Modèles Statiques

Diagrammes d'objets et de classes

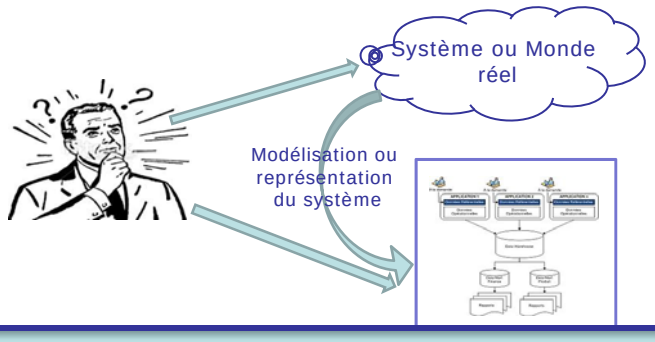
Pr. Larbi Kzaz

Octobre 2018

Introduction

Paradigme ou Approche Objet: de quoi s'agit-il?

- ✓ C'est une façon de Voir les choses ou d'aborder des problèmes. Et qui repose sur un fondement défini: modèle théorique, courant de pensée, etc.
- ✓ En *Géni Logiciel*, le paradigme ou encore *Approche Objet*, traduit une façon particulière de voir et d'aborder un *Système*, appelé aussi *Monde réel*, par les différents *Acteurs* du GL: *Analystes, Concepteurs, Développeurs*.



Introduction

Paradigme ou Approche Objet:

Le paradigme *Objet* repose sur l'idée suivante:

- ✓ Tout système est une collection d'éléments et d'entités du monde réel pouvant être concrètes (chose physique, équipement, être humain) ou abstraites (chose immatérielle, concepts, notions, idées, etc.). Les entités du système ont une existence propre; elles jouissent d'une certaine autonomie, agissent et interagissent entre elles.
- ✓ Un *OBJET* est une *représentation abstraite* d'une ENTITE du monde réel.

Remarques:

- ✓ La représentation du système traduit ce que voit et perçoit un INDIVIDU en fonction de ses PROPRES CAPACITÉS COGNITIVES ET SES PRÉOCCUPATIONS.
- ✓ Des individus différents peuvent ainsi produire des représentations (modèles) différentes d'un même système.

Introduction

Exemples:

- Domaine Scolarité du Système « Ecole ou université »

  Objets: Etudiant, Professeur, Matière, Salle de cours, Séance, Examen, etc.

- Domaine Commercial du Système « Entreprise »

  Objets: Client, Fournisseur, Article, Bon de Commande, Facture, etc.

- Domaine RH du Système « Entreprise »

  Objets: Salarié, Grille des Salaires, Fonctions, Postes de travail, Congé, Formation, etc.

- Domaine Commercial du Système « Agence Immobilière»

  Objets: Bien immobilier, Propriétaire, Locataire, Rendez-vous de Visite, Contrat de Vente, Contrat de Location, etc.

Concepts Fondamentaux

Notion d'Objet:

C'est une représentation abstraite d'un élément concret ou abstrait du monde réel.

Modélisation

- ETUDIANT
- PROFESSEUR
- SALLE DE COURS
- MATIERE
- SEANCE
- ……

éléments du Monde réel

Représentation des objets du monde réel

Concepts Fondamentaux

Caractérisation d'un objet:

Tout objet est caractérisé par:

- ✓ *Une Identité:*

  C'est une donnée « naturelle » ou « artificielle » (créée spécialement), permettant de distinguer un objet des autres objets du système.

- ✓ *Des Attributs:*

  Ce sont des données (Variables, Propriétés) associées à un objet et dont les valeurs peuvent varier dans le temps. Les valeurs des attributs d'un objet à un instant précis constituent l'*ETAT* de l'Objet, et permettent le suivi de son évolution dans le système.

- ✓ *Un Comportement:*

  Il s'agit de l'ensemble des opérations qu'un objet est capable de réaliser. Ces opérations sont appelées aussi *Méthodes*.

Concepts Fondamentaux

Exemples:

Proposer des *attributs* et des *méthodes* pour chacun des objets suivants:

Objets Informatique (Système d'Exlpoitation) :

- Fenêtre: Window
- Fichier
- Disque

Objets Mécanique :

- Roue
- Moteur

Objet du domaine RH :

- Salarié

Objet du domaine Banque :

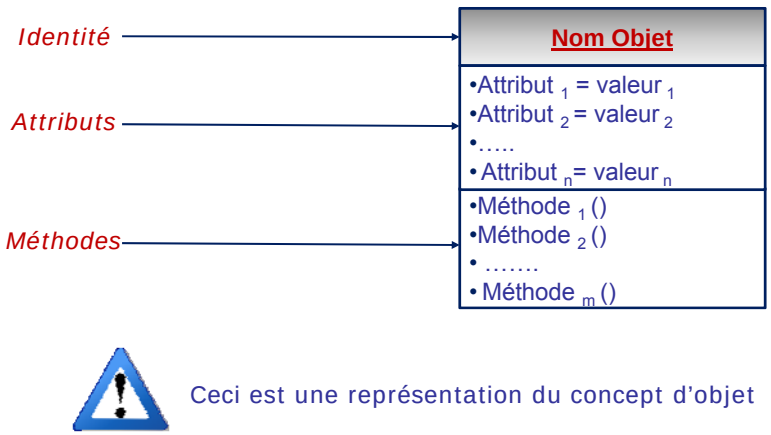
- Compte Bancaire

Objet du domaine Scolarité:

- Etudiant

Concepts Fondamentaux

Représentation d'un objet:



Ceci est une représentation du concept d'objet

Concepts Fondamentaux

Etat d'un objet:

A chaque instant chaque *attribut* d'un objet possède une *valeur*. L'ensemble des valeurs des différents attributs d'un objet définissent son *état* à un instant donné.

Objet
•Attribut $_1$ = valeur $_1$ •Attribut $_2$ = valeur $_2$ •….. • Attribut $_n$= valeur $_n$
•Méthode $_1$ () •Méthode $_2$ () • ……. • Méthode $_m$ ()

Ma voiture
•numéroMatricule $_1$ = "1478 -ب – 8 " •marque= "DACIA " •dateMiseEnCirculation= "13/04/2016 " •compteurKilométrique= 20470
•démarrerMoteur() •arrêterMoteur() •incrémenterCompteur() • calculerAger() • controlerNiveauHuile()

Objets ET Classes

Notion de Classe:

Une *Classe* regroupe un *Ensemble d'objets semblables*. Les objets d'une classe ont les *mêmes attributs* (caractéristiques) et possèdent les *mêmes méthodes* (même comportement)

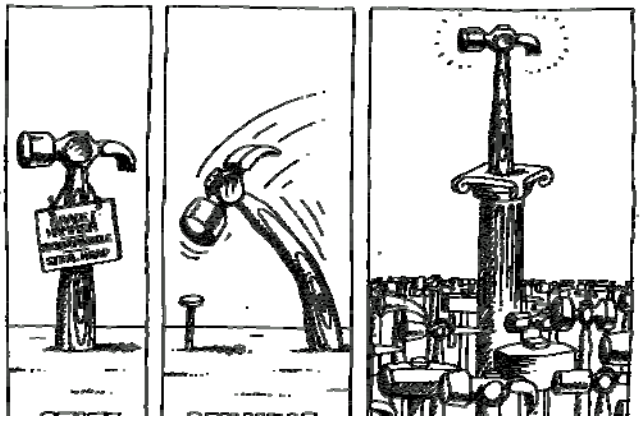
- Un objet est un *élément* d'une classe.
- On dit aussi qu'un Objet est une *Instance* d'une classe.
- Deux objets d'une même classe ont des *identités différentes*. (Principe d'Identité)

Quelle analogie pourrait-on faire avec:

- le moule,
- la statuette, et
- la fabrication d'une statuette?

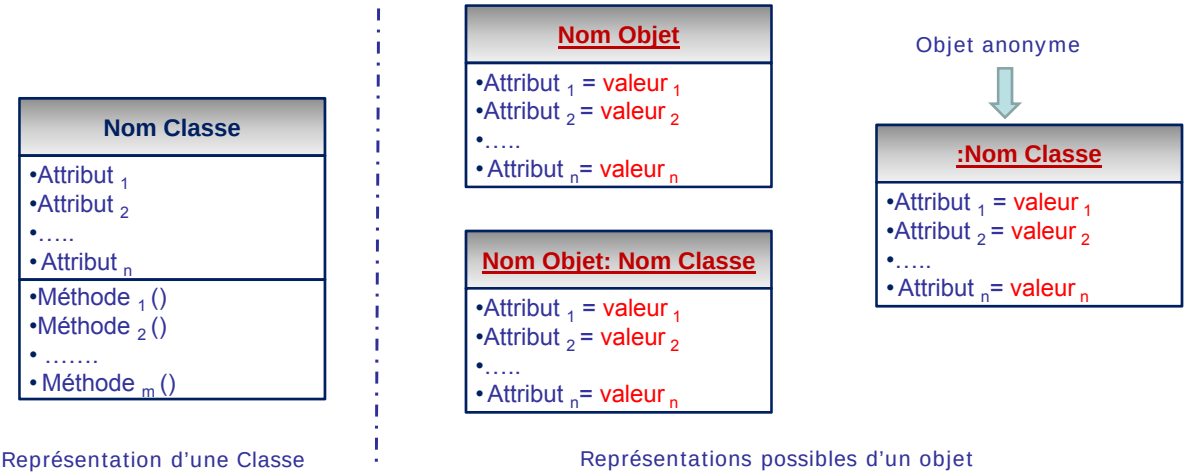
Objets ET Classes

Exemple:



Objets ET Classes

Représentation des Objets et des Classes:



Représentation d'une Classe

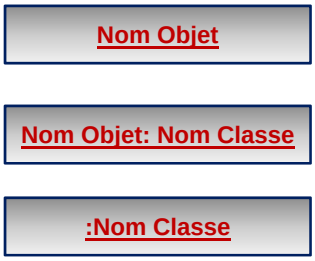
Représentations possibles d'un objet

Objets ET Classes

Représentation simplifiée des Objets et des Classes:

Durant les premières étapes de modélisation on pourra pour des raisons pratiques, se limiter aux représentations simplifiées suivantes:

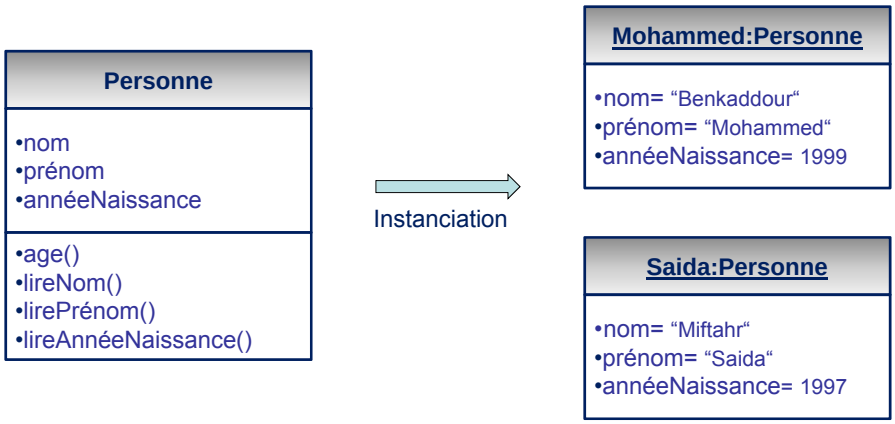
Représentation simplifiée d'une Classe



Représentations simplifiées d'un objet

Objets ET Classes

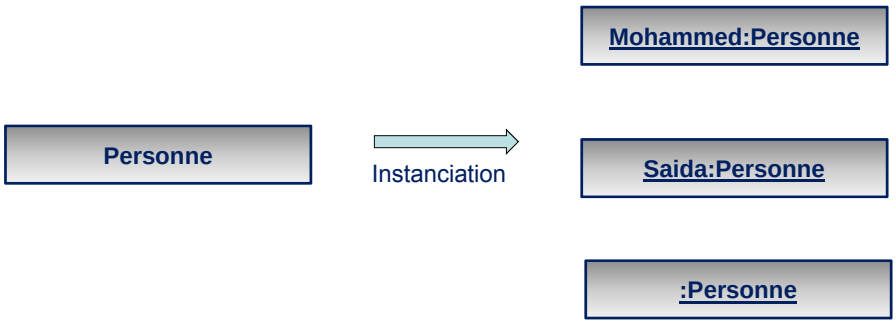
Exemple:



Deux instances de la classe Personne

Objets ET Classes

Exemple:

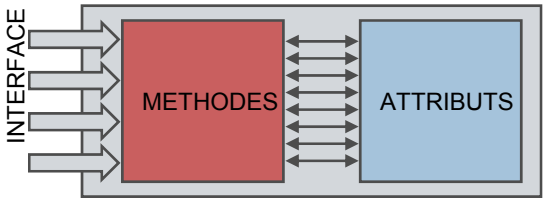


Représentations simplifiées des classes et des objets

Encapsulation et Visibilité

Principes:

- ✓ L' *Encapsulation* consiste à cacher et les attributs d'une classe et à les rendre inaccessibles aux autres classes du système.
- ✓ La seule façon d'accéder aux attributs d'une classe est de lui *envoyer un message* (*appel de méthode*), qui va déclencher l'exécution de l'une de ses méthodes.



Encapsulation et Visibilité

Principes:

- ✓ L'accès à un attribut consiste à:
  - Lire ou Récupérer sa valeur; cela permet de connaitre l'état d'un objet, instance d'une classe.
  - Ecrire ou Modifier sa valeur; cela permet de modifier l'état d'un objet, instance d'une classe.
- ✓ Des méthodes de type LireAttribut(), EcrireAttribut(), InitialiserAttribut(), AffecterAttribut() font partie des méthodes associées Classes.

Encapsulation et Visibilité

Niveaux de visibilités:

Elle définit les niveaux de visibilité, ou encore les droits d'accès, des attributs et méthodes d'un objet. On distingue trois niveaux de visibilité:

- ✓ *Publique* (+) *:* L'attribut, ou la méthode, est publique; elle est accessible aux autres objets du système.
- ✓ *Privé* (-) *:* L'attribut, ou la méthode, est privé; il n'est accessible que par les méthodes de l'objet.
- ✓ *Protégé* (#) *:* L'attribut, ou la méthode, est protégé; il n'est accessible que par les méthodes des objets héritiers.

Remarque:

- Les attributs et les méthodes publiques forment la partie Interface de la Classe.

Attributs et Méthodes de Classe

Définition:

Par défaut, chaque instance d'une classe possède sa propre copie des attributs de la classe. Les valeurs des attributs peuvent donc différer d'un objet à un autre. Chaque objet a son propre état.

Parfois, on a besoin de définir un attribut qui garde *une valeur unique* et *partagée* par toutes les instances de la classe.

Cet attribut s'appelle **attribut de classe**

Attributs et Méthodes de Classe

Définition:

Les *attributs* et *les méthodes de classe* sont des attributs et des méthodes qui *n'ont pas de sens pour les objets* lorsqu'ils sont pris individuellement; ils sont définis et sont *significatifs au niveau de la classe* dans son ensemble.

Les *attributs* et *les méthodes de classe* sont aussi appelés dans le jargon des *langages de programmation* des *attributs* ou des *méthodes statiques*. (static en Java ou en C++)

Une *méthode de classe* ne peut manipuler que des *attributs de classe* . Elle n'a pas accès aux *attributs de **la** classe* (i.e. des instances de la classe).

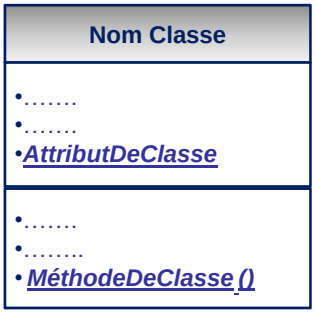
L'accès à une *méthode de classe* ne nécessite pas l'existence d'une instance de cette classe.

Exemple:

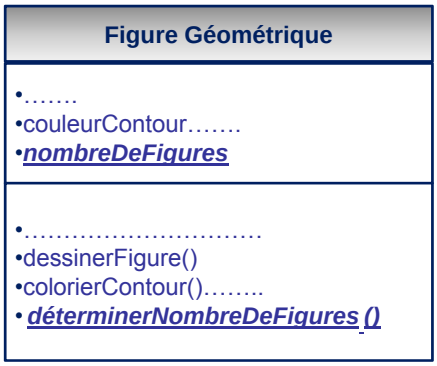
- Le nombre d'instances d'une classe est un *attribut de classe*.
- La méthode qui calcule le nombre d'instances d'une classe est une *méthode de classe*.

Attributs et Méthodes de Classe

Notation:



Exemple:



Relations

Relations entre classes:

Les Objets du système peuvent avoir des relations; on distingue quatre types de relations:

- **Héritage** http://uml.free.fr/cours/p15.html
- **Agrégation**
- **Composition**
- **Association**

Relations : Héritage

Présentation:

- *L'héritage est une relation qui permet de **modéliser des hiérarchies**, lorsqu'elles existent, entre les classes.*

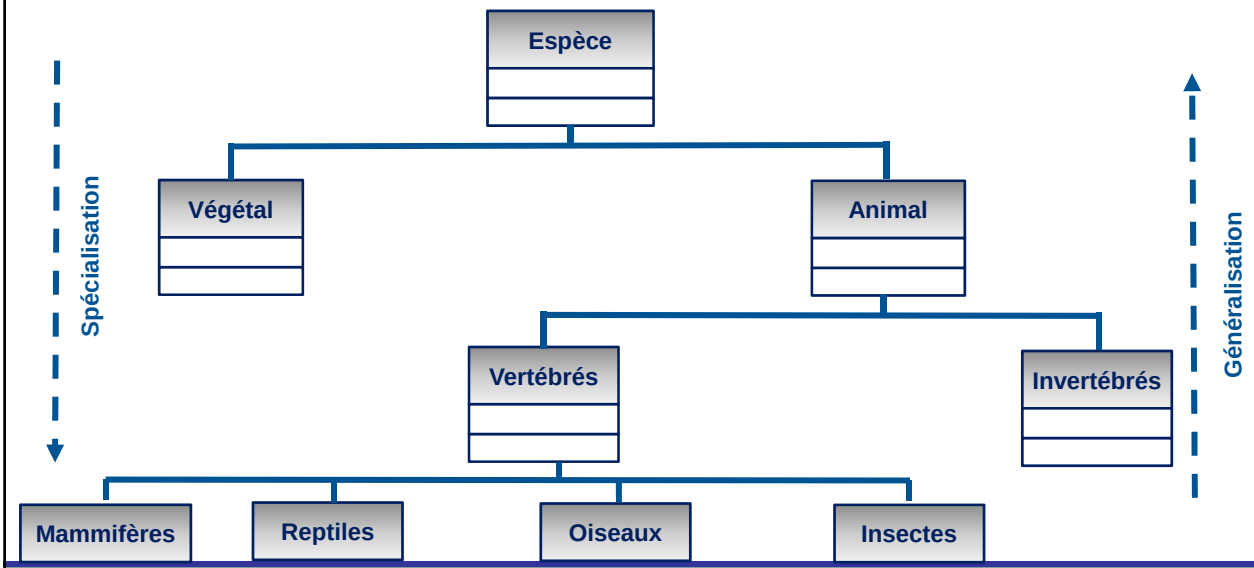
Elle permet de:

- ✓ *Gérer la complexité, en ordonnant les objets au sein d'arborescences de classes, d'abstraction croissante.*
- ✓ *Etablir une sorte de classification entre un ensemble d'objets ayant des attributs et des méthodes en communs et d'autres qui ont des attributs et des méthodes spécifiques.*

- *Deux approches sont possibles:*
  - ✓ *Approche **Ascendante**:*
    Elle Consiste à partir d'objets spécifiques pour aboutir à des objets plus généraux.
    *Cette approche porte le nom de **Généralisation.***
  - ✓ *Approche **Descendante**:*
    Elle Consiste à partir d'objets Généraux pour aboutir à des objets spécifiques.
    *Cette approche porte le nom de **Spécialisation**.*

Relations : Héritage

Exemple : Classification des espèces:

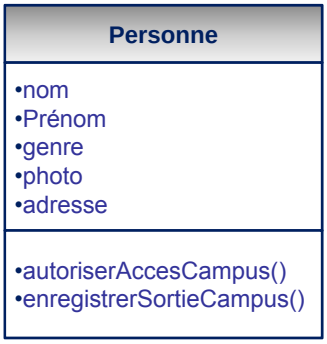


Relations : Héritage

Exemple Introductif:

Supposons qu'on souhaite modéliser l'ensemble des personnes qui fréquentent notre campus.

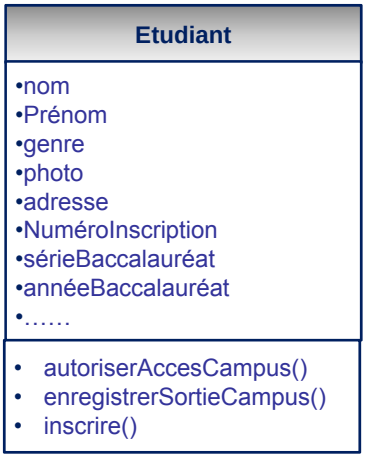
Première approche: On représente toute la population par une classe Unique « Personne »

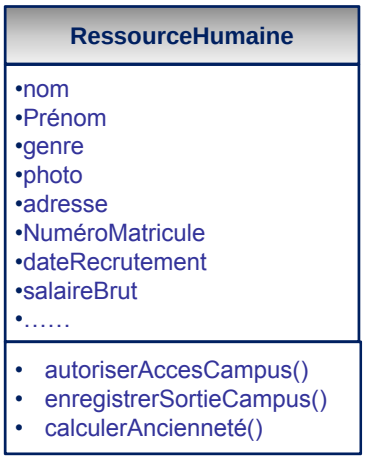


Relations : Héritage

Exemple Introductif:

✓ *Deuxième Approche: On représente cette population par **deux classes**:*





Relations : Héritage

Première approche

On remarque que parmi les personnes, certains ont des attributs, resp. des méthodes, spécifiques.

Ainsi, les étudiants ont, en plus des attributs de la classe PERSONNE, *un numéro d'inscription, une série de baccalauréat et l'année de son obtention, etc.*

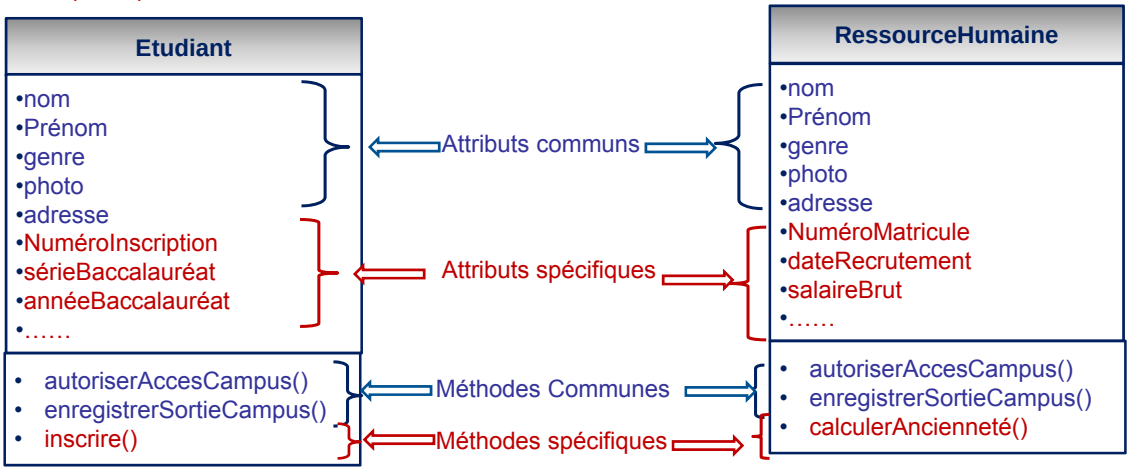
De même, le personnel administratif et enseignant, dispose d'un numéro matricule, d'une date de recrutement, d' un salaire Brut, etc.

On décide alors de créer deux nouvelles classes.

Relations: Héritage

Première approche:

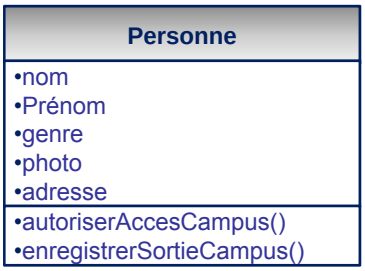
Les attributs des deux nouvelles classes comportent, en plus des attributs de la classe PERSONNE. Des attributs spécifiques.



Relations : Héritage

Première approche:

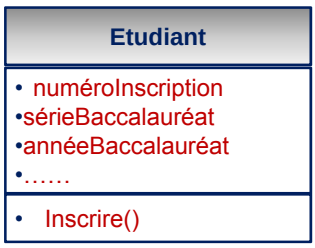
*La classe **« Personne »** ne contiendra plus que **les attributs et les méthodes communes** aux deux classes.*

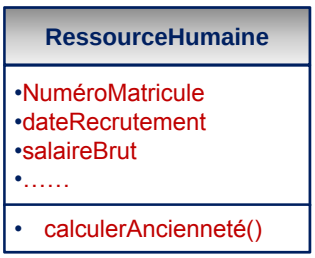


Relations : Héritage

Première approche:

Les deux classes Etudiant et Ressource Humaine contiendront les attributs et les méthodes qui leur sont spécifiques.

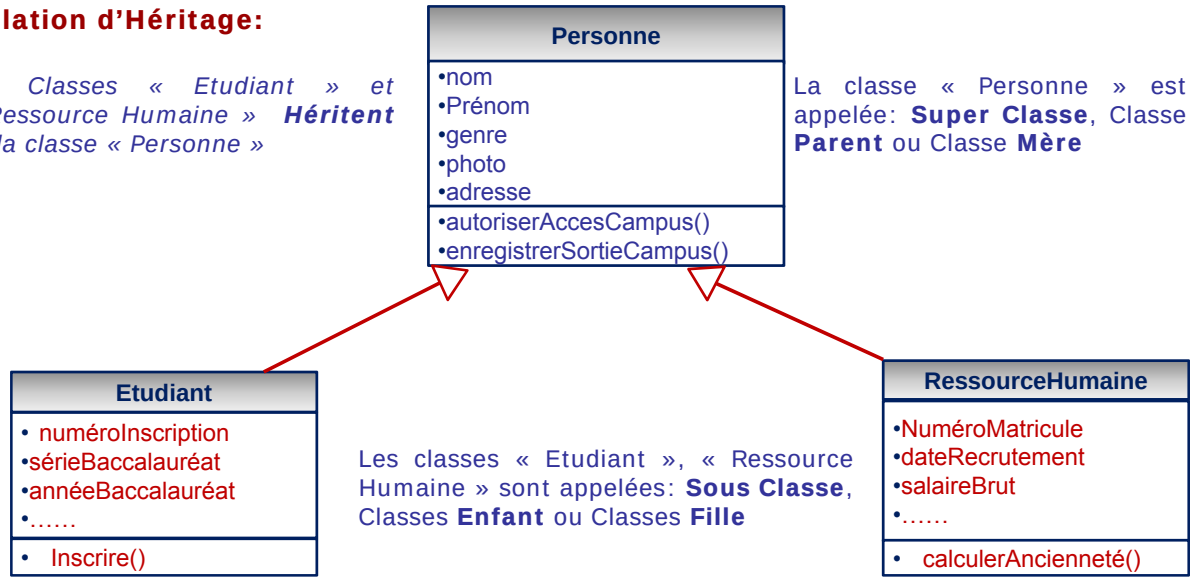




Relations : Héritage

Relation d'Héritage:

*Les Classes « Etudiant » et « Ressource Humaine » **Héritent** de la classe « Personne »*



La classe « Personne » est appelée: **Super Classe**, Classe **Parent** ou Classe **Mère**

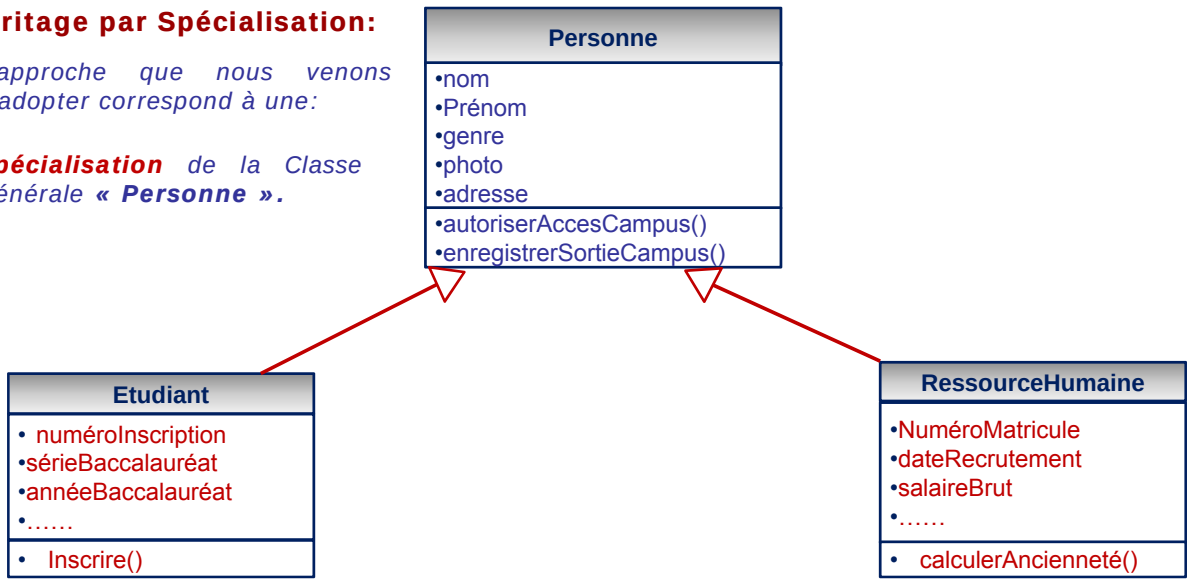
Les classes « Etudiant », « Ressource Humaine » sont appelées: **Sous Classe**, Classes **Enfant** ou Classes **Fille**

Relations: Héritage

Héritage par Spécialisation:

L'approche que nous venons d'adopter correspond à une:

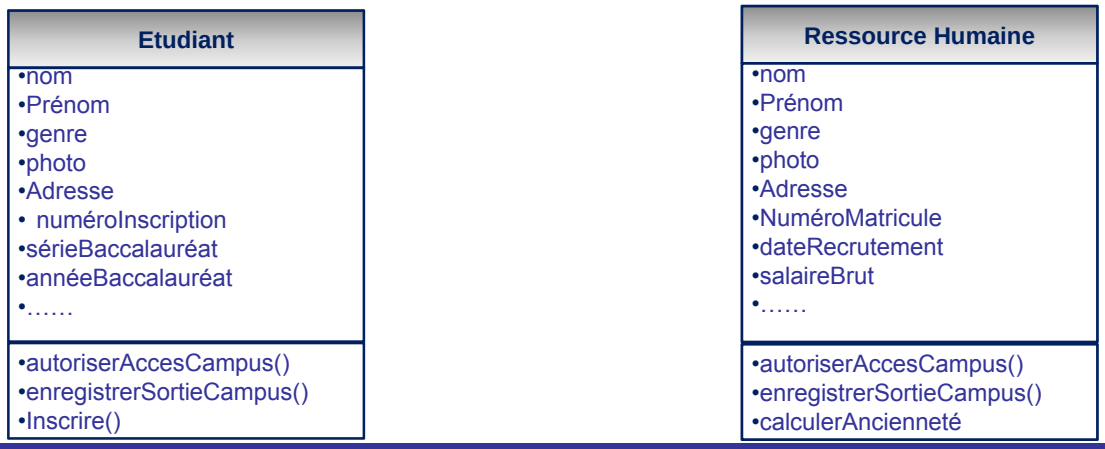
Spécialisation** de la Classe Générale **« Personne ».



Relations: Héritage

Deuxième Approche:

- ✓ *La deuxième approche applique une démarche inverse:*
- ✓ *Elle consiste à observer dès le départ que la population qui fréquente le campus est composée d'étudiants et de ressources humaines:*

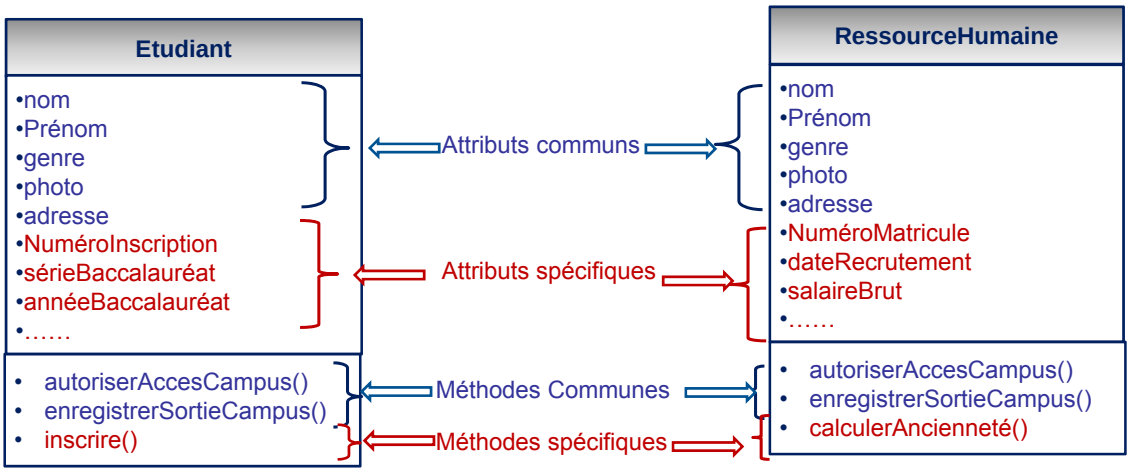


Relations: Héritage

Deuxième approche:

On constate par la suite que les deux classes ont:

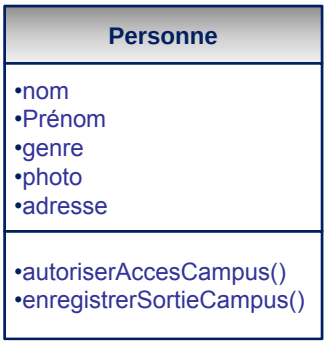
Des Attributs et des méthodes ***communes*** *Des Attributs et des méthodes* ***propres***



Relations: Héritage

Deuxième Approche:

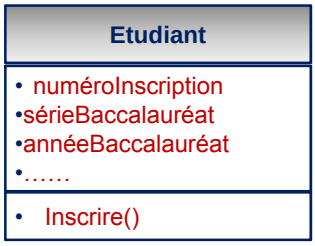
*Ce constat nous amène à créer une nouvelle classe **« Personne »** qui regroupera les attributs et les méthodes communes.*

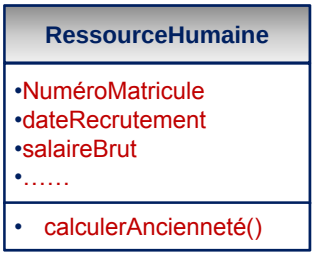


Relations : Héritage

Deuxième approche:

Et de ne conserver dans les classes « Etudiant » et « Ressource Humaine » que les attributs qui leur sont spécifiques..



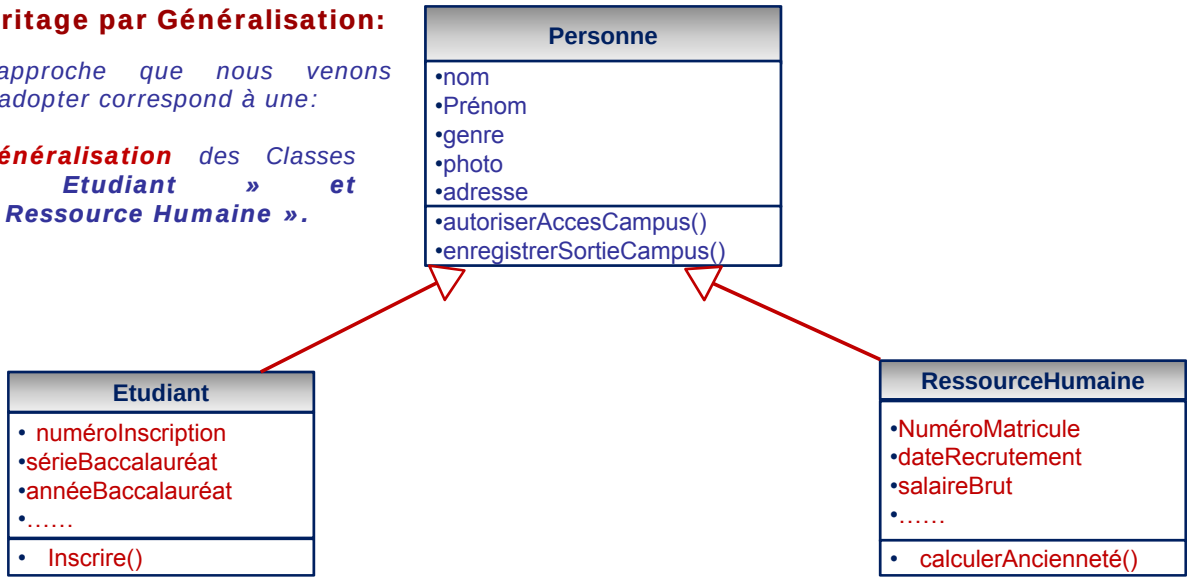


Relations: Héritage

Héritage par Généralisation:

L'approche que nous venons d'adopter correspond à une:

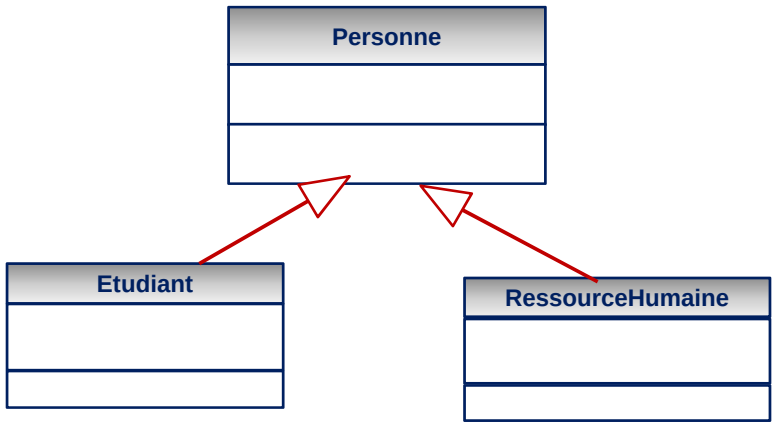
***Généralisation** des Classes « **Etudiant** » **et** « **Ressource Humaine** ».*



Relations: Héritage

Constat:

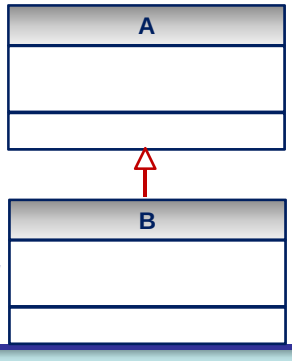
Les deux démarches aboutissent au même Modèle.



Relations: Héritage

Propriétés:

- ✓ *Une instance (Objet) de la classe B est aussi une instance (Objet) de la classe A.*
- ✓ *Les instances (Objets) de la classe B partagent les mêmes attributs et méthodes que celles de des instances (Objets) de la classe A.*
- ✓ *La classe B est une **spécialisation** de la classe A.*
- ✓ *La classe A est une **généralisation** de B.*



- ✓ *La classe A est une **Super Classe**.*
- ✓ *La classe A est une Classe **Parent**.*
- ✓ *La classe A est une Classe **Mère**.*
- ✓ *La classe B est une **Sous Classe**.*
- ✓ *La classe B est une Classe **Enfant**.*
- ✓ *La classe B est une Classe **Fille**.*

Relations: Héritage

Exercices:

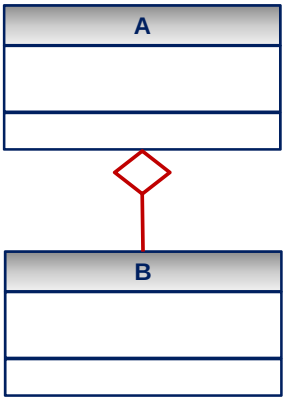
- ✓ *Modéliser le parc de véhicules d'une entreprise de location de moyens de transport.*
- ✓ *Modéliser les modes de paiement*
- ✓ *Modéliser les objets d'un Bibliothèque graphique.*
- ✓ *Modéliser le fonds documentaire d 'une Bibliothèque.*
- ✓ *Modéliser les messages échangés via un réseau social.*

Relations : Contenance (Agrégation/Composition)

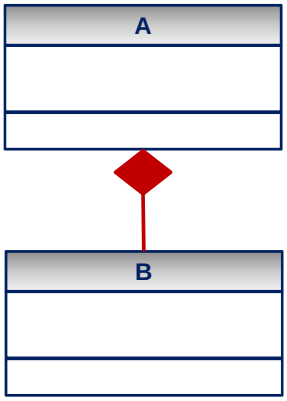
Présentation:

UML permet de représenter la relation de contenance moyennant deux types de relations:

- ✓ **Agrégation.**



- ✓ **Composition**



Relations : Contenance (Agrégation/Composition)

Présentation:

Ce type de relation exprime le fait qu'une Classe A **Contient** une autre classe B

Exemples:

- ✓ Une voiture contient (possède, est composée de):
  - Roues
  - Moteur
  - Carrosserie
  - Plaque d'immatriculation.
- ✓ Un dessin contient un ensemble de figures géométriques
- ✓ Une présentation PowerPoint est composé de Slides
- ✓ Une équipe de foot ball est composée d'un ensemble de joueurs.
- ✓ Une classe (groupe) est composée d'un certain nombre d'étudiants.

Relations : Contenance (Agrégation/Composition)

Agrégation:

L'agrégation exprime une relation de type « **avoir un** , **Possède**», une instance de A (un objet de A) peut « avoir un » une instance de B (un objet de B).

- les objets de B peuvent exister indépendamment des objets de A.
- La suppression d'une instance de A n'entraine pas nécessairement la suppression des instances de B qui lui sont rattachés.
- La même instance de B peut faire partie de (être commune à) plusieurs instances de A.

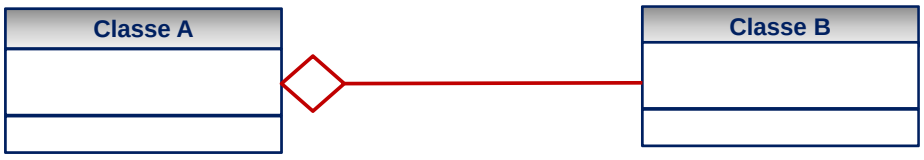
Exemple: Quel type de relation existe-t-il entre:

- Un étudiant et sa classe.
- Une voiture, son moteur et sa carrosserie.
- Un joueur et son équipe.

Relations : Contenance (Agrégation/Composition)

Représentation UML:

✓ Une instance de la classe « A » Contient des instances de la classe « B ».

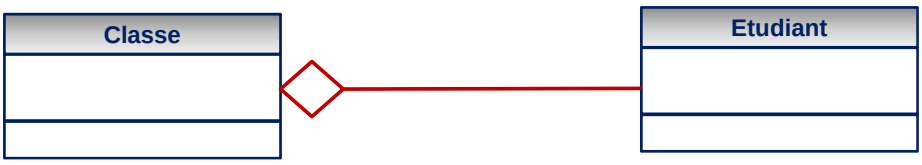


Conteneur Agrégé Ensemble Contenu Agrégat Elément

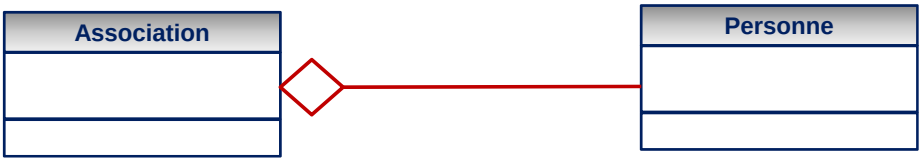
Relations : Contenance (Agrégation/Composition)

Exemples:

✓ Une classe (groupe) a un certain nombre d'étudiants.



✓ Une association est composée de personnes (ses membres).



Relations : Contenance (Agrégation/Composition)

Composition:

La composition exprime une relation de type « **Faire partie de** », une instance de la classe B (un objet de la classe B) « Fait partie » d'une instance de la classe A (un objet de la classe A).

- les objets de B ne peuvent exister indépendamment des objets de A.
- La suppression d'une instance de A entraine la suppression des instances de B qui lui sont rattachés.
- La même instance de B ne peut faire partie de (être commune à) plusieurs instances de A.

Exemples: Quel type de relation existe-t-il entre:

- Une Voiture et sa plaque d'immatriculation
- Une voiture, son moteur et sa carrosserie.

Relations: Association

Présentation:

Les Objets du système peuvent avoir liens exprimant l'existence d'une certaine relation.

Soit deux objets du système reliés par une certaine relation

Exemple:

Dans le domaine « Scolarité » on observe les faits suivants:

L'étudiant Karim est inscrit dans la filière Ingénierie

Quels sont les objets mentionnés dans ce fait et quel lien existe-t-il entre eux ?

Relations: Association

Présentation:

Dans le domaine « Scolarité » on observe d'autres faits similaires au précédents:

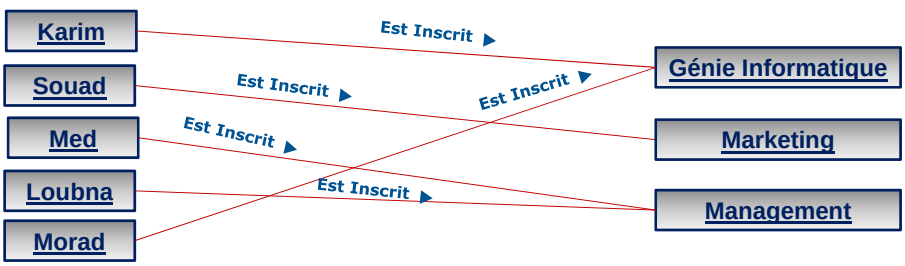
L'étudiante Souad est inscrite dans la filière Marketing

L'étudiant Mohammed est inscrite dans la filière Management

L'étudiante Loubna est inscrite dans la filière Management

L'étudiant Morad est inscrit dans la filière Génie Informatique

Représentons l'ensemble de ces faits:



Relations: Association

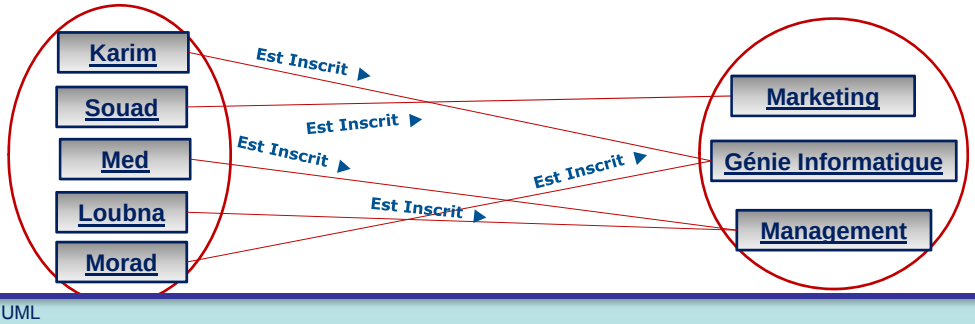
Présentation:

L'analyse de ce schéma conduit au constat suivant:

Les objets: { Karim, Souad, Med Loubna, Morad } sont semblables et peuvent être regroupés.

Les objets: { Génie Informatique, Marketing, Management }

sont également semblables et peuvent être regroupés.

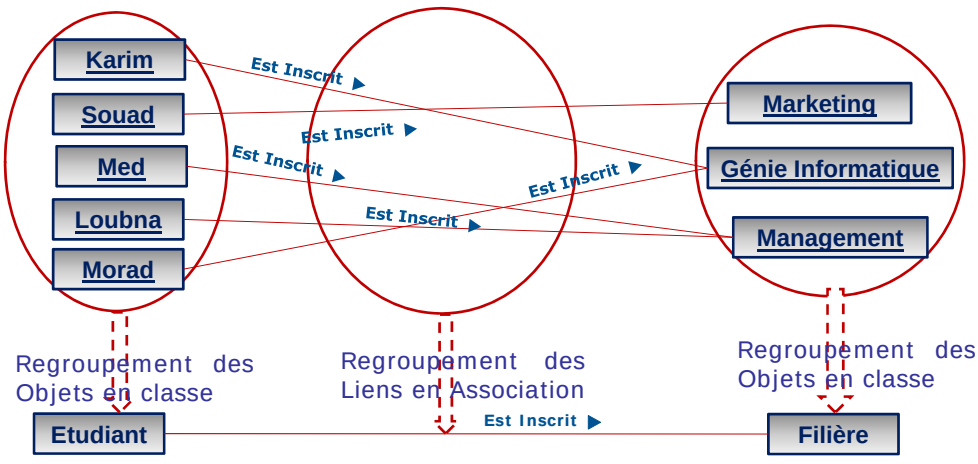


Relations: Association

Présentation:

On peut faire le même constat au niveau des liens entre les objets:

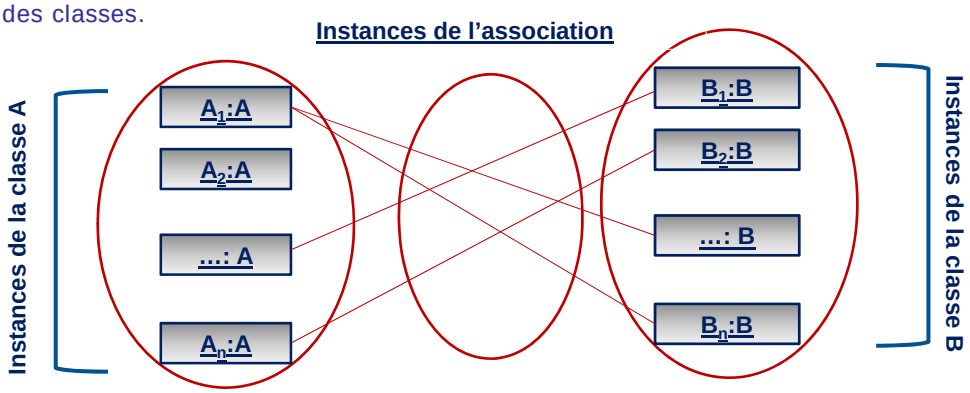
Tous les liens sont semblables et décrivent le même type de relation



Relations: Association

Présentation:

D'une manière plus générale: Une association est un ensemble de liens semblables reliant des objets, instances des classes.



Représentation ensembliste d'une association

Relations: Association

Arité des Associations: Association binaire.

Les exemples vus concernent les associations dites binaires.

Une association binaire est une association dont chaque instance relie deux objets différents.

Représentation d'une association binaire (Arité = 2)

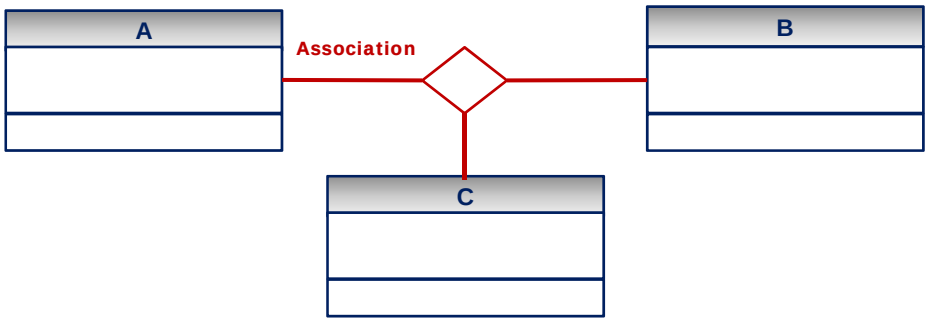
L'arité d'une association est **le nombre d'instances d'objets différents** reliés par chaque instance de l'association.

L'arité d'une association **n'est pas toujours égal au nombre de classes qu'elle relie.**

Relations: Association

Association ternaire:

Chaque instance de l'association relie trois objets (instances)



Représentation d'une association ternaire (reliant trois classes)

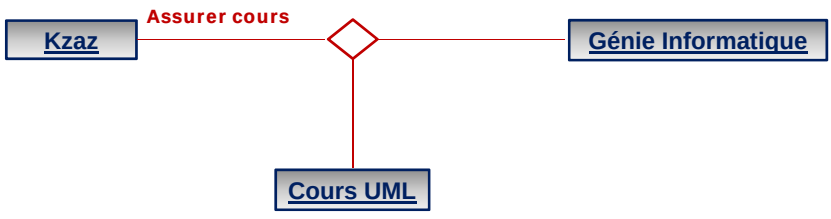
Relations: Association

Association ternaire:

Dans le domaine planification des cours on relève le fait suivant:

Le Professeur Kzaz assure le cours d'UML pour la filière Génie Informatque

Quels sont les objets mentionnés dans ce fait et quel lien existe-t-il entre eux

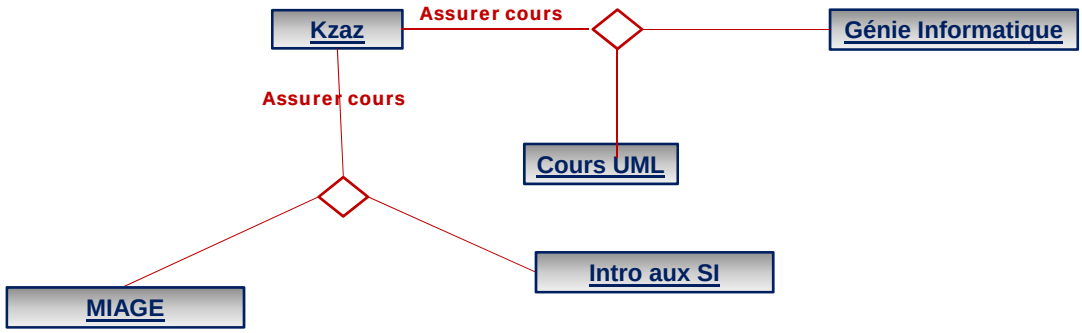


Relations: Association

Association ternaire:

Exemple; soit le fait suivant:

Le Professeur Kzaz assure aussi le cours Intro aux SI pour la filière MIAGE

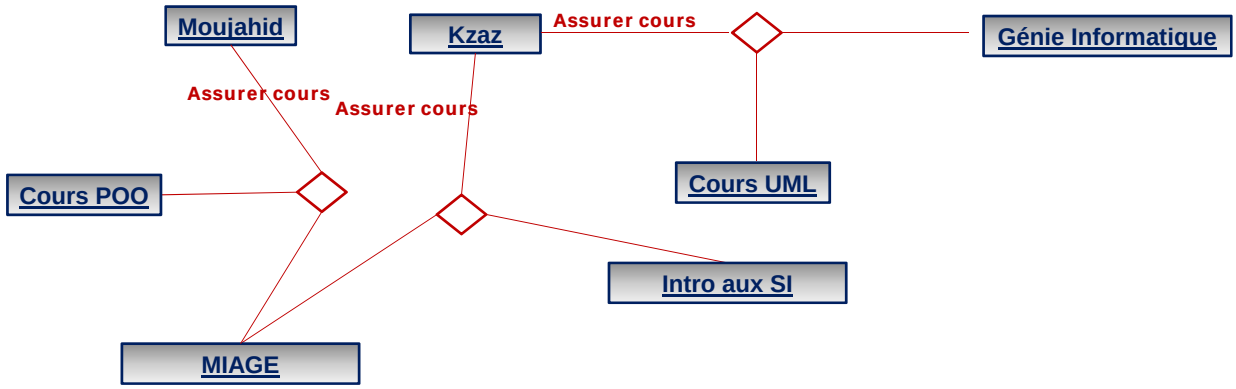


Relations: Association

Association ternaire:

Un autre fait similaire:

Le Professeur Moujahid assure le cours Programmation Orientée de la filière MIAGE



Relations: Association

Association ternaire:

Le diagramme précédent met en relation des objets qu'on peut regrouper en 3 sous- ensembles:

Les objets: { Kzaz, Moujahid } Sont semblables; ils peuvent être regroupés en une seule classe: Professeur

Les objets: { Génie Informatique, MIAGE } Sont semblables; ils peuvent être regroupés en une seule classe: Filière

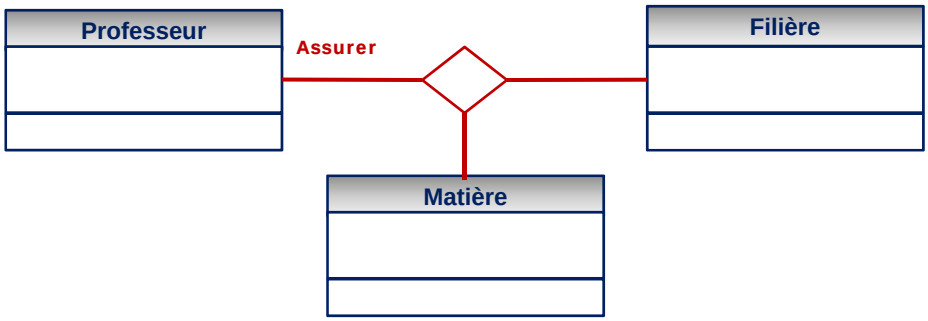
Les objets: { UML, POO } Sont semblables; ils peuvent être regroupés en une seule classe: Matière

Les objets précédents sont reliés par trois liens similaires: « Assurer Cours »; ils peuvent être regroupés en une seule association.

Relations: Association

Association ternaire:

Les diagrammes précédents donnent lieu à un seul diagramme des classes:

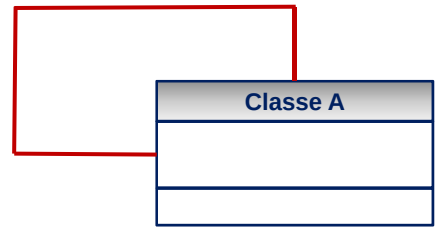


Représentation d'une association ternaire (reliant trois classes)

Relations: Association

Association réflexive:

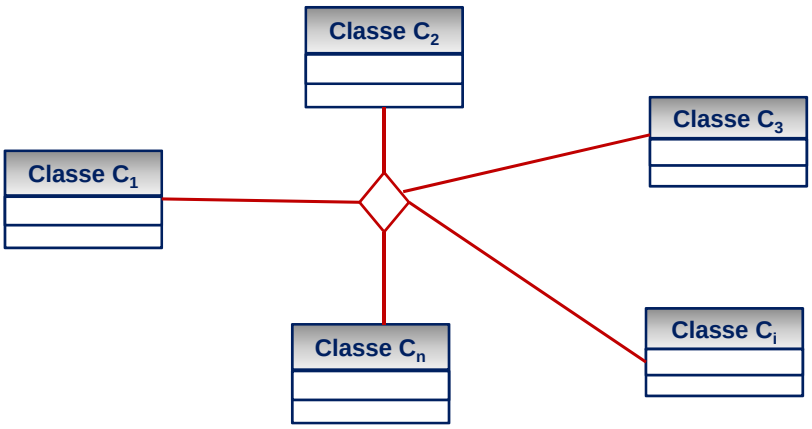
Une association réflexive relie des objets différents d'une même classe.



Deux instances de la classe A sont reliées

Relations: Association

Association n-aire:



Relations: Association

Exemples:

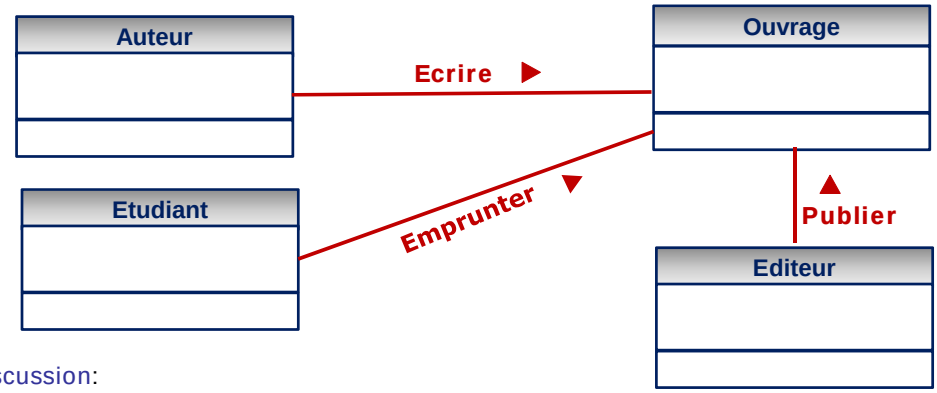
Modéliser les phrases suivantes:

- ✓ Un auteur écrit des ouvrages.
- ✓ Un éditeur publie des ouvrages.
- ✓ Un étudiant emprunte un ouvrage.
- ✓ Un professeur assure une matière pour une classe.
- ✓ Une séance de cours est assurée par un professeur pour une classe d'étudiants dans une salle pendant un certain horaire.
- ✓ Une personne propriétaire loue un appartement à une autre personne locataire.

Relations: Association

Exemples:

- ✓ Un auteur écrit des ouvrages
- ✓ Un étudiant emprunte un ouvrage
- ✓ Un éditeur publie des ouvrages.



- ✓ Discussion:

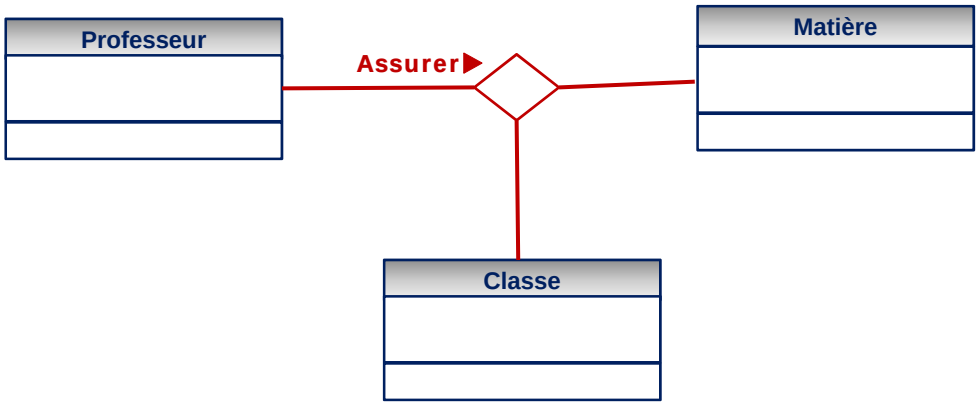
Editeur: attribut de la classe Ouvrage!

Auteur: attribut de la classe Ouvrage!

Relations: Association

Exemples:

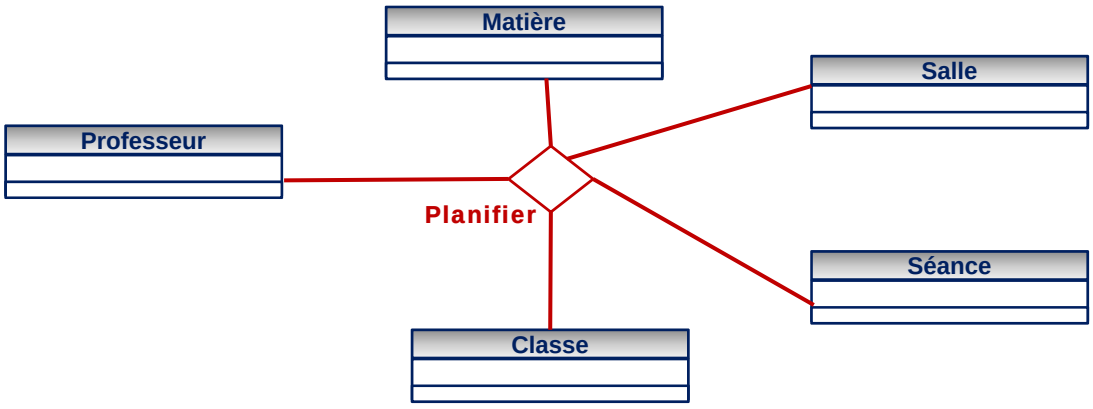
✓ Un Professeur assure une matière pour une classe



Relations: Association

Exemples:

✓ Une séance de cours est assurée par un professeur pour une classe d'étudiants dans une salle pendant un certain horaire.



Relations: Classe-Association

Exemple introductif:

✓ Un étudiant emprunte un ouvrage auprès de la bibliothèque. Lors de l'emprunt, les dates d'emprunts et de restitution sont fixées.

Soit le diagramme des classes suivant:

Placer les attributs suivants sur ce diagramme: – dateEmprunt – dateRestitution

✓ Ces deux attributs ne caractérisent ni l'étudiant ni l'ouvrage.

✓ Ils caractérisent l'emprunt. Ils ne sont définis qu'au moment de l'emprunt.

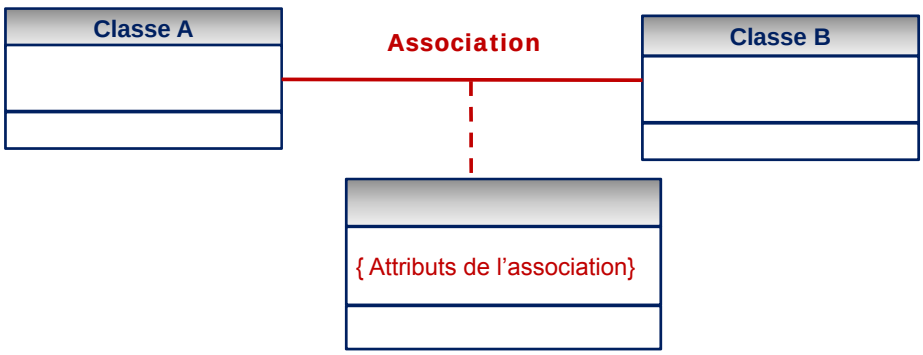
✓ Ce sont donc: des attributs de l'association « Emprunter »

des attributs de l'association « Emprunter »

Relations: Classe-Association

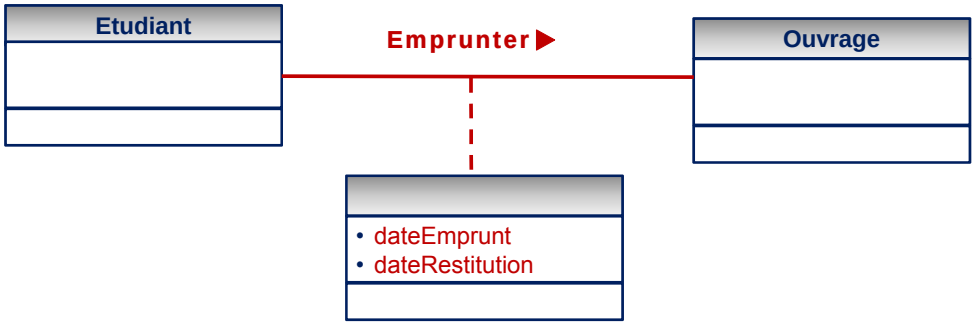
✓ Une classe-Association est une association porteuse d'attributs.

✓ Représentation des classes association.



Relations: Classe-Association

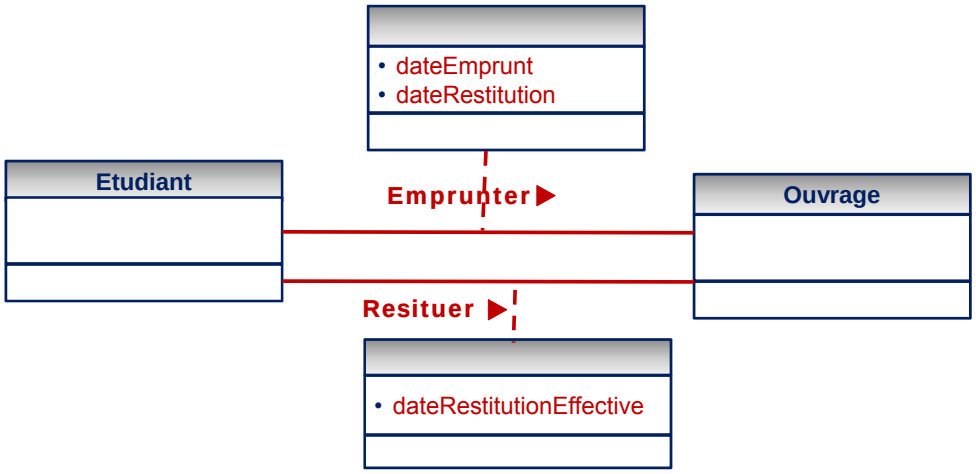
✓ Exemple:



Relations: Classe-Association

✓ Compléter le modèle pour prendre en compte la date de restitution effective de l'ouvrage.

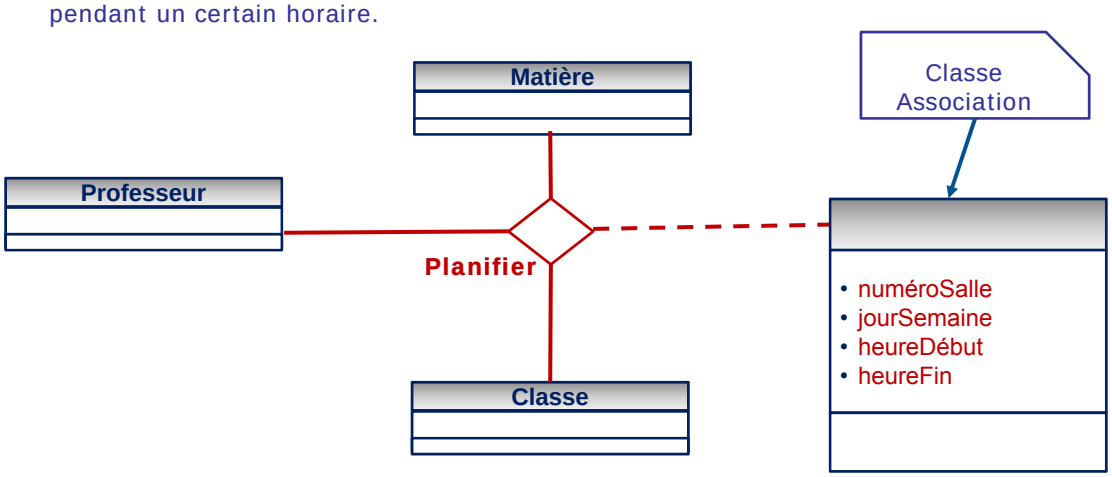
✓ La date de restitution effective peut être différente de la date de restitution réglementaire.



Relations: Classe-Association

Exemple:

✓ Une séance de cours est assurée par un professeur pour une classe d'étudiants dans une salle pendant un certain horaire.

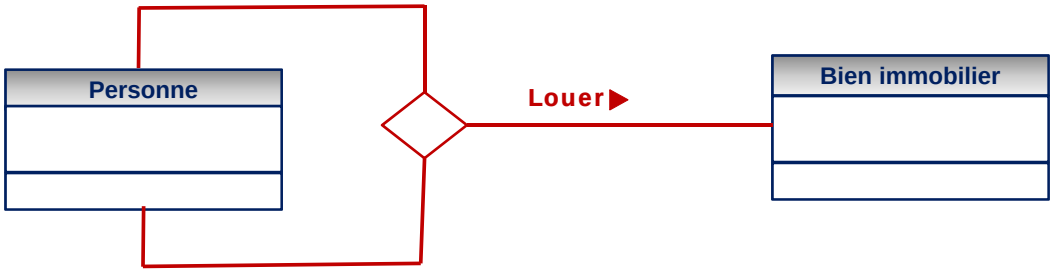


Relations: Classe-Association

Exemple:

✓ On considère le SI d'une agence immobilière. Elle reçoit de la part des propriétaires des biens à louer, et reçoit de la part des locataires des demandes de location. On souhaite modéliser le fait suivant:

Une personne, propriétaire d'un bien, loue un bien immobilier, à une autre personne, qui devient le locataire du bien.

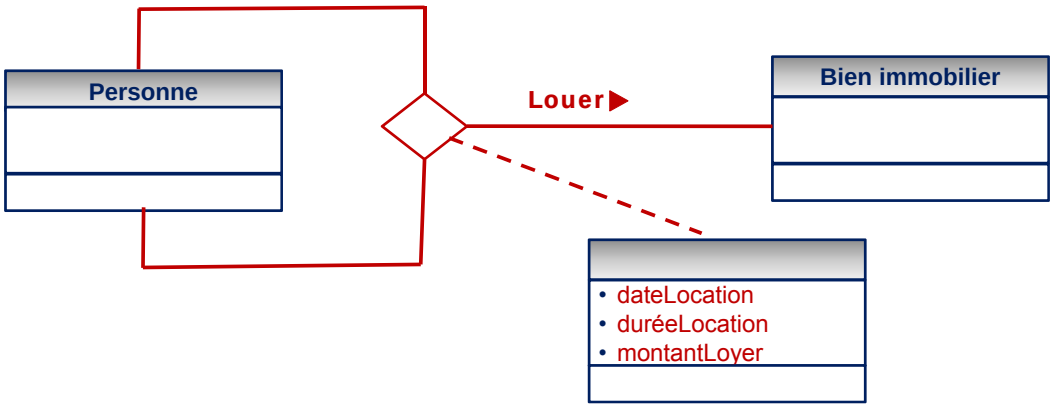


Relations: Classe-Association

Exemple:

✓ On souhaite enrichir le modèle précédent par les attributs suivants:

Date Début de location, Durée de la location, et montant du loyer.



Classe Abstraite

Définition:

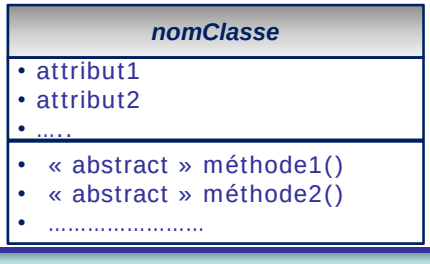
✓ Une **classe abstraite** est une classe dont l'implémentation n'est pas complète et qui ne peut être instanciée. Une classe abstraite contient des **méthodes abstraites**.
Une **méthode est dite abstraite** lorsqu'on connaît son entête, mais pas la manière dont elle peut être réalisée (i.e. on connaît sa déclaration, mais pas sa définition).

Une classe abstraite sert essentiellement à factoriser des méthodes et attributs communs à plusieurs classes, et ce dans une relation d'héritage

Une classe abstraite sert de base pour la définition d'autres classes, les sous classes.

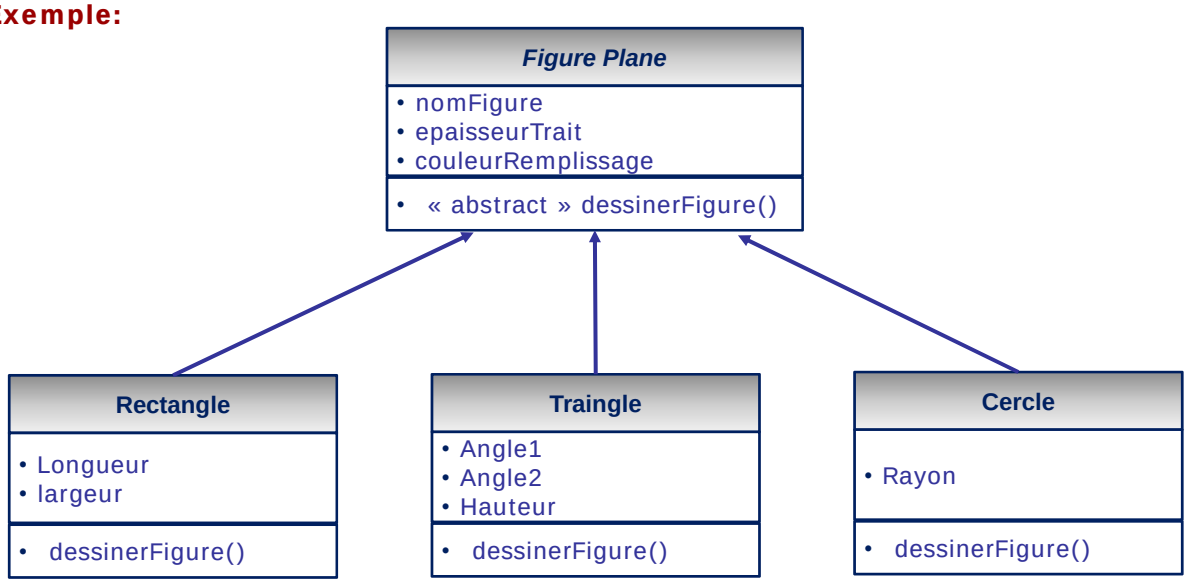
Notation: le nom d'une classe abstraite est écrit en italique

Les méthodes abstraites sont précédées du stéréotype « abstract ».



Classe Abstraite

Exemple:



Classe Interface

Définition:

✓ Une **classe interface** est une classe complètement abstraite.

Une **classe interface** ne contient pas d'attributs; elle ne contient que des méthodes abstraites, c'est à dire des méthodes non implémentées dans la classe.

Les classes interfaces sont définissent des ensembles d'opérations que d'autres classes doivent implémenter.

Une classe interface sert essentiellement à factoriser des méthodes communes à plusieurs classes, et ce dans une relation d'héritage.

Une classe interface permet de structurer et simplifier les modèles.

Notation: le nom d'une classe interface est précédé du mot « Interface ».