Chapter 5. UML Class Diagram

Reyes Grangel Seguer

EI1030. Analysis of Inf. Systems / EI1032. Software Analysis / MT1046. Integrated Inf. Systems
Computer Engineering / Computational Mathematics
Dept. of Computer Systems and Languages

February 27, 2019

Outline

1. Class Diagram
   - Overview
   - Basic Elements
   - Relationships
   - Interfaces

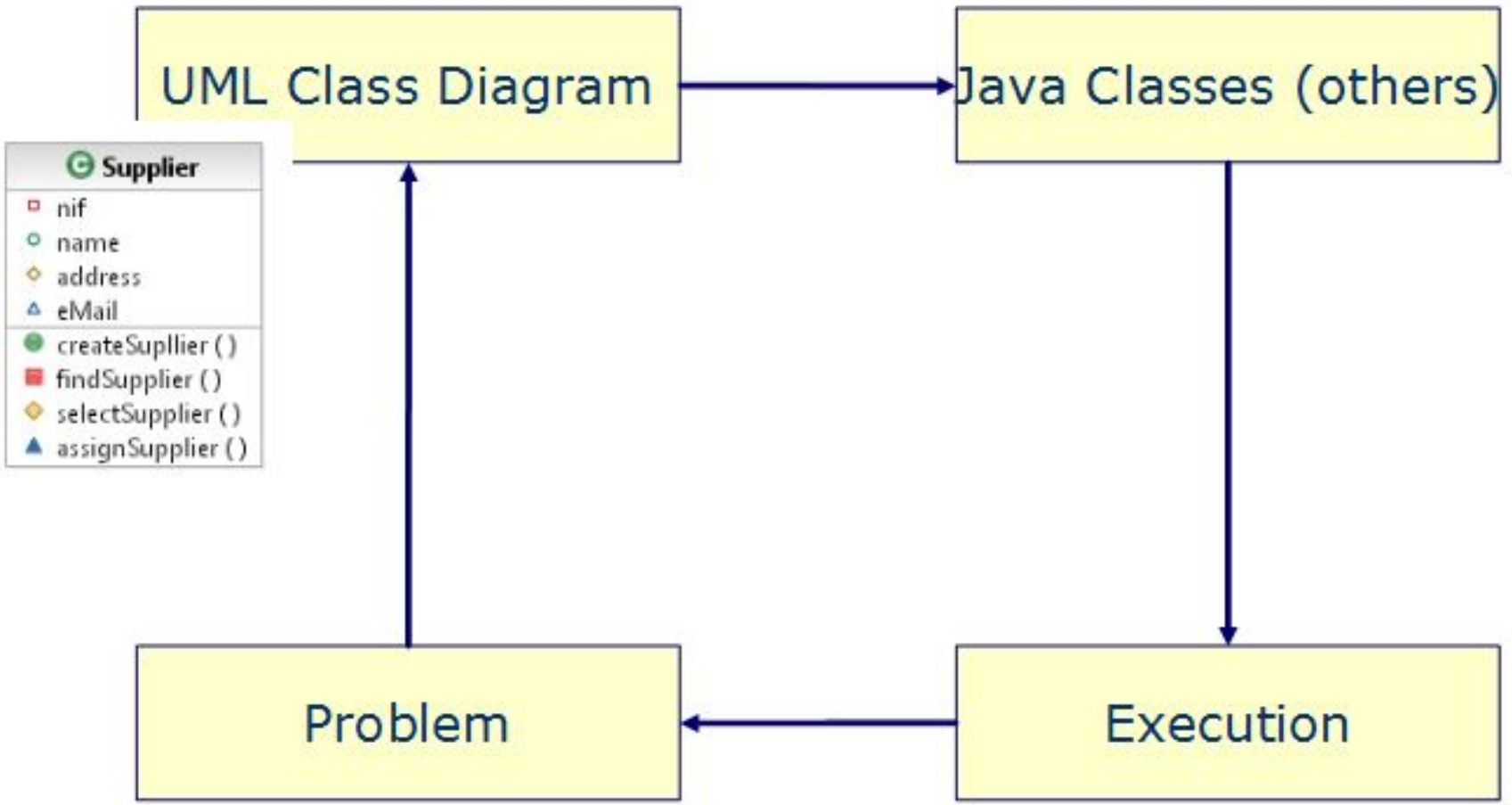
Outline

1. Class Diagram
   - Overview
   - Basic Elements
   - Relationships
   - Interfaces

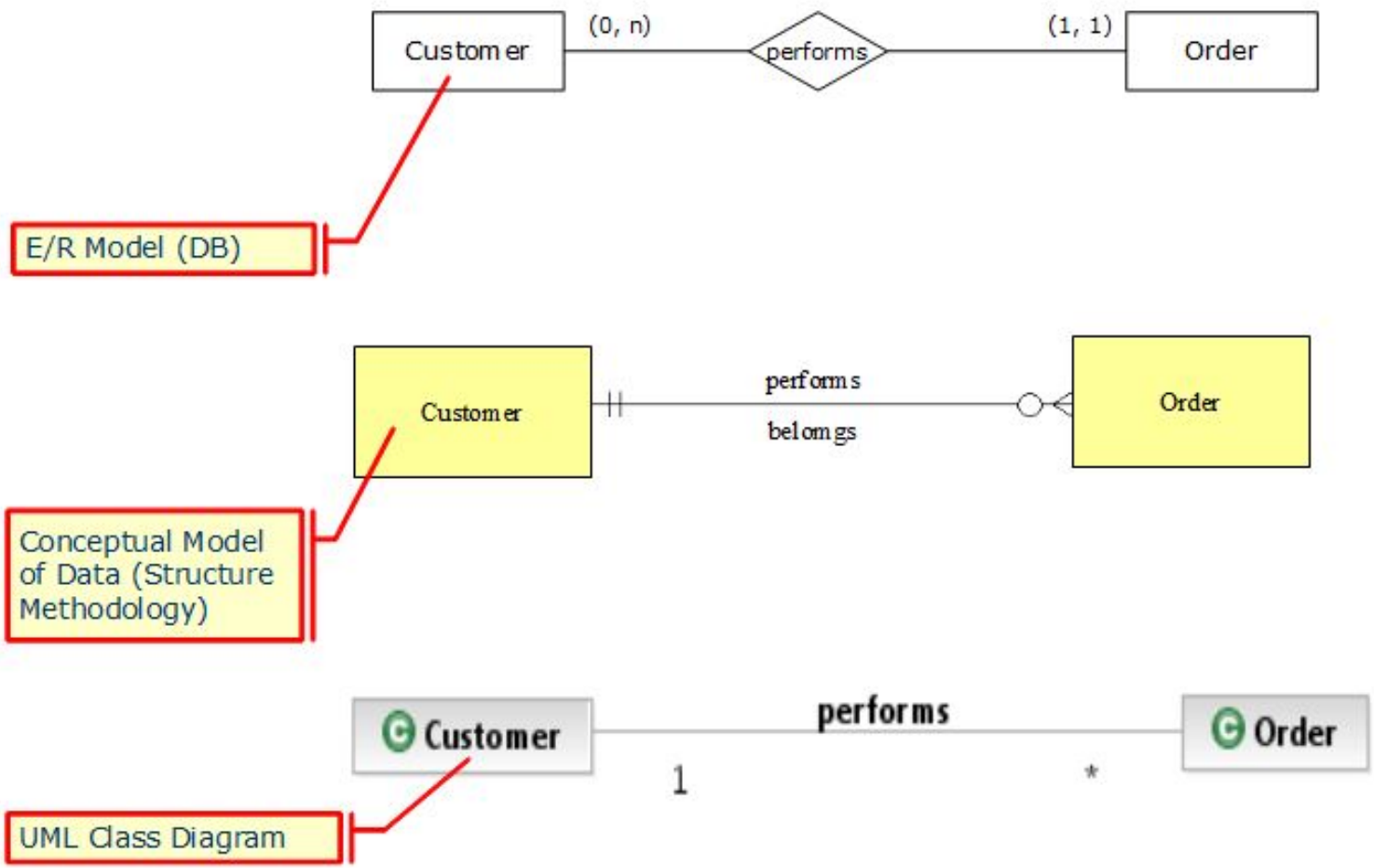
Features

- To show the information structure of the system
- Also, to show behaviour of the classes
- It can be used in different phases of the development process
- It can be refined in successive iterations of the development process
- It can be used to show
  - System vocabulary
  - The main elements of the domain
  - Information structure
  - The logic schema of the database
  - etc.

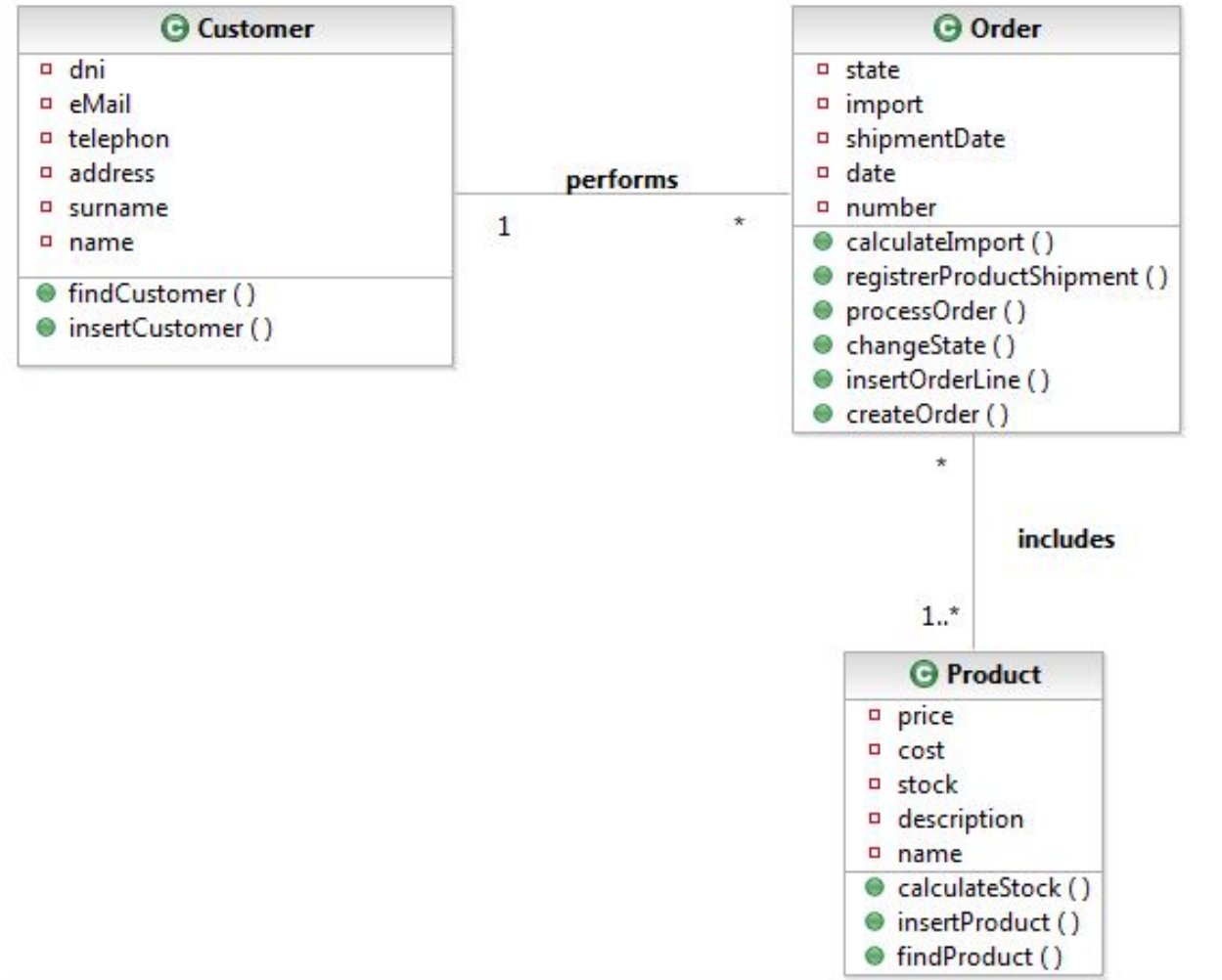
Objective


UML Class Diagram
Java Classes (others)
Problem
Execution
Supplier
nif
name
address
eMail
createSupllier ( )
findSupplier ( )
selectSupplier ( )
assignSupplier ( )

ER Model vs Class Diagram



Example: Class Diagram



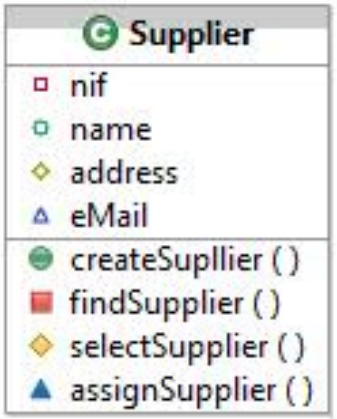
Class

What does it represent?

A set of objects with an equivalent role in the system, i.e. that they share attributes, operations, relationships and semantic

Notation

- Capital letter for the first character of the name
- Abstract classes in italic



Main Types of Relationships

- **Association**: to exchange message
  - **Association Class**: to have properties of association and class
  - **Aggregation** and **Composition**: to show structural relationship of composition
  - **Generalisation**: to show hierarchy
- **Dependency**: to show use relationship

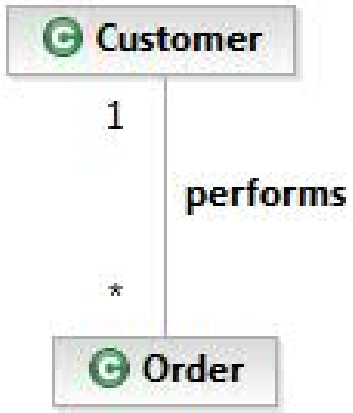
Association

What does it represent?

A nexus among two or more classes which has some specific goal

Notation

Verb in third person of singular



Multiplicity

Notation

- **Exact number**: 1, 2, 5, 10, etc.
- **minimum value .. maximum value**
  - **Value range**: 0..1, 1..10, 0..11, 1..25, etc.
  - **No specific number**: 1..*, *

Specification		Tools	
Meaning	UML2	MagicDraw	IBM RSM
1..1	1	1	1
0..*	*	0..*	*
1..*	1..*	1..*	1..*

Instance Level

It is shown in an Object Diagram by means of ...

- **Object**: is an instance of a **class**
- **Link**: is an instance of an **association**

Attributes

What does it represent?

Any property of modelled element, which is shared by all objects of the class

Notation

- Not compulsory
- Nomenclature and Data Type
- **Properties**: isReadOnly, isDerived, etc.
- **Visibility**
  - Public (+)
  - Private (-)
  - Protected (#)
  - Package (~)

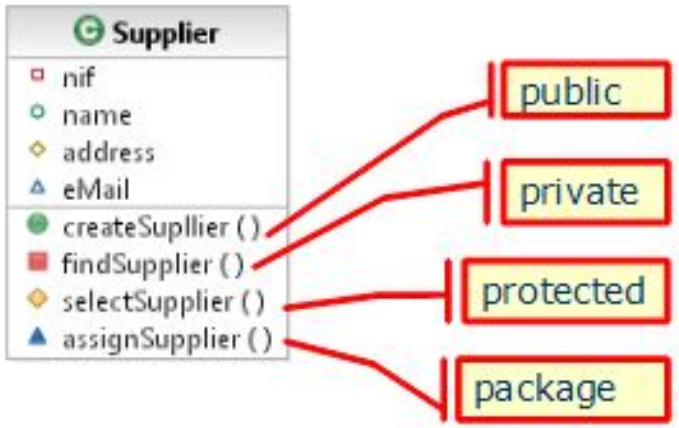
Operations

What does it represent?

Any function of modelled element, which is shared by all objects of the class

Notation

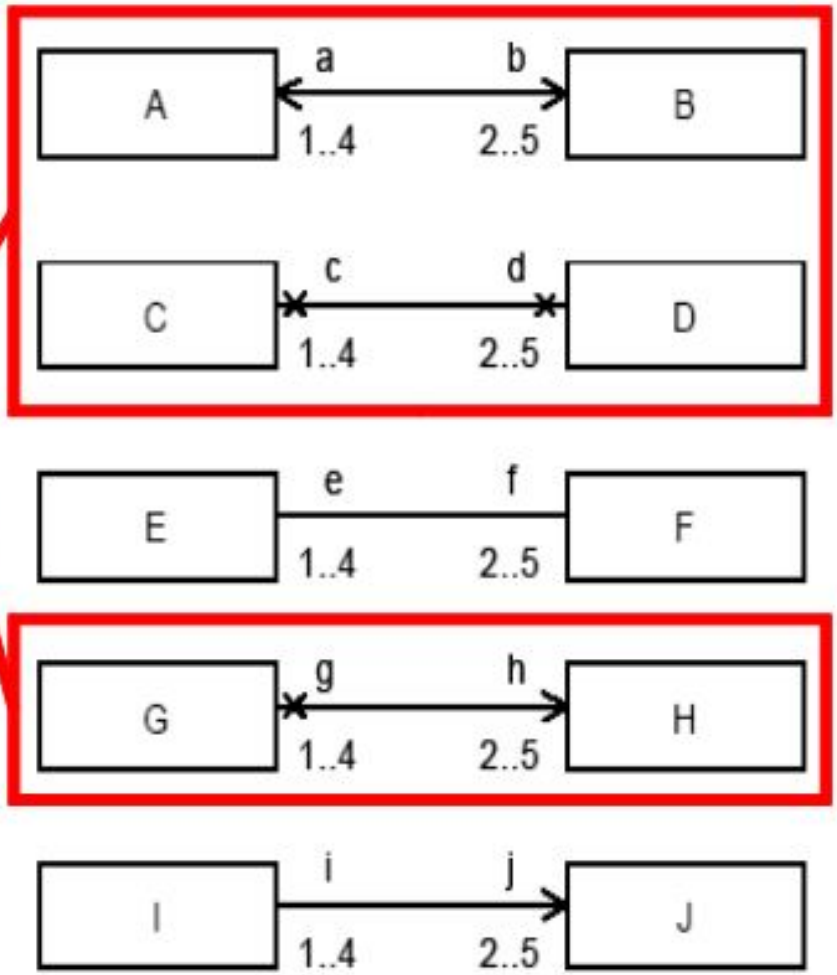
- Not compulsory
- Nomenclature and Data Type of parameters and return (**signature**)
- **Properties**: isQuery, isUnique, etc.
- **Visibility**
  - Public (+)
  - Private (-)
  - Protected (#)
  - Package (~)



Association Navigability (I)

Various options may be chosen for showing navigation arrows on a diagram. In practice, it is often convenient to suppress some of the arrows and crosses and just show exceptional situations:

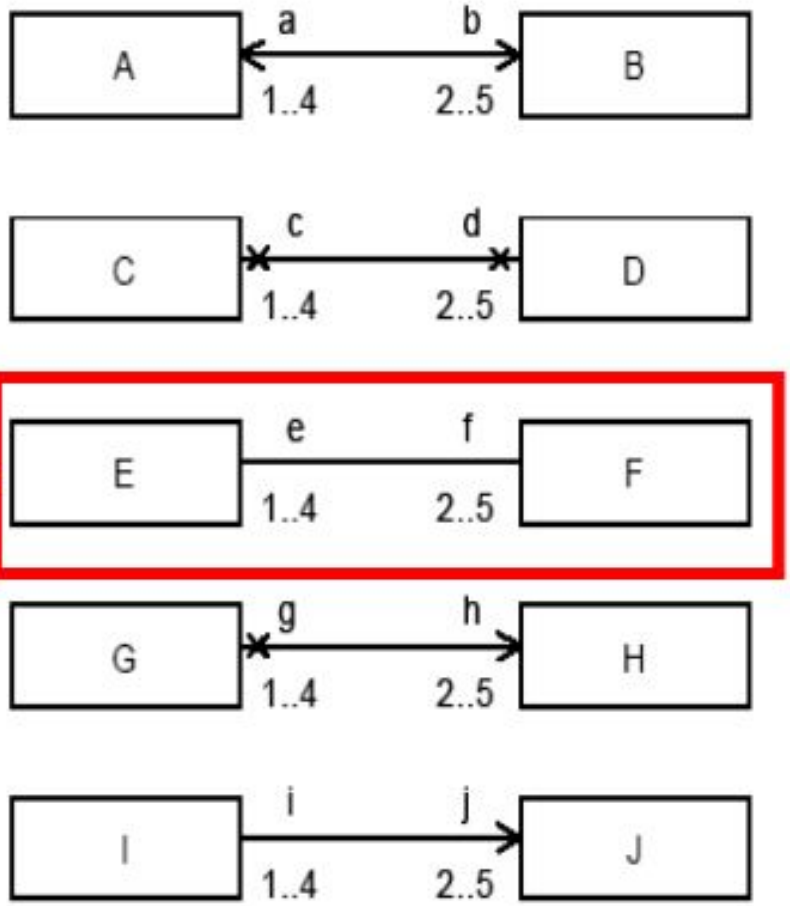
- Show all arrows and x's. Navigation and its absence are made completely explicit
- Suppress all arrows and x's. No inference can be drawn about navigation. This is similar to any situation in which information is suppressed from a view
- **Suppress arrows for associations with navigability in both directions, and show arrows only for associations with one-way navigability. In this case, the two-way navigability cannot be distinguished from situations where there is no navigation at all; however, the latter case occurs rarely in practice**



Association Navigability (II)

Various options may be chosen for showing navigation arrows on a diagram. In practice, it is often convenient to suppress some of the arrows and crosses and just show exceptional situations:

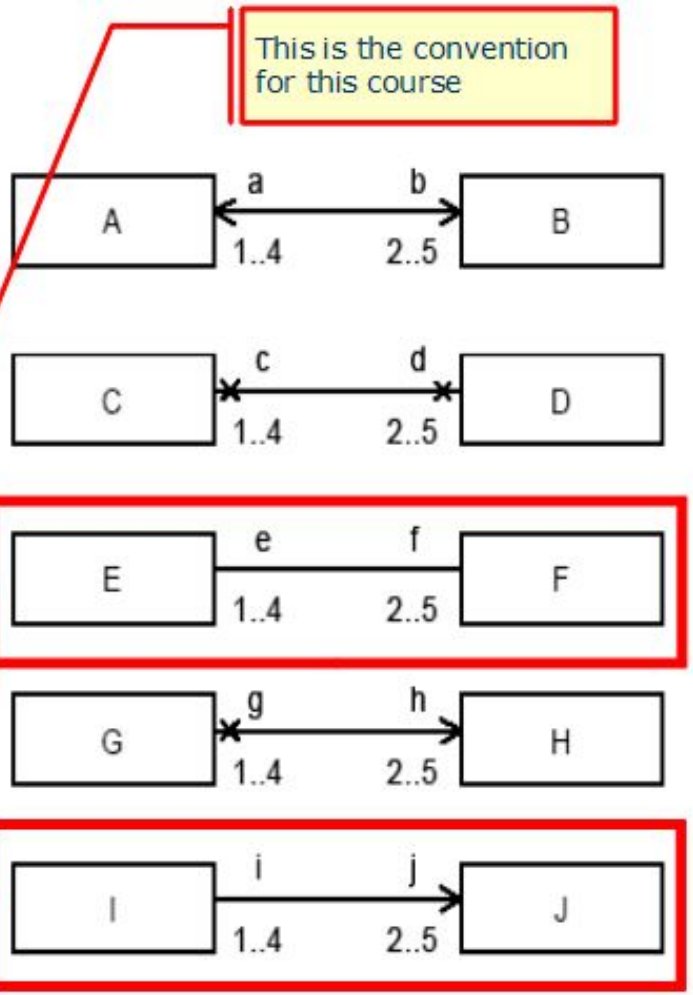
- Show all arrows and x's. Navigation and its absence are made completely explicit
- Suppress all arrows and x's. No inference can be drawn about navigation. This is similar to any situation in which information is suppressed from a view
- **Suppress arrows for associations with navigability in both directions, and show arrows only for associations with one-way navigability. In this case, the two-way navigability cannot be distinguished from situations where there is no navigation at all; however, the latter case occurs rarely in practice**



Association Navigability (III)

Various options may be chosen for showing navigation arrows on a diagram. In practice, it is often convenient to suppress some of the arrows and crosses and just show exceptional situations:

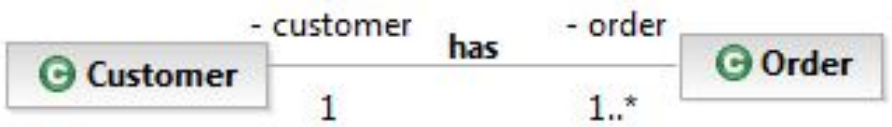
- Show all arrows and x's. Navigation and its absence are made completely explicit
- Suppress all arrows and x's. No inference can be drawn about navigation. This is similar to any situation in which information is suppressed from a view
- **Suppress arrows for associations with navigability in both directions, and show arrows only for associations with one-way navigability. In this case, the two-way navigability cannot be distinguished from situations where there is no navigation at all; however, the latter case occurs rarely in practice**

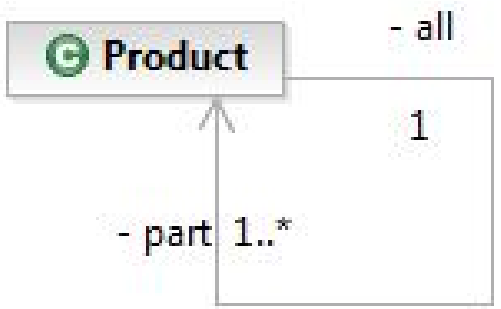


Role

What does it represent?

The function that is performed by objects in the association





Main Types of Associations

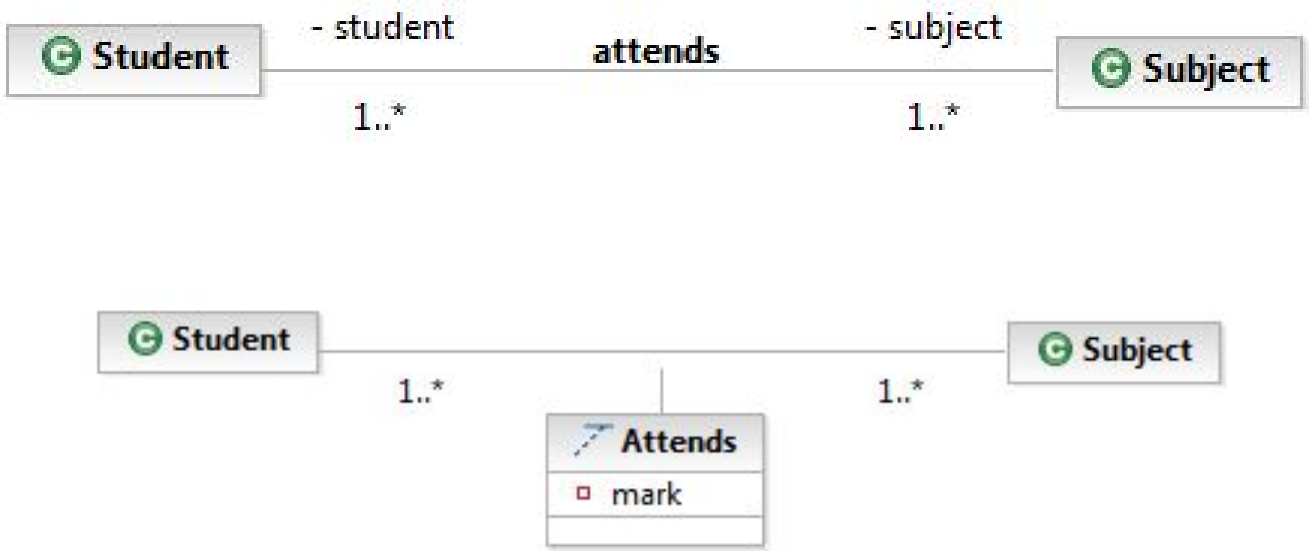
It is possible always to substitute them for an **association**

- Association Class
- Aggregation/Composition
- Generalisation

Association Class

What does it represent?

A model element that has both association and class properties, it can be seen as an association that also has class properties and vice versa



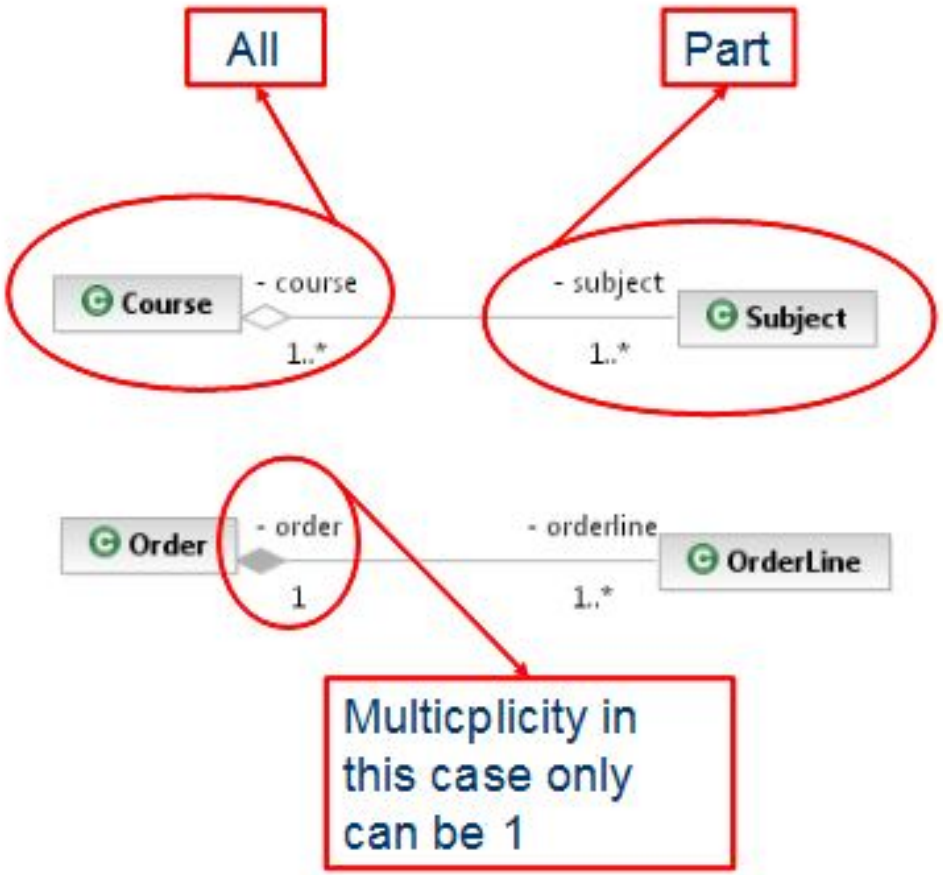
Aggregation/Composition

What does it represent?

A relationship 'all-part', in which one object of a class is a part of an object of another class

Types

- **Aggregation**: one object can be part, at once, of several objects
- **Composition**: the parts do not exist without the all, stronger relationship



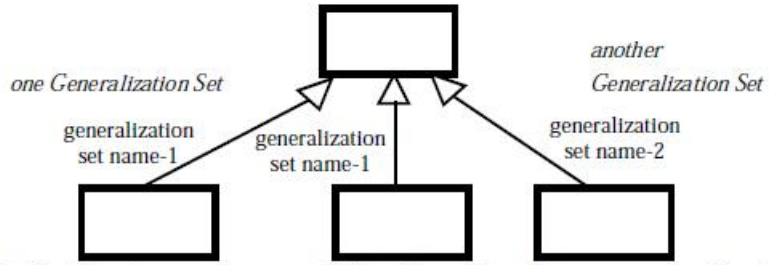
Generalisation

What does it represent?

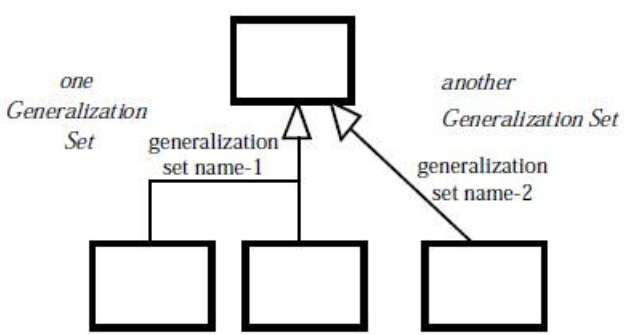
A relationship 'is a kind of', in which is defined on the top the **superclass** or father, and on the bottom the **subclasses** or children

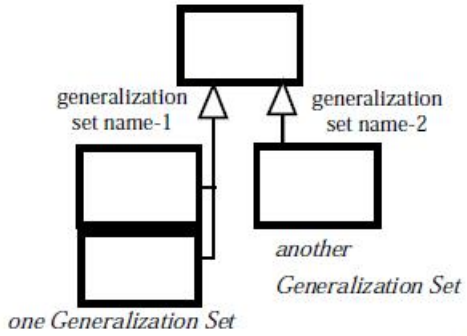
- **Subclasses**
  - Inherit the structure and behaviour of the superclass
  - Can be additional attributes and operations
  - Can modify the behaviour of the superclass
  - Cannot eliminate any attribute nor operation of the superclass
  - Its instances can be used there it can be used father's instances
- **Hierarchy**
  - It is an implementation relationship
  - Possible problem of couping
  - It should be used only if it exist a conceptual generalisation

Generalisation Set Notation

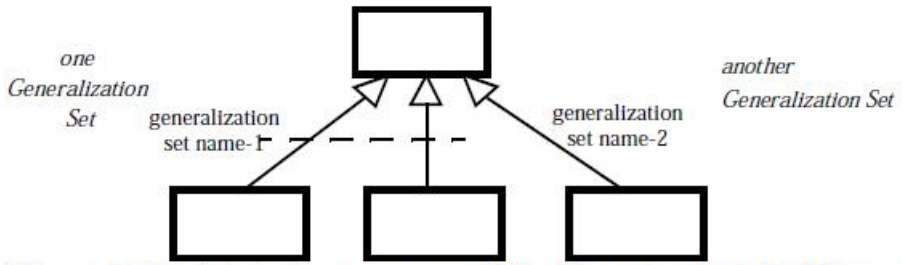


a) GeneralizationSet sharing same general Classifier using the same generalization relationship names.



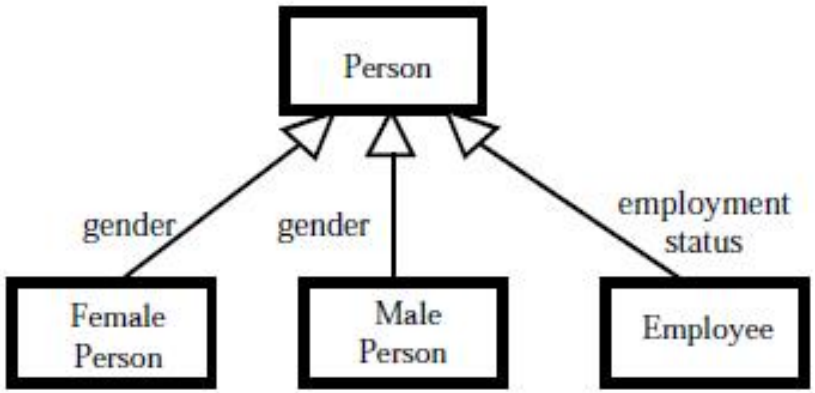


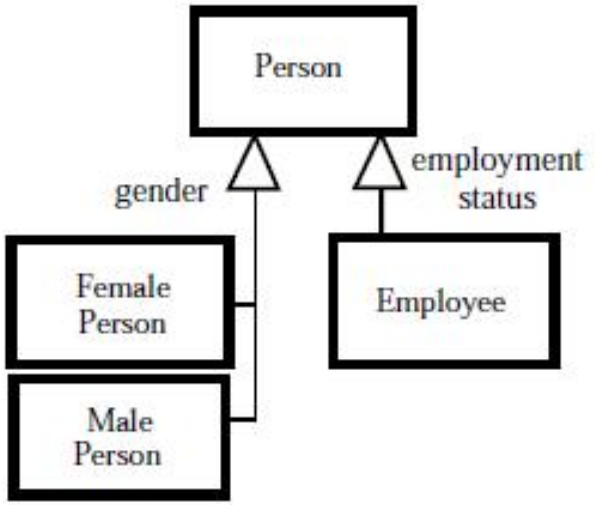
b) GeneralizationSet designation by subtypes sharing a common generalization arrowhead.

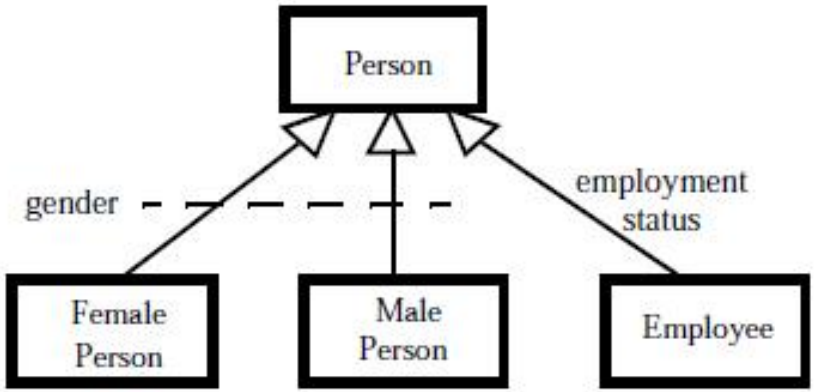


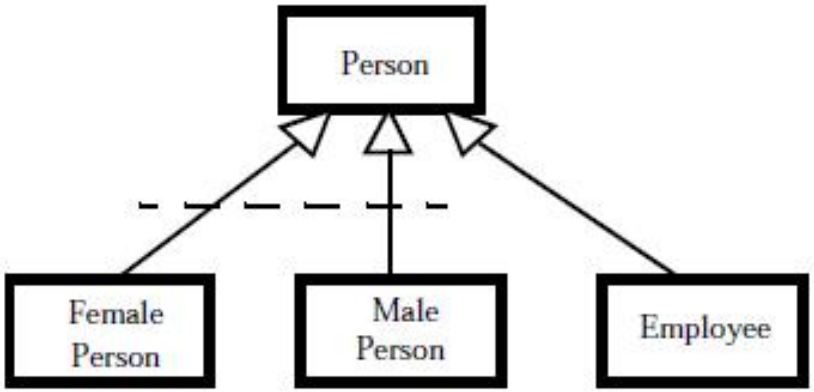
c) GeneralizationSet sharing same general Classifier using the dashed-line notation.

Example: Generalisation Set Notation

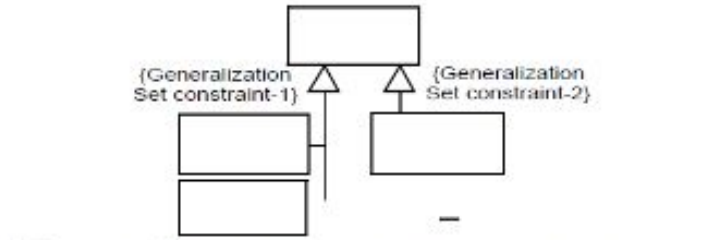




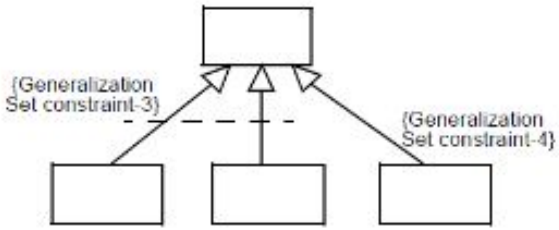




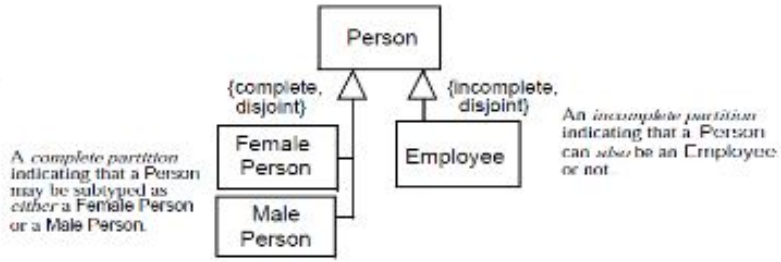
Generalisation Constraints



(a) GeneralizationSet constraint when sharing common generalization arrowhead.



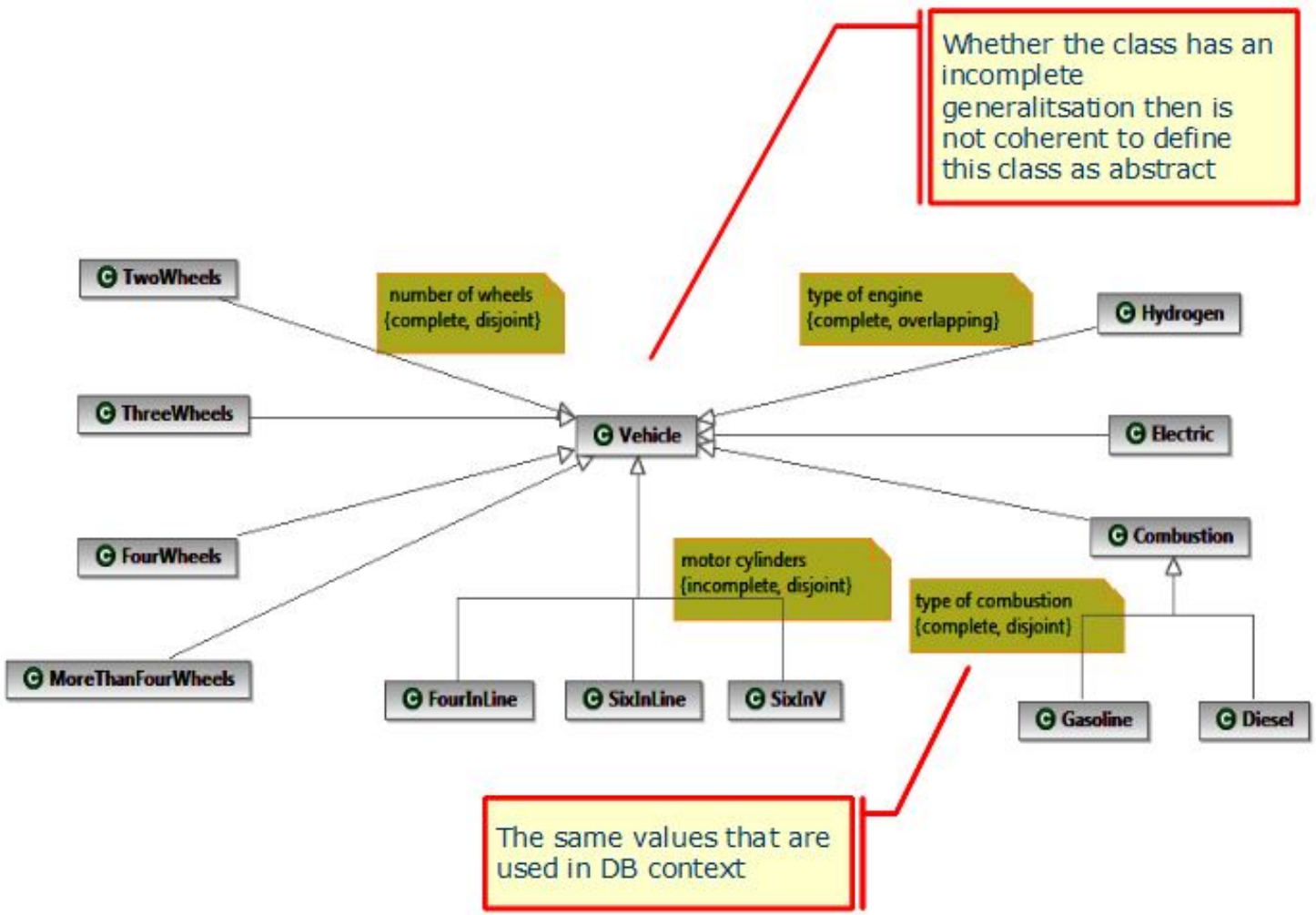
(b) GeneralizationSet constraint using dashed line notation.



{complete, disjoint} -	Indicates the generalization set is covering and its specific Classifiers have no common instances.
{incomplete, disjoint} -	Indicates the generalization set is not covering and its specific Classifiers have no common instances*.
{complete, overlapping} -	Indicates the generalization set is covering and its specific Classifiers do share common instances.
{incomplete, overlapping} -	Indicates the generalization set is not covering and its specific Classifiers do share common instances.

* default is {incomplete, disjoint}

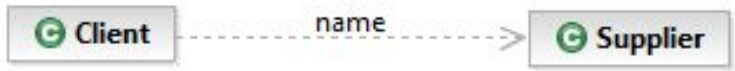
Example: Generalisation



Dependency

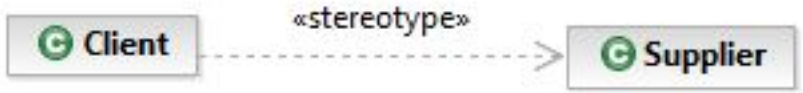
What does it represent?

A relationship 'of use', in which the modification of independent element (supplier) generates the modification of the dependent element (client)



Notation

- It is represented by a dashed arrow from client to supplier
- The arrow can be named or stereotyped



Overview

What does it represent?

A set of operations for specifying a service of a class or component, with the objective of doing visible some operations of a class

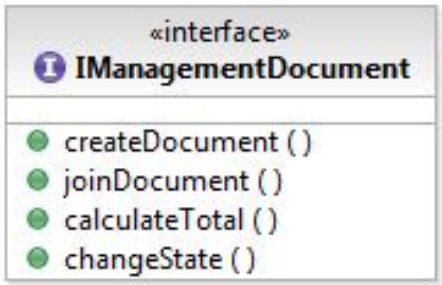
Conditions

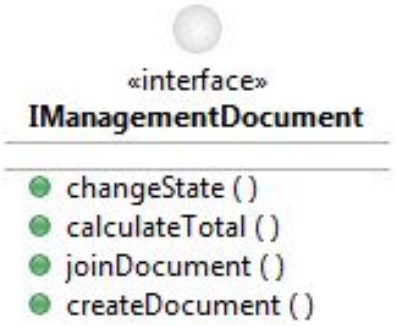
- An interface do not need to specify all the operations of one class
- One class can have more than one interface
- Classes that '**realise**' one interface need include all the operations of the interface

Representation of Interfaces

Notation

- It can be extended, icon + signature, or icon
- Its name starts with an 'I'
- It is an abstract class without attributes



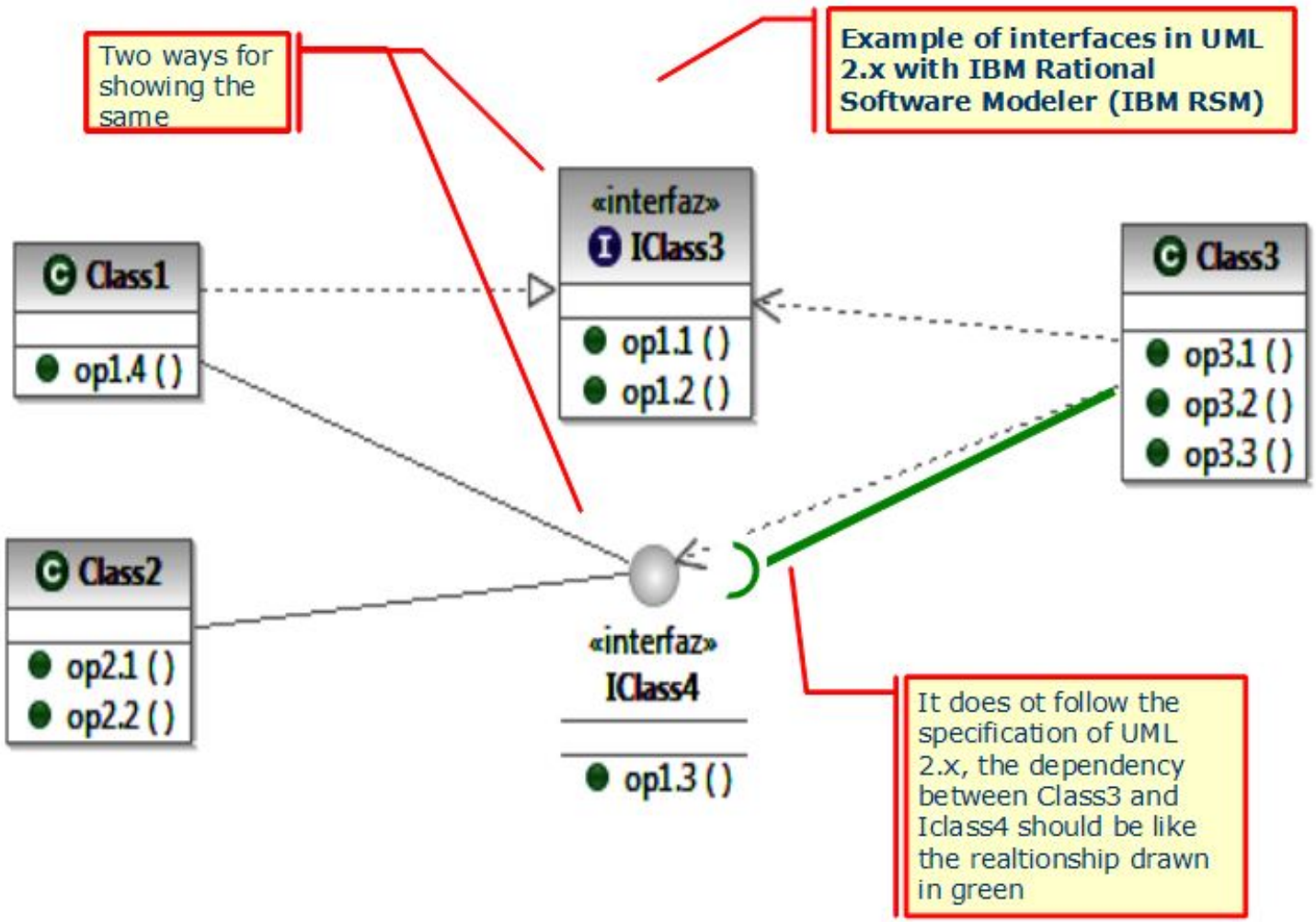


Relationships between Classes and Interfaces

Types of relationships

- **Realisation**: a contract between an interface and a class which realises it, the class is forced to provide a set of methods that implement the operations specified by the interface (**export-supplier**)
- **Dependency**: a use relationship which is used by a class and depends on the operations specified by the interface (**import-client**)

Representation of Relationships



Example: Interface