

# LOG121

## Conception orientée objet

# Diagrammes de classes UML

Enseignante: Souad Hadjres

# Plan

2

- Documenter la conception
  - Le langage UML
  - Diagramme de classes UML

# Documenter la conception

3

- Il est nécessaire de documenter et illustrer les décisions de conception
  - ▣ Cela permet de guider l'implémentation
  - ▣ Les illustrations graphiques facilitent la communication
  - ▣ Cela facilite aussi la maintenance

# Plan

4

- Documenter la conception
- Le langage UML
- Diagramme de classes UML

# UML: Unified Modeling Language

5

- Standard incontournable (normalisé en 1997 par l'OMG)
  - ▣ Unification de trois approches: OOAD (Grady Booch), OMT (Jim Rumbaugh), et OOSE (Ivar Jacobson)
- Supporté par de très nombreux outils
- Indépendant des langages d'implémentation
- Toujours en évolution

# UML: Unified Modeling Language

6

- Propose plusieurs vues complémentaires d'un système
  - ▣ Vue statique
    - Diagramme de classes
    - Diagramme de composants
  - ▣ Vue dynamique
    - Diagramme des cas d'utilisation
    - Diagramme de séquences
    - Diagramme de communications
    - Diagramme d'activités
    - Diagramme d'états
    - Etc.

# UML: Unified Modeling Language

7

- Nous utiliserons dans le cours
  - ▣ Les diagrammes de classes
  - ▣ Les diagrammes de séquences

# Plan

8

- Documenter la conception
- Le langage UML
- Diagramme de classes UML



# Diagramme de classes

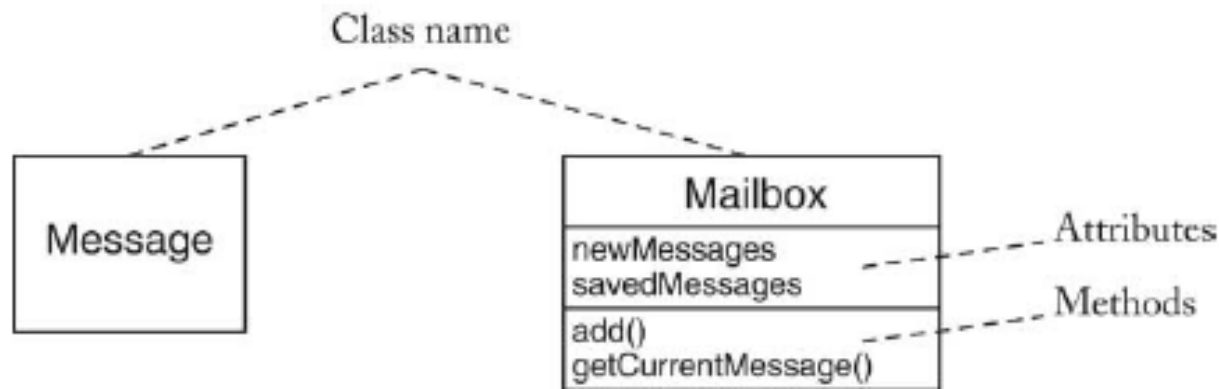
9

- « Il représente les classes et les interfaces d'un système ainsi que les différentes relations entre celles-ci. » [définition commune à plusieurs références]
- Éléments de base : les classes

# Diagramme de classes

10

- Une classe est représentée par un rectangle avec trois parties
  - ▣ 1ère Partie: le nom de la classe
  - ▣ 2ième partie: les attributs
  - ▣ 3ième partie: les méthodes



# Diagramme de classes

11

- Inclure uniquement les méthodes et les attributs les plus importants
- Considérer le regroupement des attributs dans une autre classe quand il y en a trop
  - ▣ Regrouper les attributs numéro, rue, ville et code\_postal dans une classe Adresse

# Diagramme de classes

12

## □ Attributs d'une classe:

Visibilité nom\_Attribut [multiplicité] : type\_Attribut [= Initialisation]

□ Visibilité : Public(+), Protected (#) ou Private (-)

□ Multiplicité: le nombre de fois où cet attribut peut être utilisé au sein du même objet

■ Par exemple, pour permettre de donner deux prénoms à une instance de la classe Personne, voilà la définition de l'attribut prénom:

■ # prenom [2] : String

# Diagramme de classes

13

## □ Méthodes d'une classe:

Visibilité nom\_méthode ([liste\_de\_paramètres]) : [type\_retour]

## □ Exemple

■ getMessage(index : int) : Message

## □ Quand la méthode a des paramètres, la liste de paramètres est spécifiée comme suit:

nom\_param1: type\_param1, nom\_param2: type\_param2, etc.

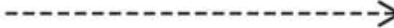
## □ Une méthode peut ne rien retourner


■ Correspond à une méthode en Java avec le mot clé « void »

# Diagramme de classes

14

## □ Relations entre classes

Dependency 

Aggregation 

Inheritance 

Composition 

Association 

Directed Association 

Interface Type Implementation 

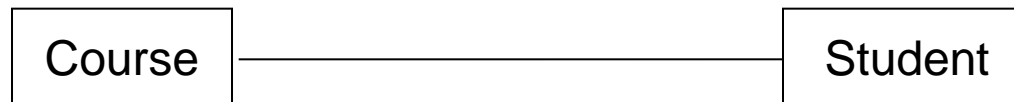
# Diagramme de classes

15

## □ Association

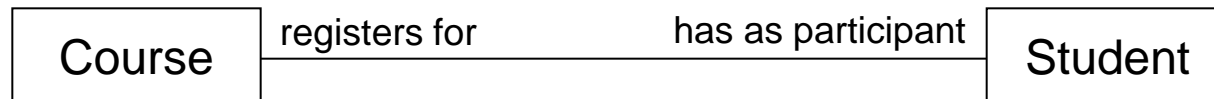
### ▣ Relation structurelle (relation logique)

#### ■ Exemple:



### ▣ Elle peut être nommée

### ▣ On peut aussi nommer ses extrémités en leur associant des rôles



# Diagramme de classes

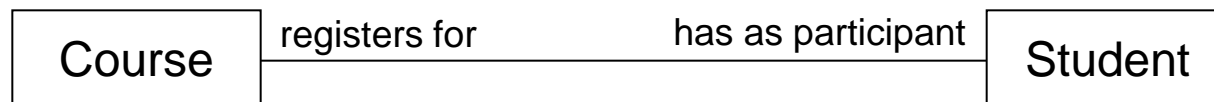
16

## □ Association

- Elle peut être bidirectionnelle ou elle peut avoir une seule direction de navigation
- Exemple de navigation unidirectionnelle: La file *MessageQueue* connaît l'ensemble des instances de *Message* qu'elle contient mais *Message* ignore tout de la file qui le contient



- Exemple de bidirection: *Course* possède un ensemble de *Students* et *Student* a un ensemble de *Courses*.





# Diagramme de classes

17

## □ Relation d'agrégation

- On spécifie les multiplicités aux deux extrémités de cette relation pour indiquer combien d'instances de l'agrégé sont contenues dans une instance de l'agrégat
  - n'importe quel nombre (zéro ou plus): \*
  - un ou plusieurs: 1..\*
  - zéro ou un: 0..1
  - exactement un: 1
- Un agrégé peut apparaître dans plusieurs agrégats.



# Diagramme de classes

18

## □ Relation de composition

- C'est un cas particulier d'agrégation
- Les objets agrégés n'existent pas en dehors du conteneur
- Si le conteneur est supprimé, les objets agrégés sont aussi supprimés
- Exemple: une file de messages est contenue de manière permanente dans une boîte vocale



# Diagramme de classes

19

## □ Relation d'implémentation

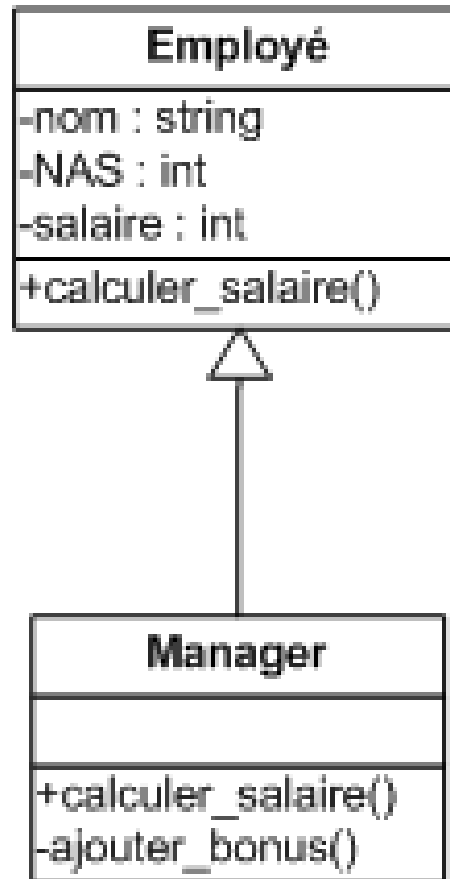
- C'est une relation entre une classe et une interface
- Une interface décrit un ensemble de méthodes sans spécifier aucune implémentation
  - À partir de java SE 8: implémentation des méthodes par défaut (default method)
- Une classe qui implémente une interface doit implémenter toutes les méthodes de cette interface
- Pour représenter une interface en UML, on ajoute le stéréotype «interface» au dessus du nom de l'interface



# Diagramme de classes

20

## □ Relation d'héritage



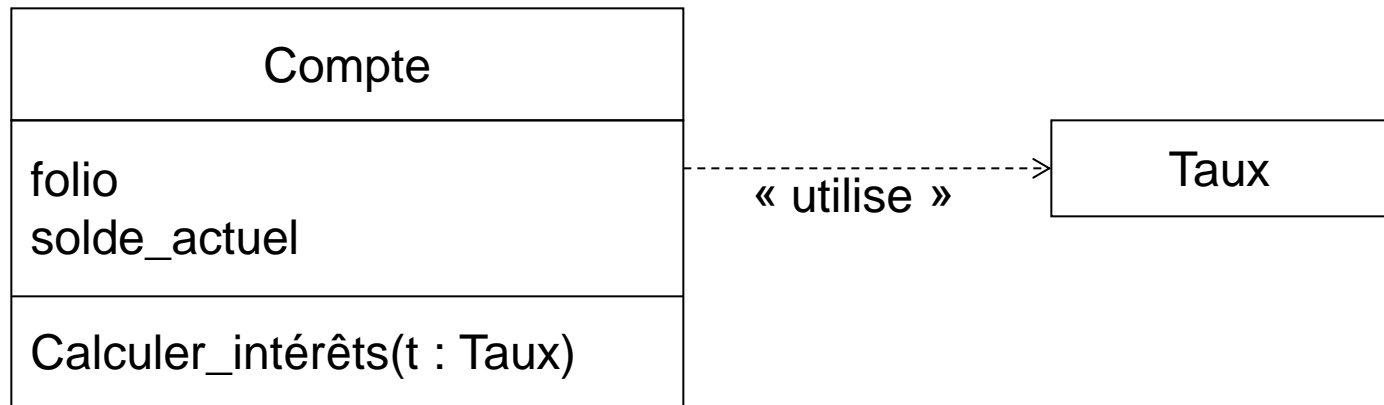
# Diagramme de classes

21

## □ Relation de dépendance

▣ Ce n'est pas une relation stable dans le temps

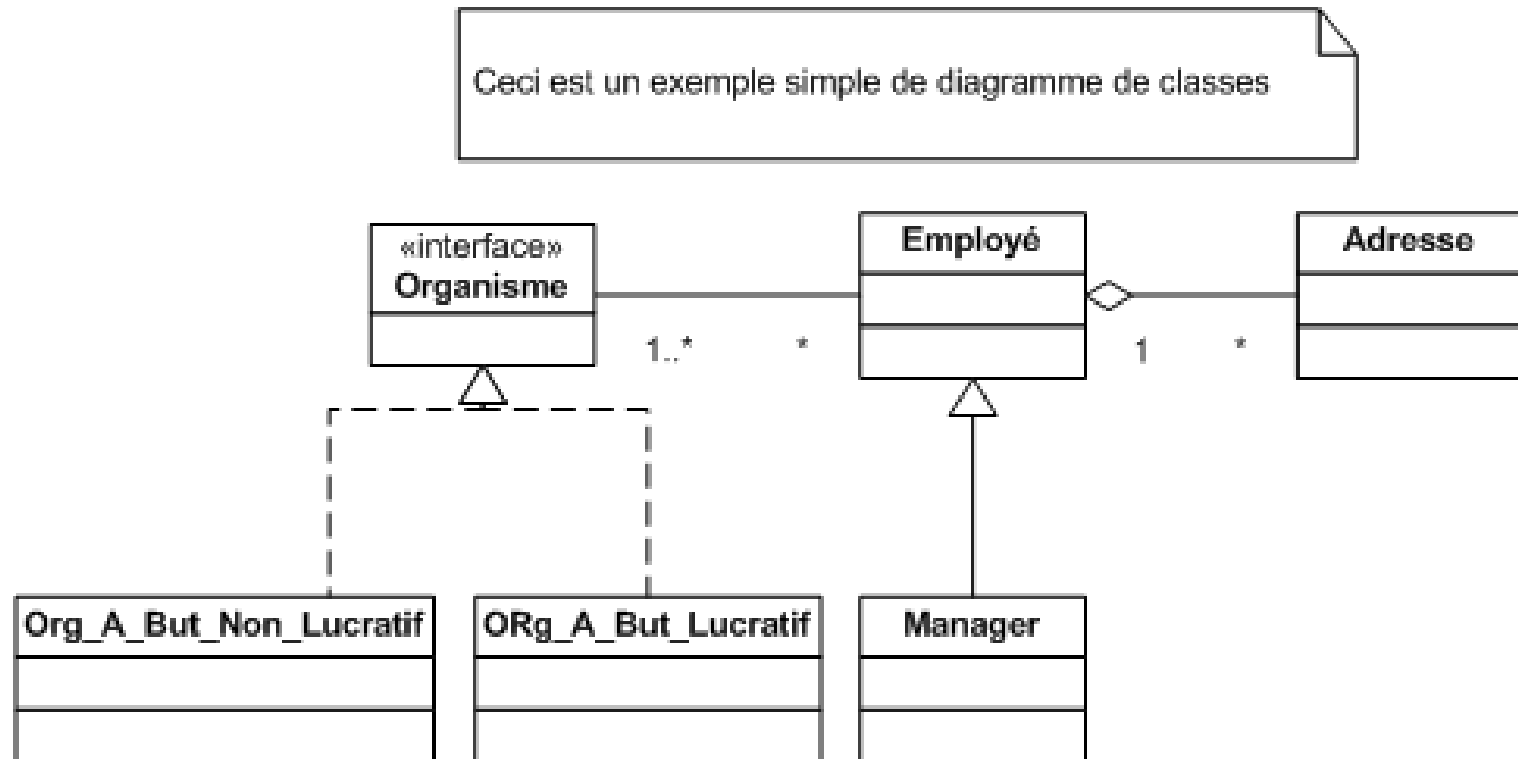
- Une instance d'une classe peut avoir besoin d'une instance d'une autre classe de façon ponctuelle.



# Diagramme de classes

22

## □ Exemple simple



# Diagramme de classes

23

## □ Quelques conseils

- ▣ Construire des diagrammes faciles à comprendre
  - Éviter de vouloir mettre toutes les classes et les relations dans un seul diagramme
- ▣ Construire des diagrammes qui communiquent adéquatement la conception
  - Inclure juste les éléments pertinents selon l'objectif du diagramme
- ▣ Accompagner les diagrammes d'explications textuelles