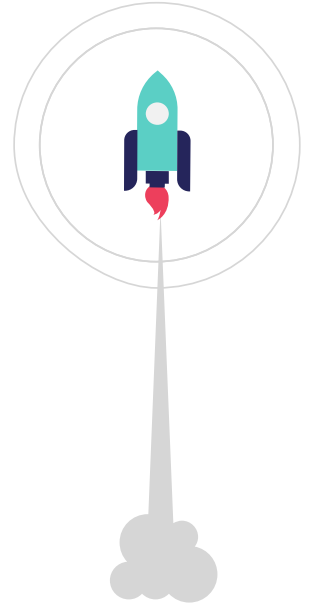




UML




Par Mariam BENLLARCH

La Notion UML



Introduction au langage de modélisation UML

UML:

U		Unified	
M		Modeling	=
L		Language	

Langage de
modélisation
unifié

Introduction au langage de modélisation UML

UML:

U → **Unified**

M → **Modeling**

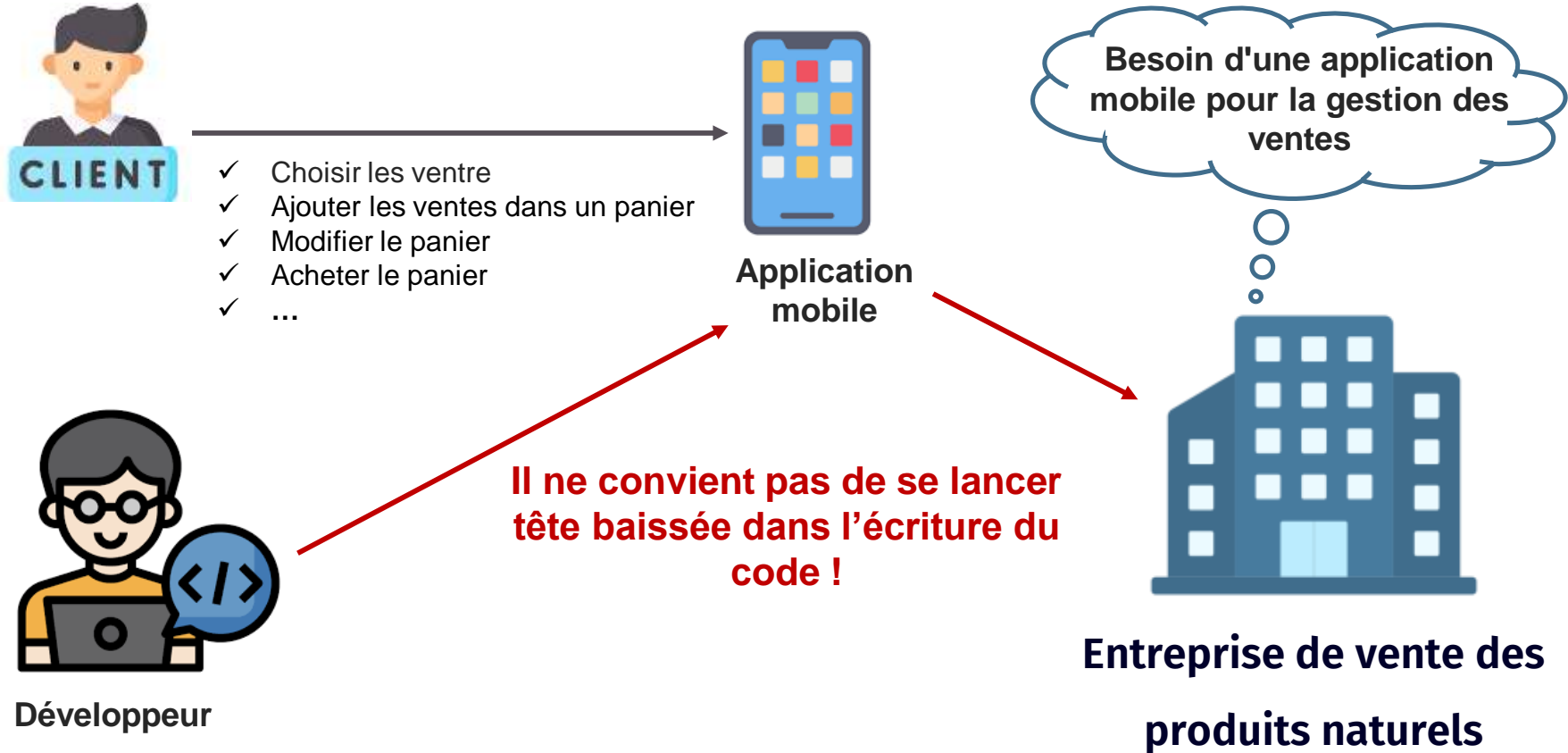
L → **Language**

Langage de

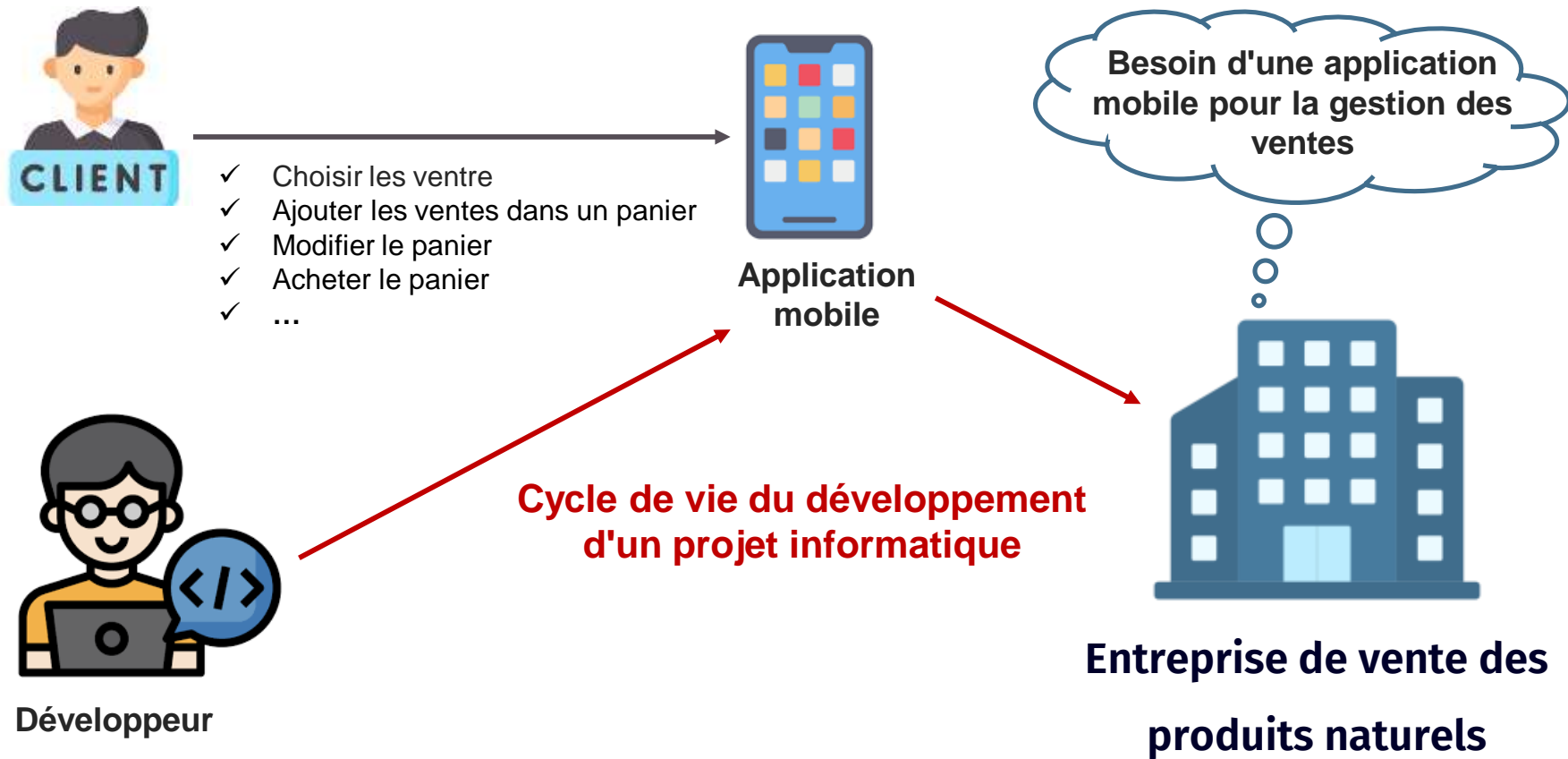
= **modélisation**

unifié

C'est quoi la modélisation ?



C'est quoi la modélisation ?



C'est quoi la modélisation ?

Développeur



Application mobile



Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** Etude de l'existant, expression des besoins, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme des modèles.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

C'est quoi la modélisation ?

Développeur



Application mobile



Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** **Etude de l'existant**, expression des besoins, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme des modèles.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

C'est quoi la modélisation ?

Existant dans le système de l'entreprise



- ✓ Les produits et les types de produits
- ✓ Les matériels informatiques, Les logiciels, et les réseaux
- ✓ La gestion des ventes
- ✓ ...

C'est quoi la modélisation ?

Développeur



Application mobile



Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** Etude de l'existant, **expression des besoins**, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme des modèles.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

C'est quoi la modélisation ?

Besoin des clients



- ✓ Choisir les ventes
- ✓ Ajouter les ventes dans un panier
- ✓ Modifier le panier
- ✓ Acheter le panier
- ✓ ...

Besoin de l'entreprise



- ✓ La méthode de paiement
- ✓ L'organisation des produits dans l'application
- ✓ ...

C'est quoi la modélisation ?

Développeur



Application mobile



Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** Etude de l'existant, expression des besoins, **définition des limites.**
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme des modèles.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

C'est quoi la modélisation ?

Les limites de l'entreprise



- ✓ Limites de ressource humaines
- ✓ Limites du budget
- ✓ Limites technologiques
- ✓ ...

C'est quoi la modélisation ?

Développeur



Application mobile



Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** Etude de l'existant, expression des besoins, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme des modèles.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

C'est quoi la modélisation ?

Développeur



Application mobile

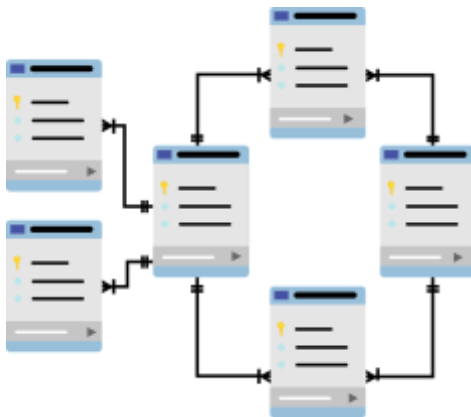


Cycle de vie du développement d'un projet informatique :

- ✓ **Analyse:** Etude de l'existant, expression des besoins, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme **des modèles**.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

C'est quoi la modélisation ?

Modèle



Un modèle est une simplification de la réalité qui permet de mieux comprendre le système à développer (ex: un plan, une carte, un schéma électronique ...).

C'est quoi la modélisation ?

Développeur



Application mobile



Cycle de vie du développement d'un projet informatique :




- ✓ **Analyse:** Etude de l'existant, expression des besoins, définition des limites.
- ✓ **Modélisation:** Création des solutions pour les besoins d'un projet informatique. Cette création est présentée sous forme **des modèles**.
- ✓ **Programmation:** Implémenter une base de données, choisir le SGBD, choisir le langage de programmation.
- ✓ **Test:** Des simulations, correction des erreurs.

Pourquoi modéliser ?

- ✓ Pour visualiser le système comme il est ou comme il devrait l'être.
- ✓ Pour valider le modèle vis à vis des clients.
- ✓ Pour spécifier les structures de données et le comportement du système.
- ✓ Pour fournir un guide pour la construction du système.
- ✓ Pour documenter le système et les décisions prises.

Introduction au langage de modélisation UML




UML:

U		Unified	
M		Modeling	=
L		Language	

Langage de
modélisation
unifié

Introduction au langage de modélisation UML

UML:

U		Unified	
M		Modeling	=
L		Language	

Langage de modélisation unifié

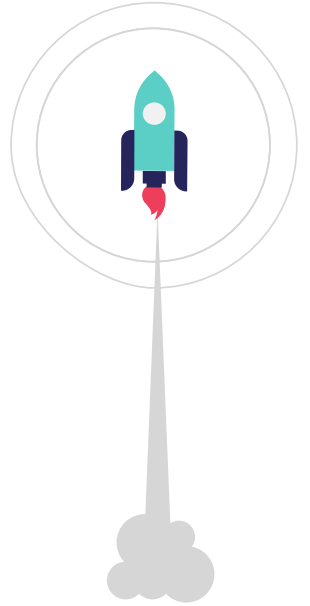
UML est un langage

- ✓ UML n'est pas une méthode,
- ✓ UML est un langage de modélisation objet,
- ✓ UML est dans le domaine public, c'est une norme.

UML un langage pour

- ✓ visualiser: chaque symbole graphique a une sémantique,
- ✓ spécifier: de manière précise et complète,
- ✓ construire: les classes, les relations SQL peuvent être générées automatiquement,
- ✓ documenter: les différents diagrammes, notes, contraintes, exigences seront présentés dans un document.

Comment modéliser avec UML ?



L' utilisation des diagrammes

- ✓ UML permet de définir et de visualiser un modèle, à l'aide de **diagrammes**.
- ✓ **Un diagramme** UML est une représentation graphique, qui s'intéresse à un aspect précis du modèle.
- ✓ Chaque type de diagramme UML possède une structure (les types des éléments de modélisation qui le composent sont prédéfinis).
- ✓ Un type de diagramme UML offre toujours la même vue d'un système (il véhicule une sémantique précise).
- ✓ Combinés, les différents types de diagrammes UML offrent une vue complète des aspects statiques et dynamiques d'un système.

Les différents types de diagrammes UML

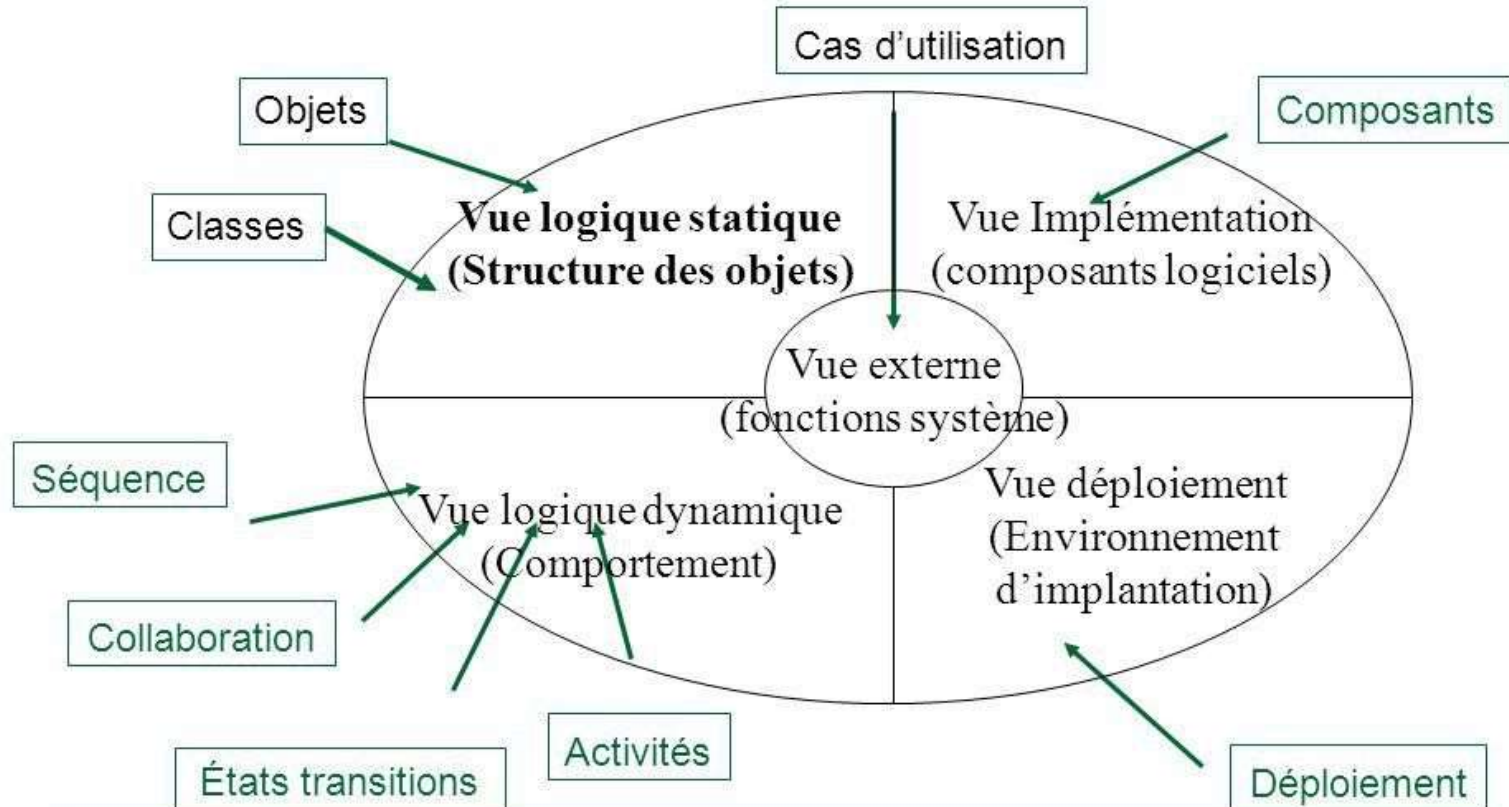
✓ **Vision statique du système :**

- ✓ diagramme de cas d'utilisation
- ✓ diagramme d'objets
- ✓ diagramme de classes
- ✓ diagramme de composants
- ✓ diagramme de déploiement

✓ **Vision dynamique du système :**

- ✓ diagramme de collaboration
- ✓ diagramme de séquence
- ✓ diagramme d'états-transitions
- ✓ diagramme d'activités

Diagrammes et vues d'architecture UML



Vision statique du système :

Diagramme de cas d'utilisation

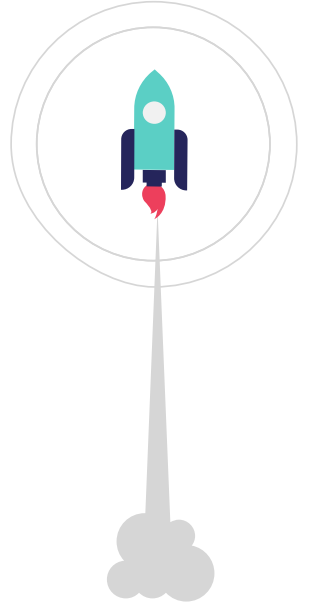


Diagramme de Cas d'Utilisation

Un diagramme de cas d'utilisation:

- Décrit:

- Les acteurs



- Les cas d'utilisation



- Le système



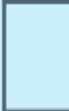
Diagramme de Cas d'Utilisation

Un diagramme de cas d'utilisation:

- Décrit:

- **Les acteurs** 

- Les cas d'utilisation 

- Le système 

Acteur

- Un acteur est une entité extérieure au système modélisé, et qui interagit directement avec lui.
- Un acteur est présenté par un petit bonhomme:



Nom
Acteur

Acteur

- Un acteur est une entité extérieure au système modélisé, et qui interagit directement avec lui.
- Un acteur est présenté par un petit bonhomme.
- Les principaux acteurs sont les utilisateurs du système.
- En plus des utilisateurs, les acteurs peuvent être :
 - Des logiciels déjà disponibles à intégrer dans le projet ;
 - Des systèmes informatiques externes au système mais qui interagissent avec lui ;
 - tout élément extérieur au système et avec lequel il interagit.



**Nom
Acteur**

Acteur

Rôles et personnes physiques

- Un acteur correspond à un rôle, pas à une personne physique :
- Une même personne physique peut être représentée par plusieurs acteurs si elle a plusieurs rôles.
- Si plusieurs personnes jouent le même rôle vis-à-vis du système, elles seront représentées par un seul acteur.



**Nom
Acteur**

Acteur

Relations entre acteurs

- Il n'y a qu'un seul type de relation possible entre acteurs : la relation de généralisation.

Exemple:

Un directeur est une sorte de commercial : il peut faire avec le système tout ce que peut faire un commercial, plus d'autres choses

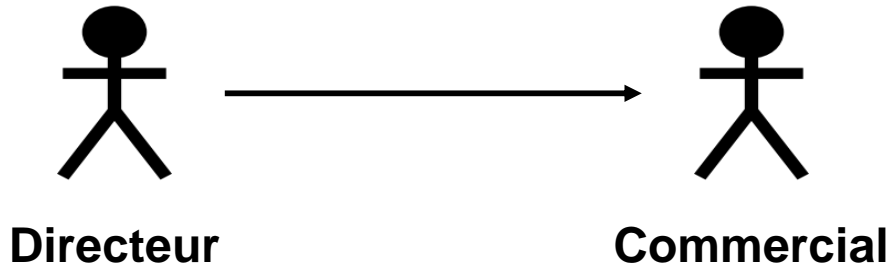


Diagramme de Cas d'Utilisation

Un diagramme de cas d'utilisation:

- Décrit:

- Les acteurs



- **Les cas d'utilisation**



- Le système



Cas d'utilisation

- Un cas d'utilisation est un moyen de représenter les différentes possibilités d'utiliser un système.
- Il exprime toujours une suite d'interactions entre un acteur et l'application.
- Il définit une fonctionnalité utilisable par un acteur.

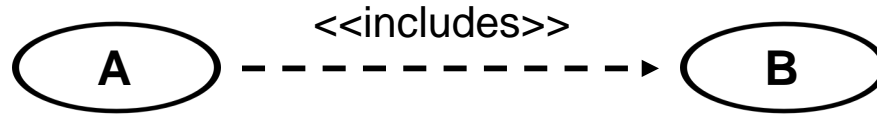


**Le cas
d'utilisation**

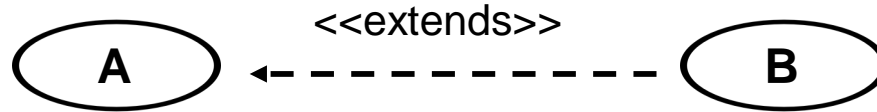
Cas d'utilisation

Relations entre les cas d'utilisations

1) **Inclusion** : B est une partie obligatoire de A et on lit A inclut B (dans le sens de la flèche).



2) **Extension** : B est une partie optionnelle de A et on lit B étend A (dans le sens de la flèche).



Cas d'utilisation

Relations entre les cas d'utilisations

- 3) **Généralisation** : le cas A est une généralisation du cas du cas B et on lit B est une sorte de A.

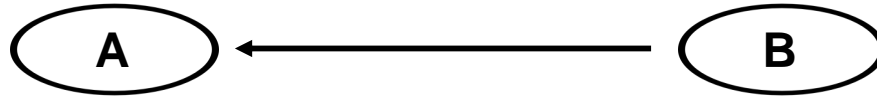


Diagramme de Cas d'Utilisation

Un diagramme de cas d'utilisation:

- Décrit:

- Les acteurs



- Les cas d'utilisation



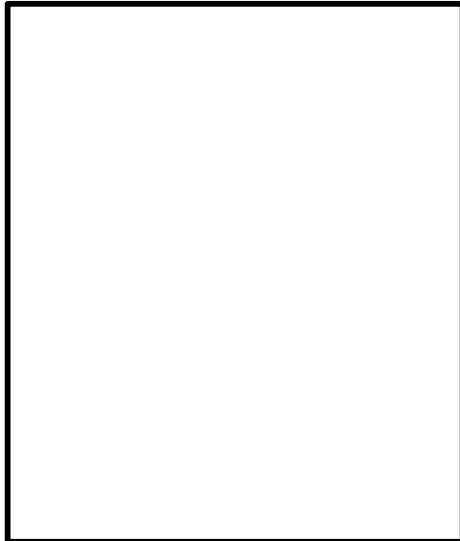
- **Le système**



Systeme

- Le système définit l'application informatique, il ne contient donc pas les acteurs, mais les cas d'utilisation et leur associations.

Nom du système



Nom du système

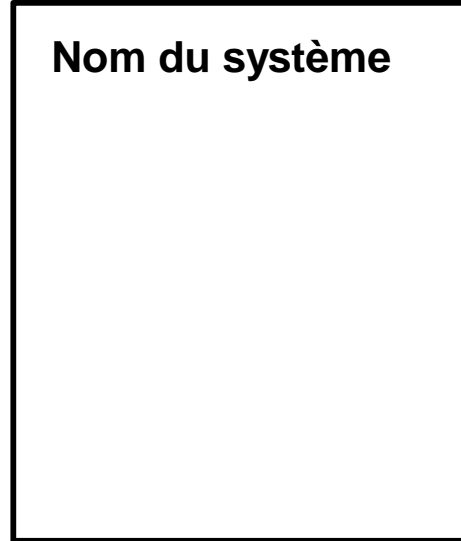
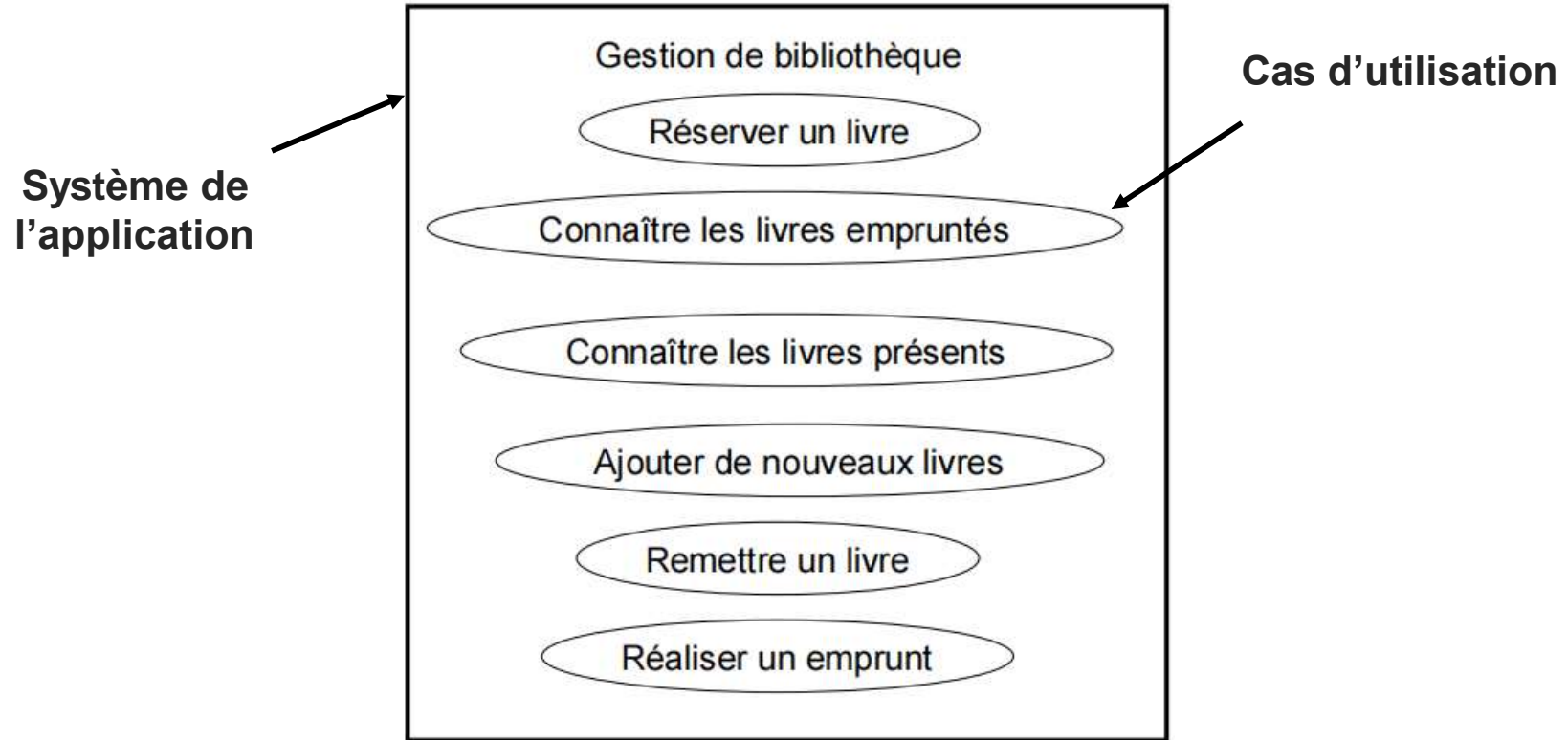


Diagramme de Cas d'Utilisation

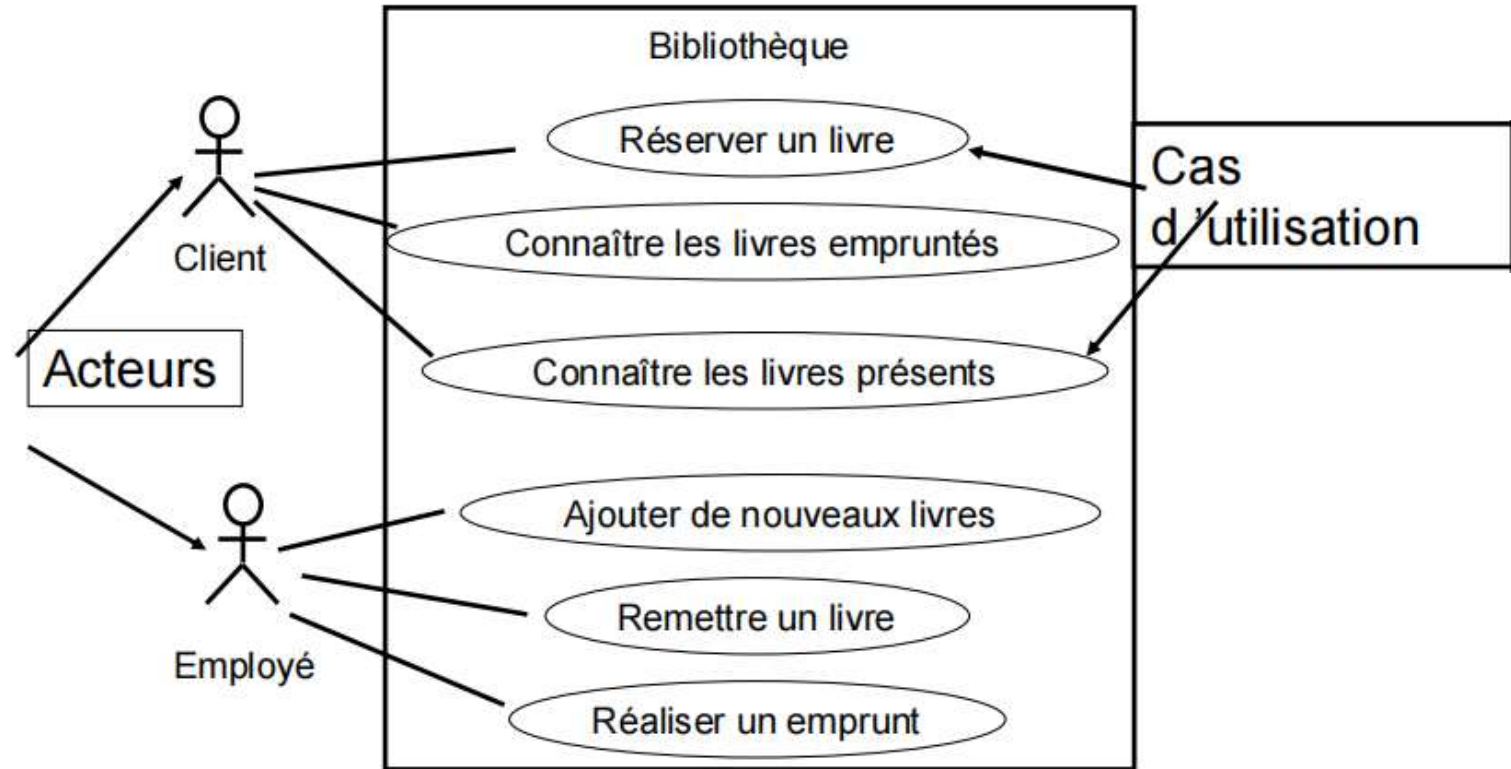
Exemple:

**Une application pour la gestion d'une
bibliothèque**

Une application pour la gestion d'une bibliothèque



Une application pour la gestion d'une bibliothèque



Une application pour la gestion d'une bibliothèque

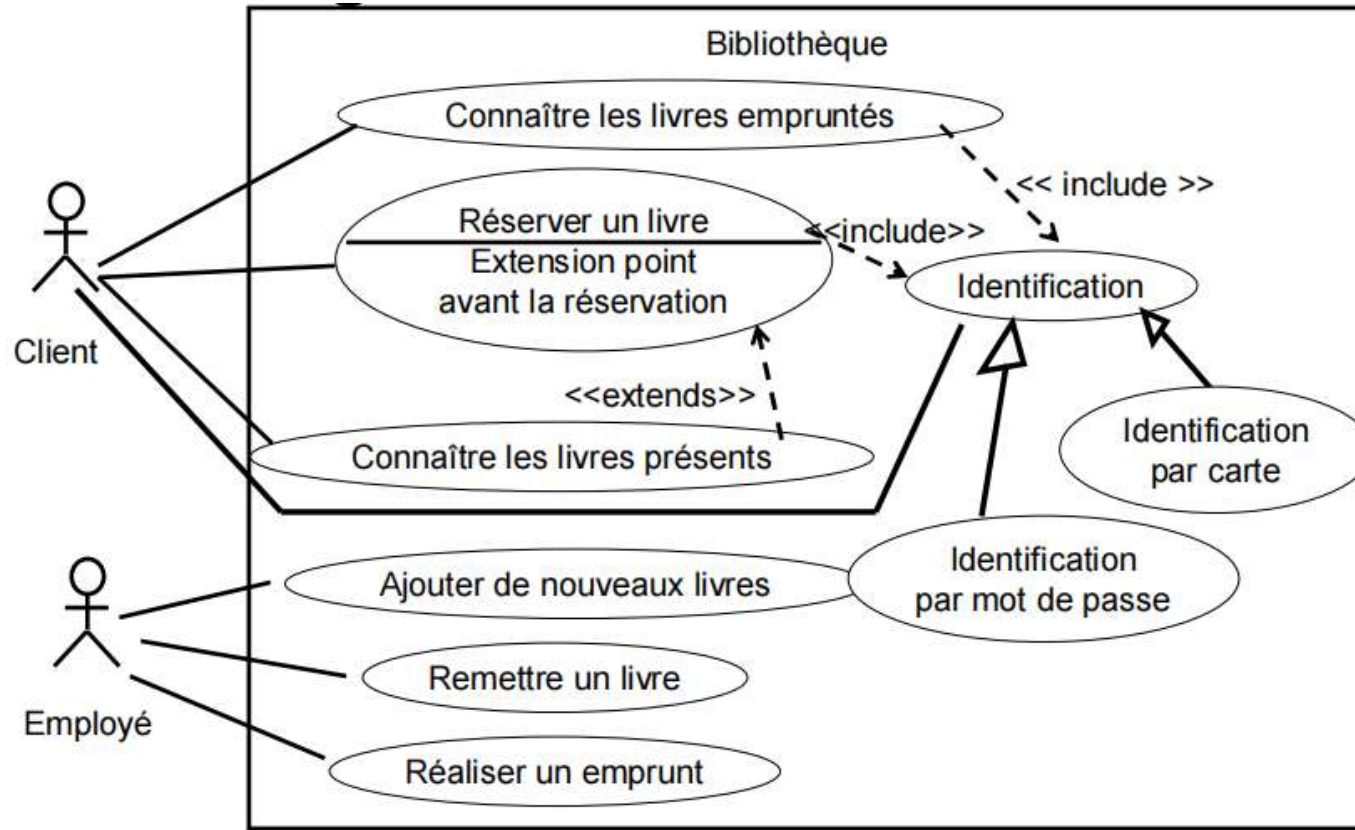


Diagramme de Cas d'Utilisation

Exercices

Exercice 1:

Dans un magasin, un commerçant dispose d'un système de gestion de son stock d'articles, dont les fonctionnalités sont les suivantes :

- Edition de la fiche d'un fournisseur,
- Possibilité d'ajouter un nouvel article (dans ce cas, la fiche fournisseur est automatiquement éditée. Si le fournisseur n'existe pas, on peut alors le créer),
- Edition de l'inventaire. Depuis cet écran, on a le choix d'imprimer l'inventaire, d'effacer un article ou d'éditer la fiche d'un article.

Modéliser cette situation par un diagramme de cas d'utilisation

Exercice 2:

Dans un établissement scolaire, on désire gérer la réservation des salles de cours ainsi que du matériel pédagogique (ordinateur portable ou/et Vidéo projecteur). Seuls les enseignants sont habilités à effectuer des réservations (sous réserve de disponibilité de la salle ou du matériel). Le planning des salles peut quant à lui être consulté par tout le monde (enseignants et étudiants). Par contre, le récapitulatif horaire par enseignant (calculé à partir du planning des salles) ne peut être consulté que par les enseignants. Enfin, il existe pour chaque formation un enseignant responsable qui seul peut éditer le récapitulatif horaire pour l'ensemble de la formation.

Modéliser cette situation par un diagramme de cas d'utilisation.

Vision statique du système :

Diagramme de classes

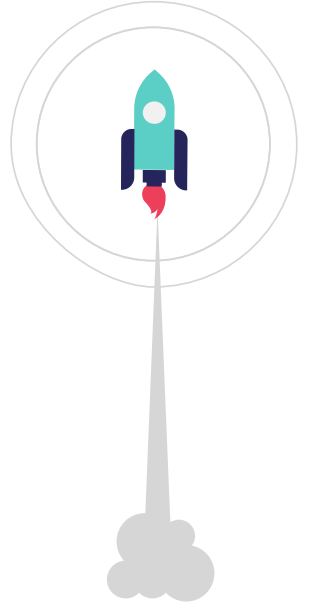


Diagramme de classes

Définition

- Le diagramme de classes exprime la structure statique du système en termes de classes et de relations entre ces classes.
- L'intérêt du diagramme de classe est de modéliser les entités du système d'information.
- Le diagramme de classe permet de représenter l'ensemble des informations finalisées qui sont gérées par le domaine. Ces informations sont structurées, c'est-à-dire qu'elles ont regroupées dans des classes.

Diagramme de classes

Le diagramme de classes comporte 5 concepts :

- ✓ Classe
- ✓ Attribut
- ✓ Identifiant
- ✓ Relation
- ✓ Opération

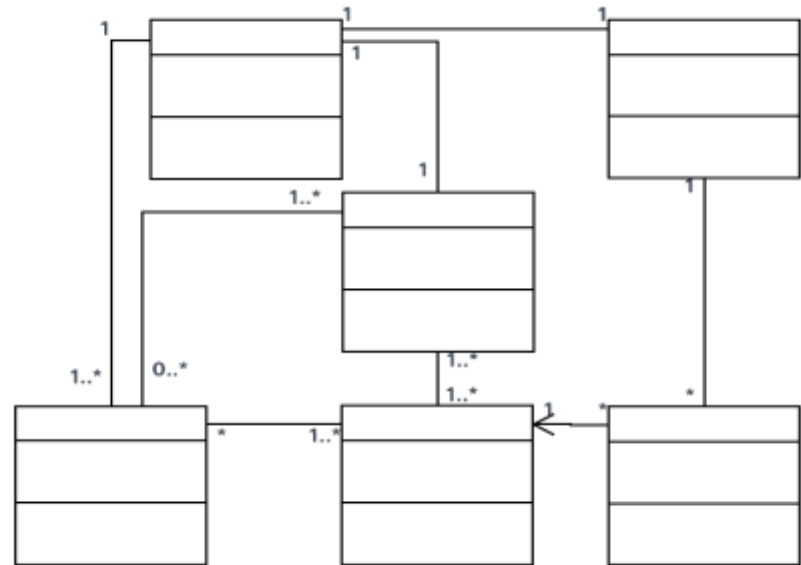
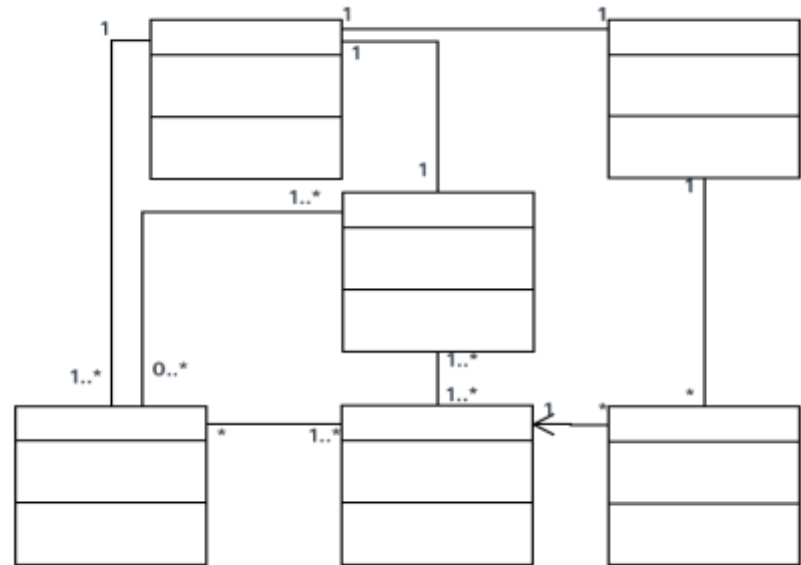


Diagramme de classes

Le diagramme de classes comporte 5 concepts :

- ✓ **Classe**
- ✓ Attribut
- ✓ Identifiant
- ✓ Relation
- ✓ Opération



Classe

Définition :

- une classe est une description d'un ensemble d'objets d'un système (un domaine d'application) : elle définit leur structure, leur comportement et leurs relations.
- Les objets peuvent être des objets concrets ou abstraits.

Exemple:

Une université est un système. Et dans l'université, il y a des objets qui font partie de son système :

Enseignant, étudiant, classe, cours, planning, etc ...

Les objets concrets sont: Enseignant et étudiant.

Les objets abstraits sont: classe, cours, planning.

Classe

Représentation :

les classes sont représentées par des rectangles compartimentés :

- le 1er compartiment représente le nom de la classe
- le 2ème compartiment représente les attributs de la classe
- le 3ème compartiment représente les opérations de la classe

Nom Classe
Attribut 1: int Attribut 2: int Attribut 3: int
Opération 1(): void Opération 2(): void

Classe

Représentation :

Les compartiments d'une classe peuvent être supprimés (en totalité ou en partie) lorsque leur contenu n'est pas pertinent dans le contexte d'un diagramme. La suppression des compartiments reste purement visuelle : elle ne signifie pas qu'il n'y a pas d'attribut ou d'opération.

Formalisme de représentation simplifiée :

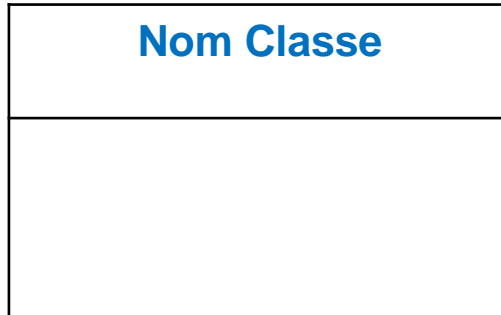
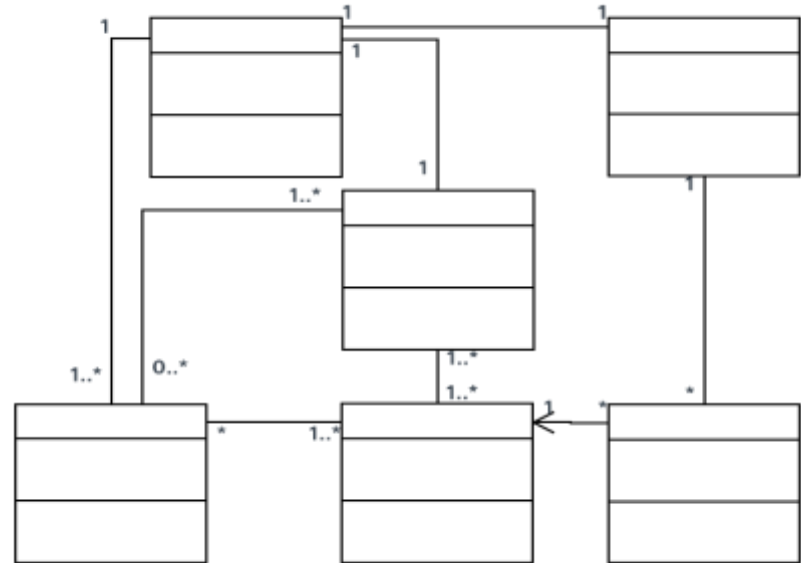


Diagramme de classes

Le diagramme de classes comporte 5 concepts :

- ✓ Classe
- ✓ **Attribut**
- ✓ Identifiant
- ✓ Relation
- ✓ Opération



La notion d'attribut

Une classe correspond à un concept global d'information et se compose d'un ensemble d'informations élémentaires, appelées attributs de classe.

Un attribut représente la modélisation d'une information élémentaire représentée par son nom et son format.

Nom Classe
Attribut 1: int Attribut 2: int Attribut 3: int
Opération 1(): void Opération 2(): void

La notion d'attribut

UML définit 3 niveaux de visibilité pour les attributs :

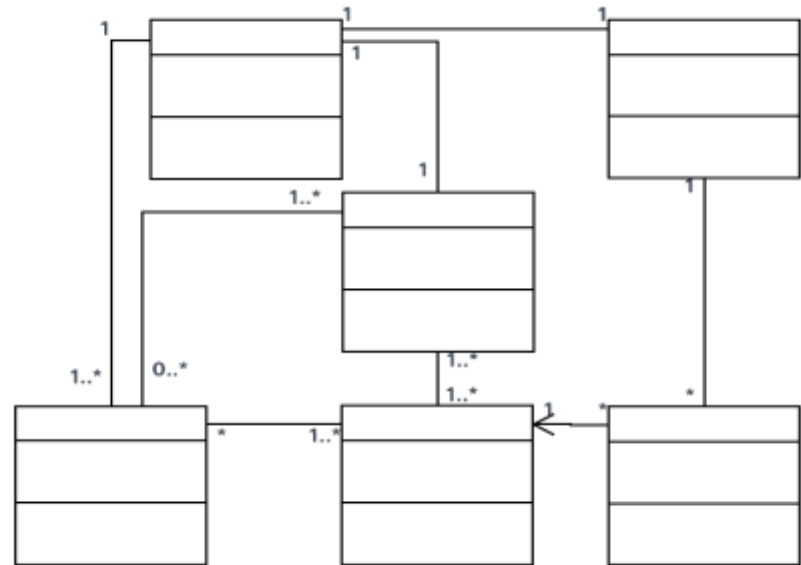
- public (+) : l'élément est visible pour tous les clients de la classe
- protégé (#) : l'élément est visible pour les sous-classes de la classe
- privé (-) : l'élément n'est visible que par les objets de la classe dans laquelle il est déclaré.

Nom Classe
+ Attribut public: int # Attribut protégé: int - Attribut privé: int

Diagramme de classes

Le diagramme de classes comporte 5 concepts :

- ✓ Classe
- ✓ Attribut
- ✓ **Identifiant**
- ✓ Relation
- ✓ Opération



La notion d'identifiant

L'identifiant est un attribut particulier, qui permet de repérer de façon unique chaque objet, instance de la classe.

Exemple:

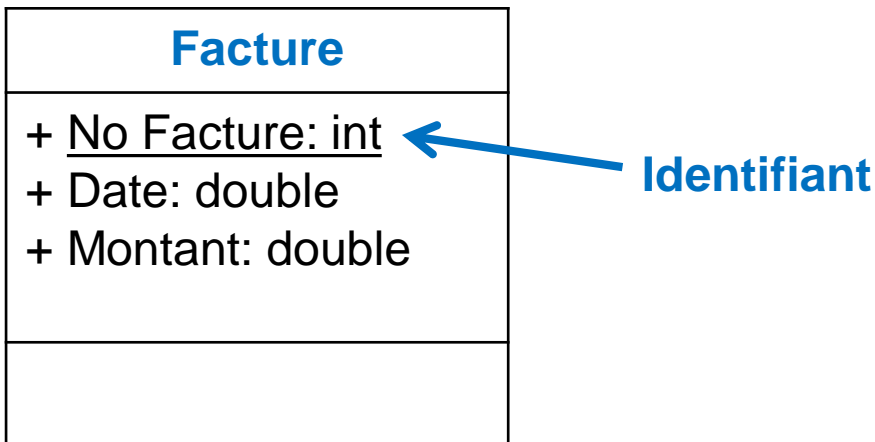
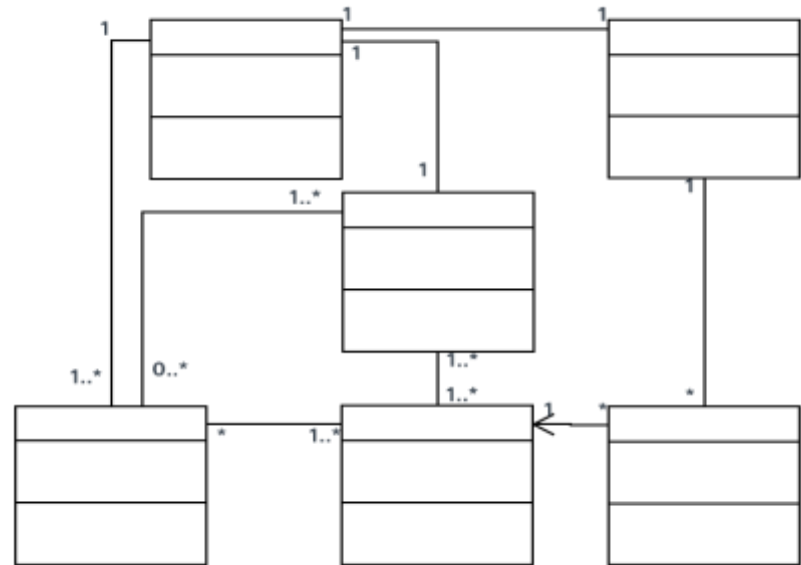


Diagramme de classes

Le diagramme de classes comporte 5 concepts :

- ✓ Classe
- ✓ Attribut
- ✓ Identifiant
- ✓ **Opération**
- ✓ Relation



La notion d'opération

l'opération représente un élément de comportement des objets, défini de manière globale dans la classe.

Une opération est une fonctionnalité assurée par une classe. La description des opérations peut préciser les paramètres d'entrée et de sortie ainsi que les actions élémentaires à exécuter.

Facture	
+ <u>No Facture</u> : int + Date: double + Montant: double	
Editer() :	void()
Créer() :	void()
Consulter() :	void()

La notion d'opération

Comme pour les attributs, on retrouve 3 niveaux de visibilité pour les opérations :

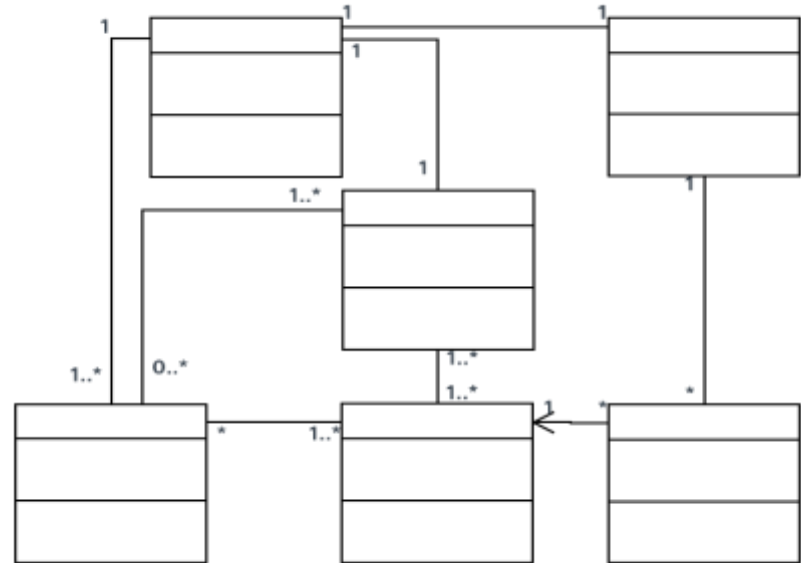
- **public (+)** : l'opération est visible pour tous les clients de la classe,
- **protégé (#)** : l'opération est visible pour les sous-classes de la classe,
- **privé (-)** : l'opération n'est visible que par les objets de la classe dans laquelle elle est déclarée.

Facture
<u>+ No Facture: int</u> + Date: double + Montant: double
+ Op public () # Op protégée () - Op privée ()

Diagramme de classes

Le diagramme de classes comporte 5 concepts :

- ✓ Classe
- ✓ Attribut
- ✓ Identifiant
- ✓ Opération
- ✓ **Relation**



Relation

S'il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives.

Les liens entre les objets doivent être considérés comme des instances de relations entre classes.

Il existe plusieurs types de relations entre classes :

- l'association
- la généralisation
- la dépendance

Relation

S'il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives.

Les liens entre les objets doivent être considérés comme des instances de relations entre classes.

Il existe plusieurs types de relations entre classes :

- **l'association**
- la généralisation
- la dépendance

Relation: Association

- L'association est la relation la plus courante et la plus riche du point de vue sémantique.
- L'association existe entre les classes et non entre les instances : elle est introduite pour montrer une structure et non pour montrer des échanges de données.
- Chaque classe qui participe à l'association joue un rôle. Les rôles sont définis par 2 propriétés :
 - la multiplicité.
 - la navigabilité.

Relation: Association

la multiplicité

Elle définit le nombre d'instances de l'association pour une instance de la classe. La multiplicité est définie par un nombre entier ou un intervalle de valeurs.

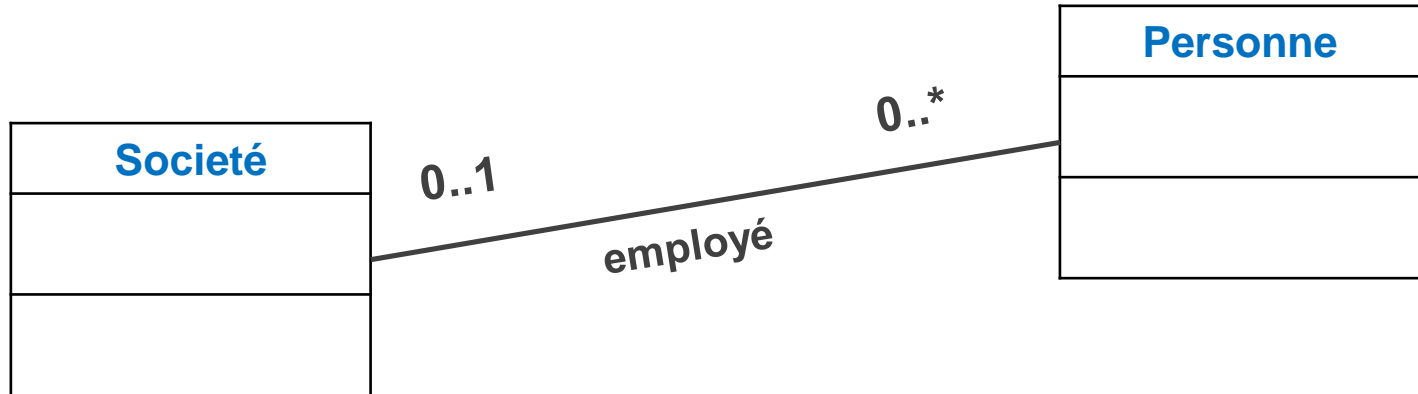
La multiplicité est notée sur le rôle (elle est notée à l'envers de la notation MERISE).

1	Un et un seul
0..1	Zéro ou un
N ou *	N (entier naturel)
N..M	De M à N (entiers naturels)
0..*	De zéro à plusieurs
1..*	De 1 à plusieurs

Relation: Association

la multiplicité

Formalisme et exemple en employant le noms de l'association et la multiplicité des rôles :

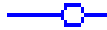


Relation: Association

la multiplicité

La multiplicité peut également s'exprimer par des symboles :

1 

0..1 

* 

0..* 

1..* 

Relation: Association

la navigabilité

- La navigabilité n'a rien à voir avec le sens de lecture de l'association.
- Une navigabilité placée sur une terminaison cible indique si ce rôle est accessible à partir de la source.
- On représente graphiquement la navigabilité par une flèche du côté de la terminaison navigable et on empêche la navigabilité par une croix du côté de la terminaison non navigable.



Relation: Association

la navigabilité

Exemple:



- La terminaison du côté de la classe **Commande** n'est pas navigable : cela signifie que les instances de la classe **Produit** ne stockent pas de liste d'objets du type **Commande**.
- La terminaison du côté de la classe **Produit** est navigable : chaque objet commande contient une liste de produits.

Relation: Association

Les classes-association:

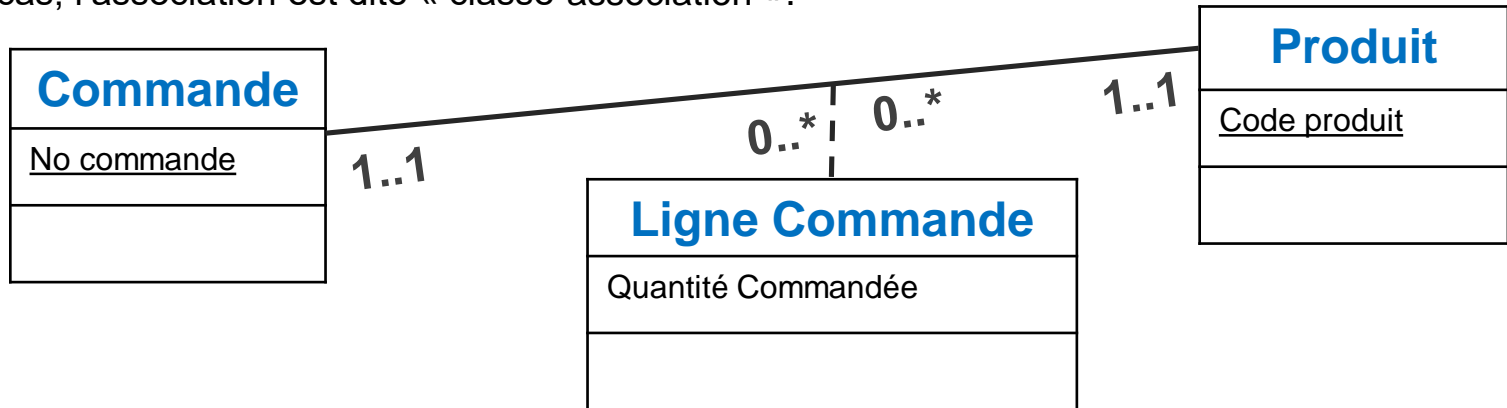
Les attributs d'une classe dépendent fonctionnellement de l'identifiant de la classe. Parfois, un attribut dépend fonctionnellement de 2 identifiants, appartenant à 2 classes différentes.

Exemple:

L'attribut « quantité commandée » dépend fonctionnellement du numéro de commande et du code produit.

On va donc placer l'attribut « quantité commandée » dans l'association « comporter ».

Dans ce cas, l'association est dite « classe-association ».

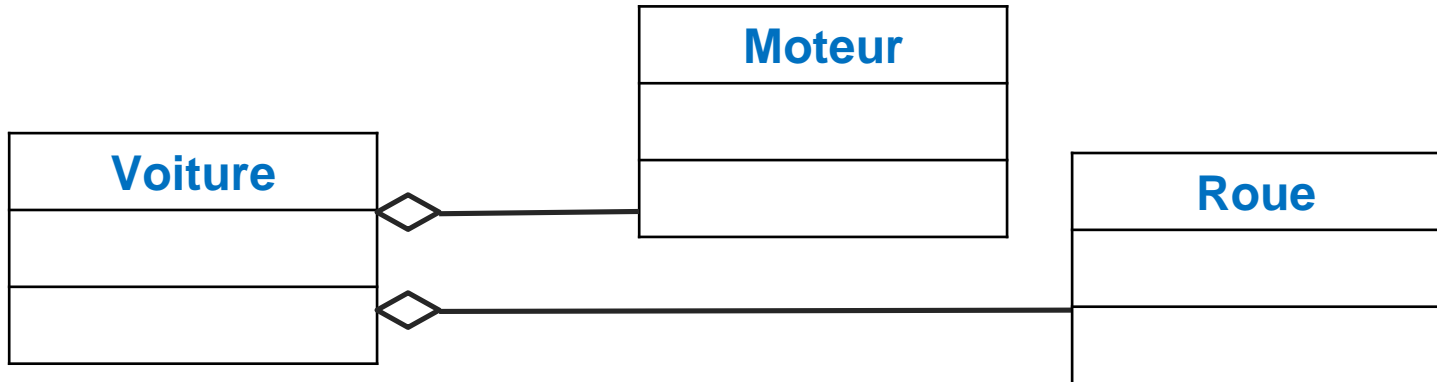


Relation: Association

L'agrégation:

- Une agrégation représente une association non symétrique dans laquelle une des extrémités joue un rôle prédominant par rapport à l'autre extrémité.
- L'agrégation se représente toujours avec un petit losange du côté de l'agregat.

Formalisme et exemple :

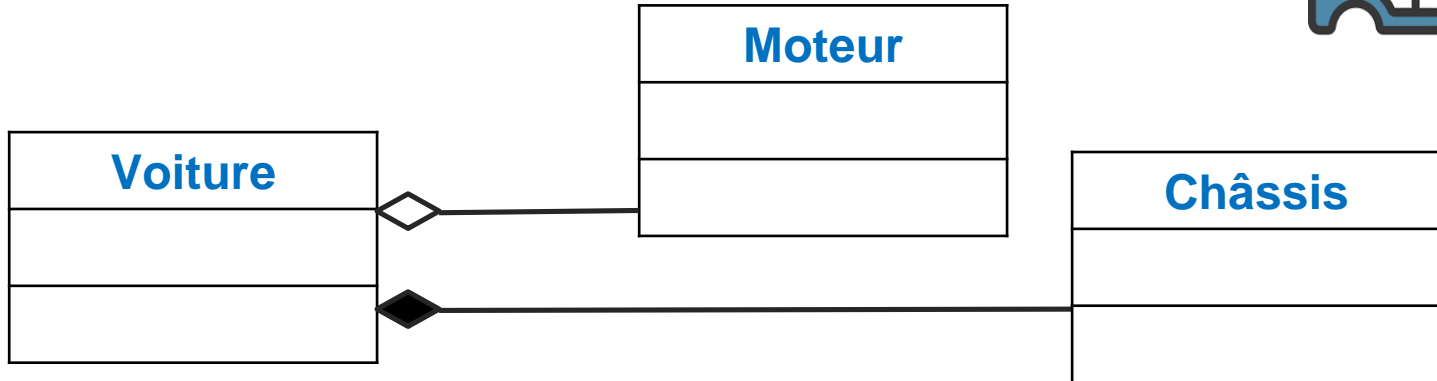


Relation: Association

La composition:

- Il s'agit d'une appartenance forte. La vie de l'objet composant est liée a celle de son composé.
- La composition se modélise par un losange noir côté composé.

Formalisme et exemple :



Relation

S'il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives.

Les liens entre les objets doivent être considérés comme des instances de relations entre classes.

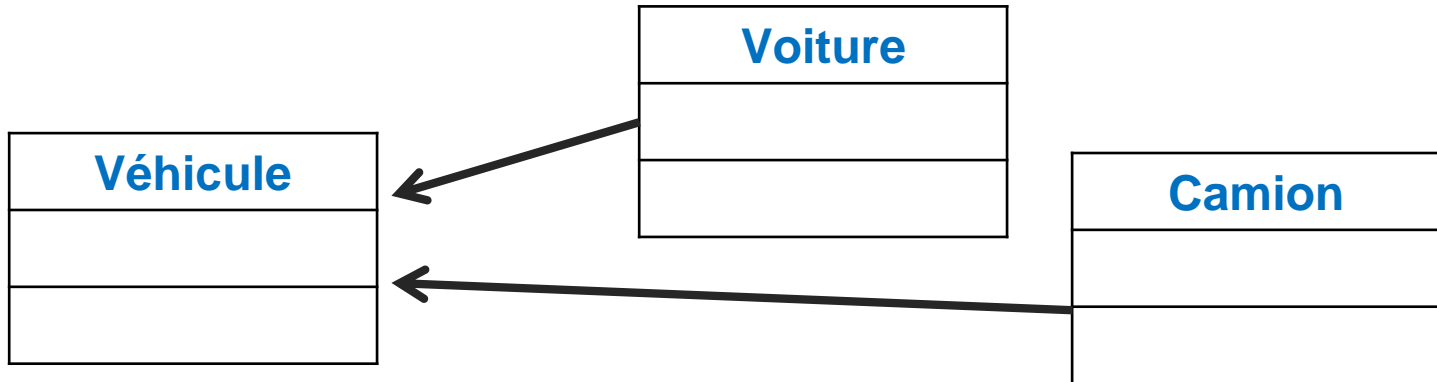
Il existe plusieurs types de relations entre classes :

- l'association
- **la généralisation**
- la dépendance

Relation: Généralisation

Une relation de généralisation est une relation dans laquelle un élément de modèle (l'enfant) est basé sur un autre élément de modèle (le parent).

Formalisme et exemple :



Relation

S'il existe des liens entre objets, cela se traduit nécessairement par des relations qui existent entre leurs classes respectives.

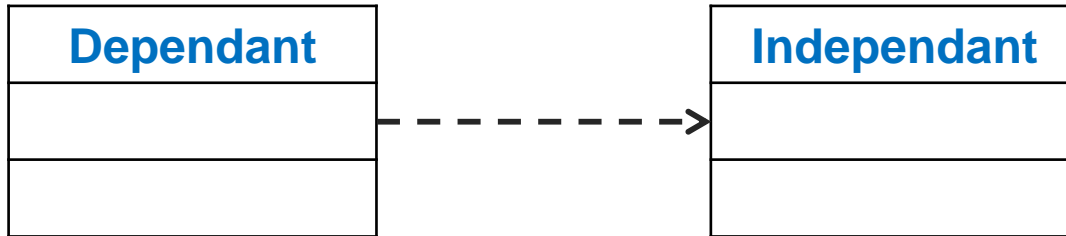
Les liens entre les objets doivent être considérés comme des instances de relations entre classes.

Il existe plusieurs types de relations entre classes :

- l'association
- la généralisation
- **la dépendance**

Relation: Dépendance

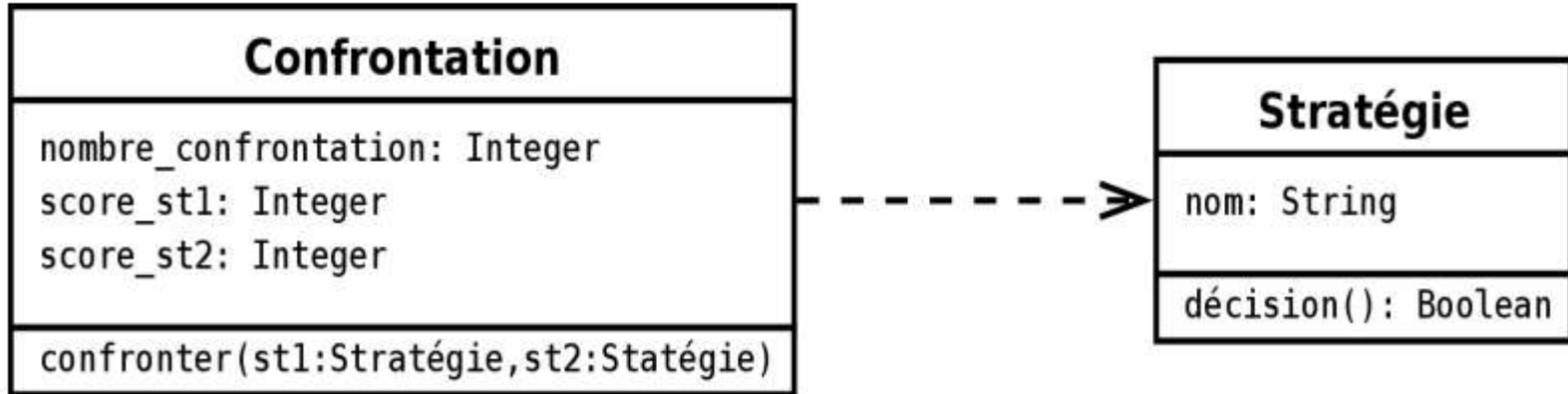
- Une dépendance est une relation entre deux éléments dans laquelle une modification à un élément provoque des modifications sur l'autre élément .
- Une dépendance est représentée graphiquement sous la forme d'une ligne dirigée de pointillés, dirigée vers l'élément.



Relation: Dépendance

Exemple :

une modification à la stratégie (l'élément indépendant) provoque des modifications sur la confrontation (l'élément dépendant).



Relation: Dépendance

- La dépendance peut être stéréotypée pour préciser le lien sémantique entre les éléments du modèle.

Exemple :

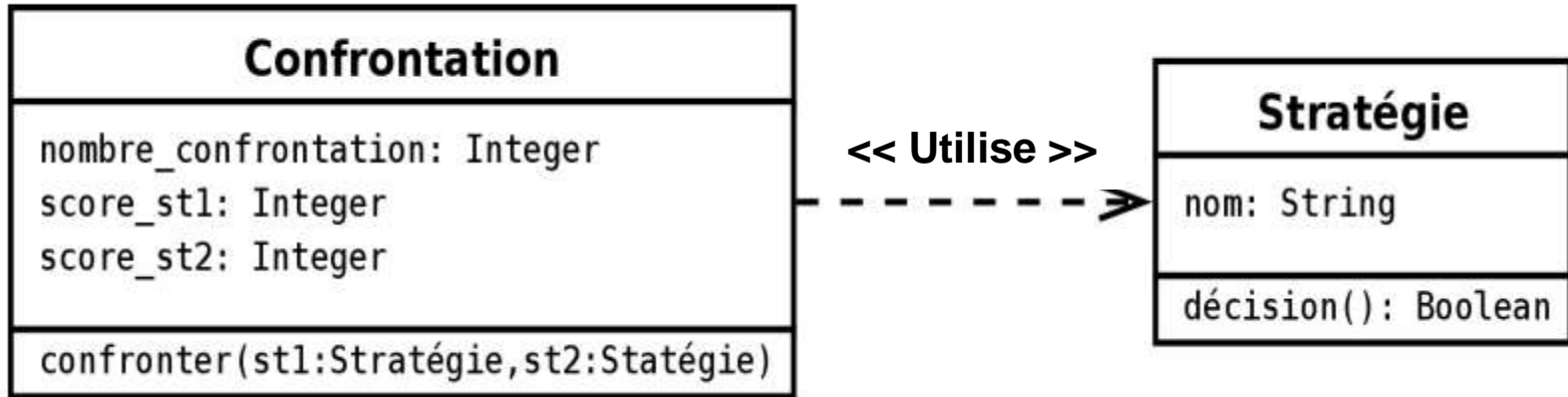


Diagramme de classes

Exemple:

**Une application pour la gestion d'une
bibliothèque**

Diagramme de classes

Étape 1 : Identifier les noms de classes

La première étape consiste à identifier les principaux objets du système.

Utilisateur

Commande

Livre

roman

Livre scientifique

Auteur

Résumé

livres pour enfants

livres de recettes

Diagramme de classes

Étape 2 : Distinguer les relations

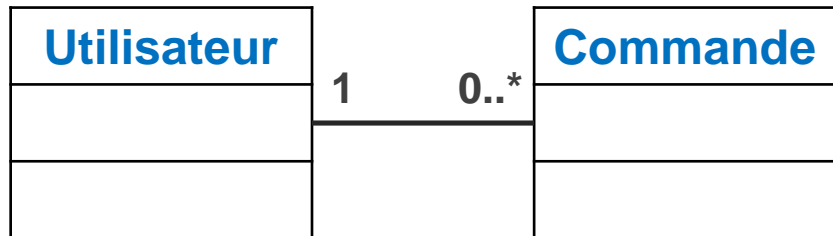


Diagramme de classes

Étape 2 : Distinguer les relations

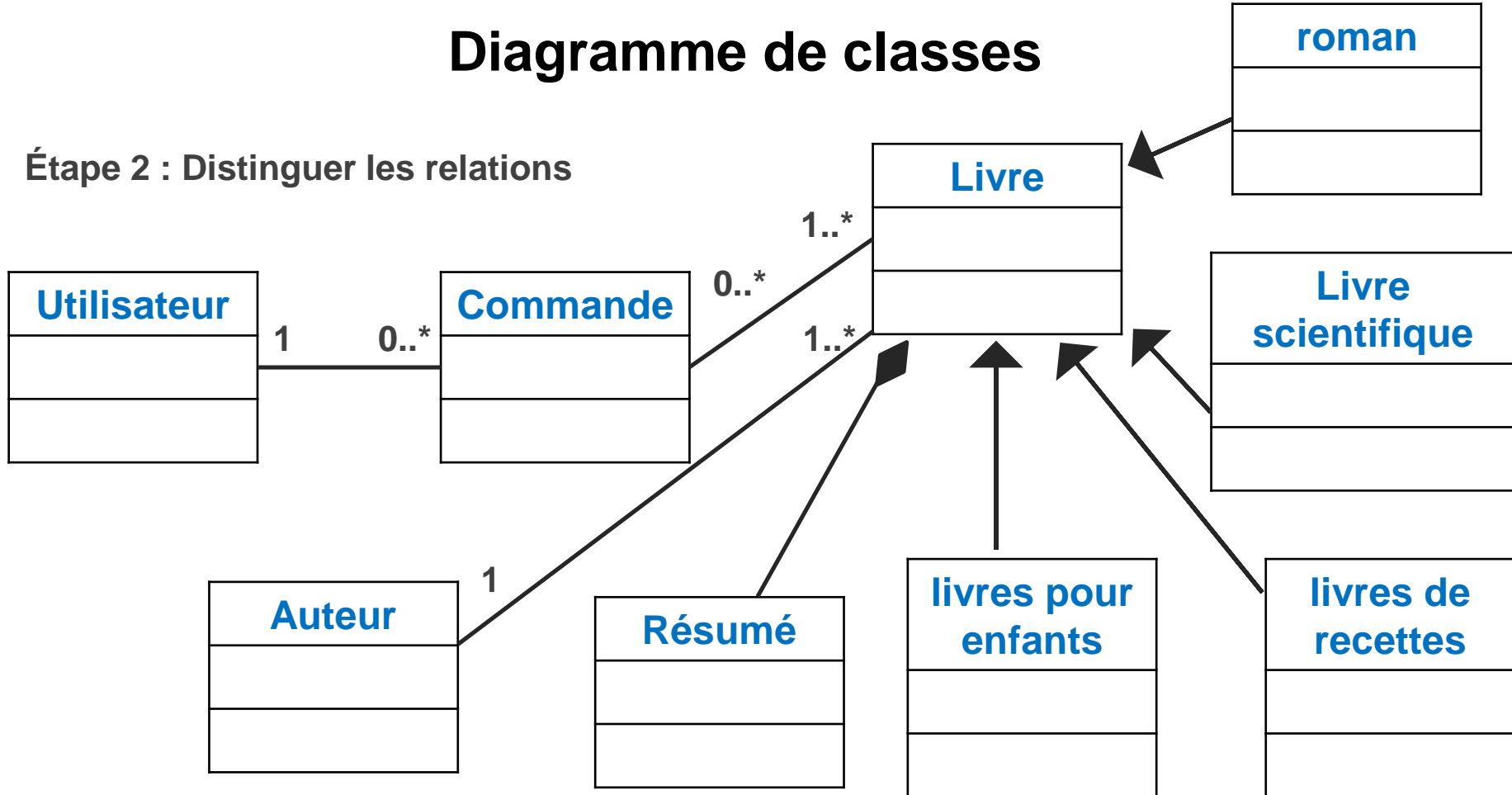


Diagramme de classes

Étape 3 : Créer la structure

Ajouter des attributs et des fonctions/méthodes/opérations.

Diagramme de classes

Exercices

Gestion d'un hôtel

Un hôtel est composé d'au moins deux chambres. Chaque chambre dispose d'une salle d'eau : douche ou bien baignoire. Un hôtel héberge des personnes. Il peut employer du personnel et il est impérativement dirigé par un directeur. On ne connaît que le nom et le prénom des employés, des directeurs et des occupants. Certaines personnes sont des enfants et d'autres des adultes (faire travailler des enfants est interdit). Un hôtel a les caractéristiques suivantes : une adresse, un nombre de pièces et une catégorie.

Une chambre est caractérisée par le nombre et de lits qu'elle contient, son prix et son numéro. On veut pouvoir savoir qui occupe quelle chambre à quelle date. Pour chaque jour de l'année, on veut pouvoir calculer le loyer de chaque chambre en fonction de son prix et de son occupation (le loyer est nul si la chambre est inoccupée). La somme de ces loyers permet de calculer le chiffre d'affaires de l'hôtel entre deux dates.

Gestion d'un hôtel

Un hôtel est composé d'au moins deux chambres. Chaque chambre dispose d'une salle d'eau : douche ou bien baignoire. Un hôtel héberge des personnes. Il peut employer du personnel et il est impérativement dirigé par un directeur. On ne connaît que le nom et le prénom des employés, des directeurs et des occupants. Certaines personnes sont des enfants et d'autres des adultes (faire travailler des enfants est interdit). Un hôtel a les caractéristiques suivantes : une adresse, un nombre de pièces et une catégorie.

Une chambre est caractérisée par le nombre et de lits qu'elle contient, son prix et son numéro. On veut pouvoir savoir qui occupe quelle chambre à quelle date. Pour chaque jour de l'année, on veut pouvoir calculer le loyer de chaque chambre en fonction de son prix et de son occupation (le loyer est nul si la chambre est inoccupée). La somme de ces loyers permet de calculer le chiffre d'affaires de l'hôtel entre deux dates.

Gestion des défauts pour un fabricant de moteurs

Une usine de fabrication des moteurs cherche à organiser la gestion des défauts observés sur des moteurs confrontés à des tests en situation réelle.

Pour chaque moteur on relève tous les défauts observés à des dates différentes. Pour chacun de ces défauts on note le kilométrage et on diagnostique la gravité.

Les défauts sont classés par type, chaque type de défaut étant codifié (on peut imaginer par exemple, la casse, la fuite, etc.).

Les défauts sont relevés par des opérateurs, employés du groupe et affectés à une usine.

Les moteurs sont d'un modèle particulier, fabriqué dans une usine, et sous la responsabilité d'un responsable également employé du groupe.

Gestion des défauts pour un fabricant de moteurs

Une usine de fabrication des moteurs cherche à organiser la gestion des défauts observés sur des moteurs confrontés à des tests en situation réelle.

Pour chaque moteur on relève tous les défauts observés à des dates différentes. Pour chacun de ces défauts on note le kilométrage et on diagnostique la gravité.

Les défauts sont classés par type, chaque type de défaut étant codifié (on peut imaginer par exemple, la casse, la fuite, etc.).

Les défauts sont relevés par des opérateurs, employés du groupe et affectés à une usine.

Les moteurs sont d'un modèle particulier, fabriqué dans une usine, et sous la responsabilité d'un responsable également employé du groupe.