UML et diagramme de classes

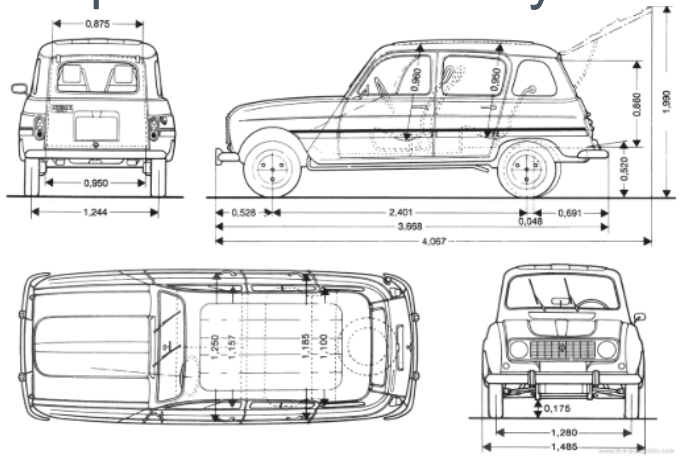
420-4C4-JR

La modélisation

- La modélisation vise à représenter d'une manière simplifiée un problème du monde réel.
  - maîtriser la complexité
  - abstraire la réalité
  - mieux comprendre le système à réaliser.
- La modélisation s'étale sur 2 étapes principales:
  - ***L'analyse***: Étude et compréhension du problème.
  - ***La conception***: donne une représentation possible du système depuis un point de vue donné en utilisant une ***méthode de conception***.

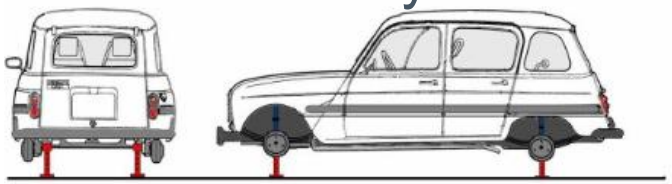
Pourquoi modéliser?

Spécifier la structure et le comportement d'un système

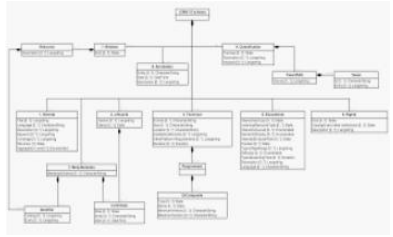


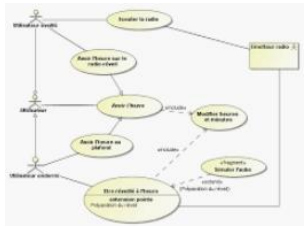
Aider à la construction d'un système

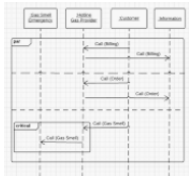
Visualiser un système

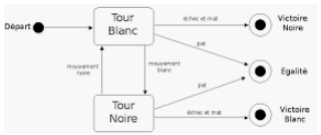


Documenter les décisions









Méthode de conception

- On part d'un énoncé informel ainsi que l'analyse de l'existant.
- Une méthode de conception d'un SI est un procédé permettant de:
  - ***Guider*** *le développeur*
  - ***Formaliser*** les étapes de développement

  afin de rendre ce développement le plus fidèle possible aux besoins du client.

Avantages de la conception

- Un bon diagramme peut souvent aider à communiquer des idées sur une conception.
- Dans une équipe de développement, les diagrammes aident à la fois à comprendre et à communiquer cette compréhension.
- Les besoins des clients changent et le personnel peut aussi changer ➔ il faut garder une trace
- Équipes différentes en développement et en maintenance.
- La nécessité de discuter de façon compréhensible avec le client.

Unified Modeling Language (UML)

- UML est un standard adopté par l'OMG (Object Management Group).
- Selon l'OMG, « UML est un langage visuel dédié à la spécification, la construction et la documentation d'un système logiciel ».
- La dernière version d'UML est 2.5.1 (diffusée par l'OMG en décembre 2017, 800 pages de spécification).
  - Définit 14 diagrammes standards.

Diagrammes de UML 2.5.1

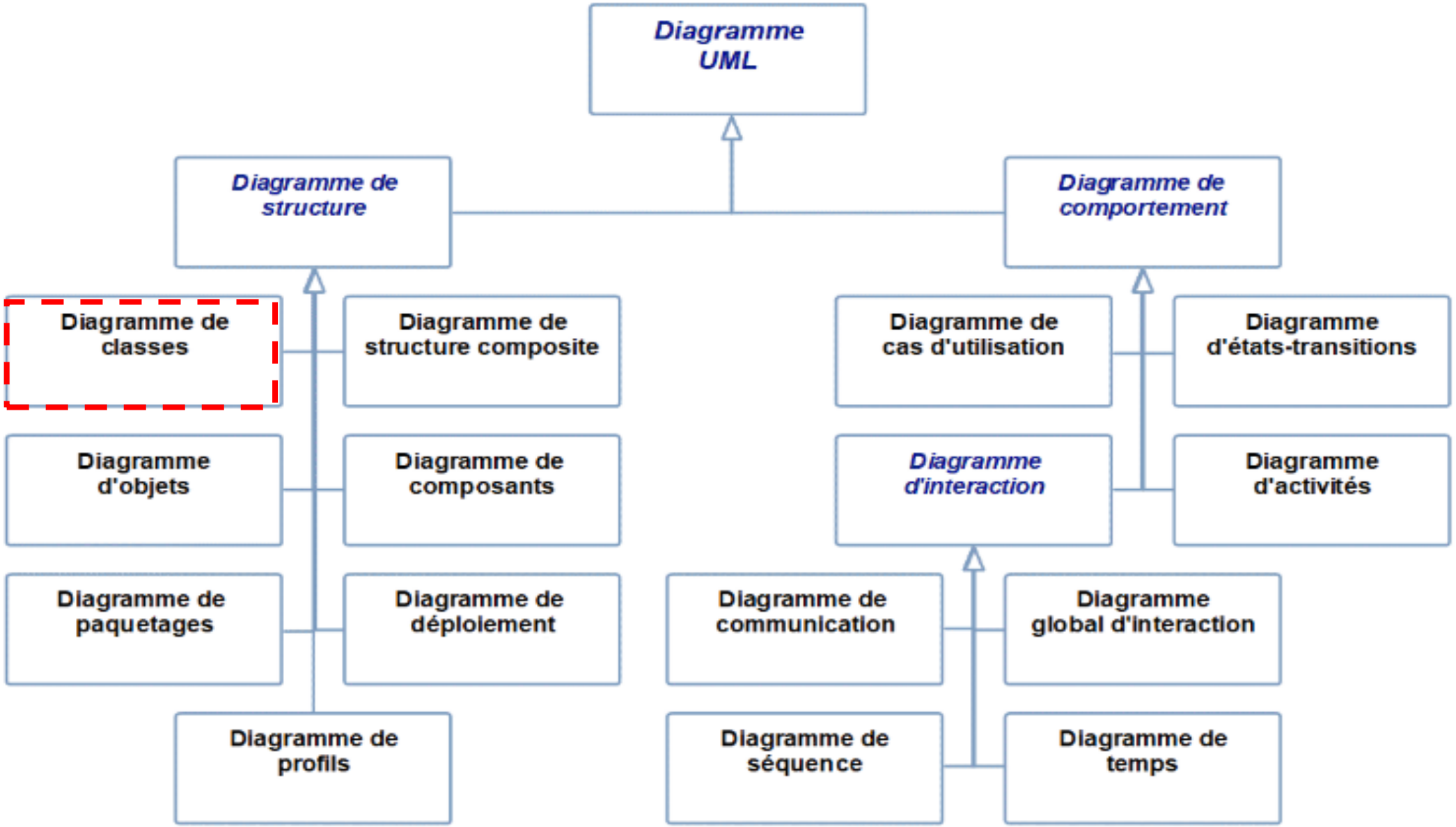


Diagramme de classes

Diagramme de classes

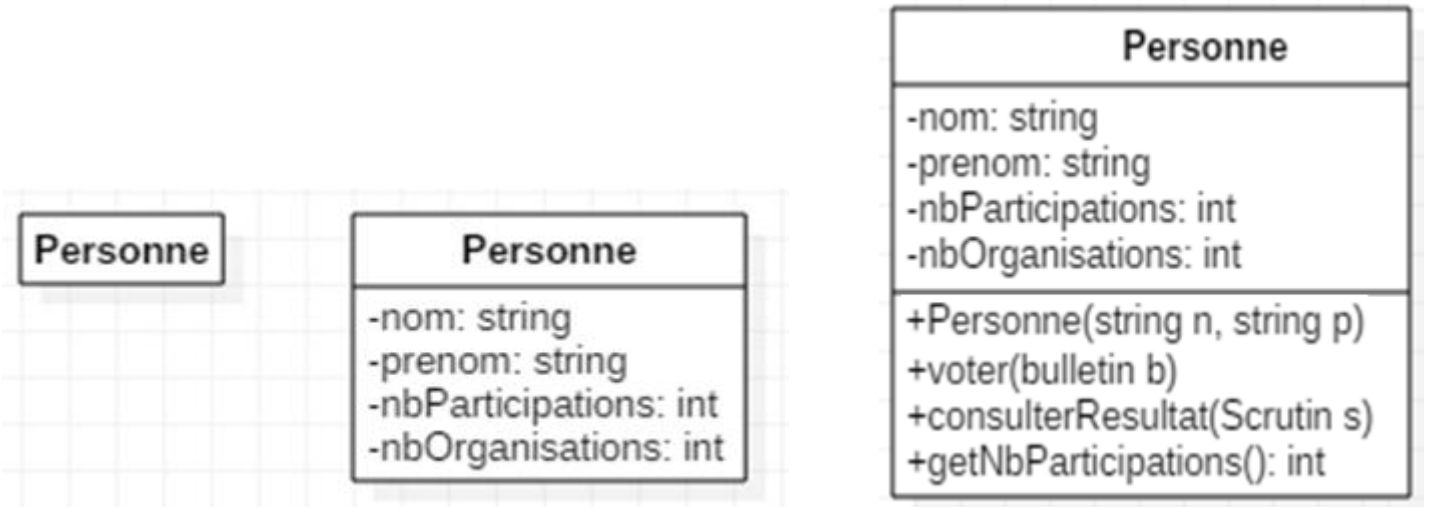
- Présente la structure statique d'un système car il ne prend pas en compte des aspects dynamiques et temporels.
- C'est un graphe formé par:
  - ***Des classes***: chacune représente un nœud du graphe.
  - ***Des relations*** entre les classes: chacune représente un arc du graphe.
  - Des contraintes,
  - Des cardinalités,
  - Des rôles,
  - Etc.
- Dans un diagramme de classes, on ne présente que les classes métiers.

Classe

- Une classe est représentée par une boite qui contient:
  - au moins le ***nom de la classe***,
  - peut contenir aussi ***des attributs***
    - On donne un nom explicatif, pas un nom de variable
    - Un attribut doit être une information simple (valeur numérique, un texte, une date, etc.)
    - Quand l'information est complexe, il faut en faire une classe.
      - Exemple: Avoir une classe Vol contenant destination comme attribut, versus avoir une Classe Vol et une classe Aéroport.
  - Peut contenir aussi ***des opérations*** (méthodes).
- Accessibilité:
  - Public: +
  - Private: -

Classe

› Exemple:



Représentation des attributs dans UML

- Format:

  `[vis] nom_attr ":" typeAttr`

  - Visibilité: + (public), - (private), # (protected), ~ (package).
  - typeAttr: type primitif (int, string, float, …) ou classe
- Élément obligatoire: nom_attr
- Signification des visibilités:
  - public ou + : tout élément qui peut voir le conteneur peut également voir l'élément indiqué.
  - protected ou # : seul un élément situé dans le conteneur ou un de ses descendants peut voir l'élément indiqué.
  - private ou - : seul un élément situé dans le conteneur peut voir l'élément.
  - package ou ~ ou rien (par défaut) : seul un élément déclaré dans le même paquetage peut voir l'élément.

Représentation des opérations dans UML

- Format:

```
[vis] nom_op ["(" liste_param ")"] [":" type_ret]
```

  - *Visibilité*: + (public), - (private), # (protected), ~ (package).
  - *liste_param*: le (ou les) argument(s) selon le format:

```
NomArgument1 : TypeArgument1, NomArgument2 : TypeArgument2
```

  - *type_ret*: le type de la valeur retournée (un type primitif ou une classe).
- Éléments obligatoires: nom_op et liste_param (peut être vide).

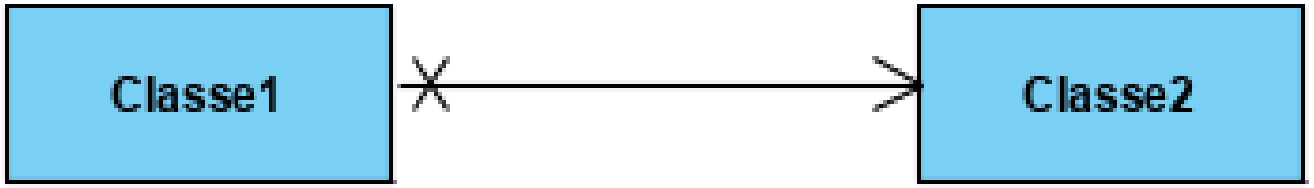
Association binaire (entre 2 classes)

- L'association définit une relation structurelle entre pairs.
- Les 2 classes sont de ***même niveau conceptuel***: aucune des 2 classes n'est plus importante que l'autre.
- Elle spécifie qu'une classe peut en utiliser une autre.
- *Exemples*:

- *Remarque*: la présence des rôles n'est pas obligatoire dans une association.

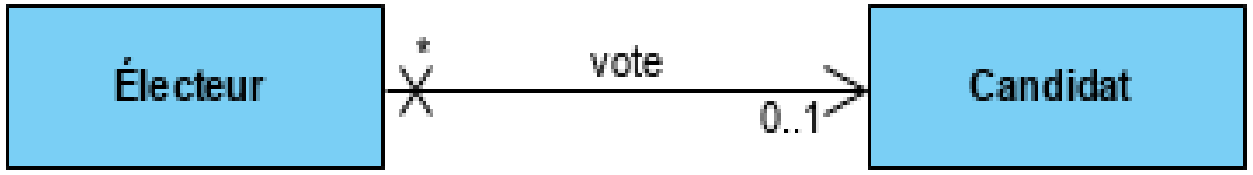
Association à navigabilité restreinte

- Par défaut, une association est navigable dans les deux sens (bidirectionnelle).
- Pour réduire la portée de l'association et ***indiquer que les instances d'une classe ne peuvent être identifiées par les instances de l'autre***, on restreint la navigabilité.



Association à navigabilité restreinte

› Exemples:



- Étant donné un Électeur, on peut identifier le Candidat pour lequel il a voté.
- Étant donné un candidat, on ne peut pas retrouver les Électeurs ayant voté pour lui.

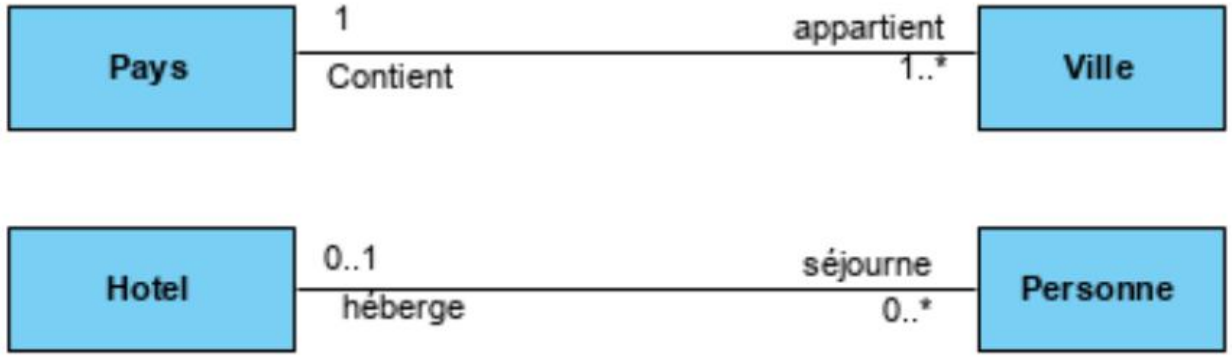
➔ L'association est donc navigable seulement de Électeur vers Candidat.

- Étant donné une Commande, il est possible de connaître la liste de ses Produits.
- Étant donné un Produit, on ne peut pas connaître la liste de ses commandes.

➔ L'association est navigable seulement de Commande vers Produit.

Cardinalités dans une association entre 2 classes

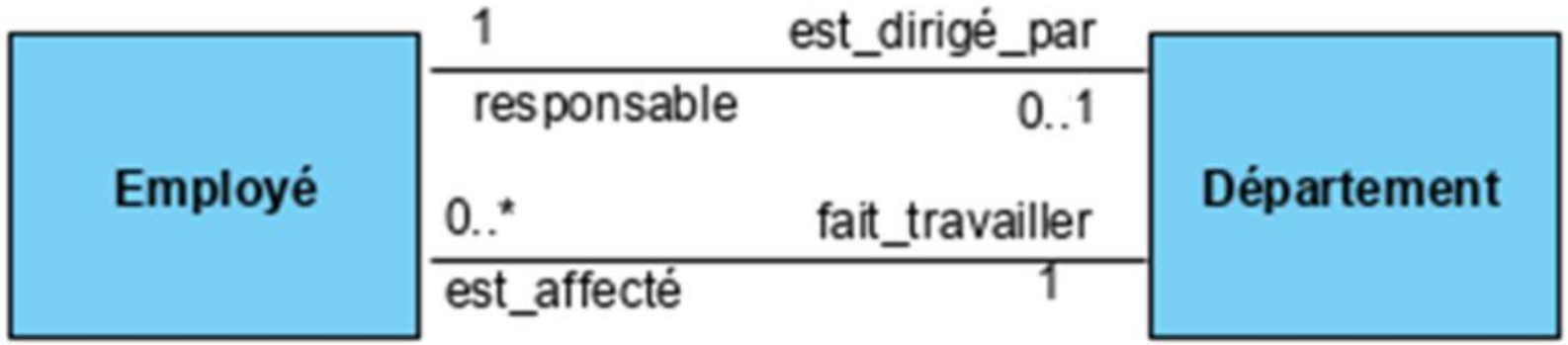
- ***Cardinalité*** (multiplicité): précise le nombre d'instances qui participent à une relation.
- Les types de cardinalités d'une relation en UML:
  - **n**: exactement « n » avec « n » est un entier naturel > 0
  - **n .. m**: avec « n » et « m » deux entiers et n < m
  - * : équivalent à « 0..* » et à « 0..n »
  - **n..***: « n » ou plus avec « n » un entier
  - La valeur par défaut est 1
- La cardinalité d'une classe A s'écrit du côté de la classe B
- Exemples:



Associations entre 2 classes

› Il peut y avoir plus d'un lien entre deux mêmes classes.

› Exemple:

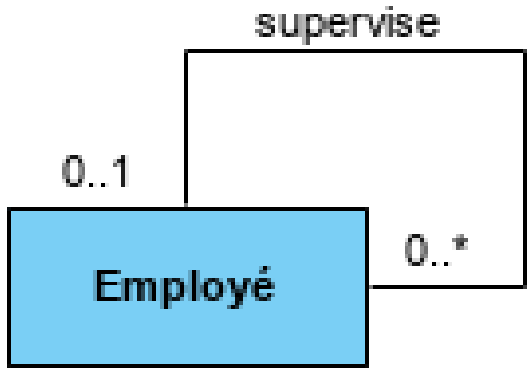


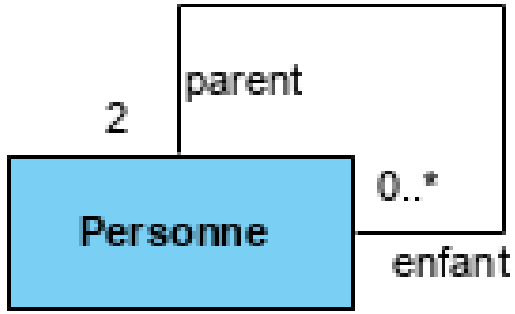
Un Employé est_affecté à un seul Département. Un Département fait_travailler 0 ou plusieurs Employés.

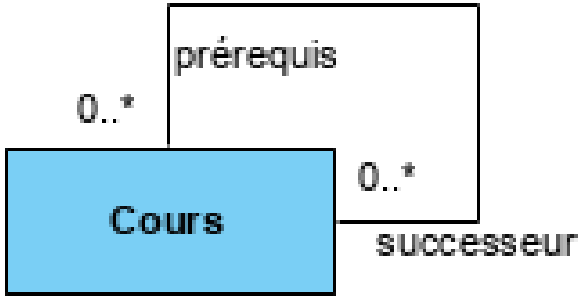
Un Département est dirigé par un seul Employé. Un Employé est responsable de 0 ou 1 Département.

Association réflexive (Self association)

› Une association est dite réflexive quand les deux extrémités de l'association pointent vers la même classe.

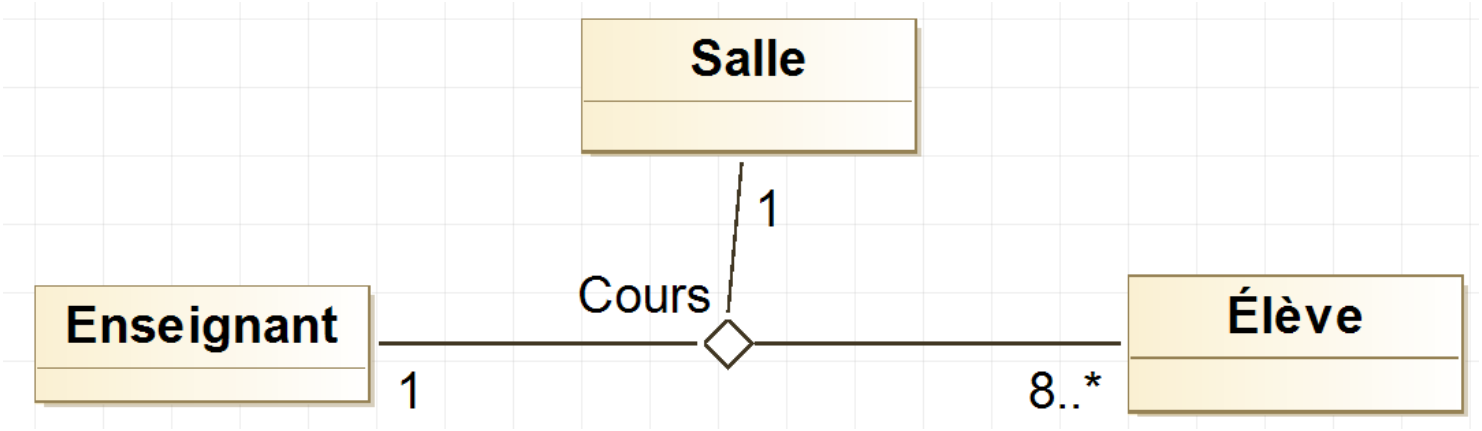






Association n-aire

- Une association n-aire relie plus de 2 classes
- Les cardinalités entre les classes et l'association doivent être soit 0..* ou 1..*
- La cardinalité d'un lien entre une association n-aire et une classe, représente le nombre minimal et maximal de participation de la classe dans l'association. Les autres entités reliées à l'association ne jouent aucun rôle.
- Contrairement aux associations binaires, les multiplicités sont « à l'endroit » pour les associations n-aires avec *n*>2.

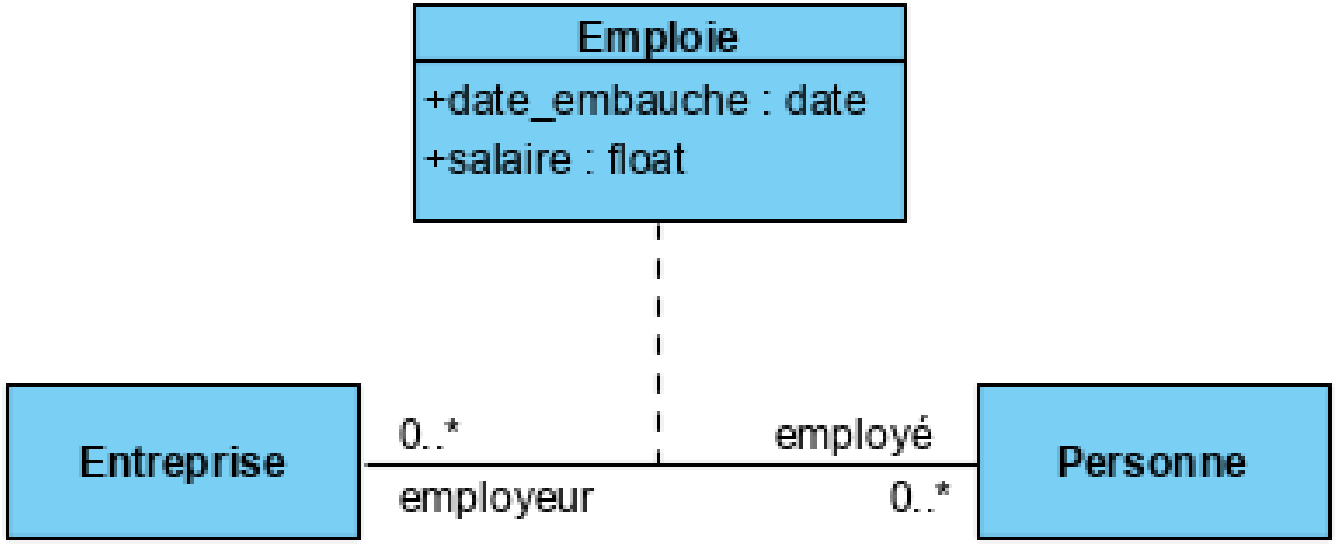


Classe-association

- Une classe-association est une association qui doit posséder des propriétés.
- Une classe-association possède les caractéristiques des classes et des associations: elle se connecte à 2+ classes et possède également des attributs et des opérations.
- Une classe-association est caractérisée par un trait discontinu entre la classe et l'association qu'elle représente.

Classe-association (Exemple)

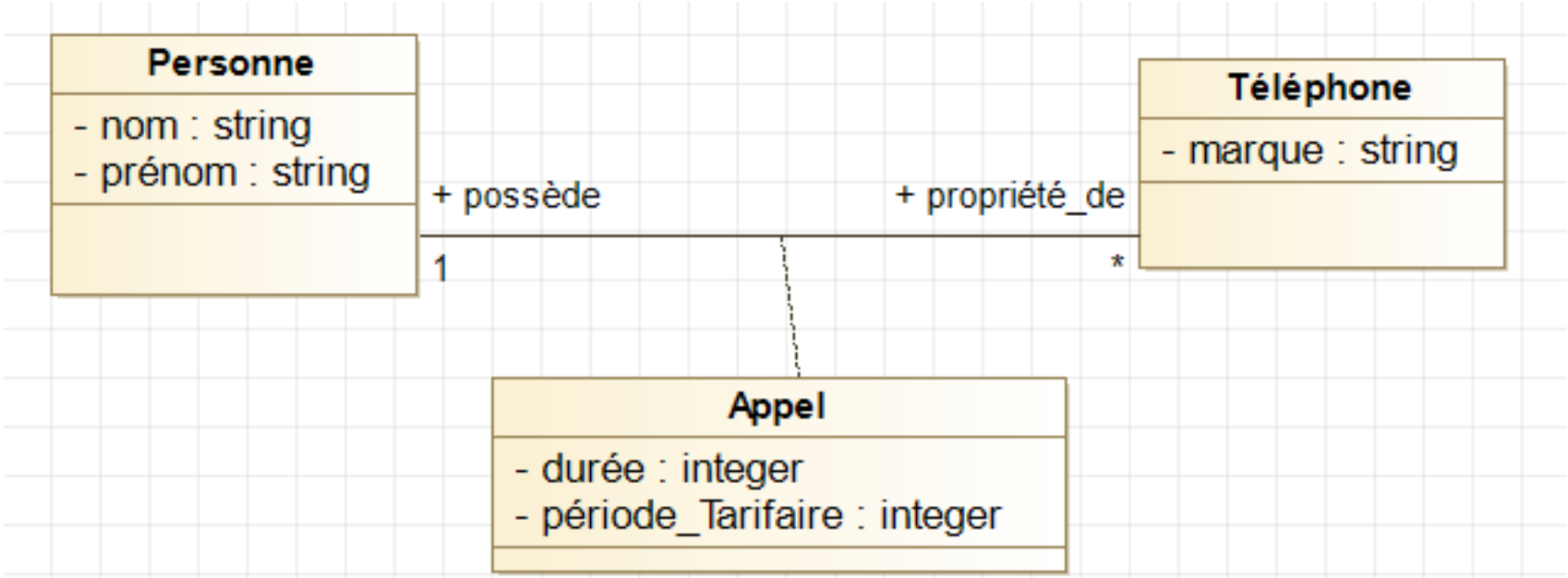
- Classe-association ***Emploie*** entre les classes ***Entreprise*** et ***Personne***.
- Elle possède les attributs ***date_embauche*** et ***salaire***.
- Ces attributs n'appartiennent ni à entreprise qui peut employer plusieurs personnes, ni à personne qui peut avoir plusieurs emplois
- Valeurs de ces attributs diffèrent selon la combinaison {Entreprise, Personne}.



Classe-association (Exemple)

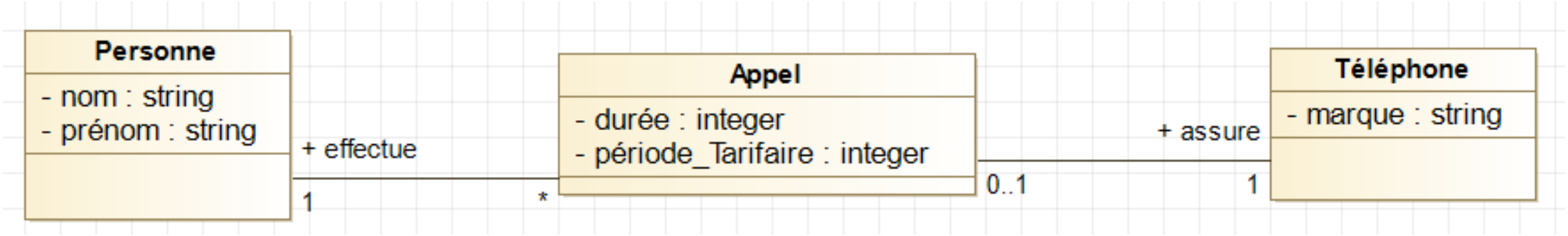
- Lorsqu'une personne utilise un téléphone, on devrait connaître à quel moment il a eu lieu et savoir sa durée. Ceci doit se faire pour chaque appel téléphonique.
- Ces 2 attributs ne doivent pas être dans la classe « Téléphone » ni dans la classe « Personne ».

  ➔ Nouvelle classe « Appel » attachée à l'association



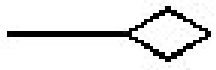
Classe-association (Remarque)

- Il est possible de convertir la classe association en un ensemble d'association binaires.



Agrégation

- Association de composition entre 2 classes dont l'une représente ***"une partie de"*** l'autre ➔ Lien composant/composite
- L'existence de la classe composante ***n'est pas obligatoire*** pour la classe composite.
  - La destruction de l'un n'entraîne pas la destruction de l'autre.
- Symbole graphique: ***losange vide*** du côté de la classe composite.



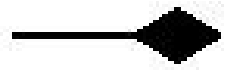
Exemples d’agrégation

- Ensemble Table + chaises: si on supprime la table, les chaises contenues à exister.
- Avion + sièges: Un avion peut bien exister sans des sièges, par exemple, dans le cas d’un avion-cargo.

	Le polygone est l’agrégat de 3 points ou plus de l’espace. La suppression du polygone n’entraîne pas la suppression des points.
	Le courriel est l’agrégat. Un fichier lui est attaché. Un fichier est autorisé à être attaché à plus d’un même courriel en même temps (ce qui ne serait pas le cas s’il s’agissait d’une composition).

Composition (Agrégation forte)

- Association de composition entre 2 classes dont l'une est le composant (une partie de) et l'autre est le composite.
- Leurs cycles de vie sont étroitement liés: La création (destruction) du composite entraîne la création (destruction) de ses composants.
- Symbole graphique: ***losange plein*** du côté de la classe composite.



Exemples de composition

- Table et Pattes: Une table sans pattes n'est pas une table
- Examen et Questions: Un examen sans questions n'est pas un examen
- Paquet de cartes et Cartes: Un paquet de carte sans carte est plutôt inutile.
- Université et Facultés: Si une université est supprimée, toutes ses facultés le seront aussi.

	Un chapitre n'appartient qu'à un seul livre. Si le livre est supprimé, le chapitre le sera aussi.
	Une molécule est composée d'atomes. Une instance d'atome n'est rattachée qu'à une seule instance de molécule. La destruction de la molécule implique la destruction de tous ses composants atomes.

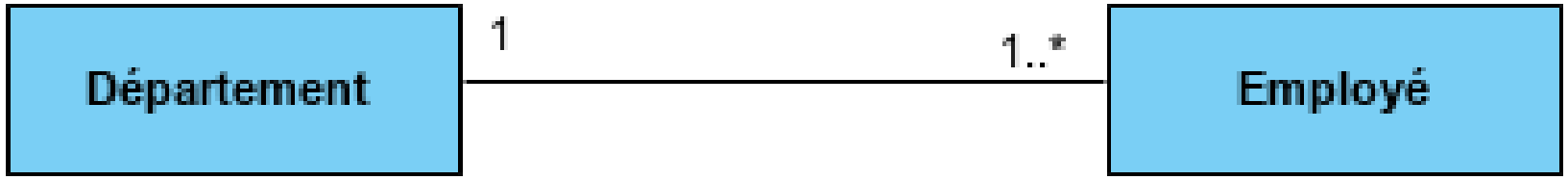
Exercice

1. Comment présenter l’association entre la classe « Département » et la classe « Employé » ?

2. Comment présenter l’association entre la classe « Contrat » et la classe « Clause » ?

Exercice

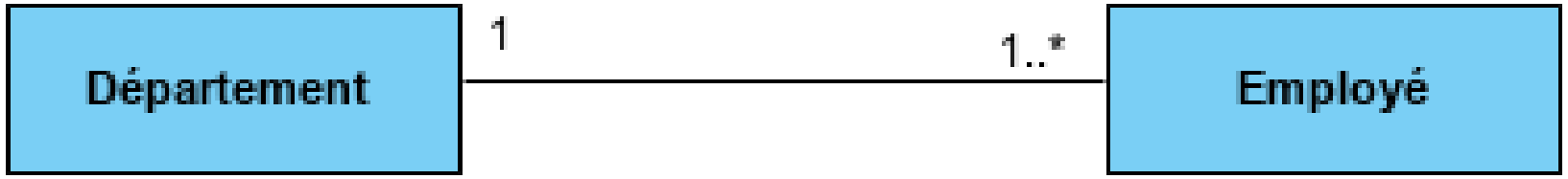
1. Comment présenter l'association entre la classe « Département » et la classe « Employé » ?



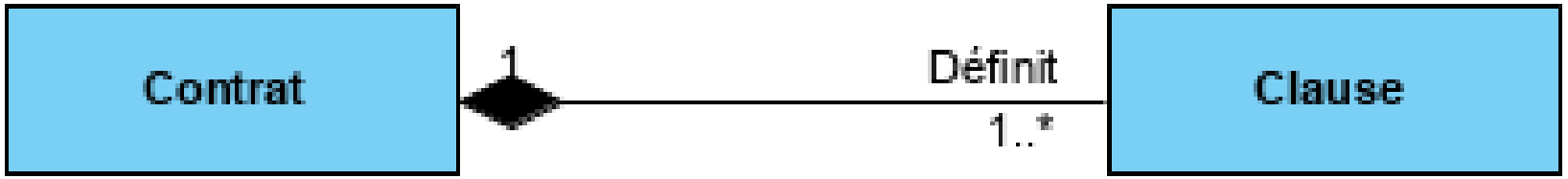
2. Comment présenter l'association entre la classe « Contrat » et la classe « Clause » ?

Exercice

1. Comment présenter l'association entre la classe « Département » et la classe « Employé » ?

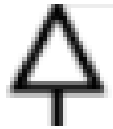


2. Comment présenter l'association entre la classe « Contrat » et la classe « Clause » ?



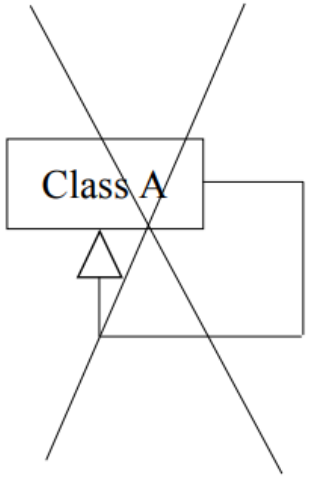
Héritage

- Décrit une relation entre une classe de base (générale, parent) et une sous-classe (spécialisée).
- La sous-classe:
  - possède toutes les caractéristiques de ses classes de base.
  - contient des informations supplémentaires à la classe de base.
  - ne peut pas accéder aux propriétés privées de la classe de base.
  - peut redéfinir des méthodes de la classe de base.
- Symbole graphique: triangle vide du côté de la classe de base.

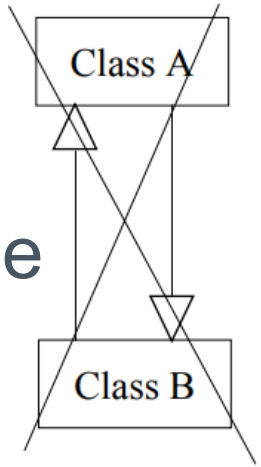


Héritage

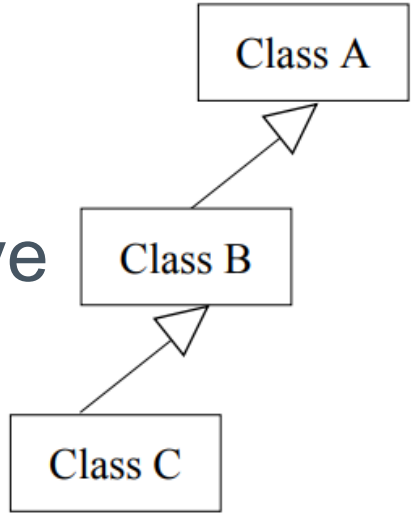
› Relation **non** réflexive



› Relation **non** symétrique



› Relation transitive



Principe de cohésion

- La cohésion est un principe de conception informatique qui concerne la ***cohérence des différents éléments*** d'une classe (attributs, méthodes).
- Une **M*étrique*** qui mesure l'application des principes orienté objets de masquage de l'information et de l'encapsulation des données.
- Dans une classe, la cohésion peut être ***faible*** comme elle peut être ***forte***.
- Plus la cohésion est forte plus les éléments de la classe sont ***cohérents*** et ***homogènes***.

Cohésion faible

- Les éléments d'une classe sont disparates et ils ont peu ou rien de commun.
- Exemples:
  - Une classe « FourreTout » qui contient une fonction de calcul de paye, une fonction de calcul des frais de vacances, une fonction de traduction d'acronymes, une fonction de traduction de code ASCII, etc.
  - Un attribut « Adresse » avec numéro civique, rue, secteur, ville, région, province, pays, continent, code postal, longueur et largeur du terrain, couleur, arbre (si présent ou pas), servitude (oui ou non et positionnement), etc.

Cohésion forte

- Les éléments d'une classe se ressemblent et s'assemblent. La classe est dédiée à une seule et unique tâche bien spécifique.
- Exemple: Adresse avec numéro civique, rue, ville, province, code postal.
- Remarque : Il faut noter qu'il n'y a pas de solution parfaite pour tous les cas possibles. Une cohésion forte dans un contexte peut être considérée faible dans un autre contexte.
  - Exemple: la division des informations de l'adresse dépend du contexte. Si on ne s'en sert que pour l'afficher, une chaine complète peut faire l'affaire. Si on est Poste Canada, il est possible qu'on veuille la diviser dans ses moindres détails. Comme il y a des niveaux intermédiaires (entre les deux).

Principe du couplage

- Le couplage est une métrique mesurant l'échange d'information entre des classes.
  - Elle s'occupe du nombre de liens entre les classes.
  - Plus il y a de liens, plus une modification a de l'impact sur les autres.

- ***Couplage faible***: Quelques liens reliant toutes les classes, le minimum possible pour être efficace et fonctionnel.
  - Une *bonne architecture logicielle* utilise le couplage le plus faible possible entre les classes.
  - ***Exemple***: Pour un Employé qui travaille à une Adresse dans un Département. Il suffirait de lier la classe Employé à la classe Département, et la classe Département à la classe Adresse.

Principe du couplage

- ***Couplage fort***: Beaucoup de liens entre les classes.
  - Produit l'antipatron « plat de spaghetti ».
  - Une classe reliée à beaucoup d'autres classes: fort probable antipatron « Blob ».
  - Dans la plupart des cas, le couplage fort émane d'une mauvaise conception même si les liens ne sont souvent pas faux et sont juste excédentaires.
  - **Exemples:**
    - Classe **Employé** en lien avec les classes **Département**, **Local**, **Adresse**, **Compagnie**. Certains de ces liens sont de trop et on peut les déduire.
    - **Client** est l'acheteur d'une **Facture**, **Facture** liste des **produits** ➔ Aucune nécessité de relier Client à Produit.
- ***Couplage nul***: classe « solo ». Aucun lien vers d'autres classes. Possible, mais à éviter.
  - Signifie que l'utilité de la classe pour le système est trop basse.

Couplage / cohésion

- Le couplage idéal et la cohésion parfaite sont impossibles.
- Il faut viser un équilibre. Pour cela, il faut se poser les bonnes questions comme:
  - Le concept est-il suffisamment divisé? Ou trop divisé?
  - La division est-elle utile (réutilisable, cohésion) ou superflue?
  - Le lien entre 2 concepts est-il nécessaire (ou peut-on trouver l'information comme-même)?
  - Un lien essentiel a-t-il été oublié?