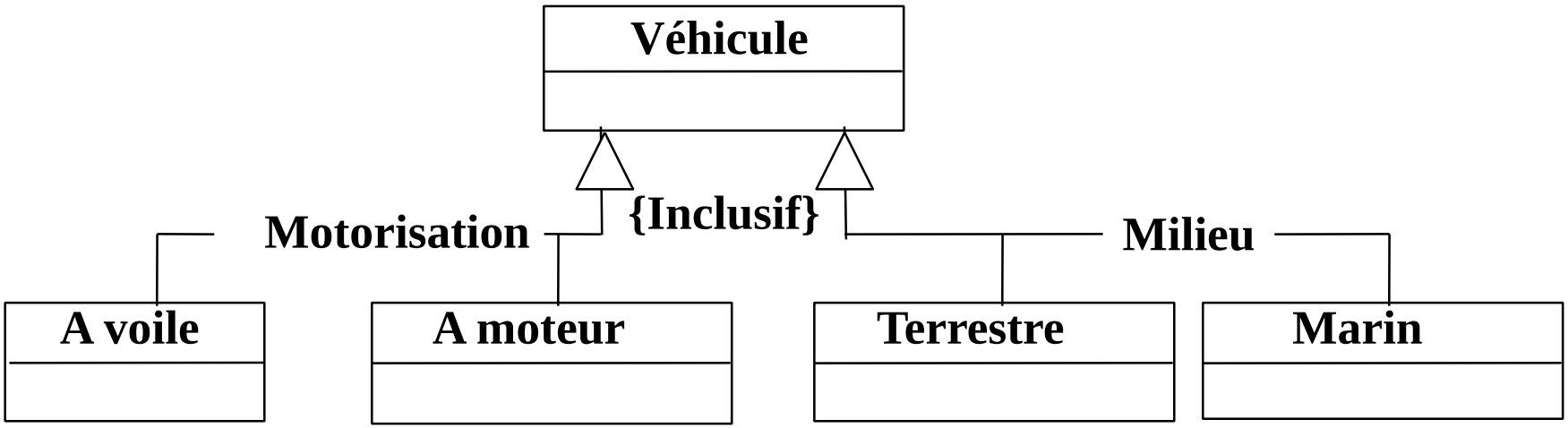
La généralisation, Contraintes

- La contrainte **{Disjoint}** ou **{Exclusif}** indique que les instances d 'une sous-classe ne peuvent pas être incluses dans une autre sous classe de la même classe



La généralisation, *Contraintes*

- La contrainte **{Chevauchement}** ou **{Inclusif}** indique que deux sous classes peuvent avoir des instances communes



La généralisation, Contraintes

- La contrainte **{Complète}** indique que la généralisation est terminée et qu'il n'est pas possible de rajouter des sous-classes. Inversement, la contrainte **{Incomplète}** désigne une généralisation extensible.

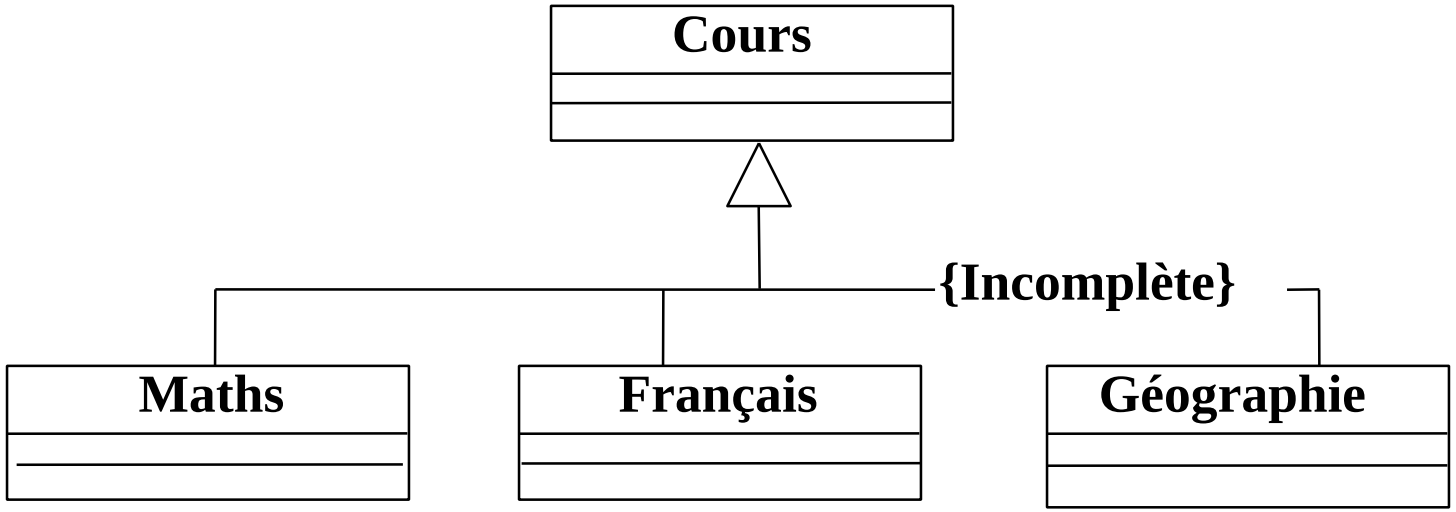
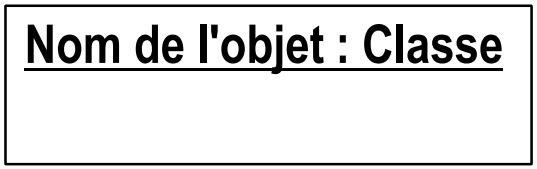
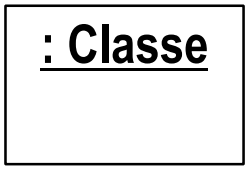


Diagramme d'objets

- Un diagramme sert à montrer un contexte. Ils facilitent la compréhension des structures de données complexes.





exemple

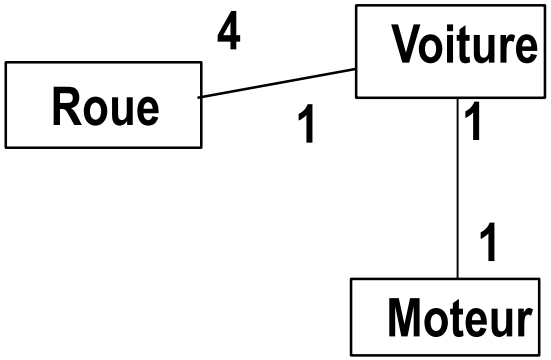


Diagramme de classes

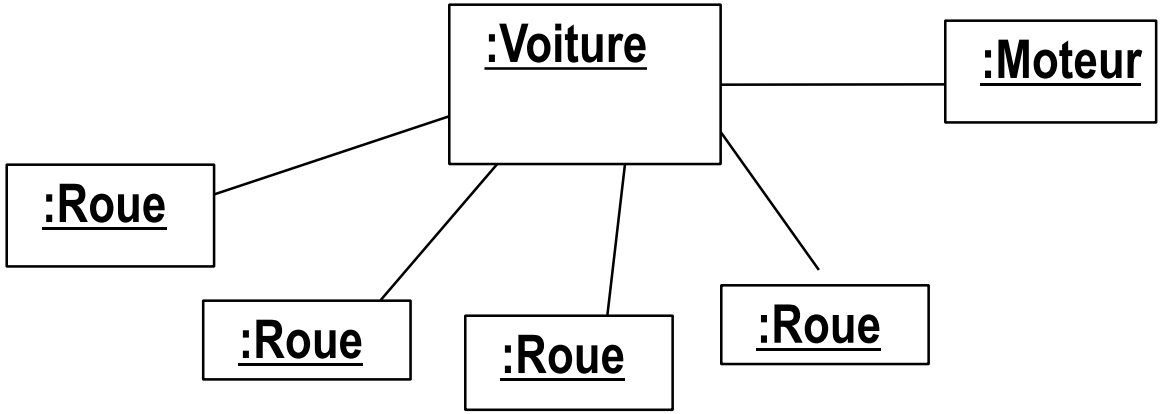
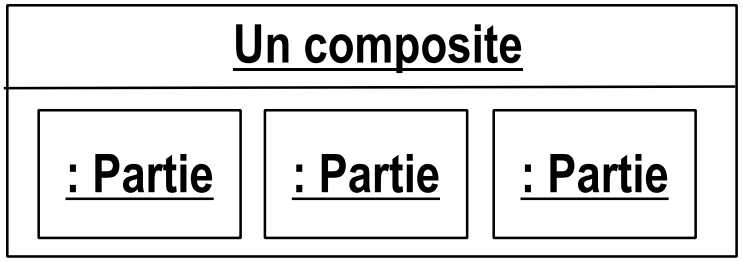


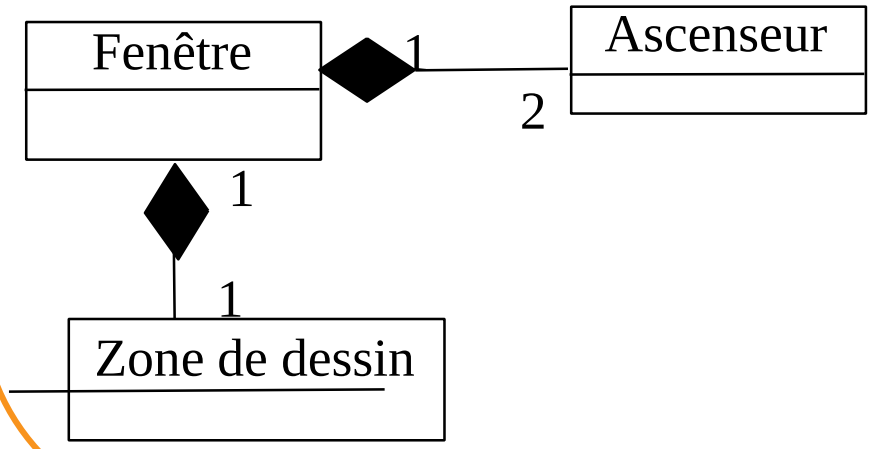
Diagramme d'objets

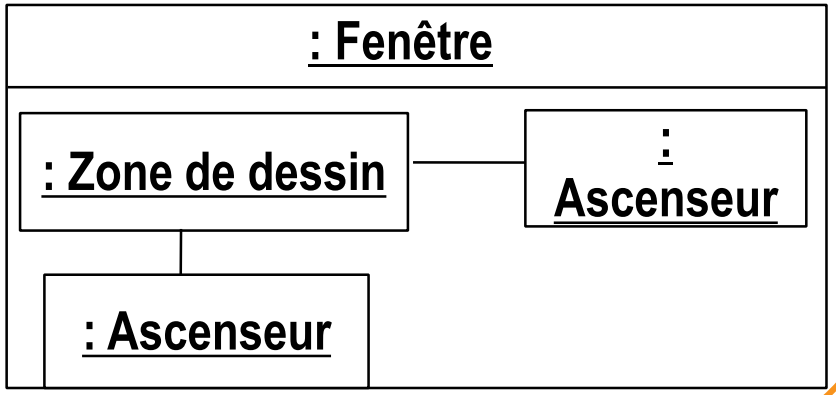
Objet composite

- Les objets composés de sous-objets peuvent être représentés au moyen d'un objet composite.



• Les objets composites sont instances de classes composites:

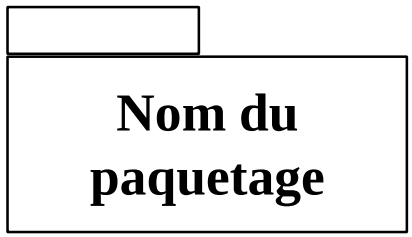




Paquetage

- Les systèmes impliquent la manipulation d'un nombre élevé de classes et autres éléments du modèle.
- En UML, un paquetage est un mécanisme générique pour organiser des éléments de modélisation en groupes
- Les paquetages permettent de présenter différentes vues de l 'architecture du système

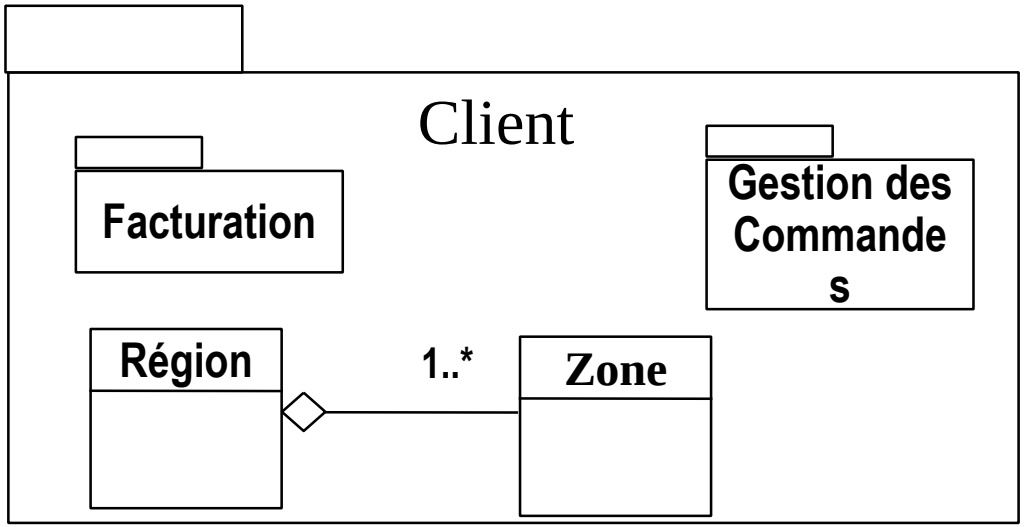
Représentation graphique



Paquetage

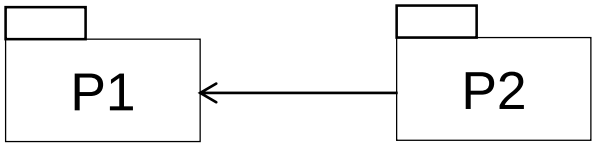
- Un paquetage peut contenir d'autres paquetages
- Chaque élément du modèle appartient à un paquetage. Le paquetage de plus haut niveau est le paquetage racine de l'ensemble d'un modèle
- Le système complet est une hiérarchie de paquetages

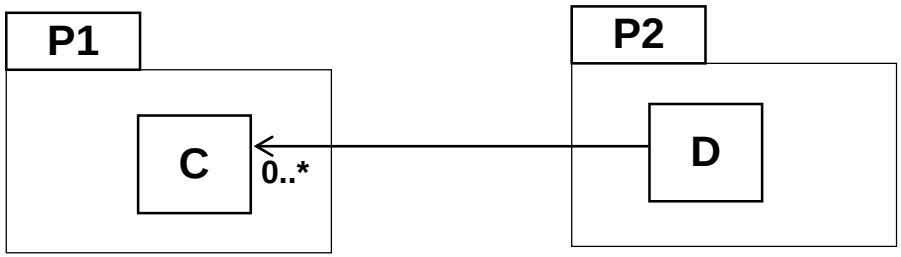
Exemple



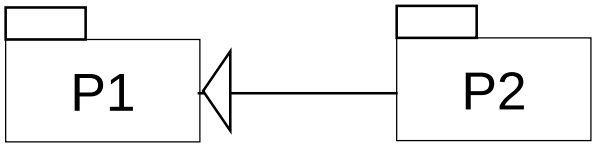
Paquetage: dépendances structurelles

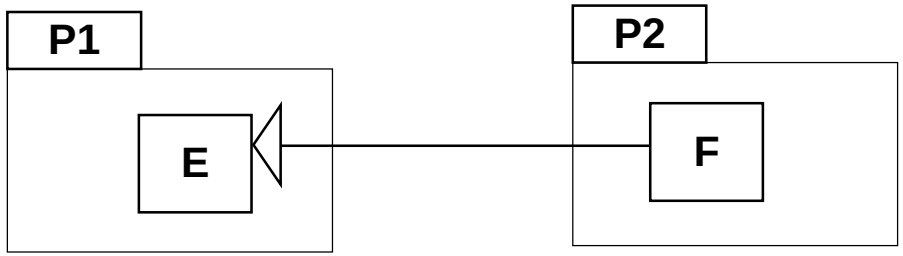
- Association



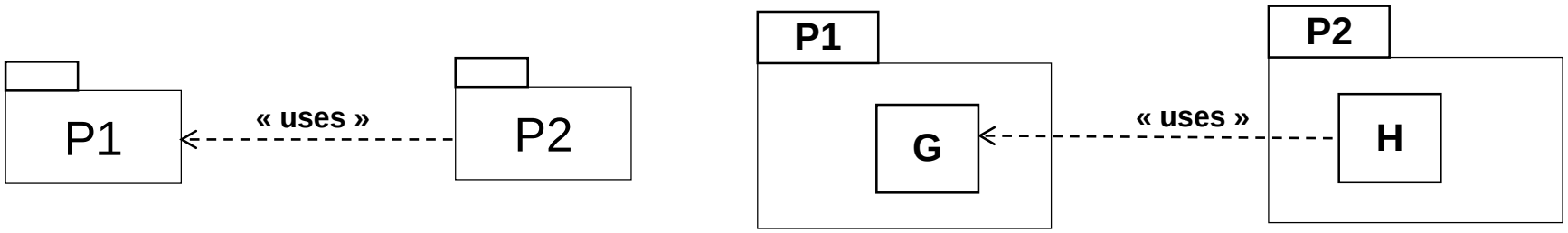


- Spécialisation





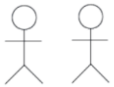
Paquetage: dépendances d'utilisation



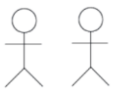
- H nécessite G pour sa mise en oeuvre ou son fonctionnement.
- La nature de l'utilisation peut être:
  - l'invocation d'une opération
  - la création d'un objet

Paquetage

- **Un modèle objet est constitué de un ou plusieurs paquetages**
  - Un nom de classe est unique dans un paquetage
  - Une classe peut appartenir à plusieurs paquetages: elle réalise le lien
  - vue comme une classe importée
  - Il est préférable qu'une association (ou une généralisation) n'existe que dans un seul paquetage
  - Les liens entre paquetages doivent être limités : couplage minimum

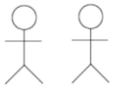


Chapitre 2
UML : Unified Modelling
EXERCICE



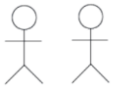
Le Modèle Dynamique

- **Décrit l'évolution au cours du temps du système**
  - Description de la vie de chaque objet dans le temps
    - **Changement d'états des objets**
      - **Modification des valeurs des attributs qui caractérise l'objet**
      - **Modification des liens entre objets**
- **Montre le flux de contrôle dans le temps**
  - Séquences d'opérations à exécuter en réponse à des événements extérieurs
- **Interactions dans le temps entre objets**
  - Envois de messages et réponses aux événements
  - Appels d'opérations



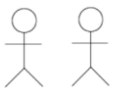
Le Modèle Dynamique : Les diagrammes

- **Le modèle dynamique repose sur un ensemble de diagrammes**
  - Diagrammes de séquence et/ou Diagramme de collaboration pour chaque scénario
    - Modéliser la collaboration de plusieurs objets dans un cas d'utilisation
  - Un diagramme d'états pour certaines classes
    - Modéliser le comportement d'un objet
    - Les diagrammes d'états sont reliés par des événements "partagés"
  - Diagrammes d 'activités pour la description des traitements ou les synoptiques de tâches (workflow)



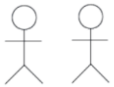
Scénario : Définition

- **Une séquence de messages qui correspond à une utilisation du système**
- **Les scénarios sont créés pour des cas d'utilisation du système**
- **Les scénarios montrent comment est utilisé le système**
  - ce qu'il va faire en réaction à ces messages
- **Eviter les alternatives (séquences conditionnelles) dans un scénario**
  - si besoin d'une alternative : faire plutôt deux scénarios



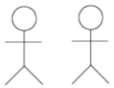
Exemples de scénarios

- **Appel téléphonique**
- **Scénario : le numéro appelé est occupé**
  - L'appelant décroche le téléphone
  - L'appelant commence à composer le numéro
  - L'appelant termine de composer le numéro
  - La tonalité "occupée" commence à sonner
  - L'appelant raccroche le téléphone
- **Scénario : le numéro appelé n'est pas occupé**
  - L'appelant décroche le téléphone
  - L'appelant commence à taper le numéro
  - L'appelant termine de taper le numéro
  - Le téléphone commence à sonner
  - L'appelé décroche
  - La conversation se déroule
  - L'appelé raccroche le téléphone



Le modèle dynamique

- La brique de base de la modélisation de la dynamique est l'échange (interaction) entre objets, les concepts principaux: l'événement et le message
- Une interaction est un ensemble de messages échangés au sein d'un groupe d 'objets pour atteindre un objectif
- Un message est la spécification d 'une communication entre objets qui transporte des informations et qui s'effectue pour déclencher une activité



Le modèle dynamique

Exemple

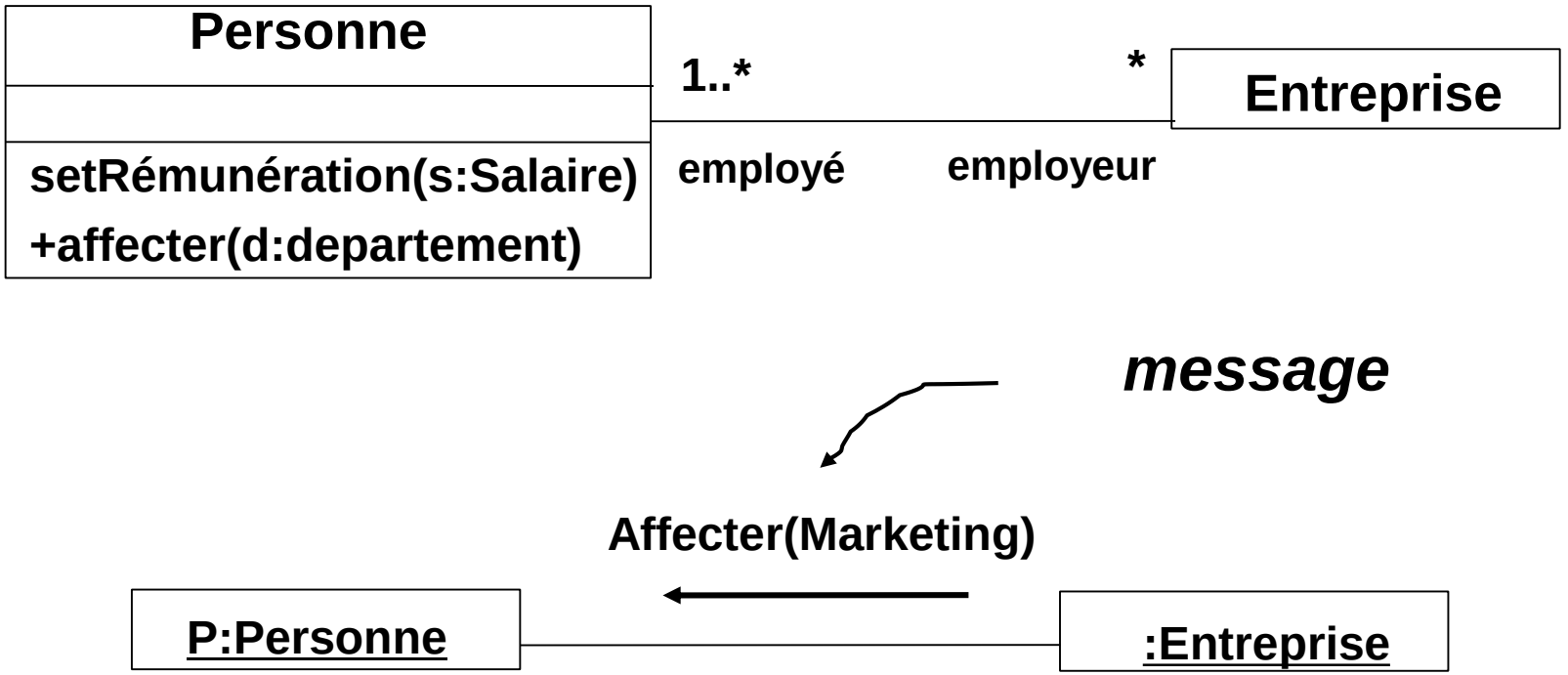


Diagramme de séquence

- **Représentation graphique du scénario montrant**
  - le séquencement des messages (opération, signal) entre objets dans le scénario
  - Modéliser la collaboration de plusieurs objets dans un cas d 'utilisation
- **Met en évidence les messages concurrents du scénario**
- **L 'envoi d 'un message est atomique**
  - insécable
  - la durée de l'envoi est considérée nulle comparée à la granularité de l 'interaction
- **Un objet est représenté par sa ligne de vie**
  - une ligne de vie peut être la vue abstraite d 'un ensemble d 'objets

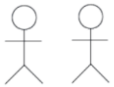


Diagramme de séquence: représentation

- Un système téléphonique

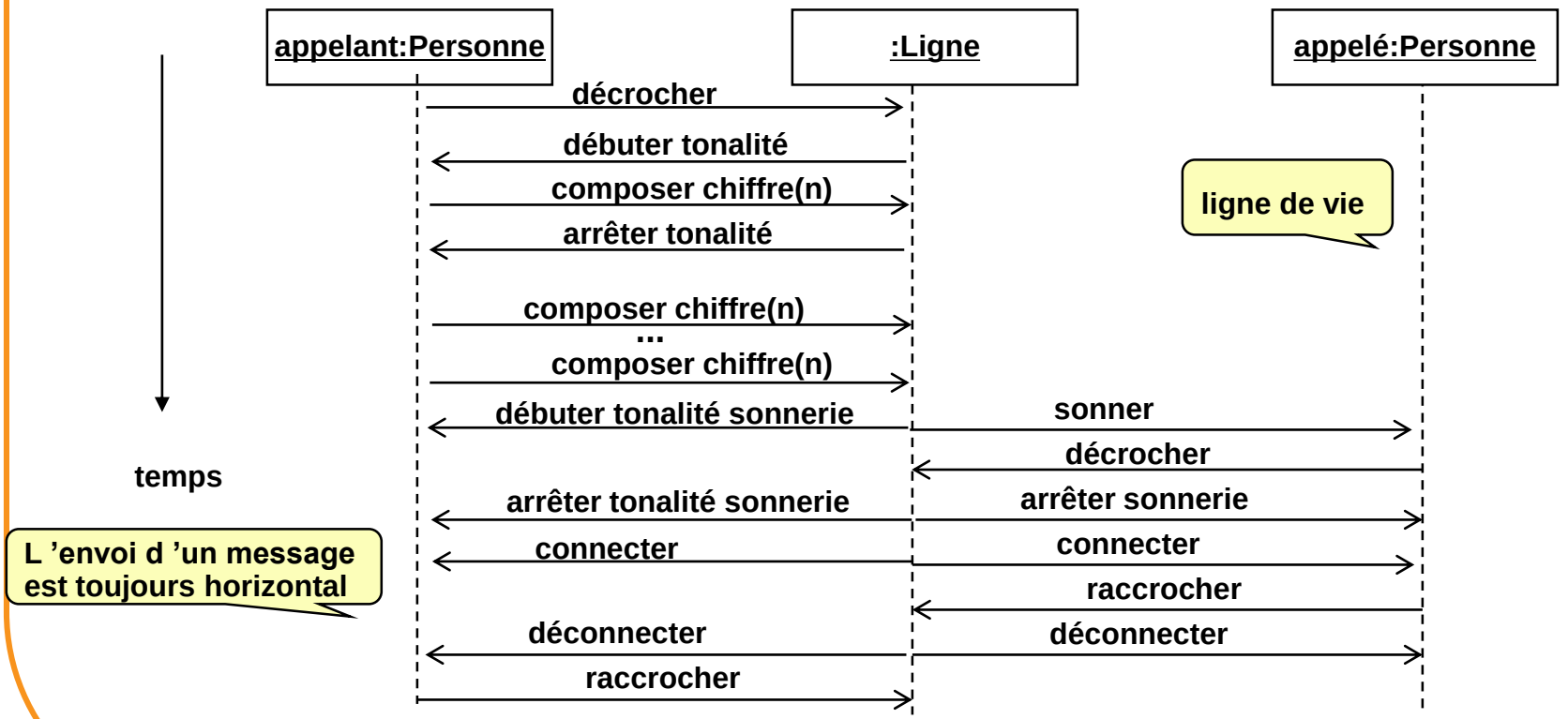


Diagramme de séquence

2 catégories d'envois de message :

- **Les envois synchrones pour lesquels l'émetteur est bloqué et attend que l'appelé ait fini de traiter le message.**
- **Les envois asynchrones pour lesquels l'émetteur n'est pas bloqué et peut continuer son exécution.**

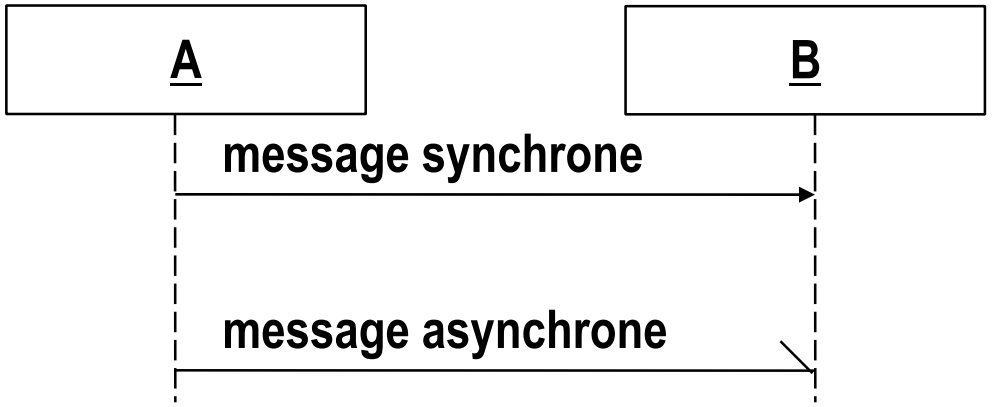


Diagramme de séquence

- La création des objets se représente en faisant pointer le message de création sur le rectangle qui symbolise l'objet créé.
- La destruction est indiquée par la fin de la ligne de vie et par une lettre X.

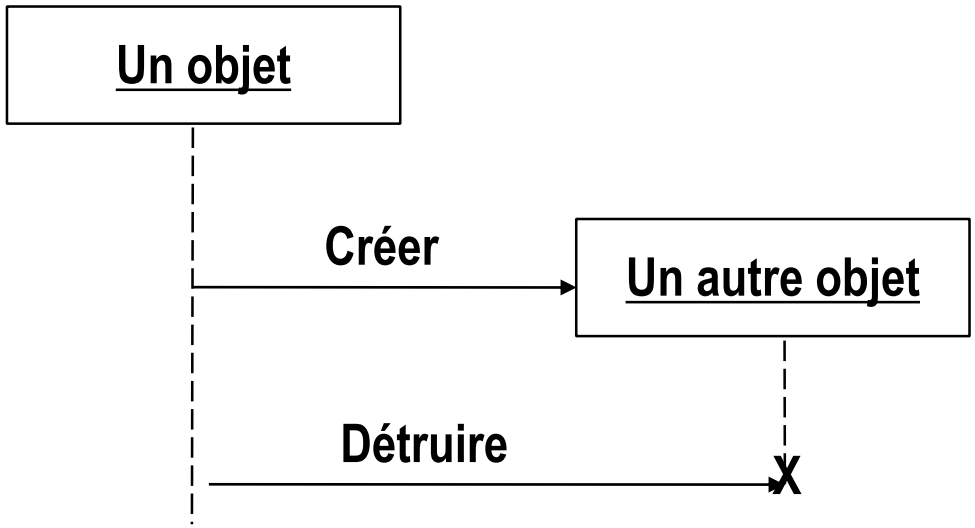


Diagramme de séquence

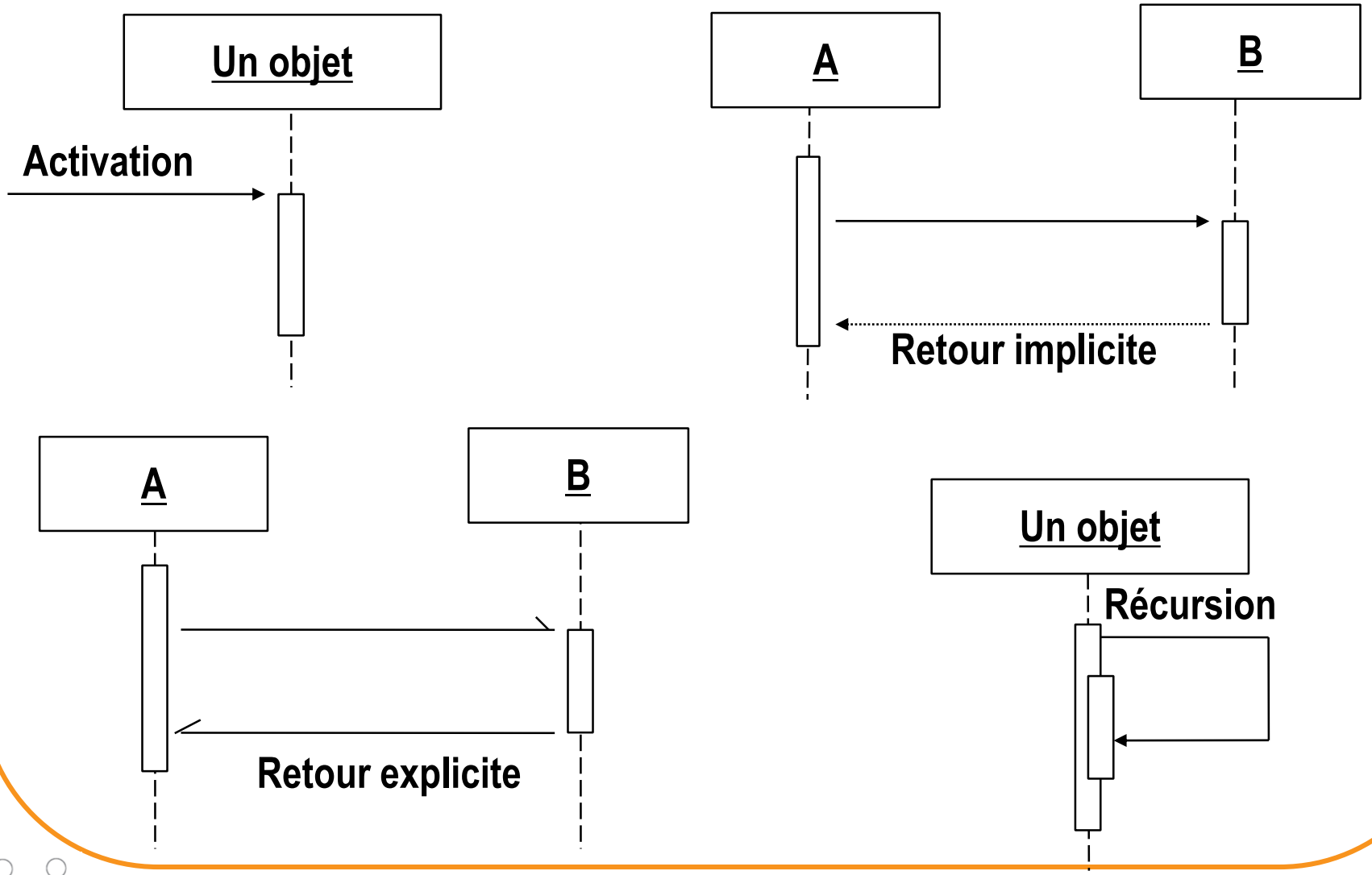
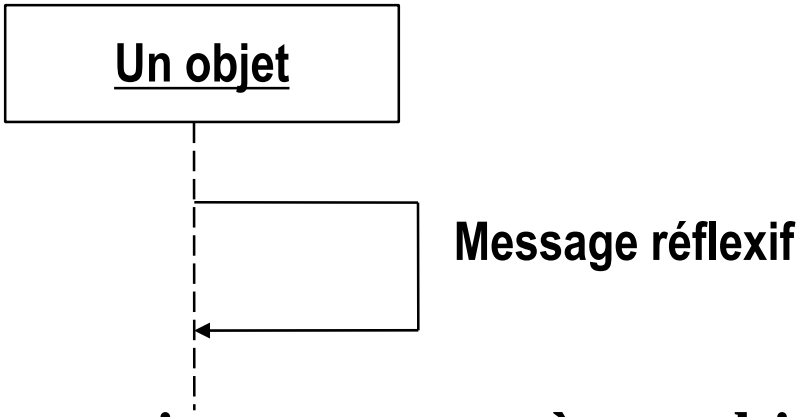


Diagramme de séquence

- **Un objet peut s'envoyer un message.**



- **Un objet composite envoie un message à son objet composant.**

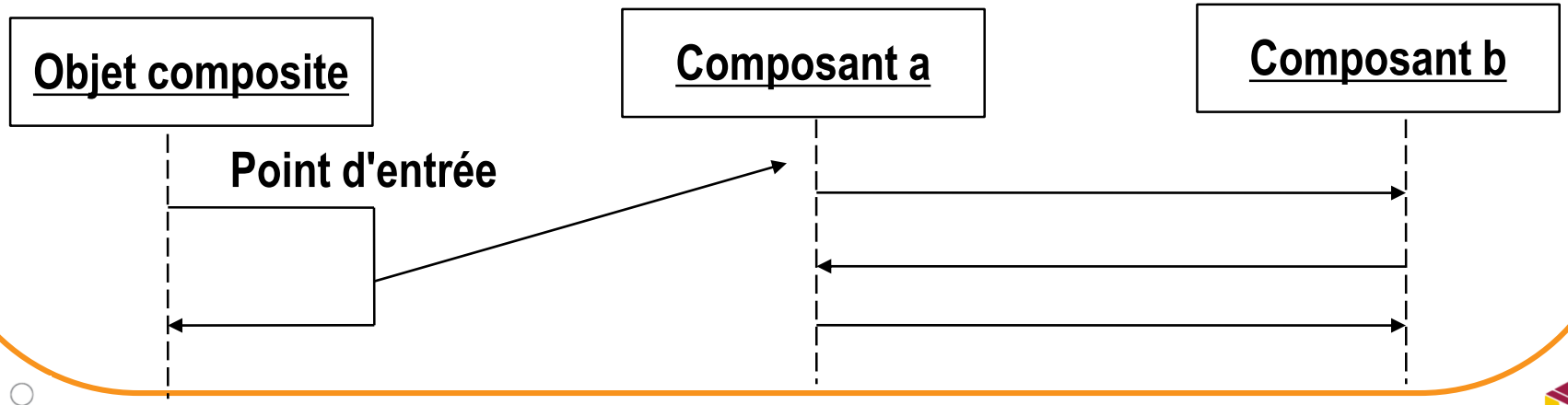
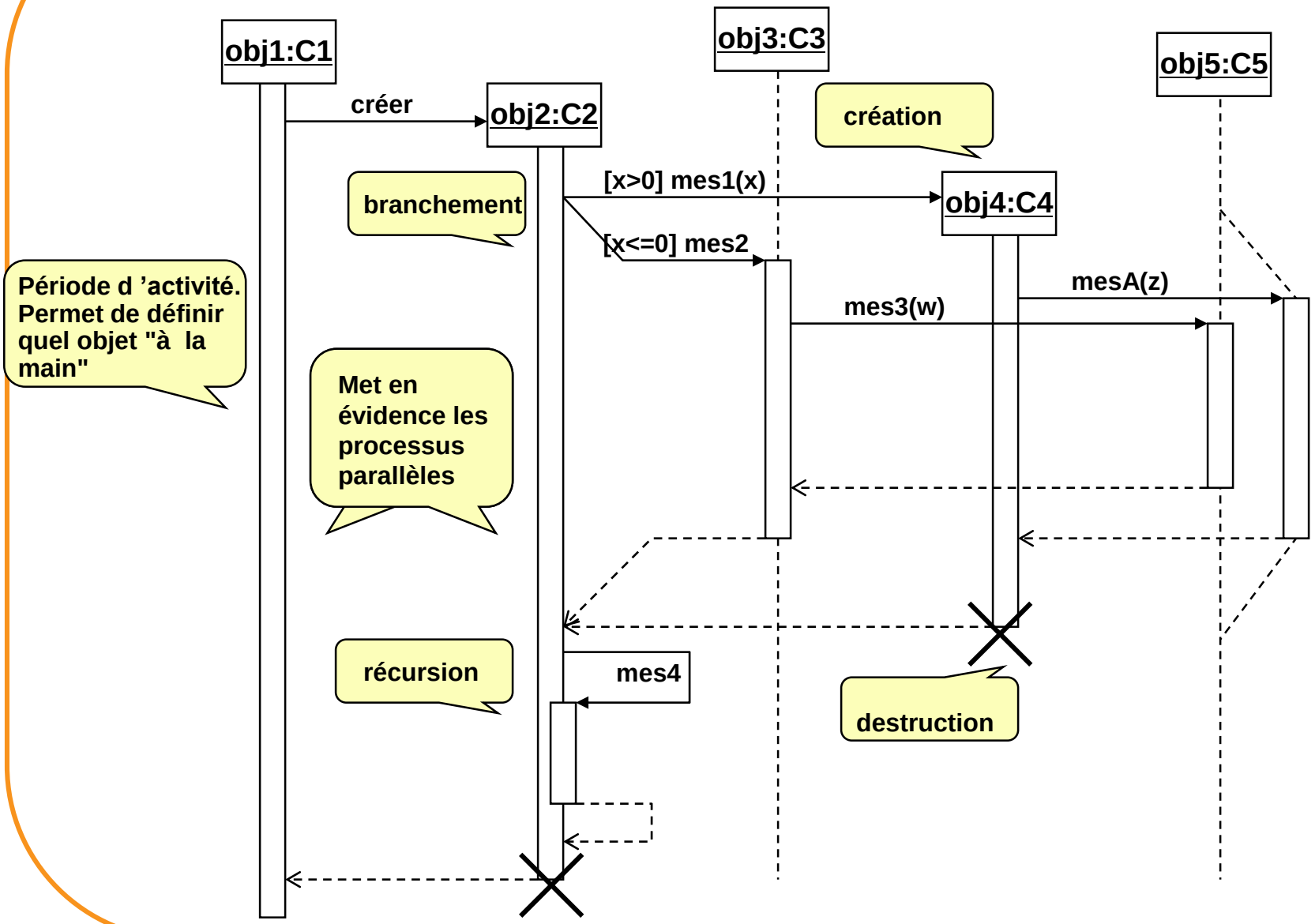


Diagramme de séquence



Chapitre 2
UML : Unified Modelling
EXERCICE

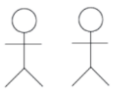


Diagramme de collaboration

- **Représente du point de vue structurel et dynamique les objets impliqués dans la mise en oeuvre "d'un but"**
  - Modèle expliquant la coopération entre les objets utilisés pour la réalisation d'une opération ou d'un cas d'utilisation
- **Contient**
  - Collaboration
    - **Contexte structurel des participants:**
      - **objets, classes, liens, associations, attributs**
  - Interaction
    - **Séquence de messages échangés entre les objets**
      - **Numérotés**
      - **Informations de contrôle du diagramme de séquence sont applicables**

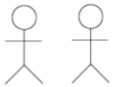


Diagramme de collaboration

- Met en évidence l ’organisation structurelle des objets qui participent à une interaction.
- Une interaction est réalisée par un groupe d'objets qui collaborent en échangeant des messages pour le réalisation d ’une fonctionnalité.

exemple

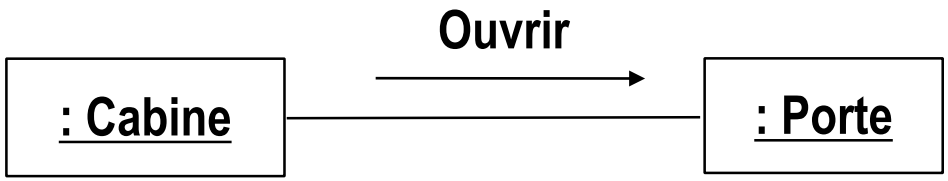
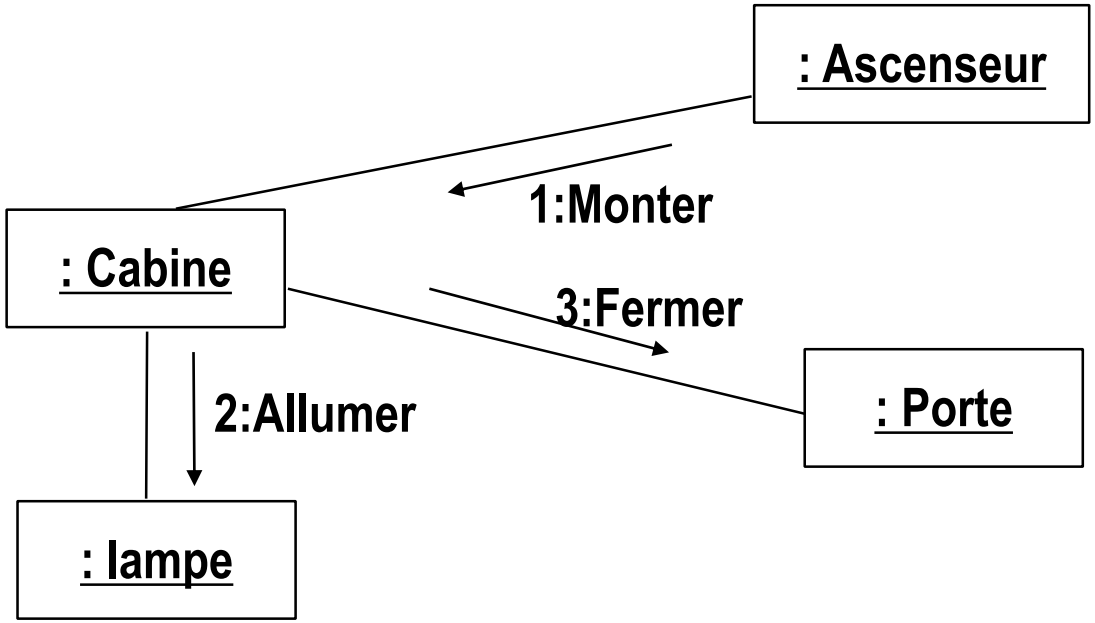


Diagramme de collaboration

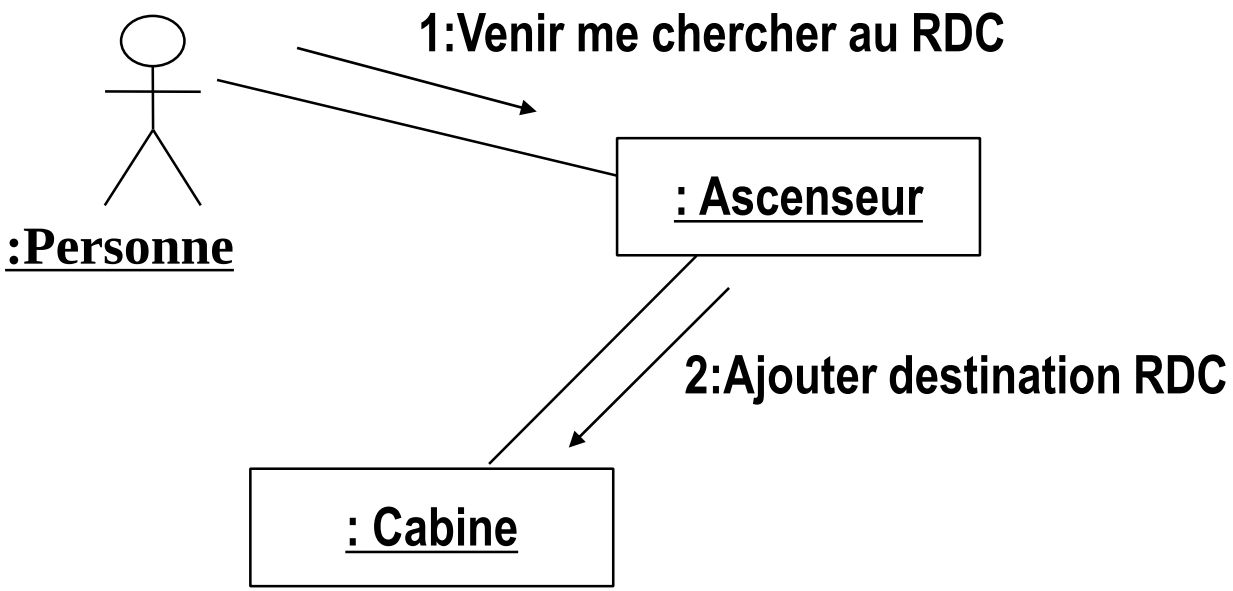
Le temps n'est pas représenté de manière implicite : les messages sont numérotés pour indiquer l'ordre des envois.



Les interactions : séquence de messages entre objets

Diagramme de collaboration

- La notation permet de faire figurer un acteur dans un diagramme de collaboration afin de représenter le déclenchement des interactions par un élément externe au système.



Synthèse des messages

- **Consolide l'ensemble des messages entre objets**
  - Ne montre pas le séquencement
  - Intègre tous les scénarios
  - Synthèse des dépendances dynamiques entre objets.

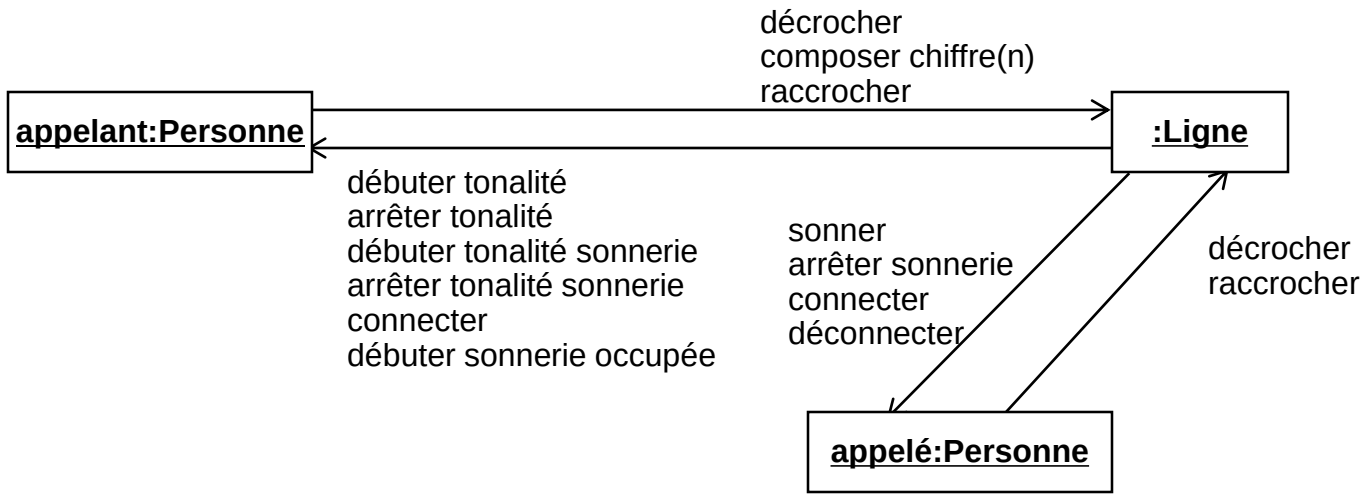
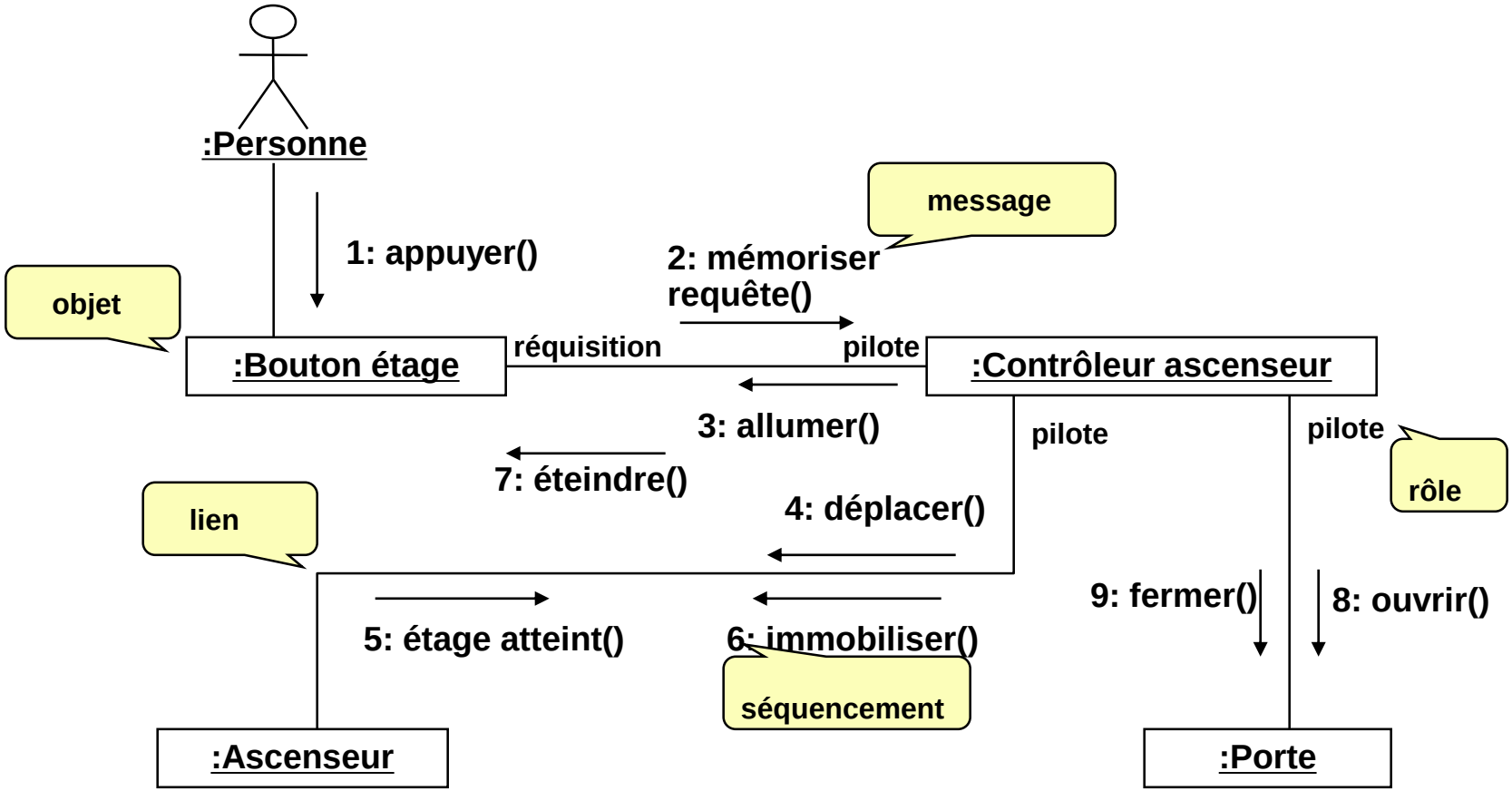


Diagramme de collaboration

Collaboration en réponse à l'appui sur le bouton d'appel d 'un ascenseur

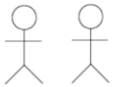


Représentation des messages dans un diagramme de collaboration

- **Un message se représente par une flèche placée à proximité d'un lien et dirigée vers l'objet destinataire du message. Un lien sert de support de transmission pour le message.**

- **Les messages respectent la forme générale suivante:**

synchronisation séquence ':' résultat ':=' nom arguments



Représentation des messages dans un diagramme de collaboration

- **Les arguments et le nom du message identifient de manière unique l'action qui doit être déclenchée dans l'objet destinataire.**
- **Les expressions suivantes donnent quelques exemples de la syntaxe d'envoi des messages:**
  - 4 : Afficher (X, Y) -- *message simple*
  - 3.3.1 : Afficher (X, Y) -- *message imbriqué*
  - 4.2 : âge := Soustraire (Aujourd'hui, DateDeNaissance)
  - [Age>= 18 ans] 6.2 : Voter () -- *message conditionnel*
  - 4.a, b.6 / c.1 : Allumer (Lampe) -- *synchronisation*
  - 1 * : Laver () -- *itération*
  - 3.a, 3.b / 4*||[i:= 1..n] : Éteindre () -- *itération parallèle*

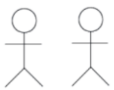
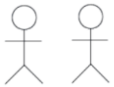


Diagramme de collaboration vs Diagramme de séquence

- **Séquence**
  - accent sur le séquencement : ordre évident des messages
  - Simplicité
    - **Mais si ajout de conditions/alternatives/boucles perd sa lisibilité**
- **Collaboration**
  - accent sur la structure : indique comment les objets sont liés
- **A propos des comportements conditionnels**
  - Intégrer les conditions dans le même diagramme ? Ou
  - Un diagramme séparé pour chaque condition (chaque scénarios) ?
  - Les diagrammes de séquence sont optimums quand le comportement est simple



Automates à états finis

- un automate d'états finis permet de modéliser le comportement d ’un objet individuel, il définit les différents états par lesquels un objet passe au cours de son existence en réponse à des événements.

- Le comportement des objets d'une classe peut être décrit de manière formelle en termes d'états et d'événements, au moyen d'un automate relié à la classe considérée.

- 2 types d ’automates à états finis
  - les diagrammes d ’états-transitions
  - les diagrammes d ’activités

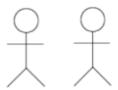
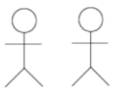


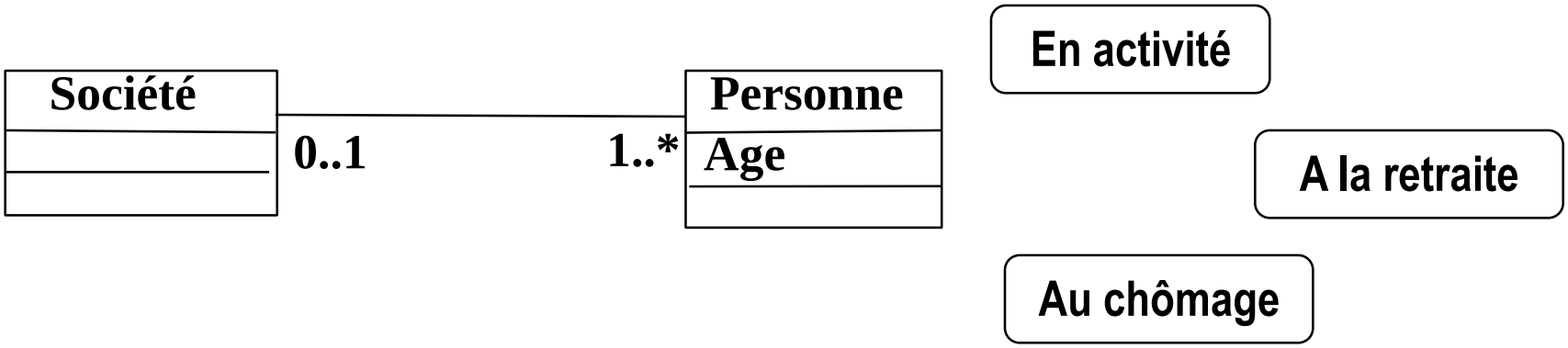
Diagramme d'Etats

- **Un diagramme d'états est associé à une classe**
  - Il est valable pour tous les objets de cette classe
    - **toutes les instances se comporteront de la même façon**
- **Il contient tous les scénarios possibles dans lesquels la classe est considérée**
- **Un scénario correspond à un chemin dans un diagramme d'état**



État

- Chaque objet est à un moment donné dans un état particulier. Chaque état possède un nom qui l'identifie.
- Les états se caractérisent par la notion de durée et de stabilité.
- Un état est toujours l'image de la conjonction instantanée des valeurs contenues par les attributs de l'objet, et de la présence ou non de liens, de l'objet considéré vers d'autres objets.



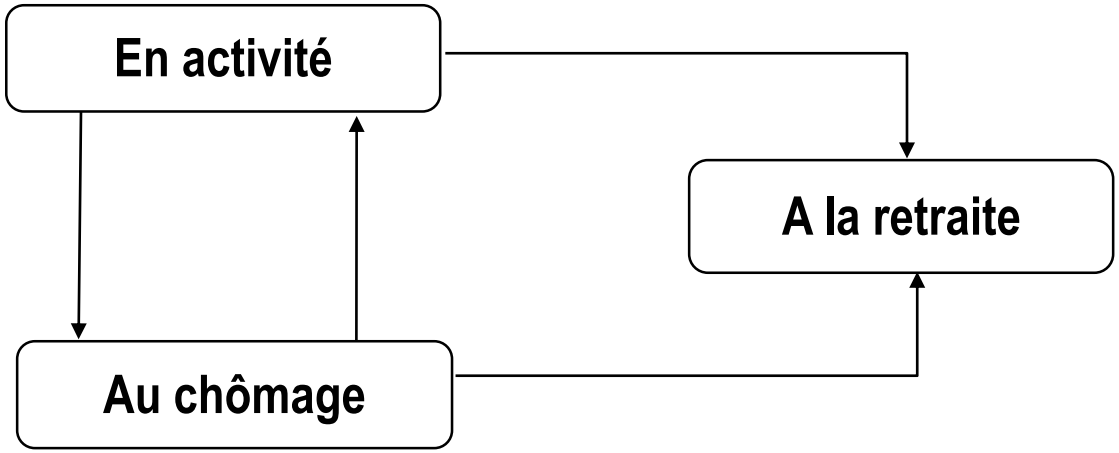
État

- Les automates retenus par UML sont déterministes. Il faut toujours décrire l'état initial du système. Pour un niveau hiérarchique donné, il y a toujours un et un seul état initial. En revanche, il est possible d'avoir plusieurs états finaux qui correspondent chacun à une condition de fin différente.

Transition

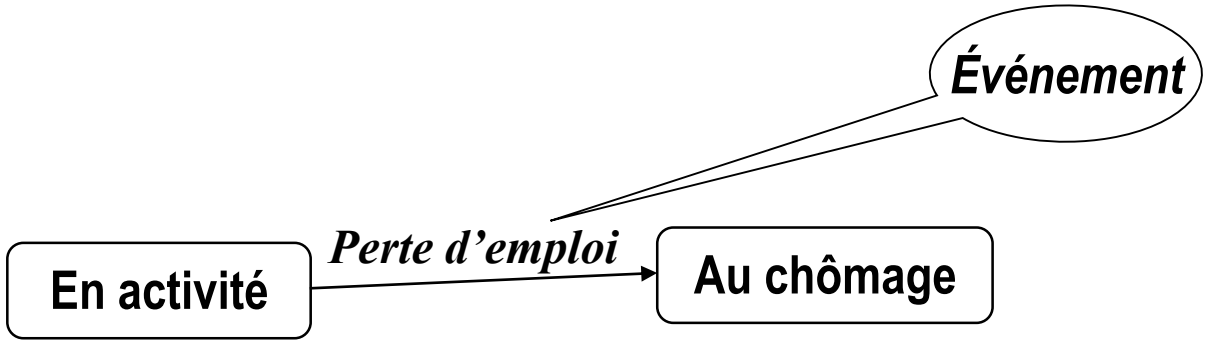
- Les états sont reliés par des connexions unidirectionnelles, appelées transition.
- Les diagrammes d'états-transitions sont des graphes dirigés.

Exemple : Transitions entre états de la classe Personne



Événement

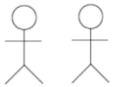
- Un événement correspond à l'occurrence d'une situation donnée dans le domaine du problème
- Un événement spécifie que quelque chose, de significatif, s'est passé, c'est une information qui doit être traitée sans délai
- Un objet, placé dans un état donné, attend l'occurrence d'un événement pour passer dans un autre état.



Un événement déclenche la transition qui lui est associée.

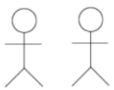
Evénement

- Représente un moyen pour transmettre de l'information
- Se produit à un instant donné
- N'a pas de durée (par rapport à la durée de l'état)
  - Peut être :
    - Événement signal causé par la réception d’un signal
    - Événement appel causé par la réception d’un appel d’opération
    - Événement temporel causé par l’expiration d’une temporisation
    - Événement modification exprimée par une expression booléenne



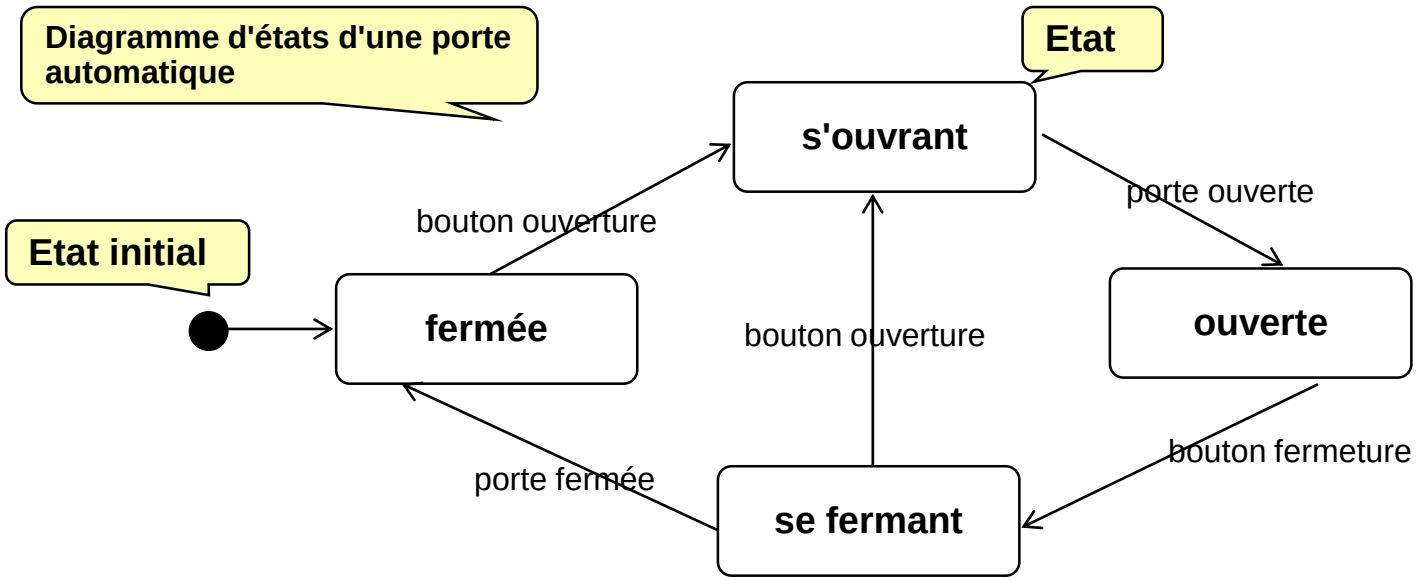
Evénement : Exemples

- **Événement signal**
  - « clic de souris », « entrée clavier »…
- **Événement appel**
  - « crée », « détruit », …
- **Événement temporel**
  - Quand (date = 1/1/2005) , Après (3 secondes) ...
- **Événement modification**
  - Quand suivi d'une condition booléenne



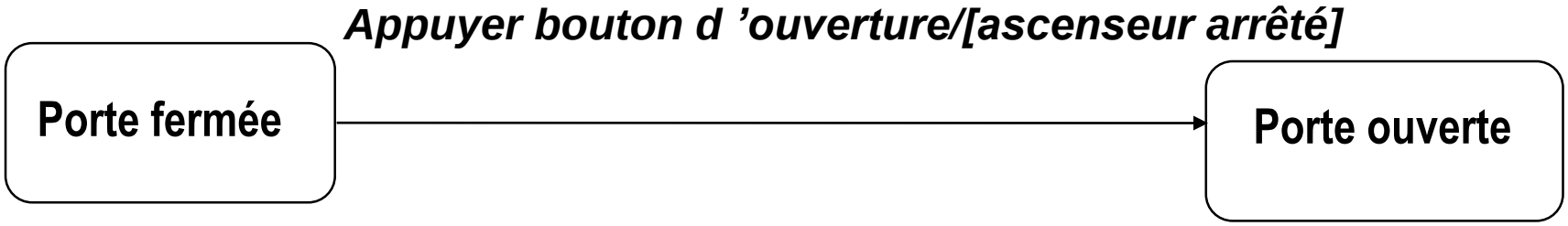
Etat d'un objet vs Evènement

- **Contexte pendant lequel l'objet est susceptible de traiter des événements reçus**
  - Un état est caractérisé par des valeurs d'attributs et/ou de liens
  - Un objet répond ou non à un événement en fonction de son état



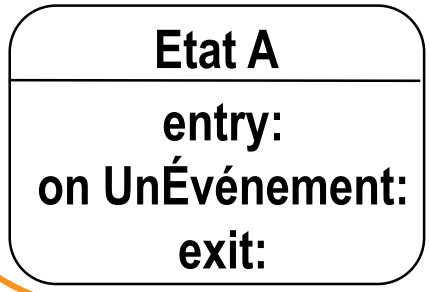
Garde

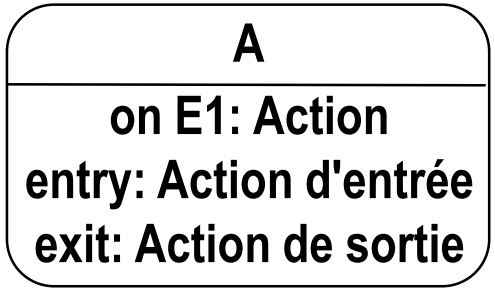
• Une garde est une condition booléenne qui valide ou non le déclenchement d'une transition lors de l'occurrence d'un événement.

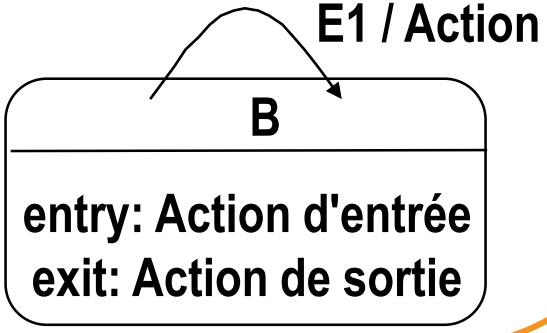


Opération, action, activité

- **Les états peuvent également contenir des actions; elles sont exécutées à l'entrée ou à la sortie de l'état ou lors de l'occurrence d'un événement pendant que l'objet est dans l'état.**
  - L'action d'entrée (entry:) est exécutée de manière instantanée et atomique dès l'entrée dans l'état.
  - L'action de sortie (exit:) est exécutée à la sortie de l'état.
  - L'action sur événement interne (on:) est exécutée lors de l'occurrence d'un événement qui ne conduit pas à un autre état.
- **Les actions correspondent à des opérations dont le temps d'exécution est négligeable. Une opération qui prend du temps correspond à un état plutôt qu'à une action.**

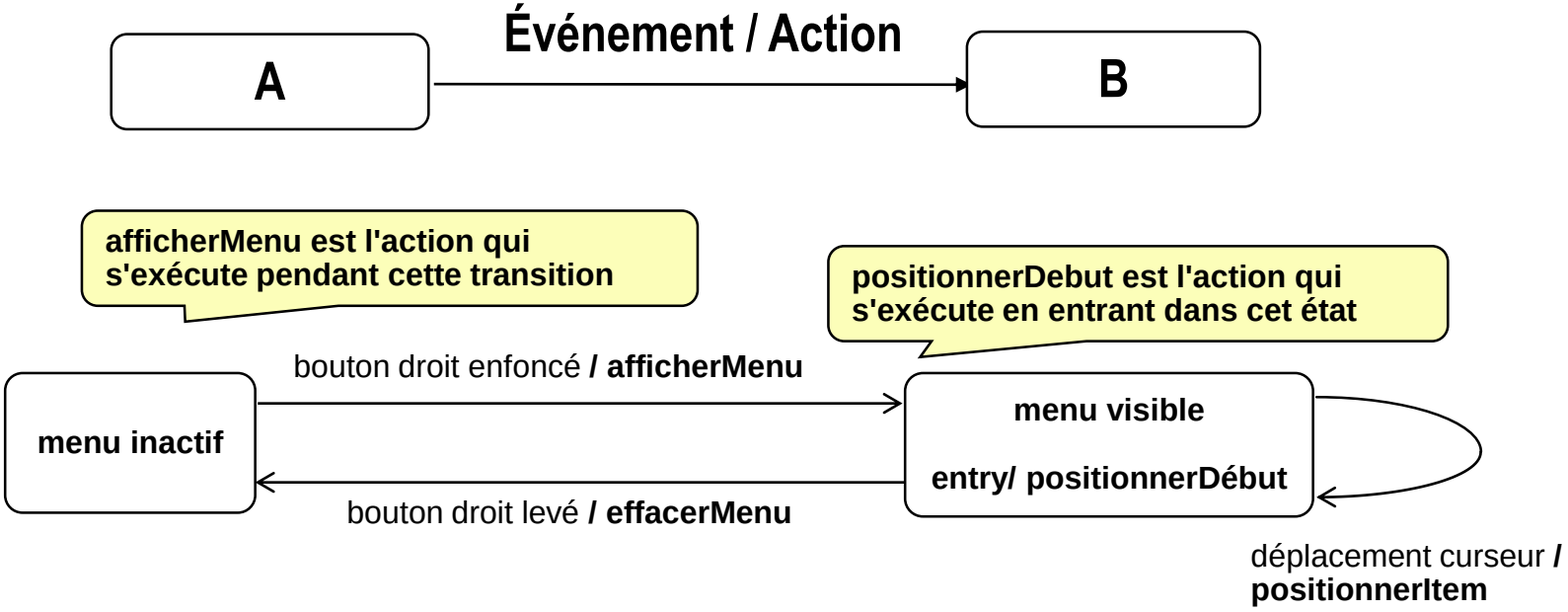






Action

- **Traitement instantané et non interruptible**
  - sa durée n'est pas significative
- **Associée à un événement et exécutée lors de la transition**
- **Associée à un état et exécutée en entrant ou en sortant**



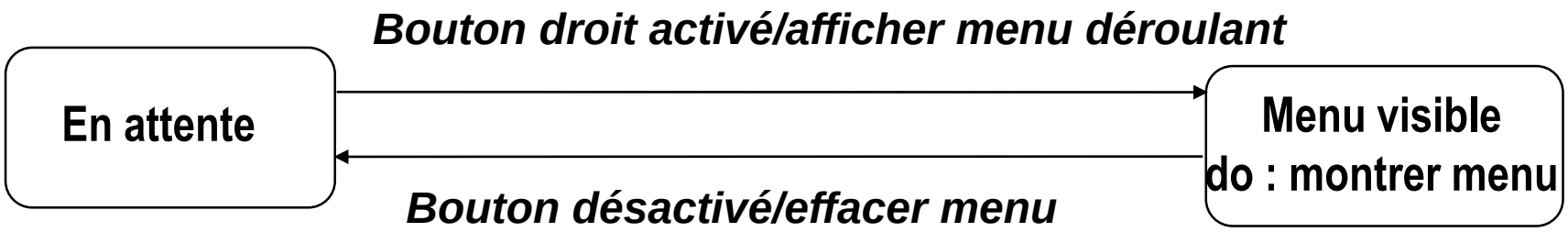
- *L'action correspond à une des opérations déclarées dans la classe de l'objet destinataire de l'événement. L'action a accès aux paramètres de l'événement, ainsi qu'aux attributs de l'objet.*

Opération, action, activité

- Le mot-clé do: indique une activité; opération qui prend un temps non négligeable et qui est exécutée pendant que l'objet est dans un état donné.

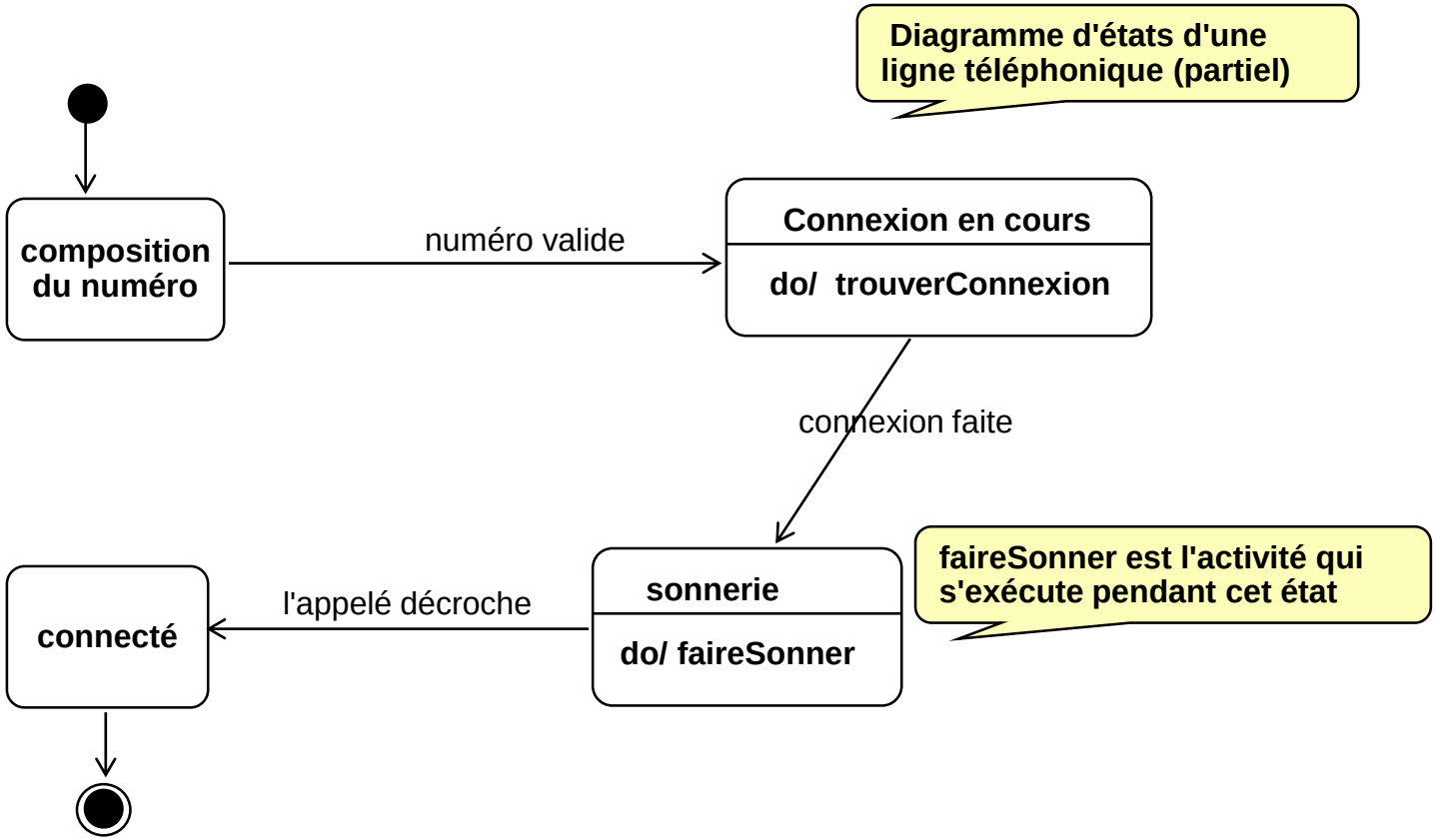
- Contrairement aux actions, les activités peuvent être interrompues à tout moment, dès qu'une transition de sortie de l'état est déclenchée.

Diagramme d'états d'un Menu Fantôme (popup)



Activité

- Traitement de durée finie ou non
- Interruptible par un événement
- Exécutée durant un état



Points d'exécution des opérations

- **6 points pour spécifier les opérations, dans l'ordre d'exécution:**
  - l'action associée à la transition d'entrée (Op1);
  - l'action d'entrée de l'état (Op2);
  - l'activité dans l'état (Op3);
  - l'action de sortie de l'état (Op4);
  - l'action associée aux événements internes (Op5);
  - l'action associée à la transition de sortie de l'état (Op6).

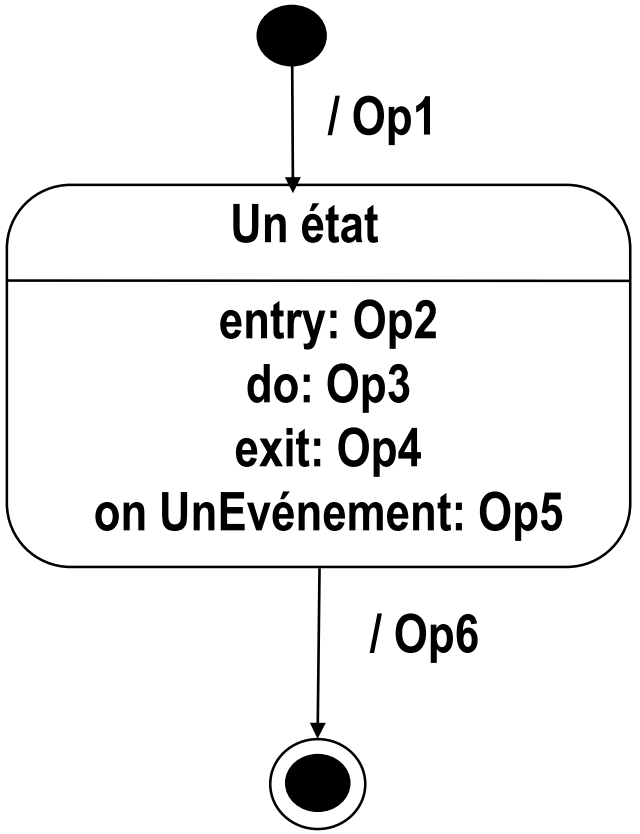
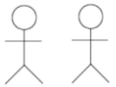


Diagramme d'états structuré

- **Structurer les diagrammes d'états**
  - Eviter l'explosion combinatoire des transitions
  - Décomposer le diagramme en sous-diagrammes
  - regrouper des états au comportement commun, dans un super-état



Etats et sous-états

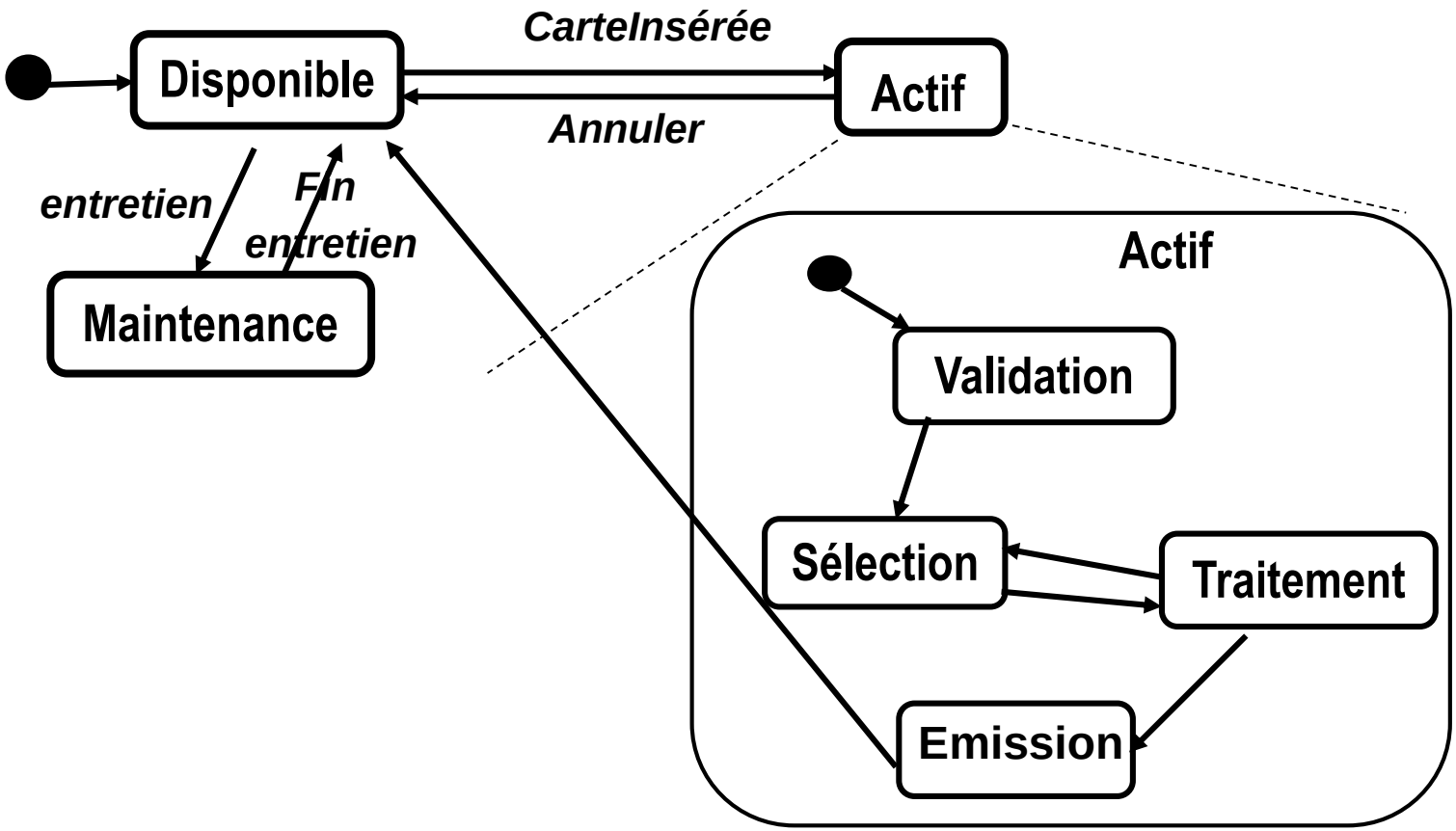
Un sous-état est un état emboîté dans un autre état, on dit que c ’est un état composé

Un état composé peut contenir soit des sous-états séquentiels (disjoints), soit des sous-états concurrents

La représentation graphique est identique à celle de l’état

Sous-états séquentiels

exemple 1 : *modélisation du comportement d'un DAB*



Sous-états concurrents

exemple : ***modélisation du comportement d'un DAB***

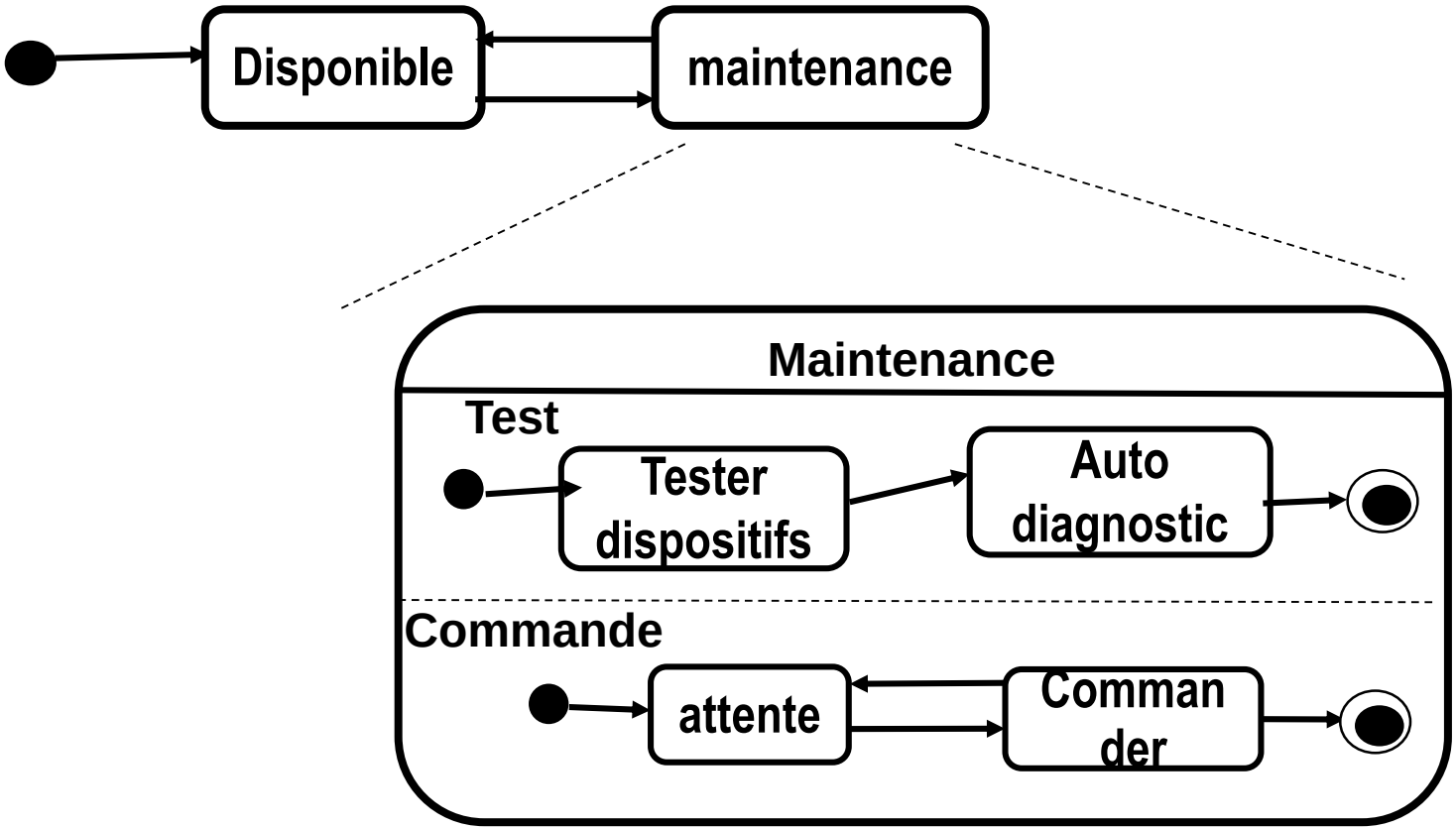


Diagramme d'états-transitions

Un diagramme d ’états-transitions permet de modéliser les aspects dynamiques d’un système à l’aide d’un automate à états finis

On utilise les diagrammes d’états-transition pour la modélisation du comportement des objets réactifs

Un objet réactif est un objet dont le comportement se caractérise surtout par sa réponse aux événements envoyés de l ’extérieur de son contexte

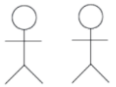


Diagramme d'activités

- **Un diagramme d ’activités est un organigramme qui montre le flot de contrôle d ’une activité à une autre, il modélise les étapes séquentielles et/ou concurrentes dans un processus de calcul, il contient:**
  - **des états d’activité et des états d’action,**
  - **des transitions**
  - **des objets**

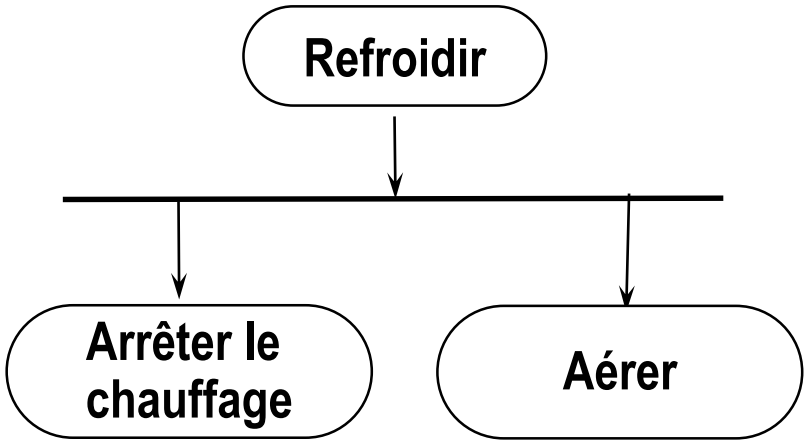


Diagramme d'activités

- Chaque activité représente une étape particulière dans l'exécution de la méthode englobante.

- Les activités sont reliées par des transitions automatiques. Lorsqu'une activité se termine, la transition est déclenchée et l'activité suivante démarre.

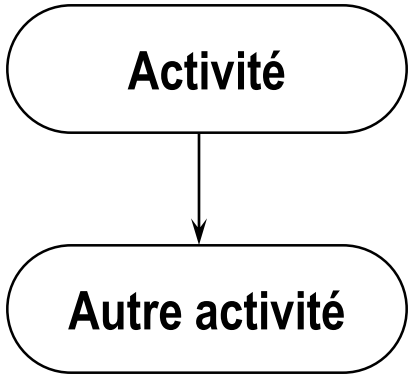


Diagramme d'activités

- Les transitions entre activités peuvent être gardées par des conditions booléennes, mutuellement exclusives. Les gardes sont à proximité des transitions dont elles valident le déclenchement.

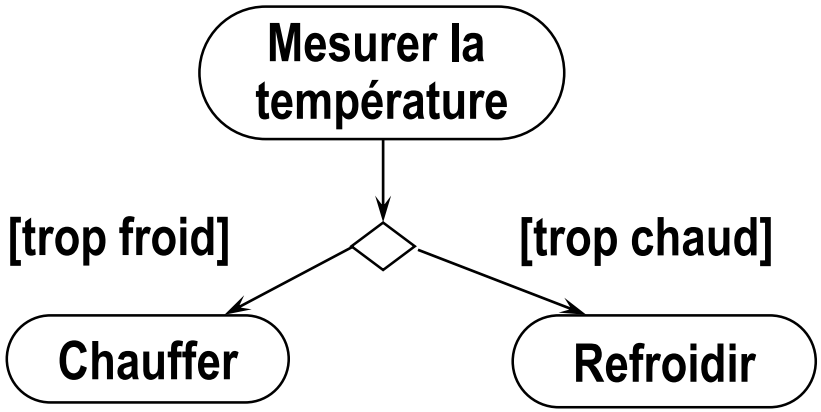


Diagramme d'activités

- Les diagrammes d'activités peuvent être découpés en couloirs d'activités (travées) pour montrer les différentes responsabilités au sein d'un mécanisme ou d'une organisation. Chaque responsabilité est assurée par un ou plusieurs objets et chaque activité est allouée à un couloir donné.

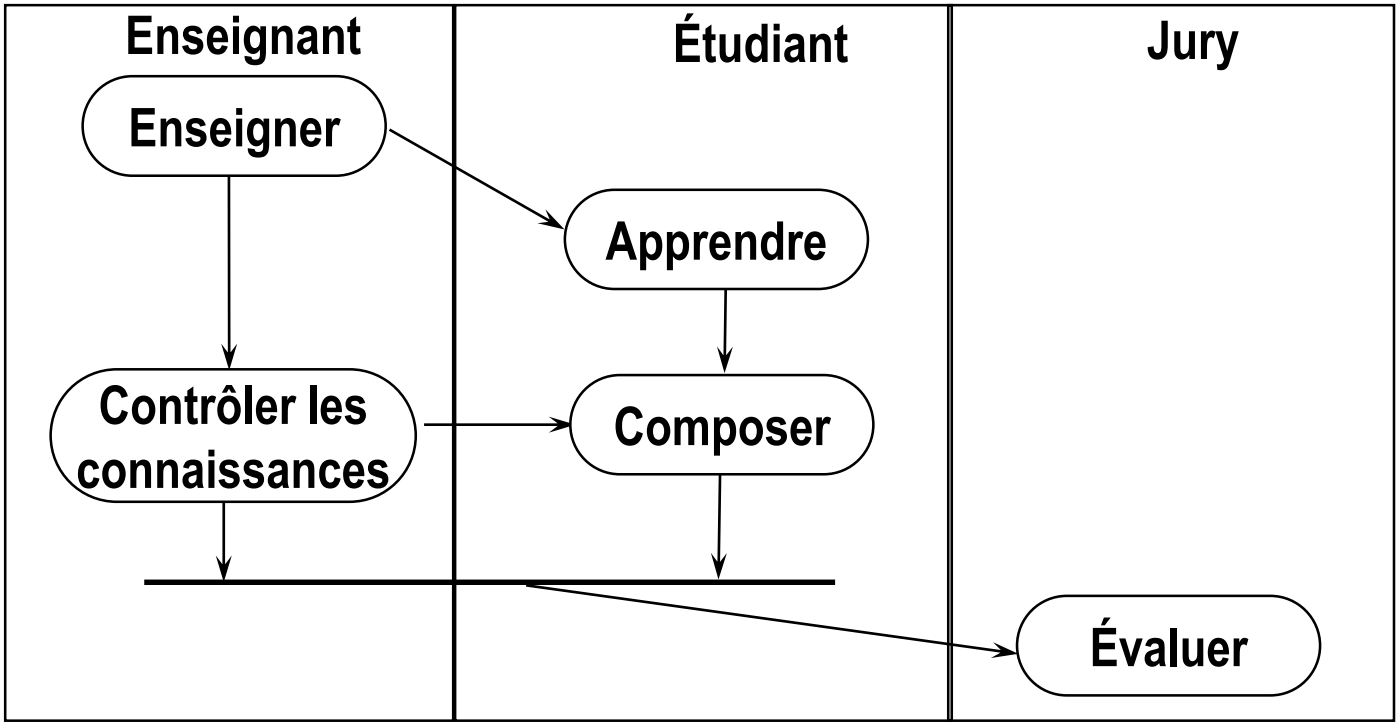
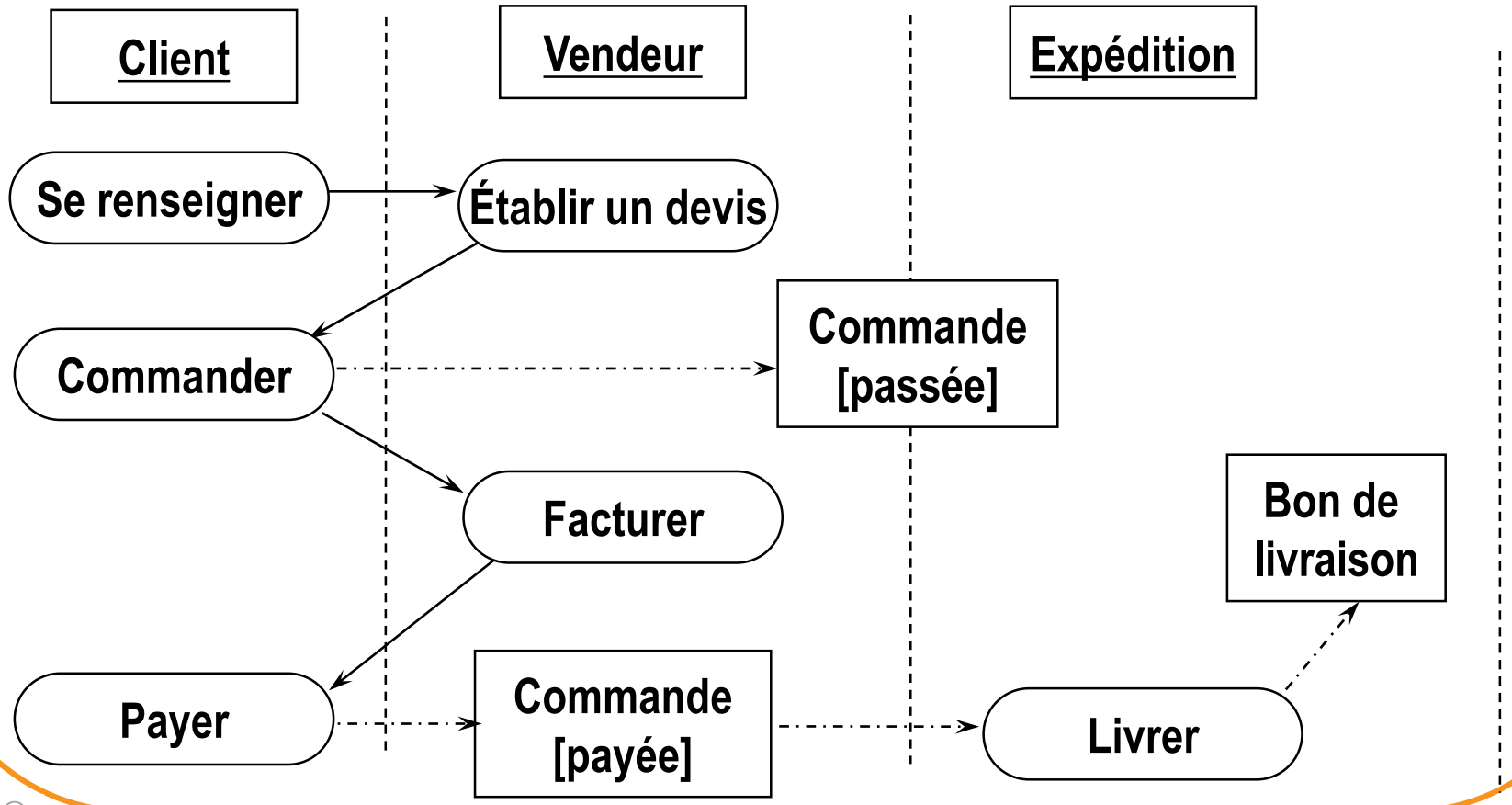


Diagramme d'activités

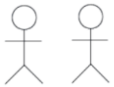
- Il est possible de faire apparaître clairement les objets dans un diagramme d'activités, soit au sein des couloirs d'activités, soit indépendamment de ces couloirs.



Les modèles de structure

- **Montre la structure statique d'un modèle**
  (e.g., classes, interfaces, composants, noeuds)

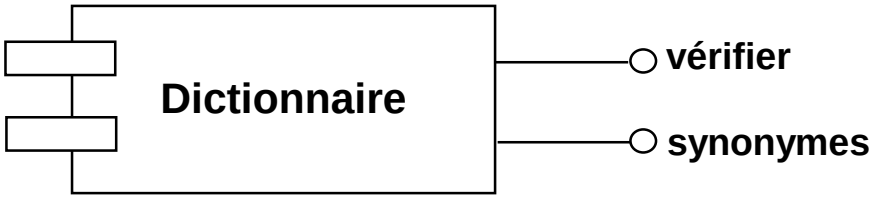
- **Types de diagrammes**
  - diagrammes structurels statiques
    - Diagramme de classes
    - Diagramme d'objets
  - Diagrammes d'implantation
    - Diagramme de composant
    - Diagramme de déploiement



Composant : définition

- **Un composant est une partie physique du système qui réalise un ensemble d'interfaces**

- Un composant avec son interface



- **Un composant représente tout élément tangible entrant dans la fabrication d'une application :**
  - code source, binaire relogeable, exécutable;
  - tout document produit ;
  - *un module est un type de composant*
- **La définition des composants est issue de la décomposition en paquetages**

Composant

3 types de composants:

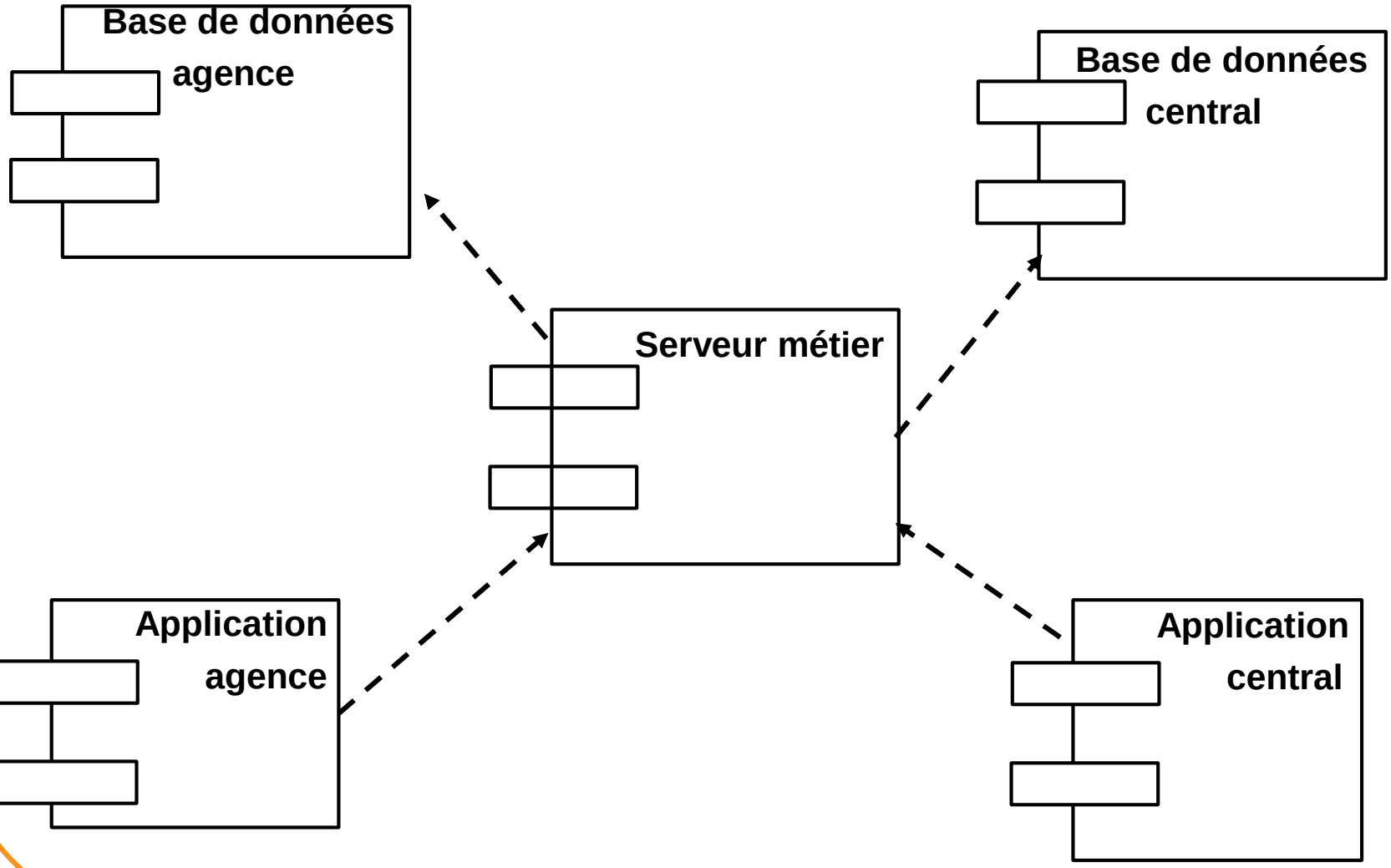
- **Les composants de déploiement comme les bibliothèques dynamiques (DLL) et les exécutables (EXE)**

- **les composants produits par le développement : ils sont le résultat du processus de développement ; ce sont les fichiers code source et les fichiers de données à partir desquels les composants de déploiement sont créés**

- **les composants d ’exécution : ils sont crées en tant que produit d ’une exécution comme par exemple un objet COM instancié à partir d’une DLL**

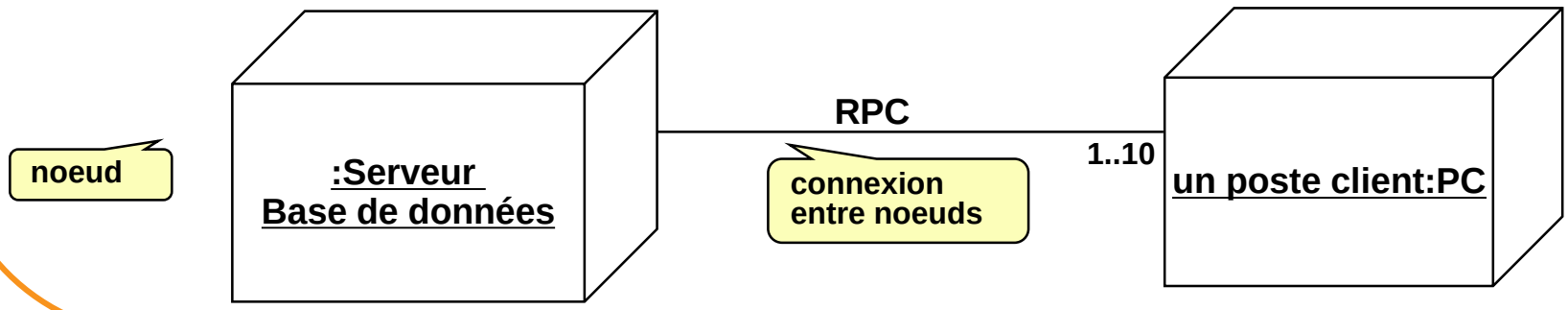
Diagramme de composants

Cas Sivex



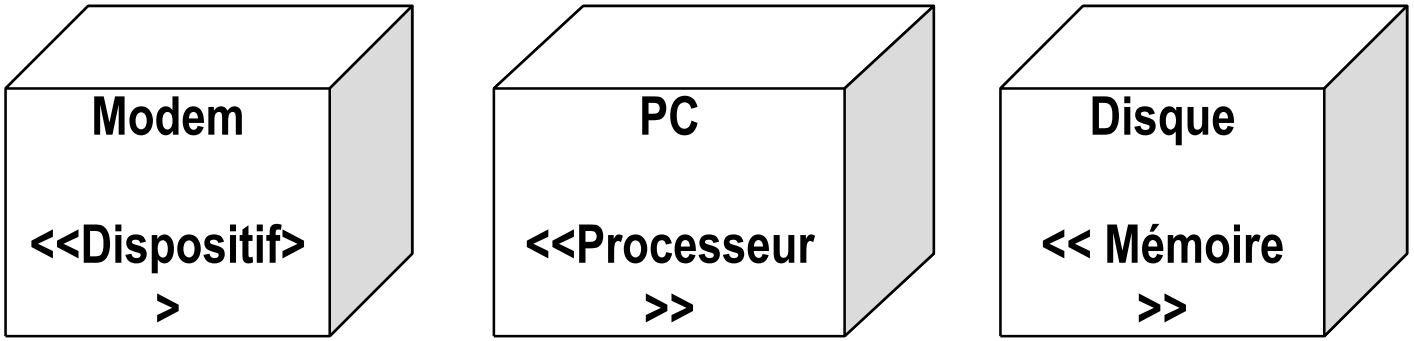
Les diagrammes de déploiement

- **Montrer**
  - la disposition physique des différents matériels entrant dans la composition d'un système informatique
    - un matériel est représenté par un noeud
  - la répartition des programmes exécutables sur ces matériels
    - décrit comment les composants et objets sont disposés dans un système distribué
- **Tout système est décrit par un petit nombre de diagrammes de déploiement; souvent un seul diagramme suffit**



Les diagrammes de déploiement

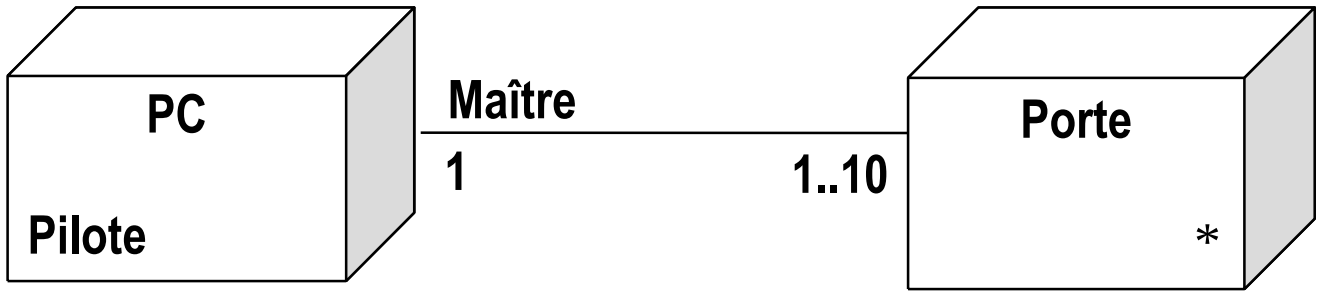
- La nature de l'équipement peut être précisée au moyen d'un stéréotype.



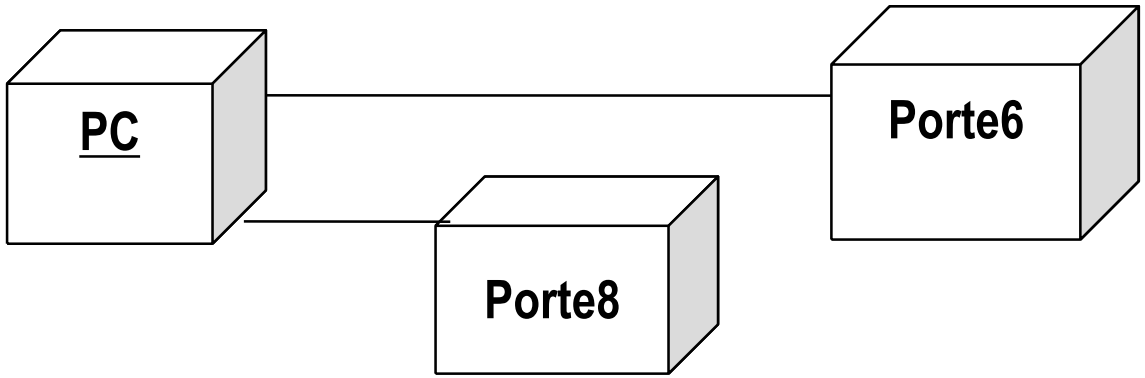
- Les nœuds sont connectés entre eux par des lignes qui symbolisent un support de communication

Les diagrammes de déploiement

- Les diagrammes de déploiement peuvent montrer des classes de nœuds ou des instances de nœuds.



- Les diagrammes de déploiement peuvent également montrer des instances de nœuds.

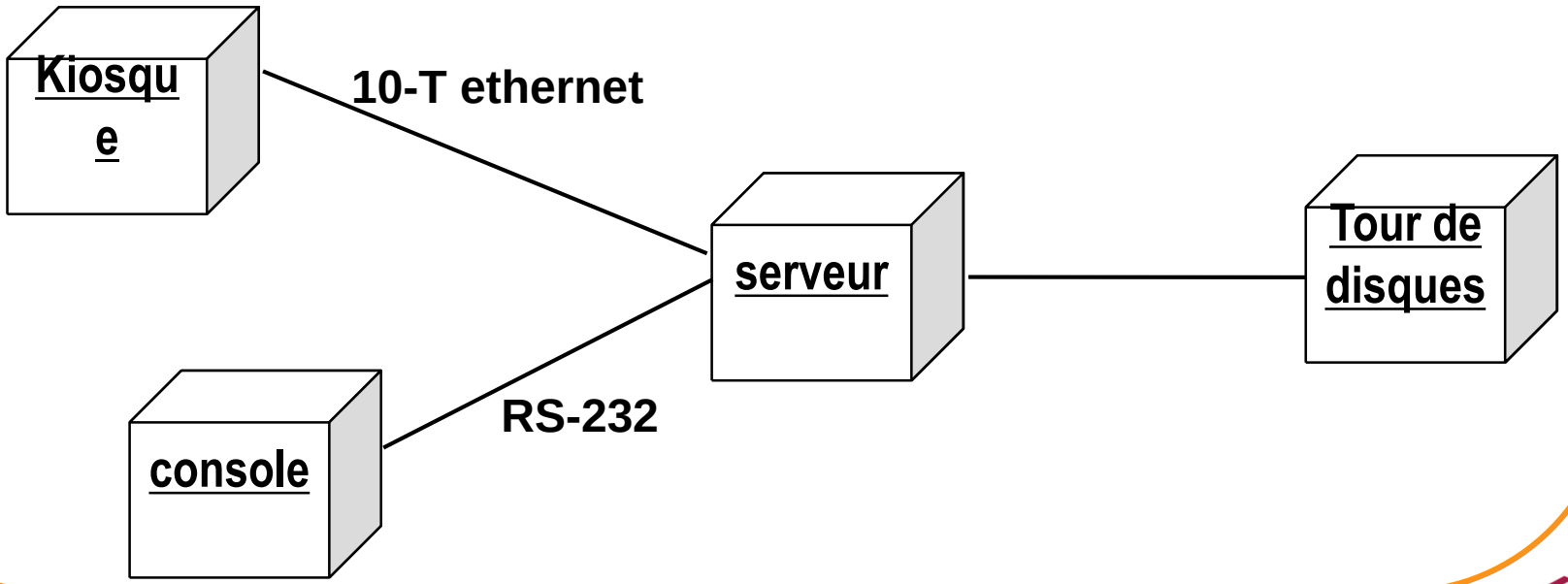


Les diagrammes de déploiement

Organisation des noeuds

Des relations de type dépendance, généralisation ou association sont utilisés pour organiser les nœuds.

La relation la plus utilisée est l 'association



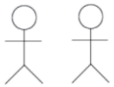
Les diagrammes de déploiement

Nœuds et Composants

- Les composants sont des éléments qui participent à l'exécution d 'un système alors que les nœuds sont des éléments qui exécutent les composants

- Les composants représentent l'emballage physique d 'éléments logiques alors que les nœuds représentent le déploiement physique des composants

Un composant est la matérialisation d'un ensemble d'éléments logiques tels que les classes et les collaborations et un nœud est l'emplacement sur lequel les composants sont déployés



Diagrammes d'UML

- Diagrammes de cas d'utilisation
- Diagramme séquence système

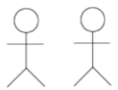
Vue utilisation
modèle d'usage

Vue statique
modèle objet

Vue dynamique
modèle dynamique

Diagrammes de classes
Diagrammes d'objets
Diagrammes de paquetages
Diagrammes de composants
Diagrammes de déploiement

Diagrammes d'interaction :
séquences ou collaborations
Diagrammes d'états
Diagrammes d'activités



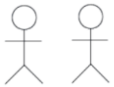
Les Diagrammes d'UML

Découverte des besoins :

- *Diagramme de cas d'utilisation:*
- **fonctions du système du point de vue de l'utilisateur**

Analyse :

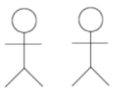
- *Diagramme de classes:*
- **structure statique en termes de classes et de relations**
- *Diagramme d'objets :*
- **Objets et leurs relations.**
- *Diagramme de séquence:*
- **représentation temporelle des objets et de leurs interactions**
- *Diagramme de collaboration:*
- **représentation spatiale des objets, des liens et des interactions**
- *Diagramme d'états-transitions:*
- **comportement d'une classe en terme d'états (Statecharts)**
- *Diagramme d'activités:*
- **comportement d'une opération en termes d'actions**



Conception :

- *Diagramme de déploiement:*
- **déploiement des composants sur les dispositifs matériels**

- *Diagrammes de composants:*
- **composants physiques d'une application**



A bientôt