Abdelhafid Boussouf University Mila

Faculty of Science and Technology

Department of Math and Computer Science

Software engineering

Chapitre 4

Class Diagram & Object Diagram

Mme. S.HEDJAZ

II. Class/Object Diagram

01 Definitions and basic principles

02 Representation of a class

03 Types of relations between classes

04 Object diagram

01 Definitions and basic principles

1- Objective:

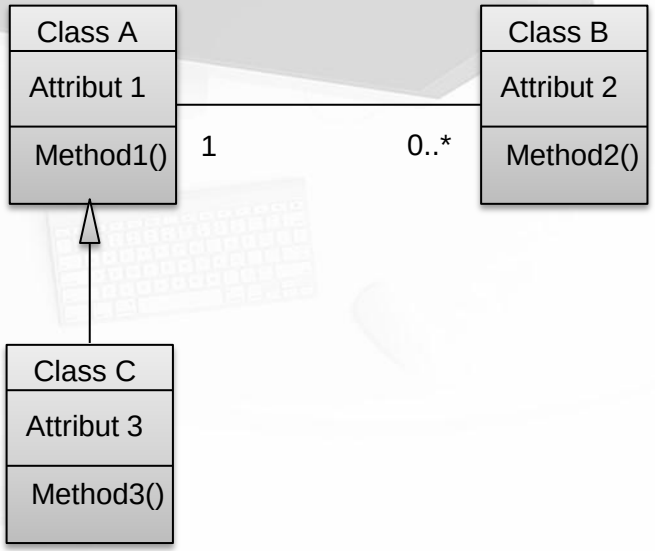
- The class diagram is probably the most important diagram to represent for object-oriented analysis methods. This is the focal point of any object-oriented development.
- The class diagram expresses the static structure of the system in terms of classes and the relationships between these classes
- The interest of the class diagram is to model the entities of the information system

2- Definition:

- A class diagram is a collection of static modeling elements that shows the structure of a model.
- A class diagram abstracts the dynamic and temporal aspects of the system
- A class is an abstract description of a set of objects in the domain of the application: it defines their structure, behavior, and relationships
- A class represents the description of a set of objects with the same characteristics.

3- Representation:

The class diagram implements classes, containing attributes and operations, and related by associations or generalizations.



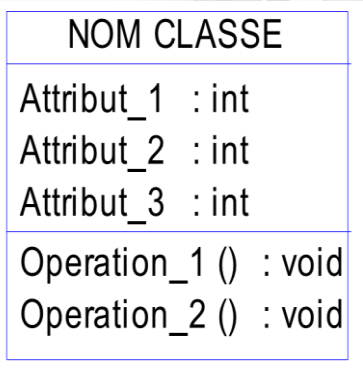
02 Representation of a class

3.1 - Class:

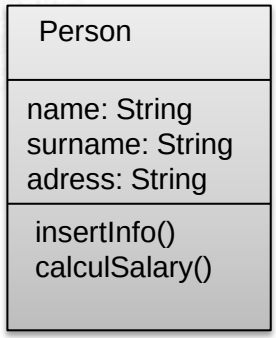
A class is represented by a rectangle separated into three parts:

- The first part represents the name of the class
- The second part represents the attributes of the class
- The third part represents the operations of the class

a. Formalism:



b. Example:



02 Representation of a class

3.2 - Attribute:

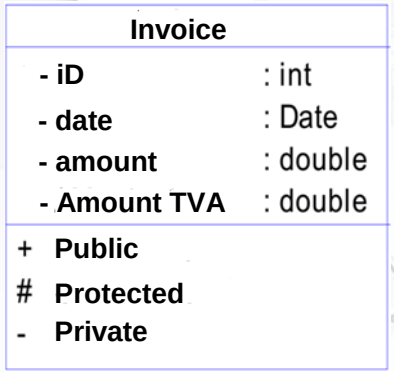
- ✓ An attribute represents the modeling of elementary information represented by its name and format
- ✓ UML defines 3 levels of visibility for attributes:
  - **public (+): The element is visible to all customers in the class**
  - **protected (#):The element is visible to the subclasses of the class**
  - **private (-): The element is visible only to objects in the class in which it is declared**

- ✓ The identifier is a particular attribute, which makes it possible to uniquely locate each object, instance of the class.
- ✓ For ease of management, it is sometimes chosen to keep in an attribute the result of a calculation made from other classes: this is called a derived attribute. To identify a derived attribute: place a "/" in front of its name.

02 Representation of a class

3.2 - Operation:

- ✓ The operation represents an element of object behavior, defined globally in the class.
- ✓ An operation is a feature that is performed by a class. The description of the operations can specify the input and output parameters as well as the basic actions to be performed.
- ✓ As with attributes, there are 3 levels of visibility for operations



03 Types of relations between classes

- ✓ There are several types of relationships between classes:

1 - Association:

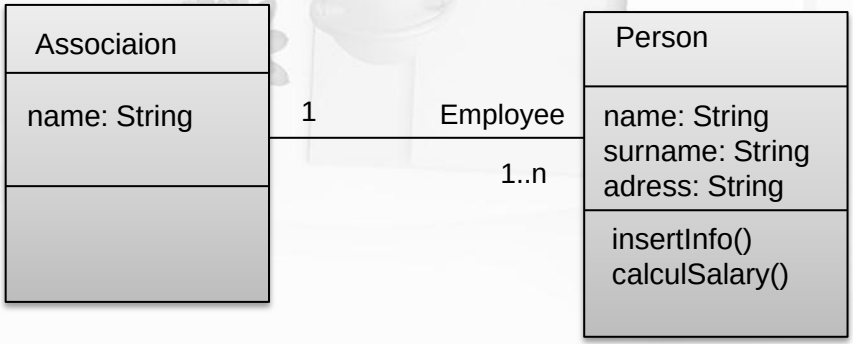
- ✓ An association is a relationship between two classes that describes the structural connections between their instances. An association therefore indicates that there may be links between instances of the associated classes.
- ✓ An association is a static n-ary relation (most often: it is binary): that is to say that it links several classes together.
- ✓ An n-ary association has n roles which are the terminal points of the association. Each class that participates in the association plays a role. Roles are defined by 2 properties:
    - Role name = indication of the class's participation in the association.
    - Multiplicity = Defines the number of instances of the association for an instance of the class.

03 Types of relations between classes

1.1 - Example:

In this diagram, the name of the association is Working, the name of the Person class role for the association is: Employee. The diagram reads as follows:

- ✓ One person works for a single company.
- ✓ In a company there are one to several people:

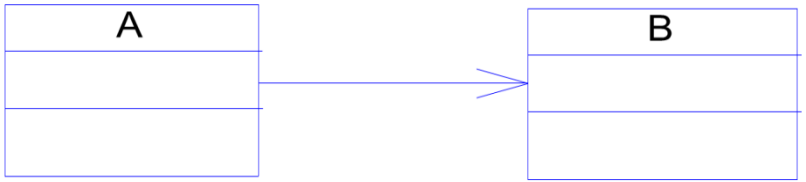


Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more
n	Only *n* (where *n* > 1)
0..n	Zero to *n* (where *n* > 1)
1..n	One to *n* (where *n* > 1)

03 Types of relations between classes

1.2 – Airworthiness:

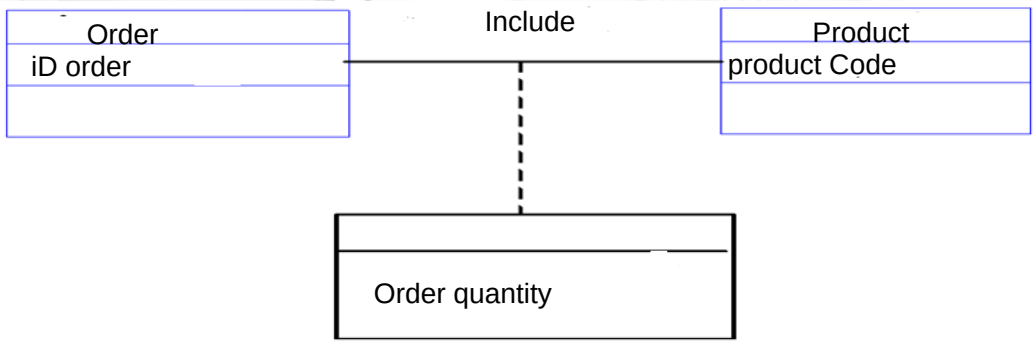
- An airworthiness placed on a target endpoint indicates whether this role is accessible from the source.
- By default, associations are navigable in 2 directions, and in some cases, only one navigation direction is useful: the end of the association towards which navigation is possible then carries an arrow.
- In the example below, instances of A see instances of B but instances of B do not see instances of A.



03 Types of relations between classes

1.3 – The association class:

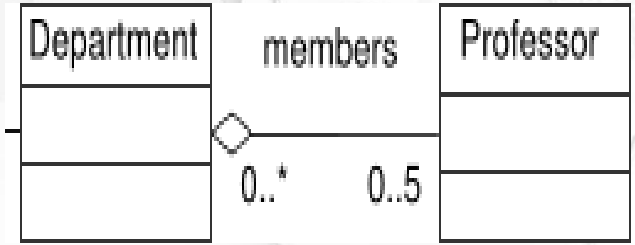
- The attributes of a class depend functionally on the class identifier. Sometimes, an attribute functionally depends on two identifiers, belonging to two different classes.
- For example, the "order quantity" attribute is functionally dependent on the order number and the "product code".
- We will therefore place the "quantity ordered" attribute in the “include" association.



03 Types of relations between classes

1.4 – Aggregation:

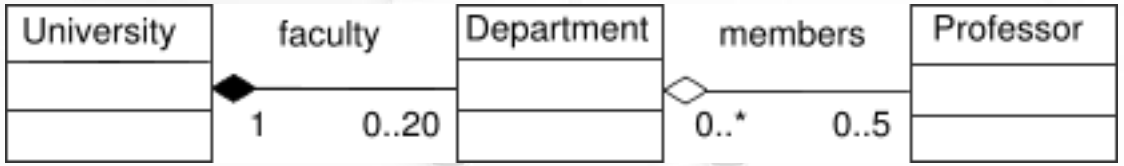
- Aggregation represents a non-symmetric association in which one of the ends plays a predominant role over the other end.
- The aggregation is always represented with a small diamond on the side aggregation.
- It is asymmetrical of the "set/element" or "container/content" type



03 Types of relations between classes

1.5– Composition:

- Composition is a special case of aggregation in which the life of the components is linked to that of the aggregates. It often refers to a physical countenance.
- Composition implies, in addition to aggregation, a coincidence of component lifetimes: the destruction of the aggregate (or container) automatically implies the destruction of all related components



03 Types of relations between classes

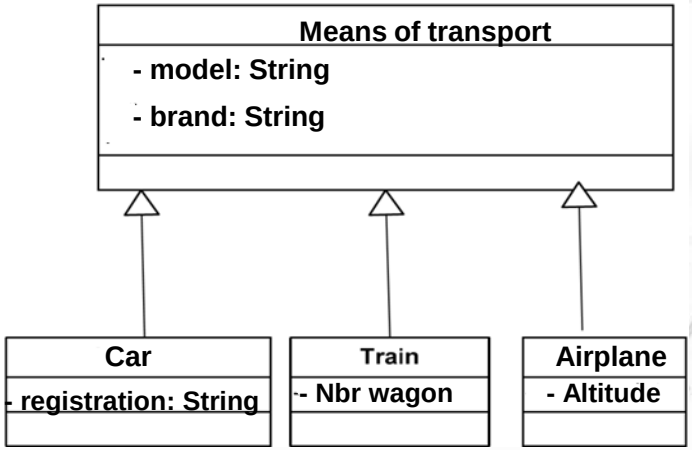
2– Generalization / specialization:

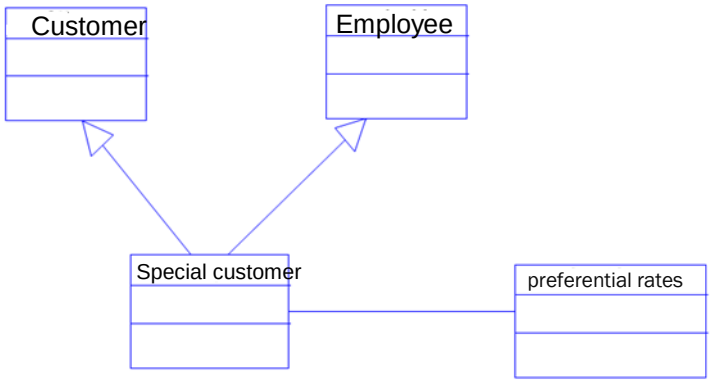
- The principle of generalization/specialization makes it possible to identify among the objects of a (generic) class subsets of objects (specialized classes) with specific definitions.
- The more specific class (also called daughter class, derived class, specialized class, descendant class...) is consistent with the more general class (also called parent class, general class...), i.e. it contains by inheritance all the attributes, members, relations of the general class, and can contain others.
- A class can have several parents, which is called multiple inheritance
- An abstract class is a class that does not instantiate directly, but represents a simple abstraction in order to factor the common properties of subclasses. It is noted in italics.

03 Types of relations between classes

2.1 - Examples:

- The "Means of transport" class is an abstract class
- Multiple generalization consists of merging several classes into a single class. The "special customer" class is a customer and employee specialization. This model makes it possible to indicate that preferential rates are granted to employees.





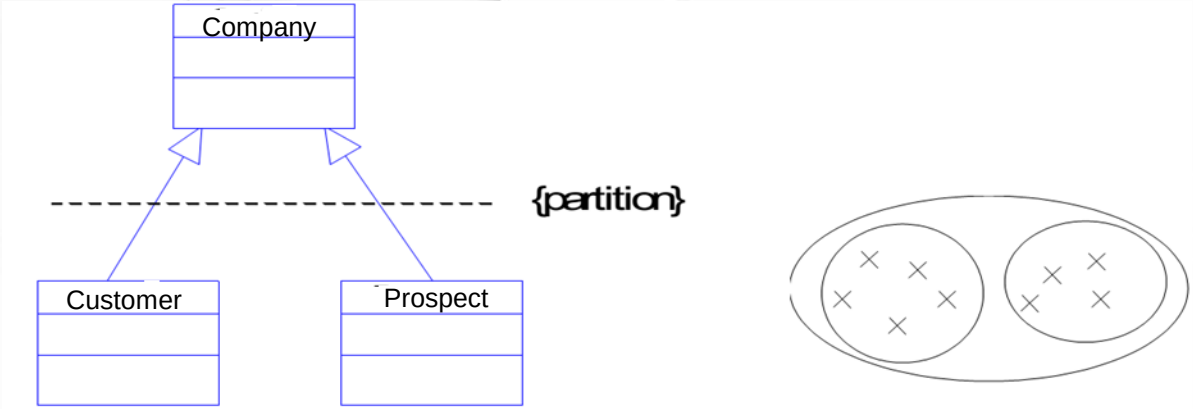
03 Types of relations between classes

2.2 – Constraints on associations:

- There are several types of constraints on associations:

a– Partition constraint:

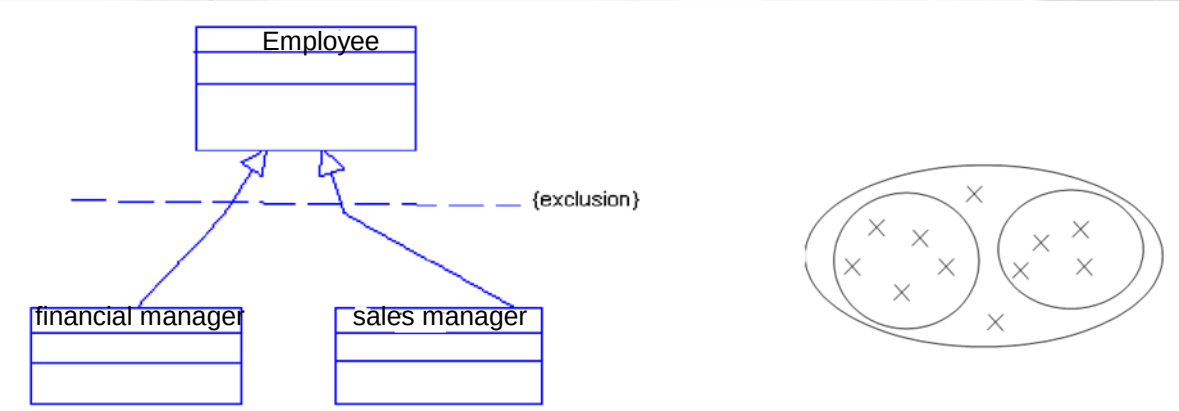
- It indicates that all instances of a class correspond to one and only one instance of the related classes.
- All companies are either customers or considered prospects.



03 Types of relations between classes

b– Exclusion constraint:

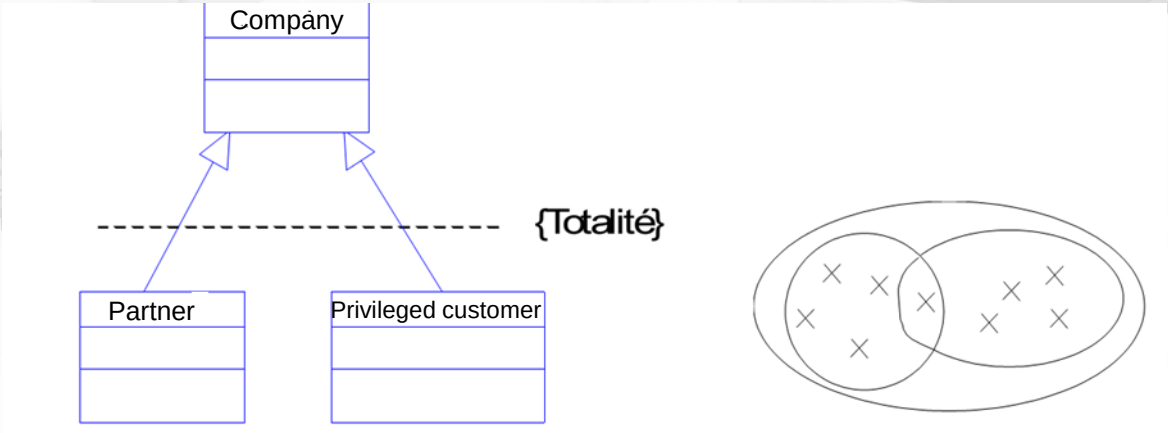
- It is used to specify that an association instance excludes another instance.
- For example, an employee cannot be both a financial manager and a sales manager.



03 Types of relations between classes

c– Constraint of totality:

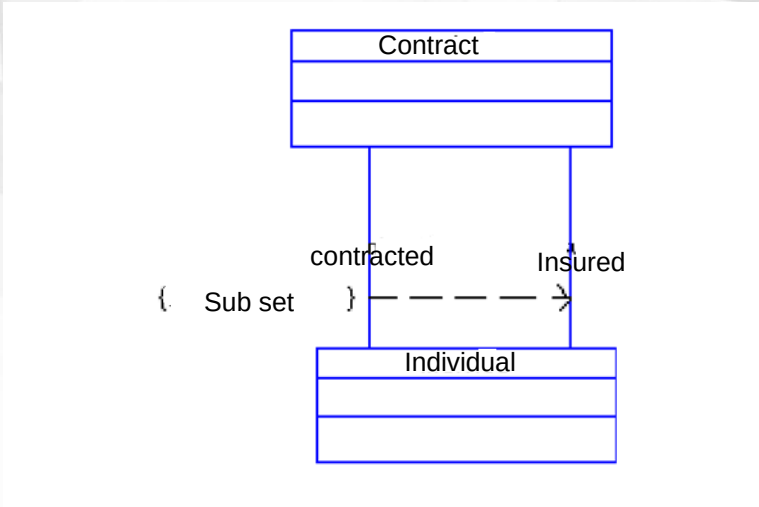
- All instances of a class match at least one of the instances of the related classes.
- Every company is at least a partner or privileged customer. And it can be both at the same time.



03 Types of relations between classes

d– Inclusion constraint:

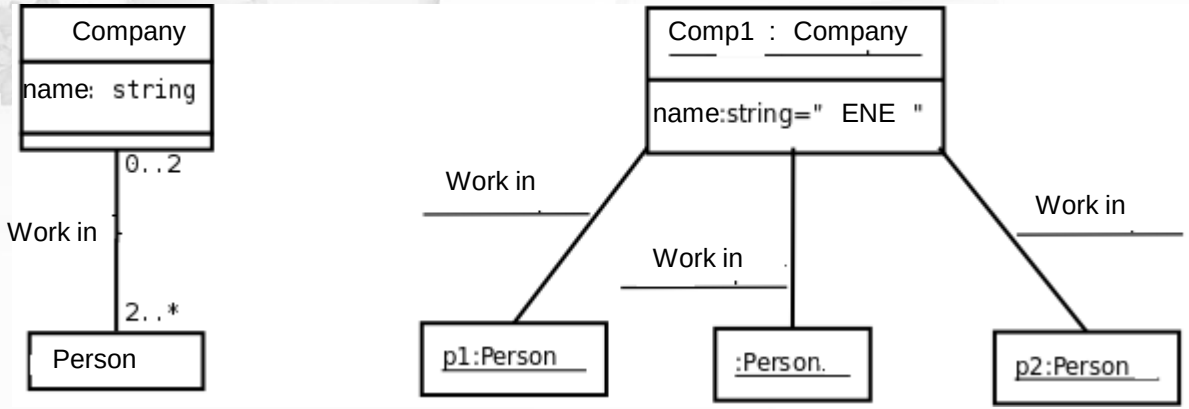
- It makes it clear that a collection is included in another collection. (the arrow of the dependency relation indicates the direction of the constraint).
- For example, it may be indicated that the contracting party to a contract is necessarily one of the insured individuals



04 Object diagram

1- Definition:

- The object diagram is used to highlight connections between objects. Objects, instances of classes, are connected by links, instances of associations.
- With the exception of multiplicity, which is explicitly stated, the object diagram uses the same concepts as the class diagram. They are basically used to understand or illustrate complex parts of a class diagram.



04 Object diagram

1- Representation:

- Graphically, an object is represented as a class. However, the operations compartment is not useful. In addition, the name of the class whose object is an instance is preceded by a <<: >> and is underlined. To differentiate between objects of the same class, their identifier can be added in front of the class name.

- In an object diagram, the relationships in the class diagram become links. The generalization relation does not have an instance, so it is never represented in an object diagram. Graphically, a link is represented as a relationship, but, if there is a name, it is underlined.

Exercise

Consider the following sentences:

1. A directory contains files
2. A room contains walls
3. Modems and keyboards are input/output devices
4. A stock market transaction is a purchase or sale
5. A bank account may be owned by a natural or legal person
6. Two people can be married
7. A person uses a programming language in a project
8. A Payer Owns a Capital
9. Mohammed programs his flight simulator in java
10. Java, Eiffel are object-oriented languages

- Determine the appropriate static relationship (generalization, instantiation, aggregation, composition, link, or association) in each sentence of the previous statement and determine the degree of the relationship in the c ase of an association.
- Draw the class diagrams corresponding to phases 4, 6, 8, 9.

Bibliographies

- **Uml 2 pratique de la modélisation**, Benoît Charroux, Yann Thierry-Mieg, Aomar Osmani https://fr.slideshare.net/nassimamine3994/uml-2-pratique-de-la-modlisation
- **Uml 2 par la pratique, Pascal roques**
- **Les cahiers du programmeur, Pascal roques**
- **Uml en action, Pascal roques**
- **.....**