# Cours Génie Logiciel (partie 3)

EILCO-ING2

ABAKARIM Fadwa

# Modélisation

La modélisation est une étape ou une activité qui consiste à représenter de manière simplifiée et abstraite un système, un processus ou une idée.

Elle se fait souvent à l’aide de diagrammes, de schémas ou de modèles conceptuels pour visualiser les besoins, les solutions ou les relations dans un projet.

# Modélisation

Dans le contexte d’un projet logiciel, modéliser revient à :

- Analyser les besoins pour comprendre ce que le logiciel doit faire.
- Représenter les solutions fonctionnelles (ce que le logiciel doit réaliser) et techniques (comment il le fera).
- Créer des diagrammes ou des schémas (ex. diagramme UML, diagramme de classes) pour clarifier et partager ces idées entre les parties prenantes (développeurs, clients, utilisateurs).

# Modélisation

Exemples de techniques de modélisation :

Les diagrammes UML incluent :

- Diagramme de cas d'utilisation : Représente les interactions entre les acteurs et le système.
- Diagramme de séquence : Montre l'échange de messages entre objets au cours du temps.
- Diagramme de classes : Décrit la structure statique du système, incluant les classes, leurs attributs et leurs relations.
- Diagramme d'états : Représente les différents états d'un objet et les transitions entre ces états.

# Différence clé : modélisation et conception

- **La modélisation** est une activité spécifique, souvent incluse dans la phase de conception, qui se concentre sur la représentation visuelle ou abstraite des idées.

- **La conception** est un processus plus large qui englobe plusieurs étapes, y compris la modélisation, pour élaborer et structurer la solution finale.

# Diagrammes

Un diagramme est une représentation graphique utilisée pour illustrer des informations, des processus, des structures ou des relations de manière visuelle.

Il sert à simplifier des idées complexes et à rendre les concepts plus compréhensibles, en organisant les éléments et leurs interactions de façon claire et intuitive.

# Caractéristiques principales des diagrammes

- Simplicité : Résume des informations complexes sous une forme facile à comprendre.
- Clarté : Utilise des formes, des flèches, des couleurs et des symboles pour organiser les éléments visuellement.
- Communication : Facilite la compréhension et l'échange d'idées en rendant les concepts accessibles à tous.

# UML

- UML (Unified Modeling Language) est un langage de modélisation visuel utilisé principalement dans le domaine de l'ingénierie logicielle.
- Son but est de spécifier, concevoir, documenter et analyser des systèmes logiciels.
- UML permet de créer des diagrammes qui représentent graphiquement différents aspects d'un système.

# UML

UML (Unified Modeling Language) comprend plusieurs types de diagrammes, qui sont regroupés en trois grandes catégories :

- **Diagrammes structurels** : Modélisent la structure du système (classes, composants, déploiement), exemple : Diagramme de classes, diagramme d’objets.
- **Diagrammes comportementaux** : Illustrent les comportements dynamiques du système (interactions utilisateurs, échanges de messages, processus), exemple : Diagramme de cas d'utilisation.
- **Diagrammes d'interaction** : Détaillent les échanges et les interactions entre les objets ou composants du système, exemple : Diagramme de séquence.

# Diagramme de classes

Le diagramme de classes est l'un des diagrammes les plus fondamentaux et les plus utilisés en UML (Unified Modeling Language).

Il permet de représenter la structure statique d'un système logiciel en montrant les **classes**, leurs **attributs**, leurs **méthodes** et les **relations entre elles**.

Ce diagramme est essentiel pour comprendre l'architecture **d'un système orienté objet** et constitue une base importante pour la conception du système.

# Diagramme de classes

Composants d'un diagramme de classes:

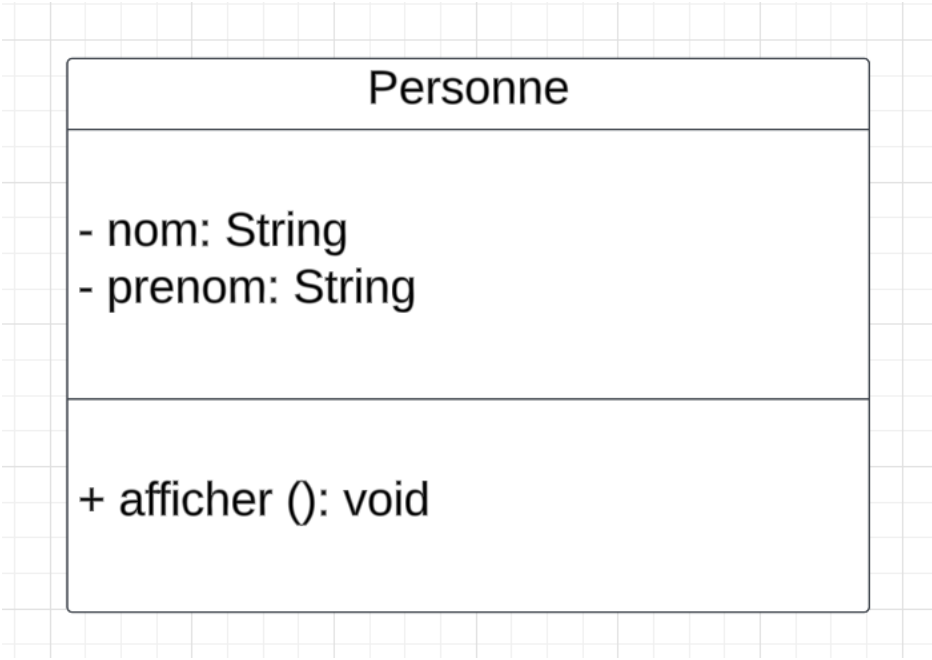
- Classe
- Attributs
- Méthodes
- Relations entre les classes

# Diagramme de classes

Classe :

Dans un diagramme de classes, une classe est représentée par un rectangle divisé en trois parties :

- La partie supérieure contient le nom de la classe.
- La partie intermédiaire liste les attributs de la classe.
- La partie inférieure contient les méthodes de la classe.



# Diagramme de classes

Attributs :

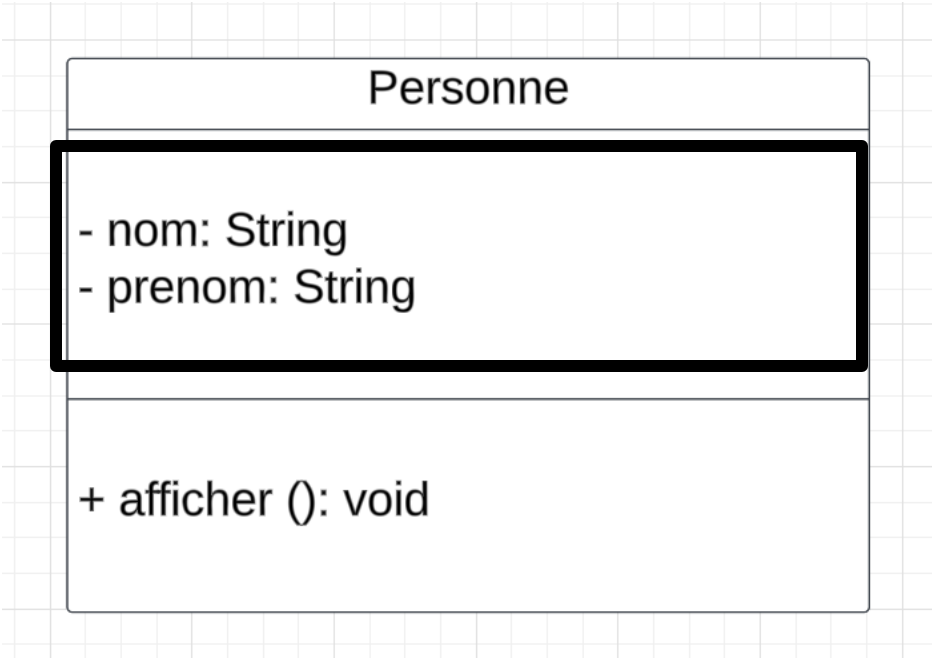
Les attributs représentent les données ou les propriétés d'un objet.

La visibilité des attributs est définie par un préfixe :

+ : public

- : privé

# : protégé

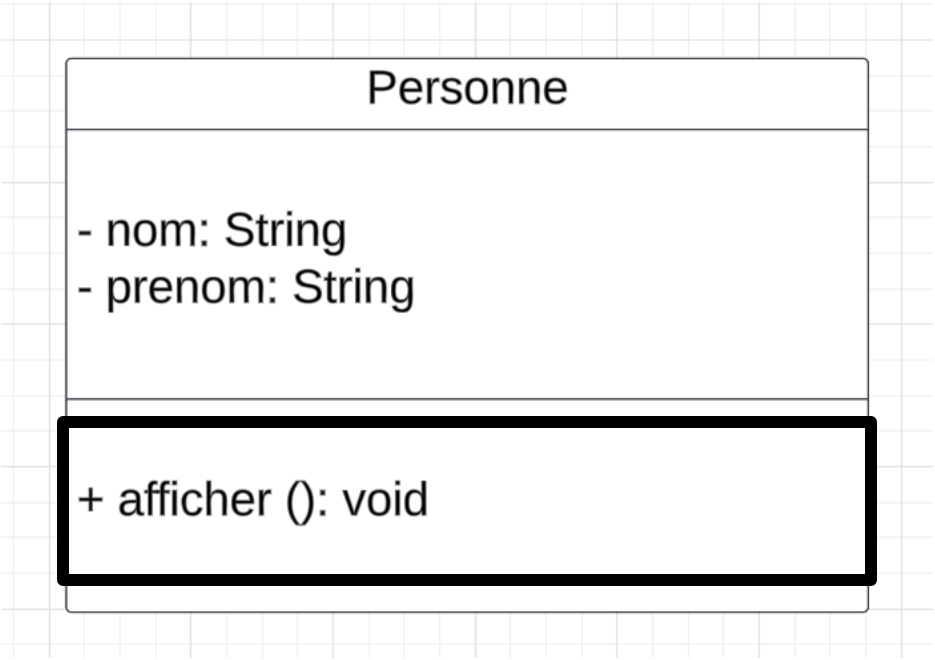


# Diagramme de classes

Méthodes :

Les méthodes (ou fonctions) définissent les comportements ou actions que les objets de la classe peuvent réaliser.

Par exemple, une classe Personne pourrait avoir des méthodes, telles que afficher(), permettant d'afficher le nom et le prénom de la personne.




Les méthodes sont également suivies de la même notation de visibilité (public, privé, etc.).

# Diagramme de classes

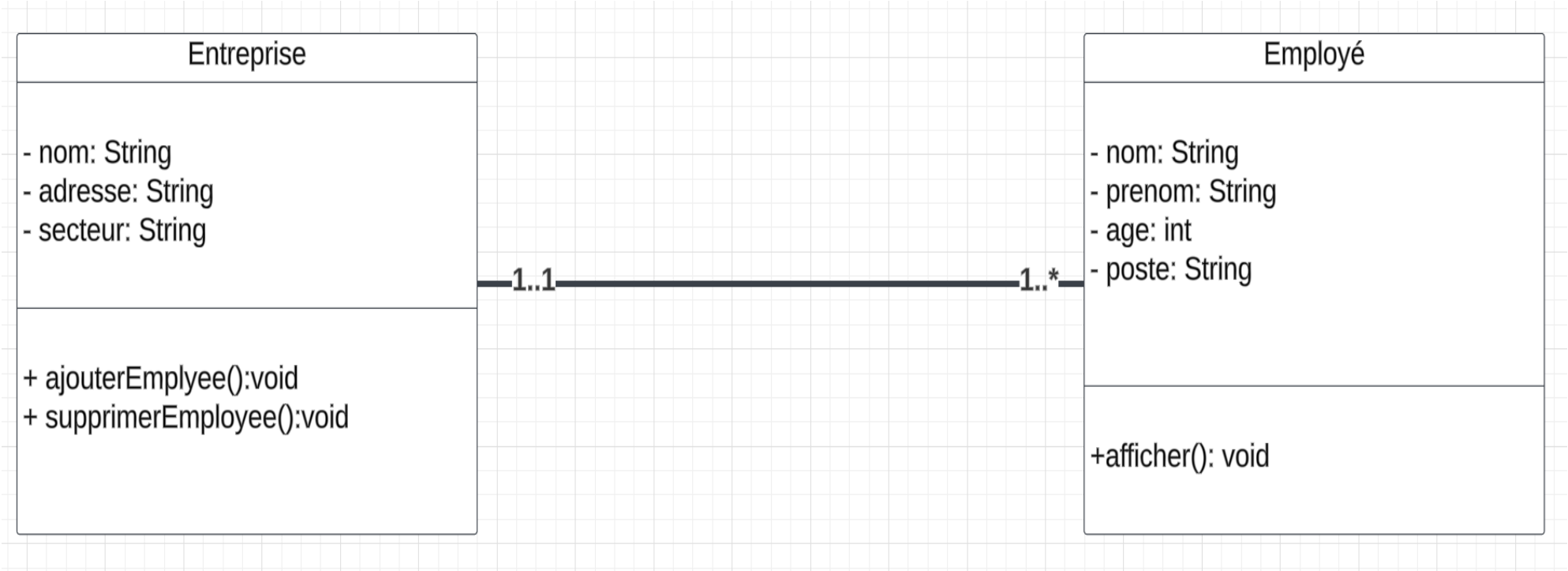
Relation entre les classes :

Les classes peuvent être liées entre elles par différentes sortes de relations :

- Association
- Héritage (ou généralisation)
- Composition et agrégation
- Dépendance

# Diagramme de classes

Relation entre les classes : Association



# Diagramme de classes

## Relation entre les classes : Association

| Cardinalité | Description | Exemple d'utilisation |
|---|---|---|
| 1 ou 1..1 | Exprime une contrainte de totalité : exactement 1 | Une personne a exactement une adresse. |
| 0..1 | Optionnel : Il peut y avoir zéro ou un élément | Une voiture peut avoir zéro ou une assurance. |
| n..m | De "n" à "m" : Il peut y avoir entre n et m éléments (où "m" > "n") | Un cours peut être suivi par 3 à 5 étudiants. |
| n..* | De "n" à plusieurs : Il peut y avoir entre n et un nombre indéfini d'éléments | Une bibliothèque peut avoir entre 10 et plusieurs milliers de livres. |
| 0..* ou * | De zéro à plusieurs : Il peut y avoir zéro ou plusieurs éléments | Un client peut avoir zéro ou plusieurs commandes. |

# Diagramme de classes

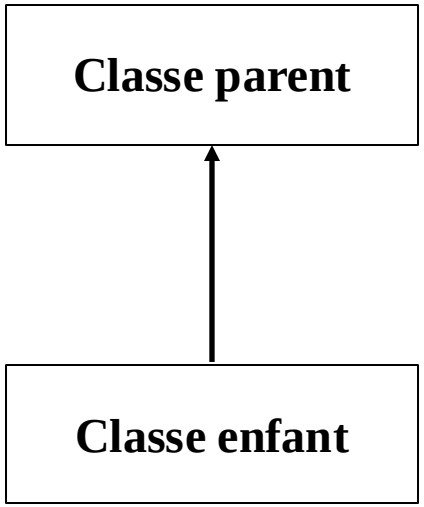
Relation entre les classes :  Héritage (ou généralisation)

L'héritage UML est une relation entre deux classes où une classe (la sous-classe) hérite des propriétés et des comportements d'une autre classe (la super-classe).

Cela permet de réutiliser les attributs et méthodes de la super-classe dans la sous-classe, tout en permettant à la sous-classe d'ajouter ou de modifier des fonctionnalités spécifiques.

# Diagramme de classes

Relation entre les classes :  Héritage (ou généralisation)



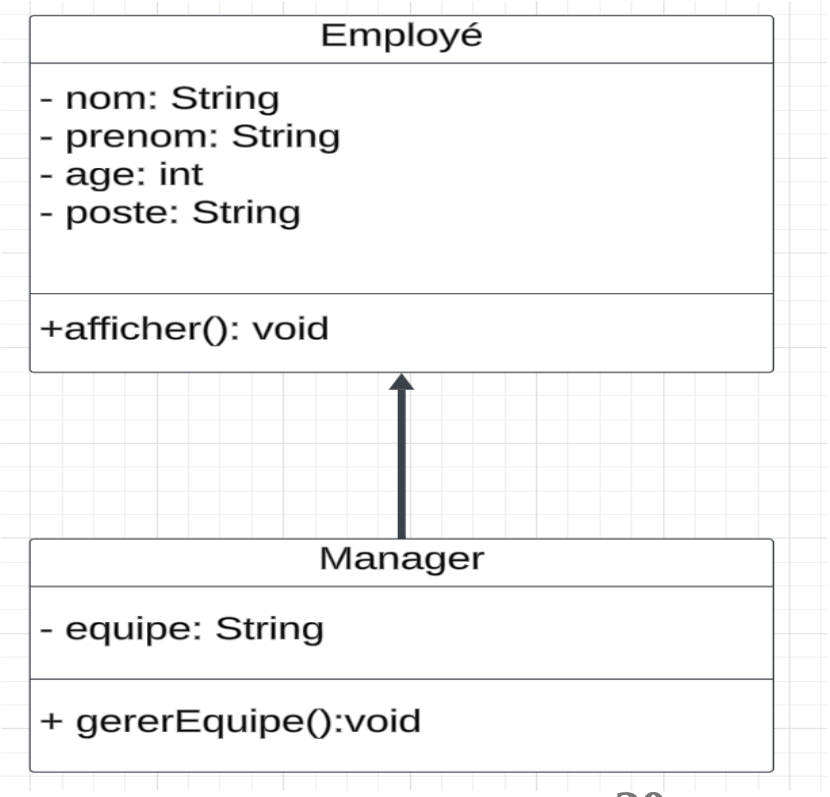
# Diagramme de classes

Relation entre les classes : Héritage (ou généralisation)

**Exemple :**

Employé (super-classe) et Manager (sous-classe).

Un manager est un type spécifique d'employé, donc Manager hérite des attributs et méthodes d'Employé.



# Diagramme de classes

Relation entre les classes :  Composition et agrégation

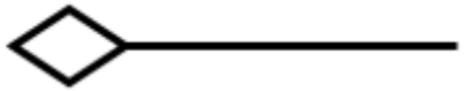
Les relations de contenance montrent comment un objet "tout" contient d'autres objets "parties".

Il existe deux types de relations de contenance : l'agrégation et la composition.

# Diagramme de classes

Relation entre les classes : Agrégation

L'agrégation est une relation de contenance faible. Cela signifie que l'objet partie peut vivre indépendamment de l'objet tout.

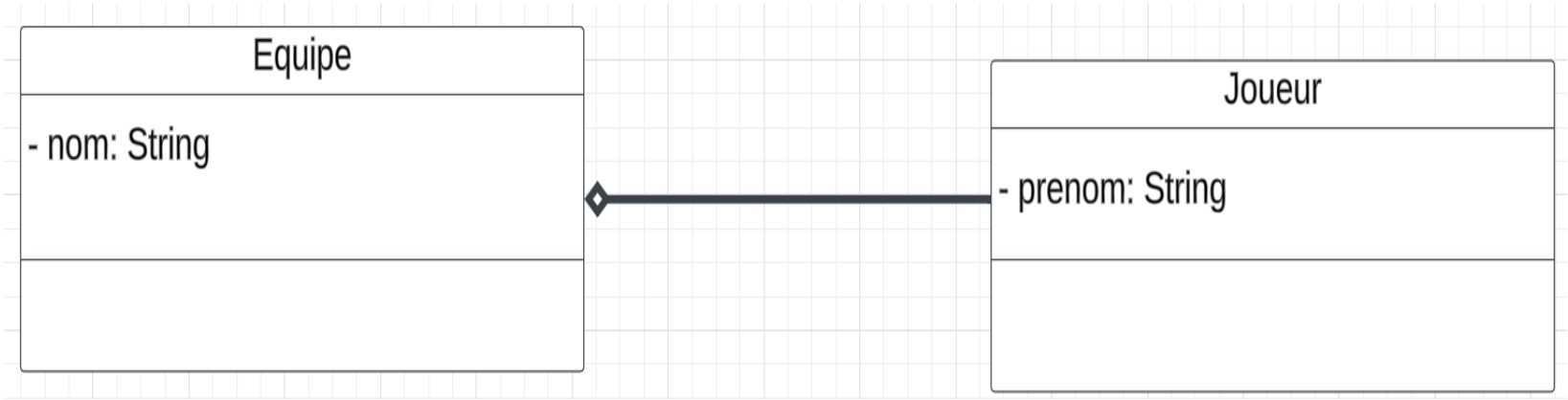


**Exemple :**

Imaginez une équipe de football. L'équipe est composée de joueurs. Mais, les joueurs peuvent exister sans l’équipe (ils peuvent jouer dans d'autres équipes ou être en dehors de l'équipe).

# Diagramme de classes

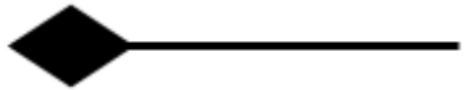
Relation entre les classes : Agrégation



# Diagramme de classes

Relation entre les classes : Composition

La composition est une relation de contenance forte. Cela signifie que l'objet partie ne peut pas exister sans l'objet tout. Si l'objet tout disparaît, les objets parties disparaissent aussi.

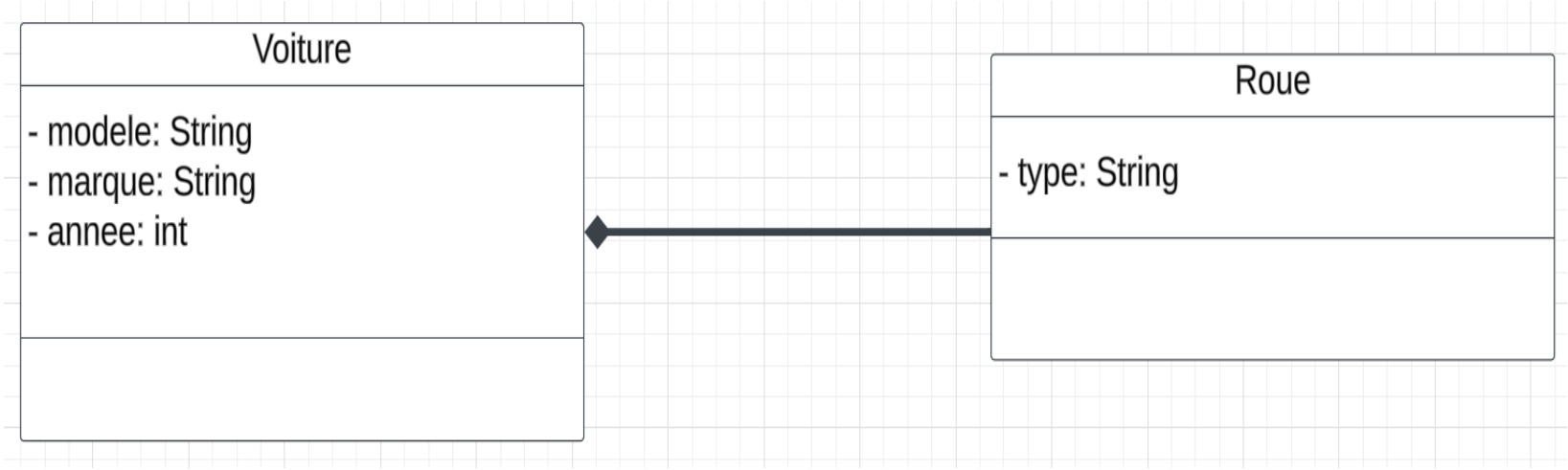


**Exemple :**

Imaginez une voiture. Une voiture est composée de roues. Mais, les roues ne peuvent pas exister sans la voiture. Si la voiture est détruite, les roues sont détruites aussi.

# Diagramme de classes

Relation entre les classes : Composition



# Diagramme de classes

Relation entre les classes : Dépendance

La dépendance est une relation faible entre deux classes en UML (Unified Modeling Language).

Elle indique qu'une classe utilise une autre classe, généralement pour accomplir une tâche ou fournir un service, mais sans en dépendre de manière permanente.

Cette relation est souvent temporaire ou ponctuelle.

# Diagramme de classes

Relation entre les classes : Dépendance

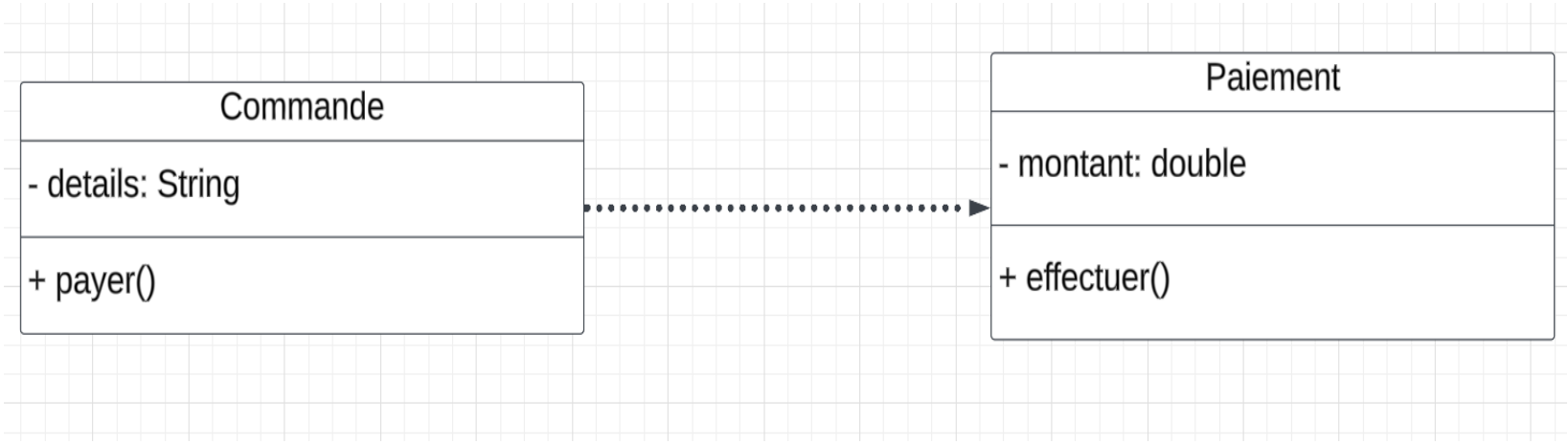
La dépendance exprime une relation temporaire où une classe utilise les services d'une autre.

Elle n’est pas aussi forte que les relations d’association, d’agrégation ou de composition.

# Diagramme de classes

Relation entre les classes : Dépendance

La classe Commande utilise temporairement Paiement pour effectuer une transaction.



# Exercice 1 : partie 1

Une personne est caractérisée par son nom, son prénom, son sexe et son âge. Les objets de classe Personne doivent pouvoir calculer leurs revenus et leurs charges. Les attributs de la classe sont privés.

- Question 1 : Donnez une représentation UML de la classe Personne, en remplissant tous les compartiments adéquats.

# Exercice 1 : partie 2

Il est possible de changer le nom et le prénom d'une personne et de les récupérer.

- Question 2 : Enrichissez encore la représentation précédente pour prendre en compte ces nouveaux éléments.

# Exercice 2

Un établissement scolaire souhaite modéliser son système d'information par un diagramme de classe. Voici les besoins :

- L'établissement a un nom, une adresse, et un ou plusieurs enseignants.
- Chaque enseignant a un matricule, un nom, un prénom, et peut enseigner une ou plusieurs matières.
- Les étudiants sont inscrits à l'établissement. Chaque étudiant a un numéro d'inscription, un nom, un prénom, une date de naissance, et est inscrit dans une seule classe.
- Une classe est identifiée par un nom (ex. : 1èreA), un niveau (ex. : 1ère année), et regroupe plusieurs étudiants.
- Les enseignants peuvent être responsables d’une ou plusieurs classes.