# Object-Oriented Analysis

## Lecture 3 : Class Diagram

Nga Nguyen, EISTI

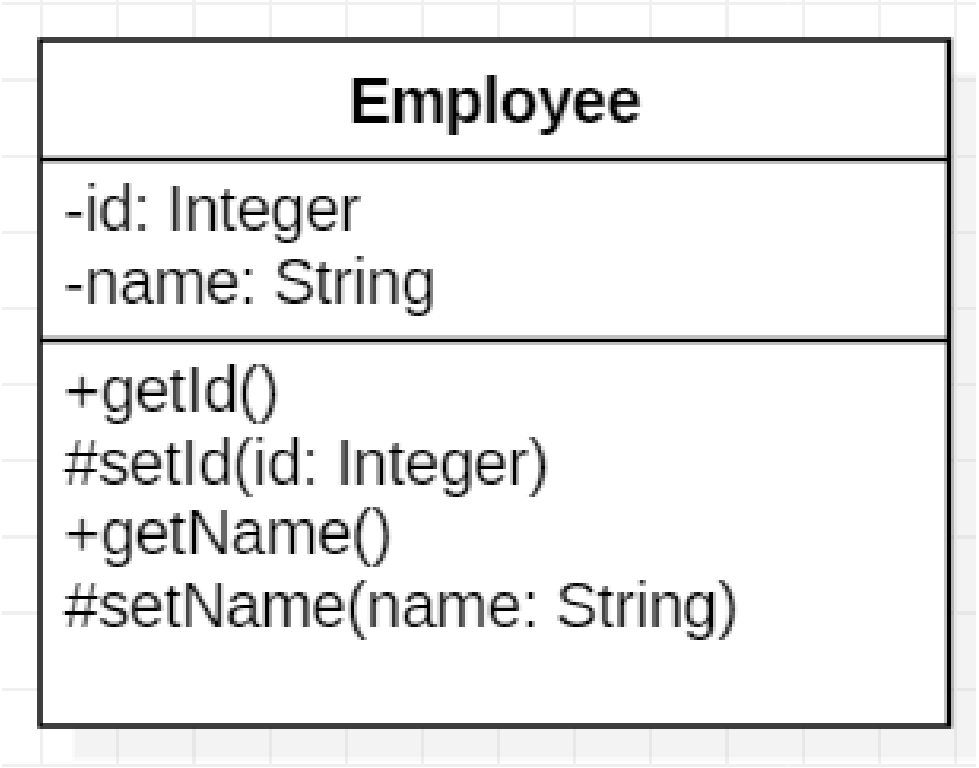
# Recall : Four Object-Oriented Concepts

- State + behavior
- Encapsulation
- Inheritance / generalization / specialization
- Polymorphism

# Class : visibility of attributes/operations

| Access right | public (+) | private (-) | protected (#) | package (~) |
|---|---|---|---|---|
| Members of the same class | yes | yes | yes | yes |
| Members of derived classes | yes | no | yes | yes |
| Members of any other class | yes | no | no | in same package |

# Example

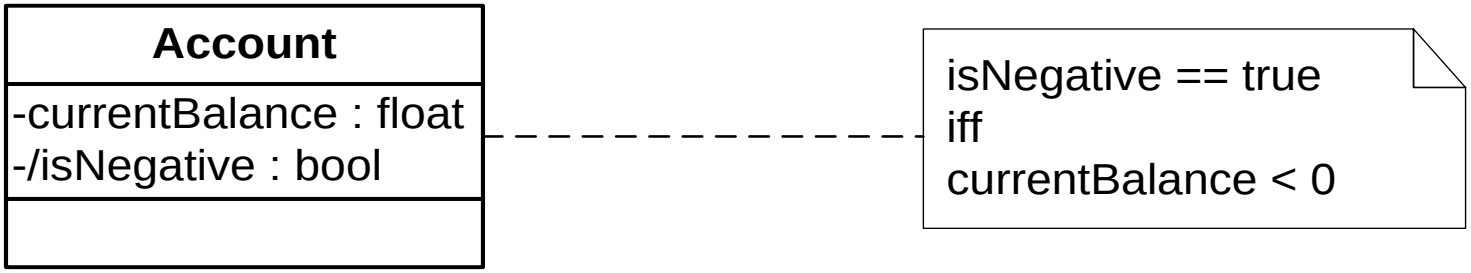



Hide the values or state of a object inside a class, preventing unauthorized parties' direct access to them.

=> Encapsulation

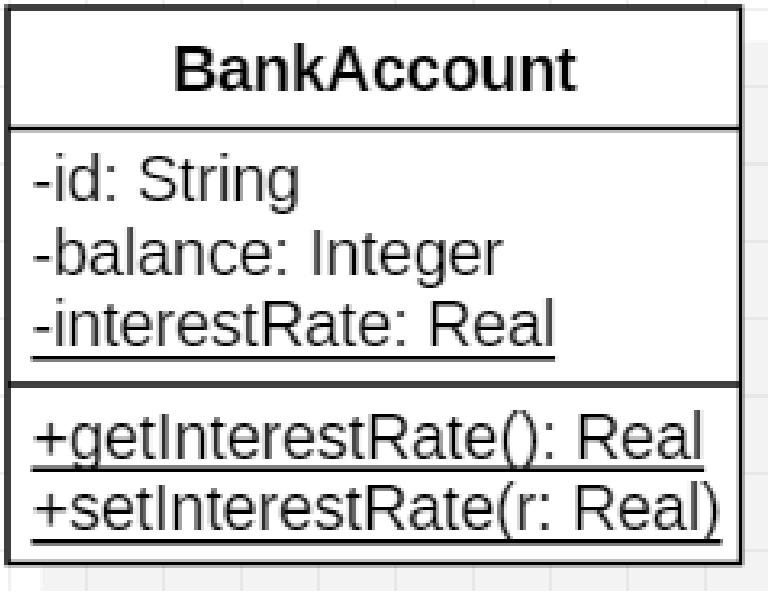
# Class : derived attributs

- A *derived attribute* is an attribute whose value can be computed from other attributes already in the model.



# Class attribute and Class operation

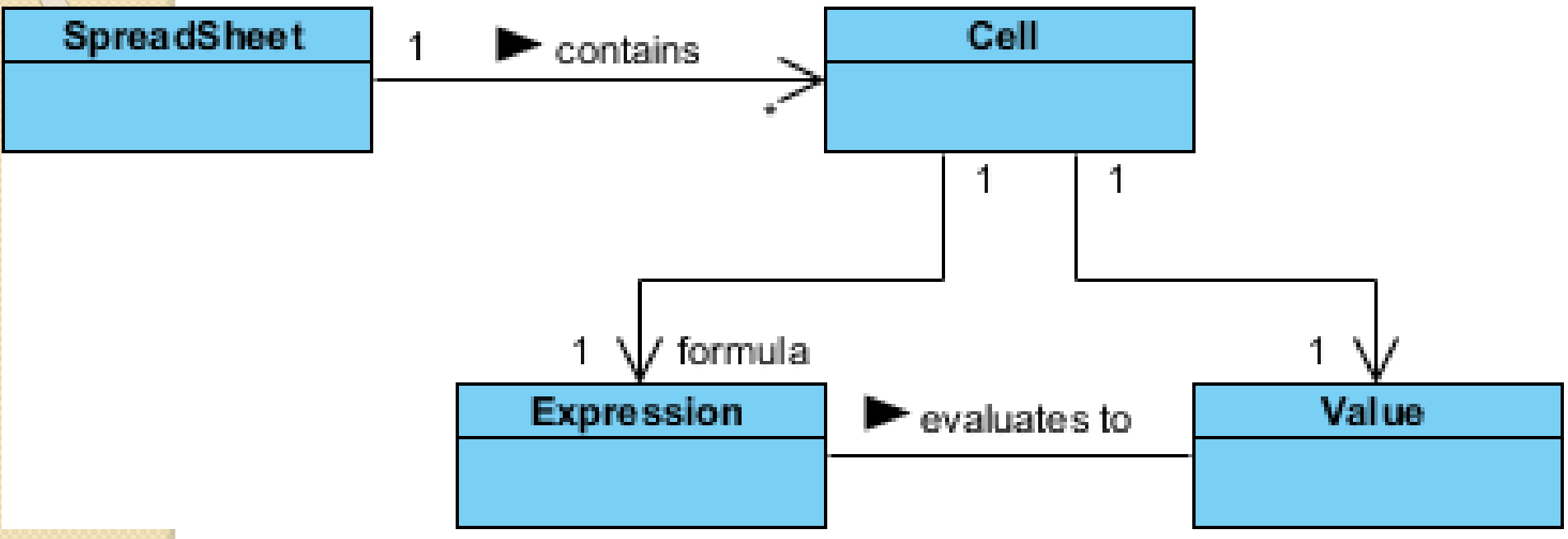
- A class attribute or class operation is an attribute or operation belonging to a class rather than the instances of the class.
- Graphically, they are underlined in UML (equivalent to static in Java or C++)
- Example



# Relationship between classes

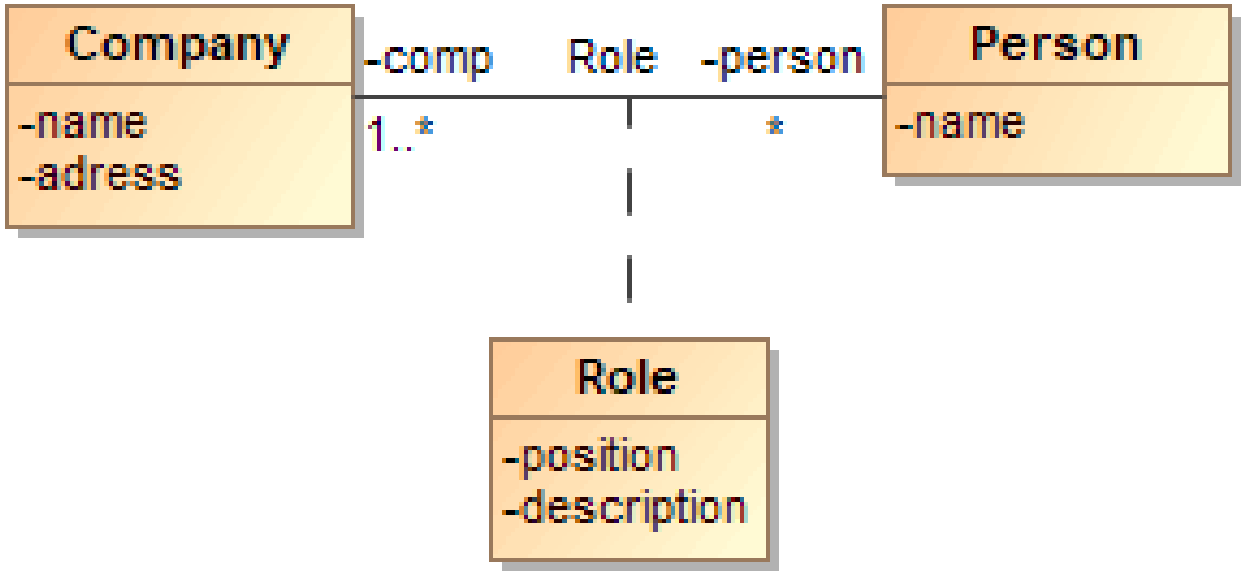
- Characteristics :
  - name
  - navigability : uni or bidirectional
  - multiplicity : how many objects of each class take part in the relationship
  - role : describes the purpose played by a class in the relationship
  - association class

# Example



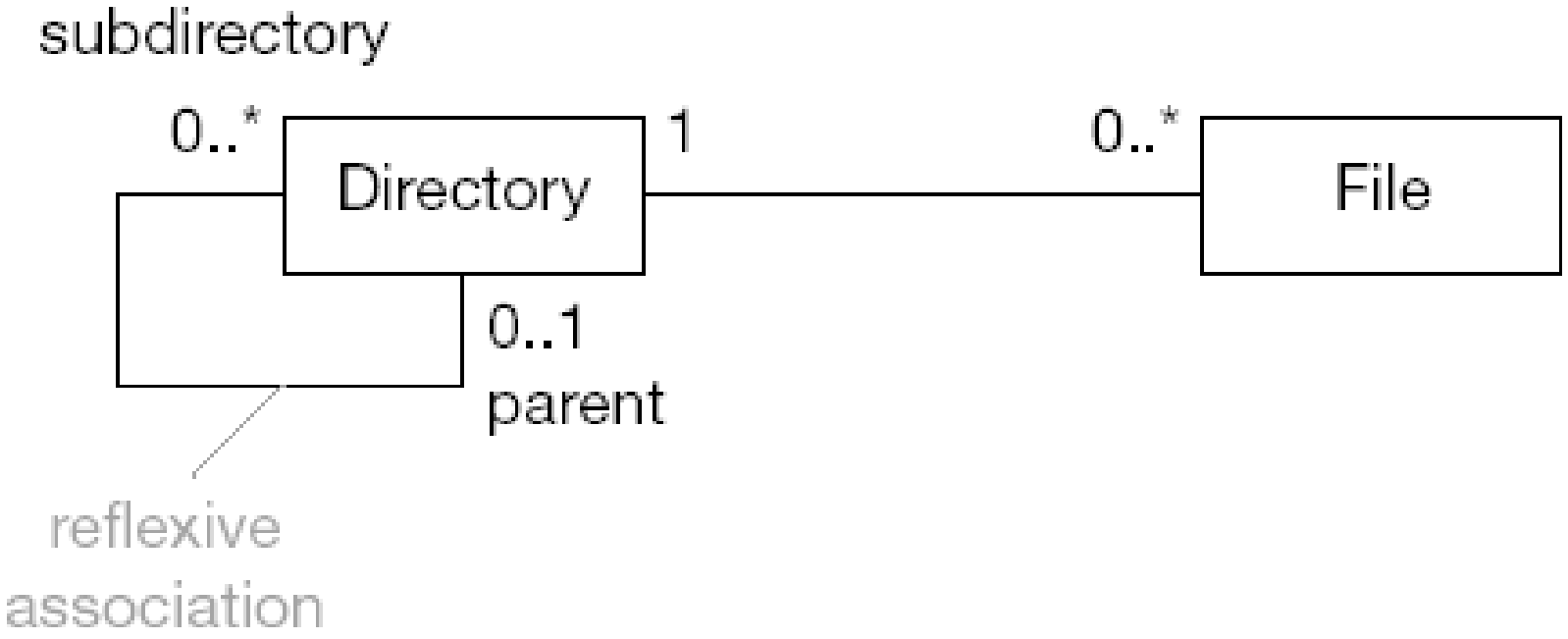

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

# Association class

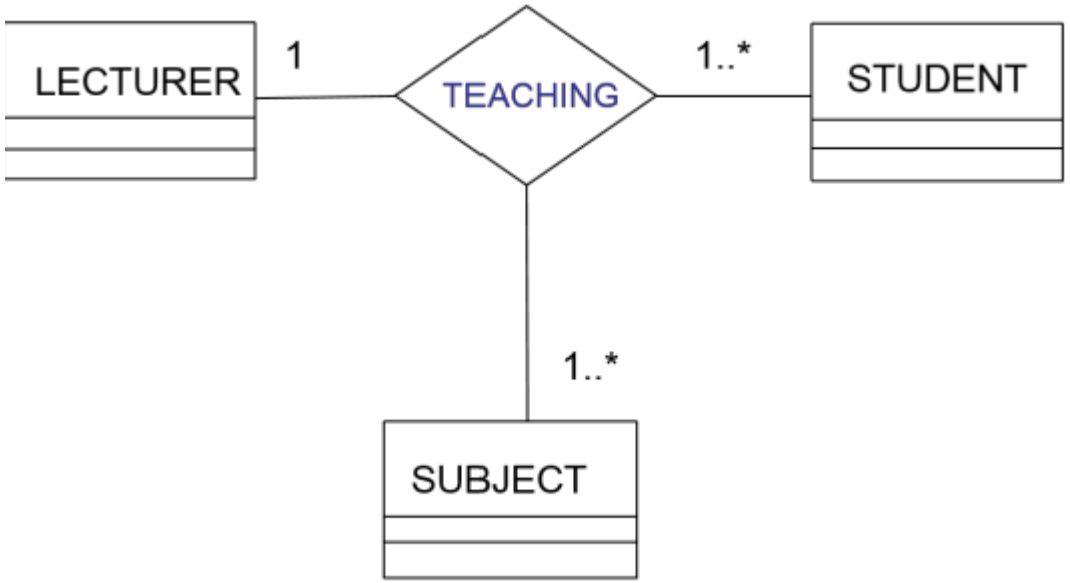



- An association class is a class that is part of an association relationship between two other classes

# Reflexive Association



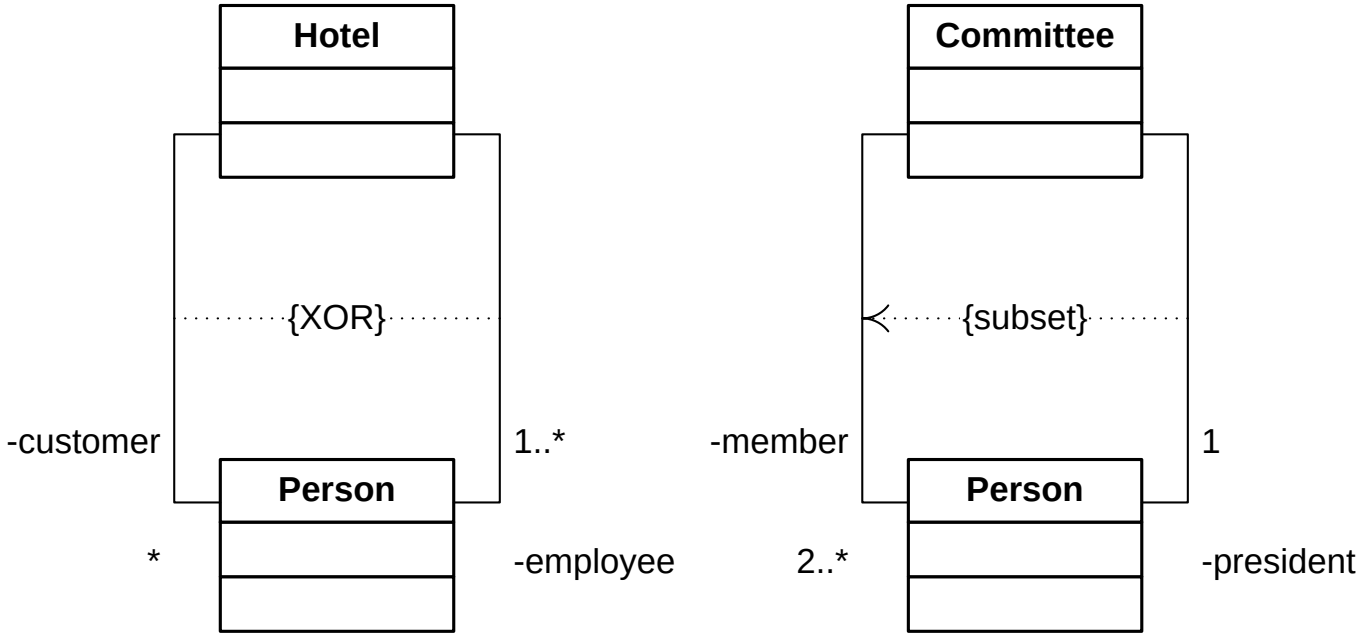
# N-ary Association



# Association Constraints

- {XOR}
- {subsets <property_name>}
- {redefines <property_name>}
- {union}
- {ordered}
- {bag}
- {sequence} ou {seq}

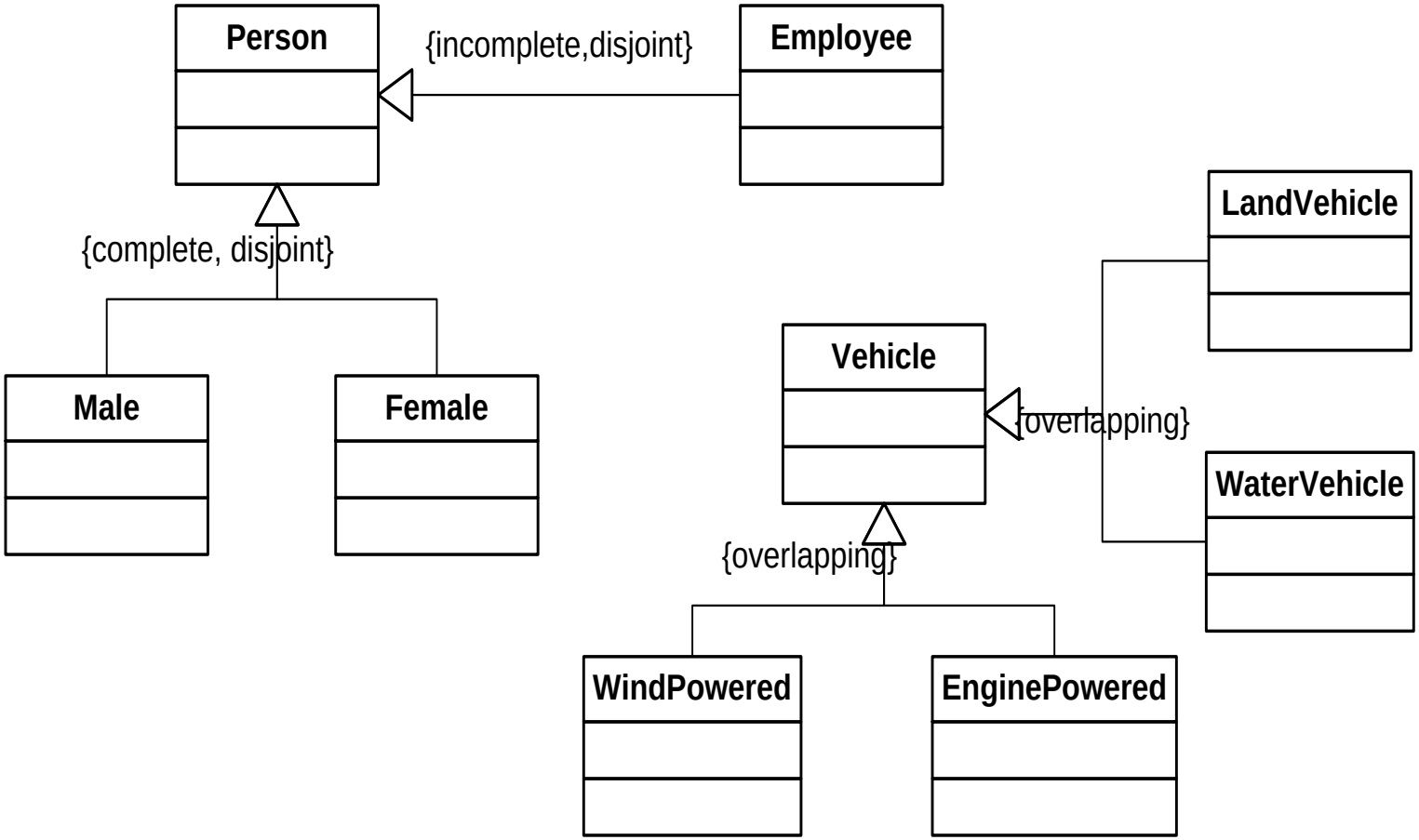
# Association Constraints : examples



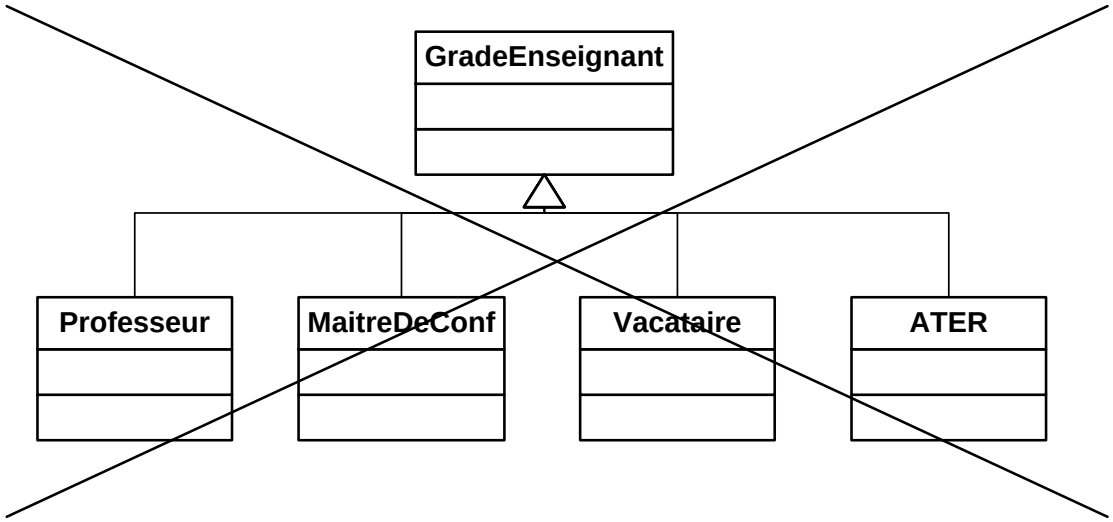
# Inheritance Constraints

- {complete, disjoint}
  - not extendable, no common instance
- {incomplete, disjoint}
  - extendable, no common instance
- {complete, overlapping}
  - not extendable, with common instances
- {incomplete, overlapping}
  - extendable, with common instances
- By default : {incomplete, disjoint}

# Inheritance Constraints : examples



# Enumerations




«enumeration»
**GradeEnseignant**

+Professeur
+MaitreDeConf
+ATER
+Vacataire

# Interface

- **Encapsulation** : an object is accessible from the outside only through its public operations.
- The declaration of an operation consists of:
  - the name
  - the parameters
  - the return type

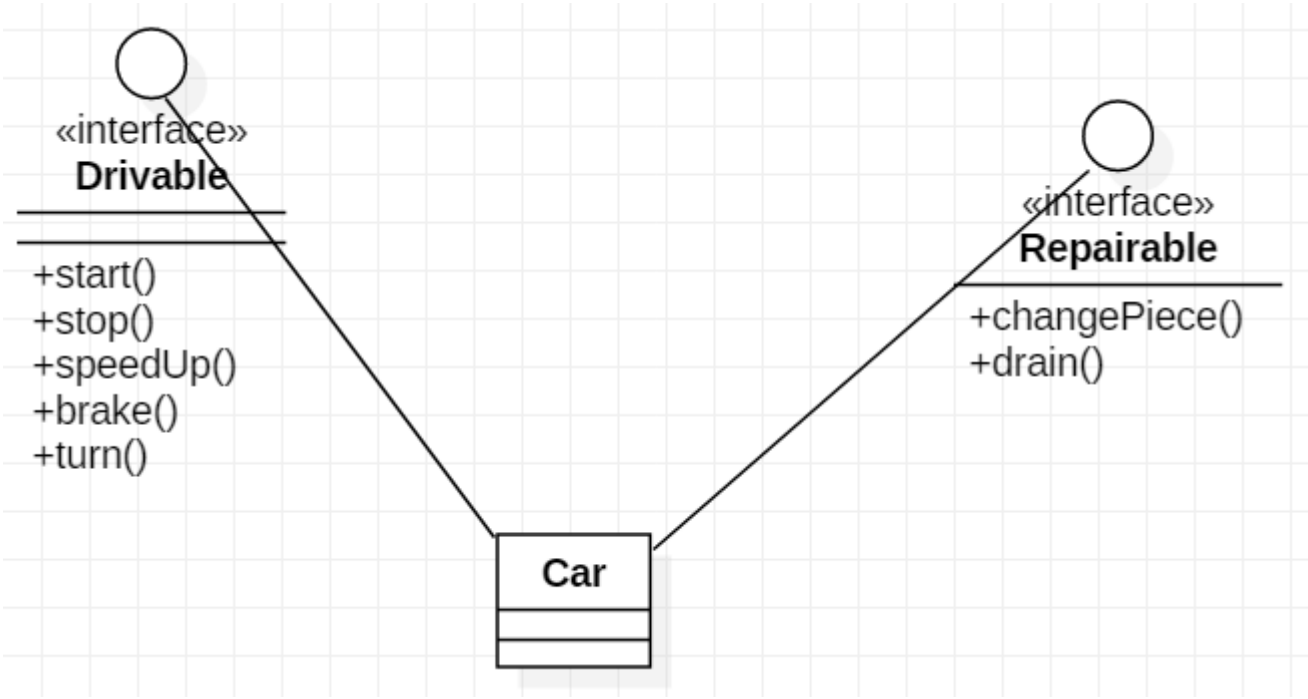
  => signature of the operation
- The set of signatures of the public operations of an object is called the interface of the object.

# Interface

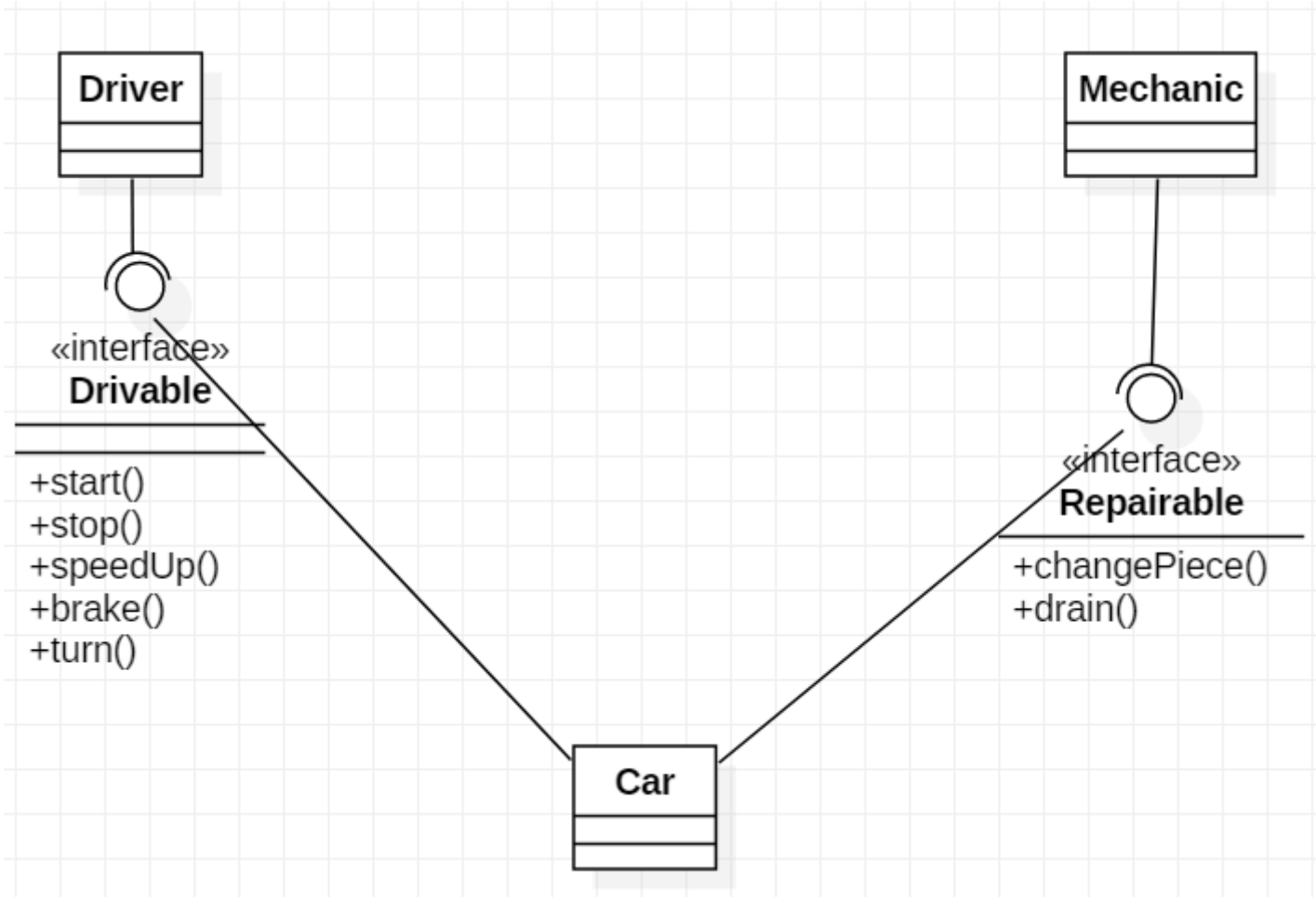
- The interface is the external view of an object, it defines the accessible services offered to the users by the object.
  - Hide the implementation details of an object (encapsulation)
  - Guarantee the integrity of data contained in the object.

# Interface : example

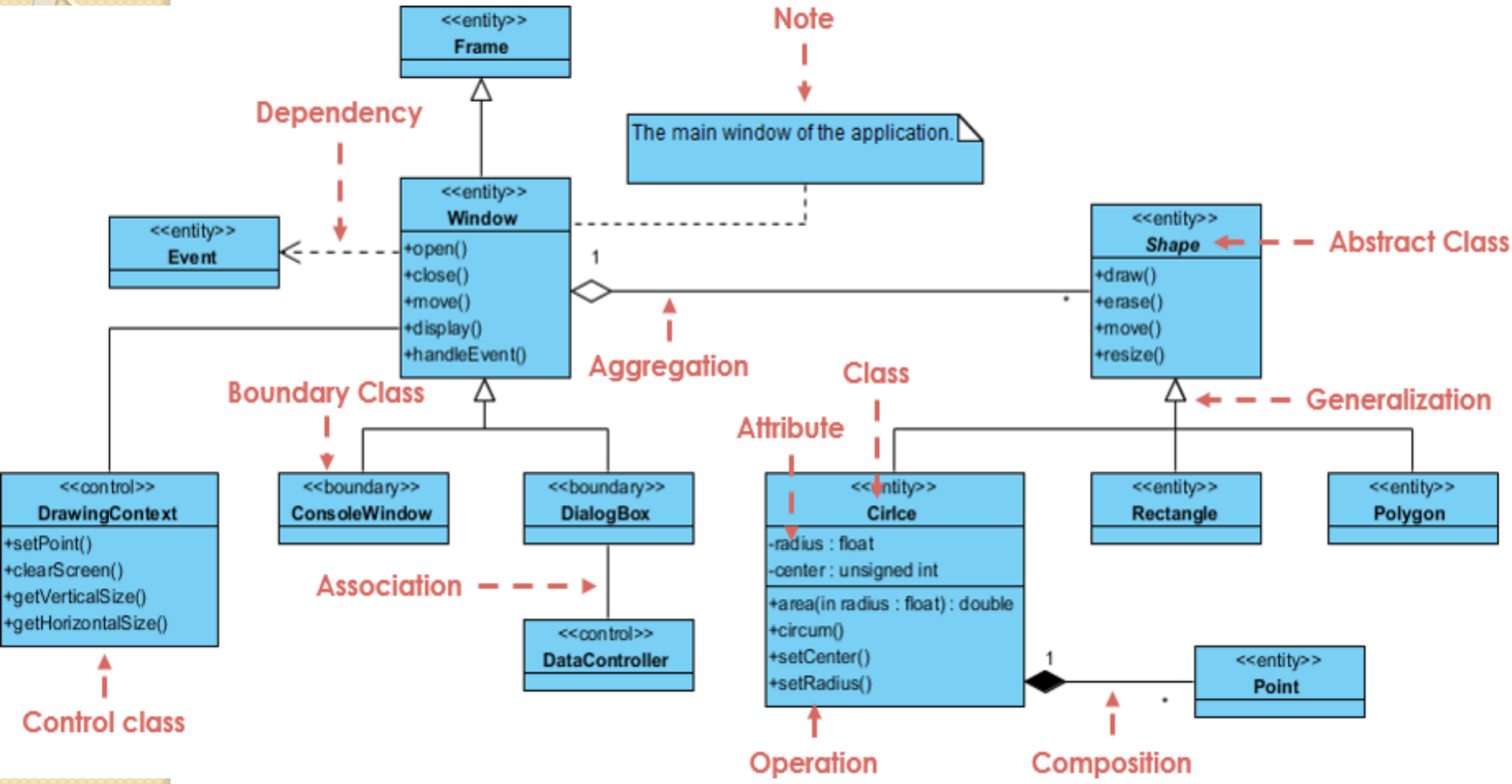
- A car :
  - **drive**: start, speed up, brake, turn, stop, …
  - **repair**: change piece, drain, …



# Interface : example



# Class Diagram




https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

# EISTI Project

- Each project has a name and takes place in a semester of a given school year.
- Among the projects, you have projects in Computer science with analysis, design and implementation parts.
- A project is divided into several different deliverables, with a number, a description and a deadline. There are at least 3 deliverables per project.
- Students form project groups of size 3 or 4. You must submit the work for each deliverable before the deadline!
- The work is evaluated by group and by deliverable.
- Propose a class diagram with
  - classes : attributes, methods
  - associations : name, multiplicity, roles, …
  - association class ?
    - aggregation, composition,  inheritance ?