UML - a tool to design applications

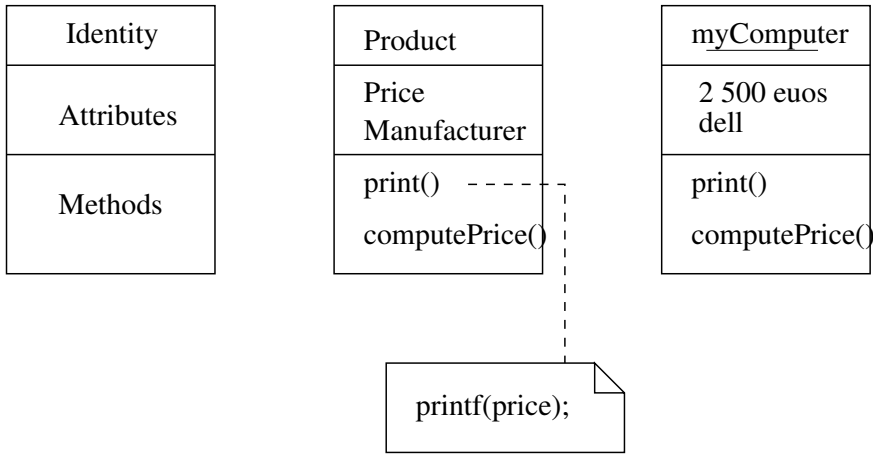
- Just a graphical language, not a method.
- 1990 - Object Management Group : standardisation.
- Unification of the methods OMT (Booch) OOSE (Jacobson) et Rumbaugh : Unified Modeling Language (version 1.0 1997, version actuelle 1.3).

Objects and classes

- Reminders :

  Object = State + Behavior + Identity

- UML graphical conventions



- Abstract classes are represented with the stereotype `<abstract>`

Classes Diagram

- Classes diagram is a **static** view of the model.
- It describes the internal structure of classes and their relations (inheritance, dependancy, composition . . .).
- The terms : *static structural diagram* and *class diagram* are equivalent in UML terminology.
- It is a collection of declarative elements of the model. These elements are classified by typing mechanism.
- **Note**: it is possible to build a diagram which contains only interfaces and abstract classes. In this case, it is called a **Meta-Model**

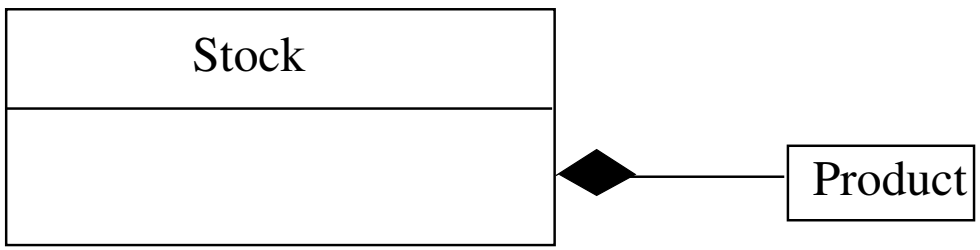
Objects Diagram

- It is the instance graph for the all object class.
- It is itself an instance of the class diagram.
- It is only used to illustrate examples.

Object composition

Object attributes can be themselves objects.

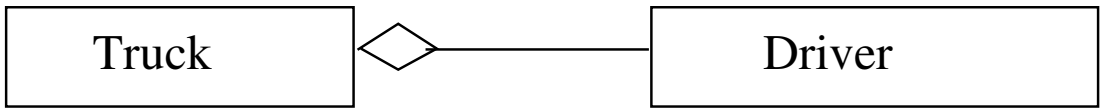
- **Composition by value** : responsability to create and to delete the object.



- Creating an object implies to create its attributes by value too.

Object composition

- **Composition by reference** : it is a reference link to an object, this object can be shared by several another object,



- Creating the container does not imply to create the referenced object.
- **NB** : the `diamond` is always at the using object side.

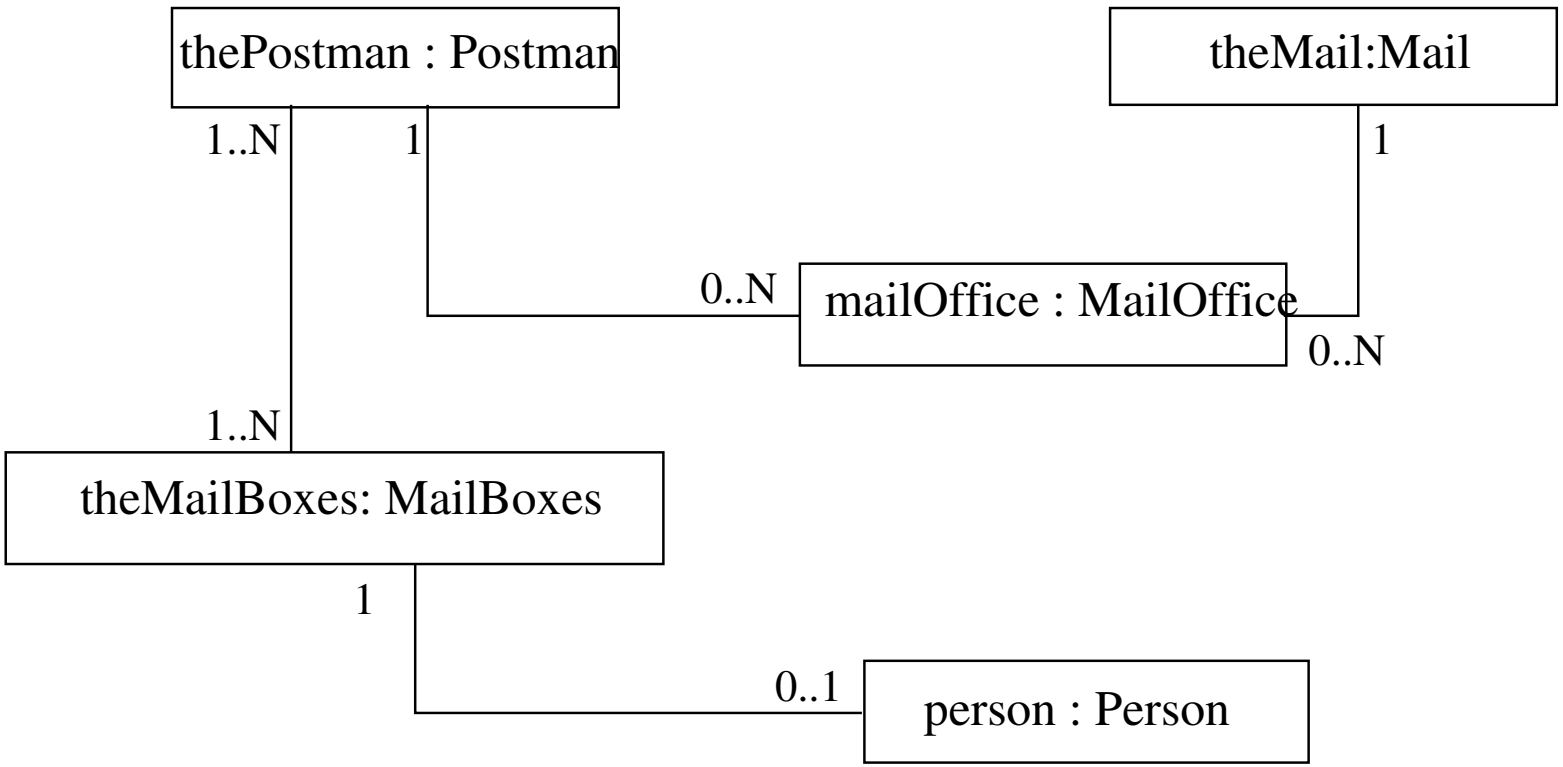
Methods and attributes visibility

- **Public**: a public attribute or method is specified with the + sign.
- **Private**: a private attribute or method is specified with the − sign.
- **Protected**: a protected attribute or method is specified with the # sign.

Association/Composition Arity

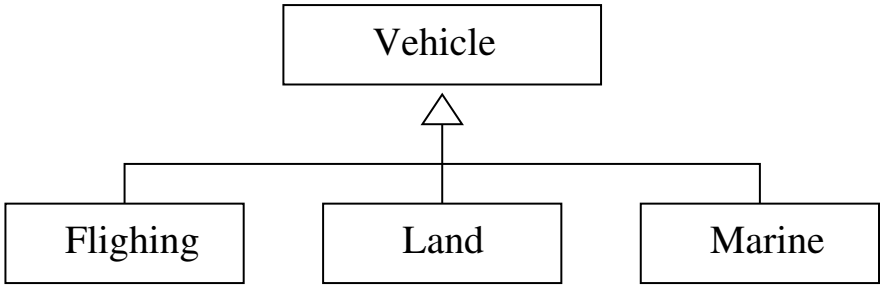
1	1 only one
0..1	0 or 1 association
M..N	from M to N (M and N integer)
$\star$	from 0 to several
0..$\star$	from 0 to several
1..$\star$	from 1 to several

Example of arities placed in diagram



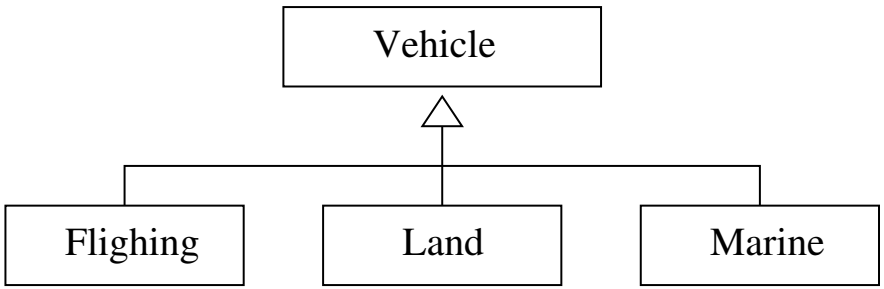
Inheritance

- **Simple inheritance**

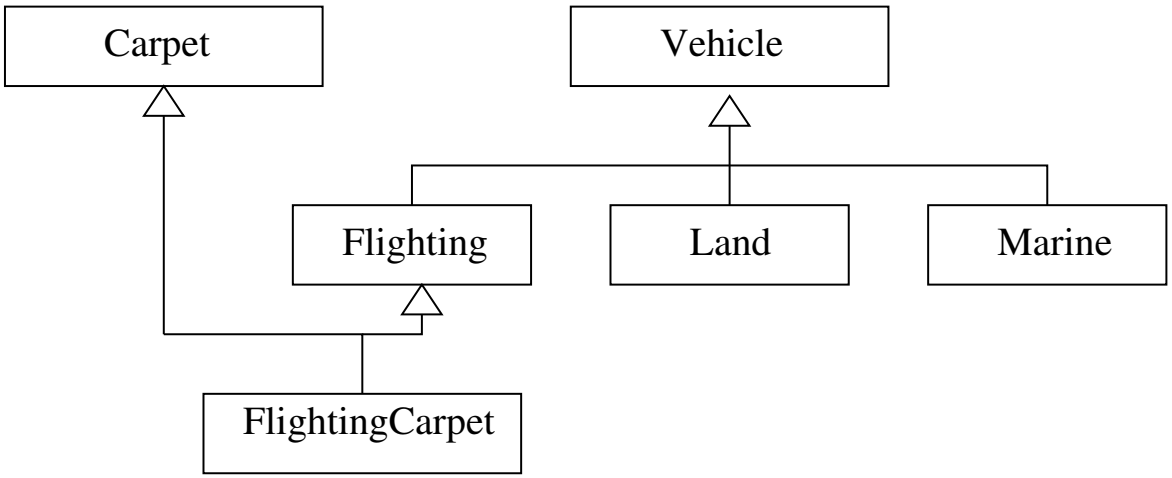


Inheritance

- **Simple inheritance**



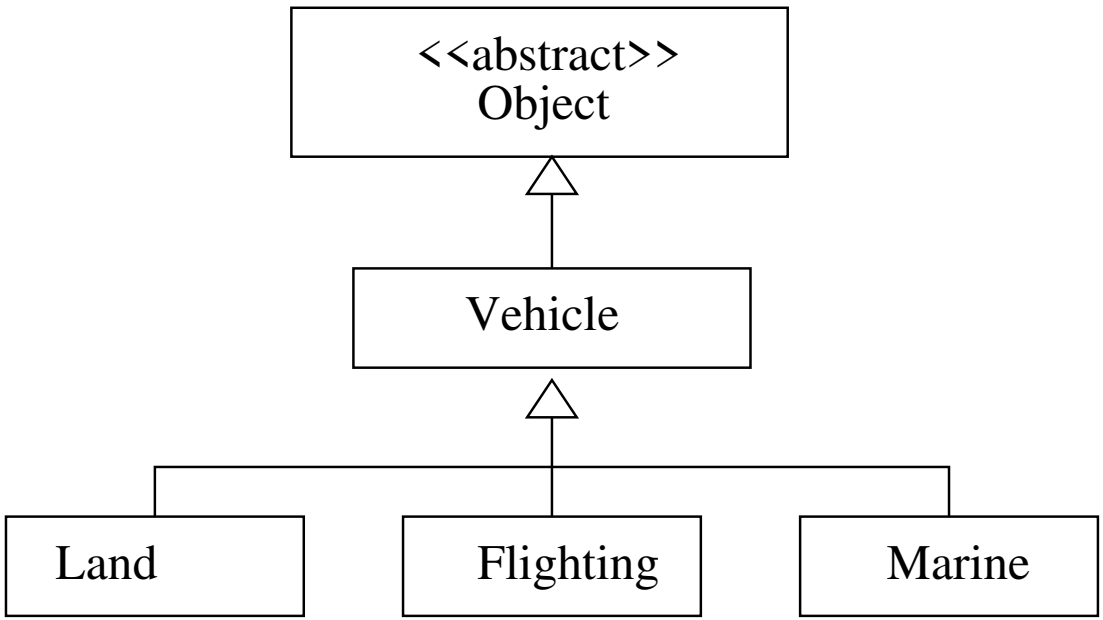
- **Multiple inheritance**



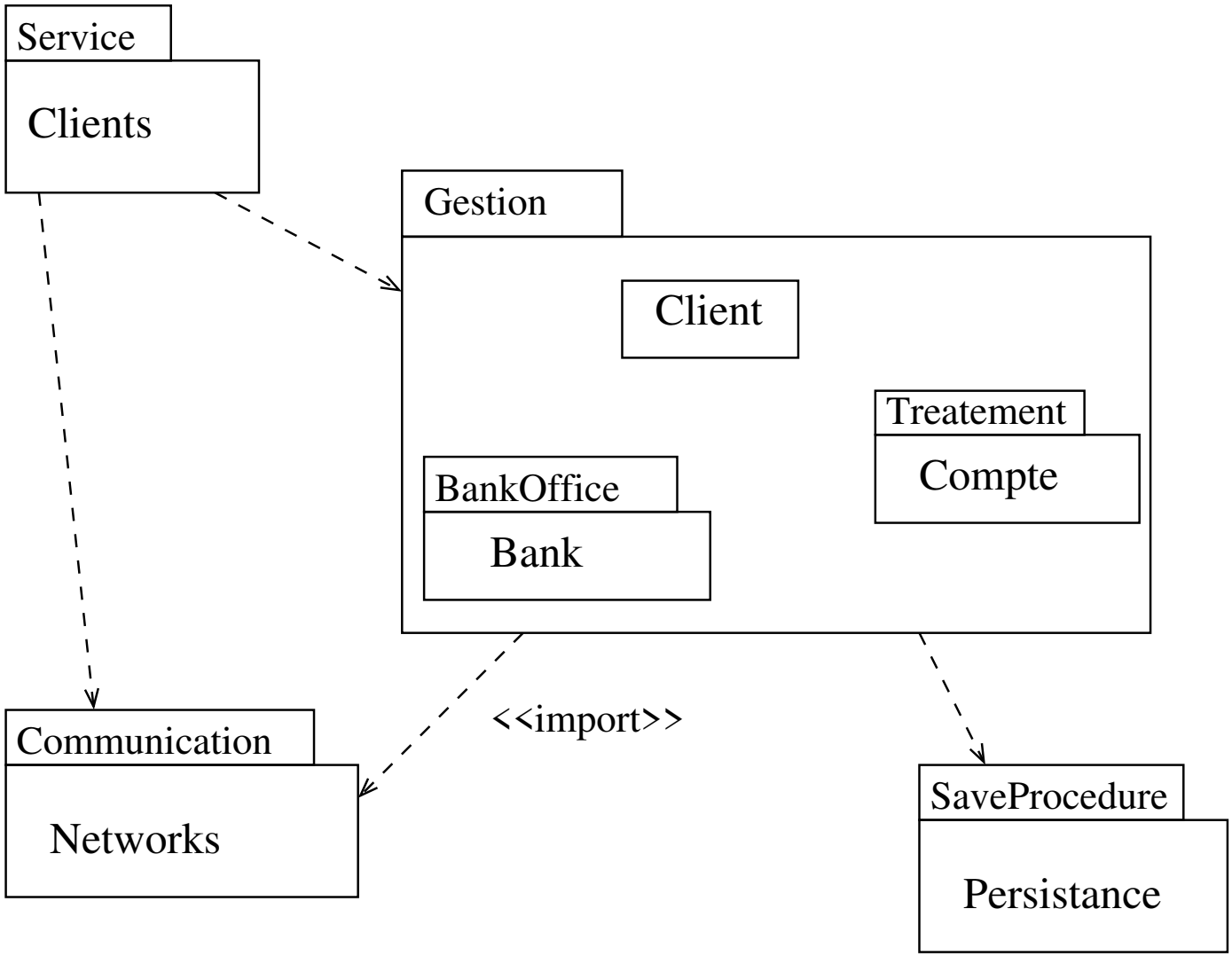
Meta-Classes

- Definition :
  - A meta-classe defines class characteristics,it is a generic model of classes (attribute or behavior).
- Example :
  - abstract class, interface, by extension each class of class.
- We note that a meta-class is also an object of which the class is the reference base class from which all objects are built (`Object` class in Java).
- At the analysis and conception step, it does not exist difference between a class and a meta-class.

Meta-Class

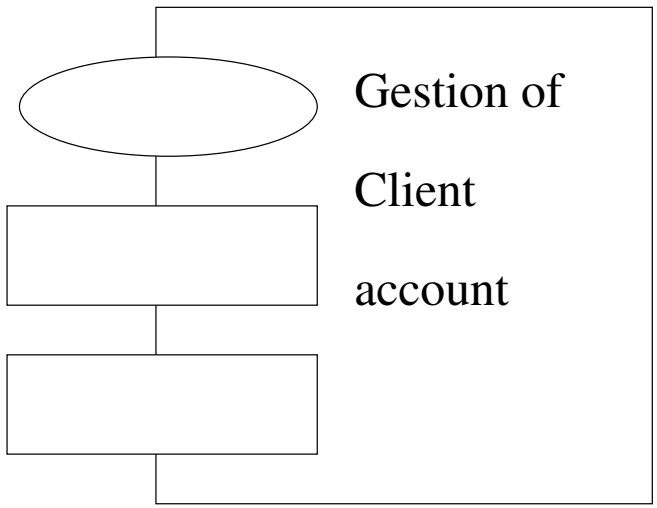


Package representation



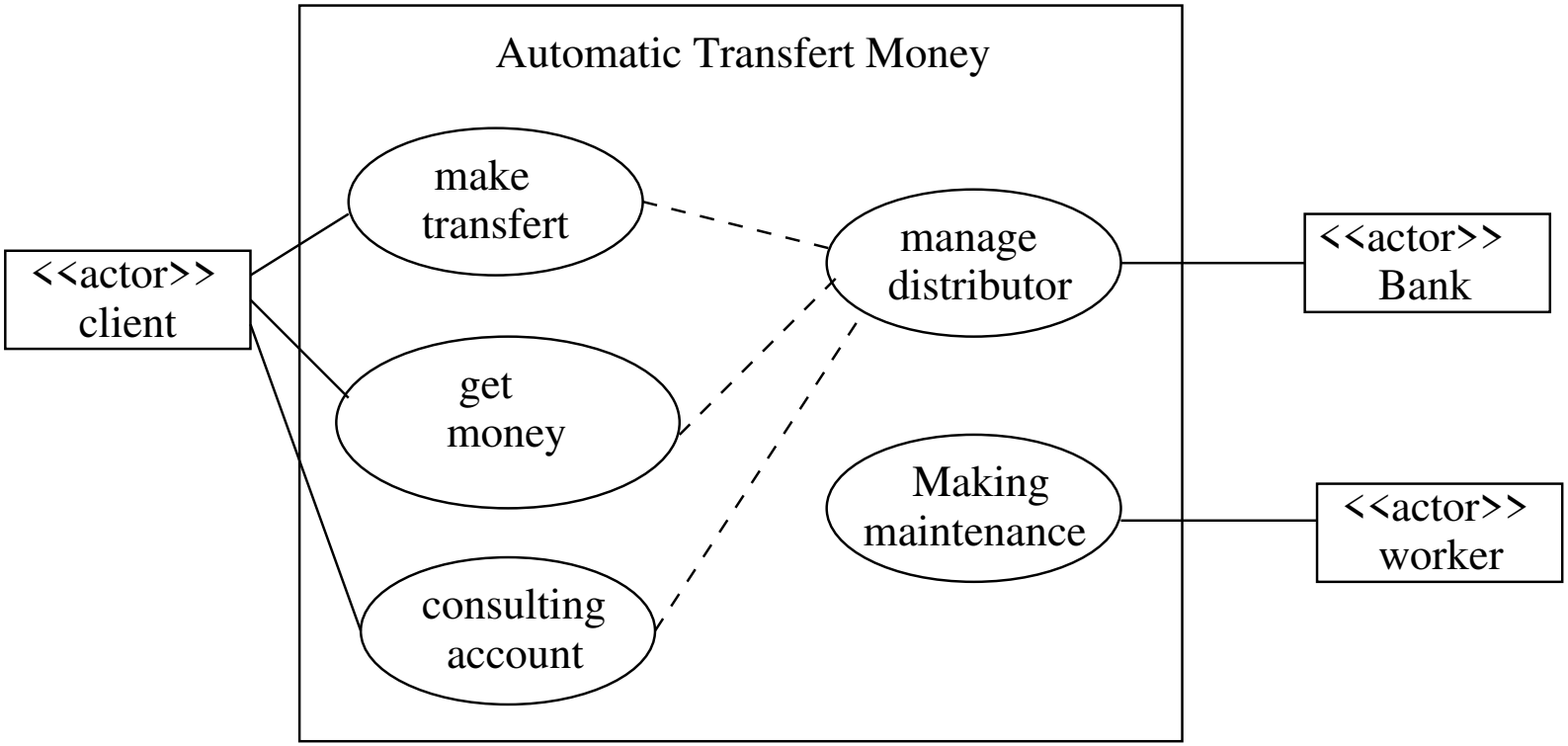
Module graph

Modules are compiling sub-units. Some programming languages do not be able to implement this concept.



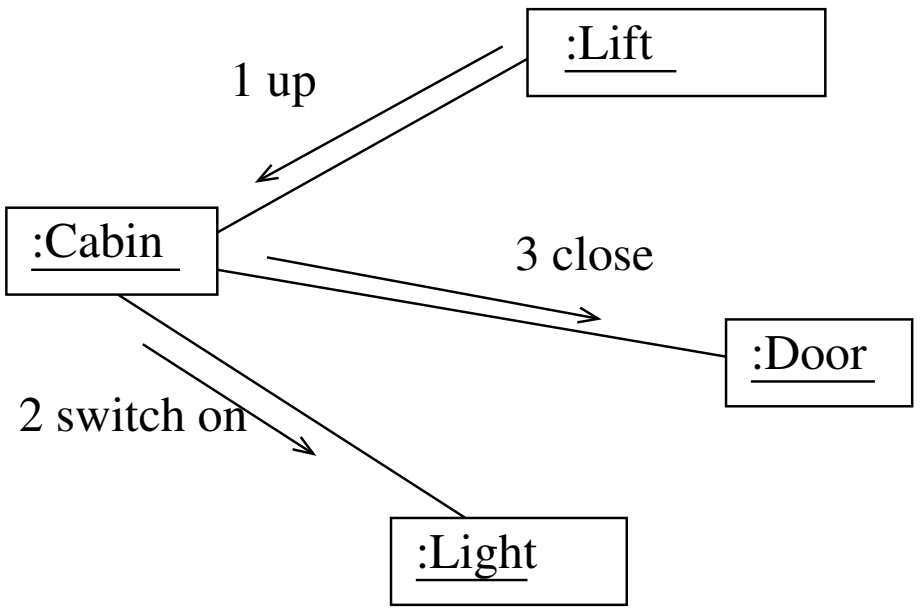
Use cases diagram

- Communication, event or data flux diagram between external entities and the system.
- Give the external interface of the system.
- Only two kinds of represented objects: external actors and composants which interact directly with the actors.



Collaboration diagram

- This diagram shows the interactions between objects and the structural relations which allows these interactions.
- The numerotation gives the order of messages.
- Time is not represented.

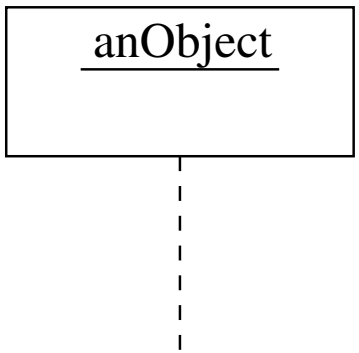


Collaboration diagram

- Give the context for an objects set and the interactions beween these objects (messages sendings).
- Messages are given on the links which associate objects, these links are oriented from the caller to the destination.
- Allows to represent actors or external element.
- The links in the collaboration diagram have not the same semantic than the composition links into the class diagram.

Sequence Diagram

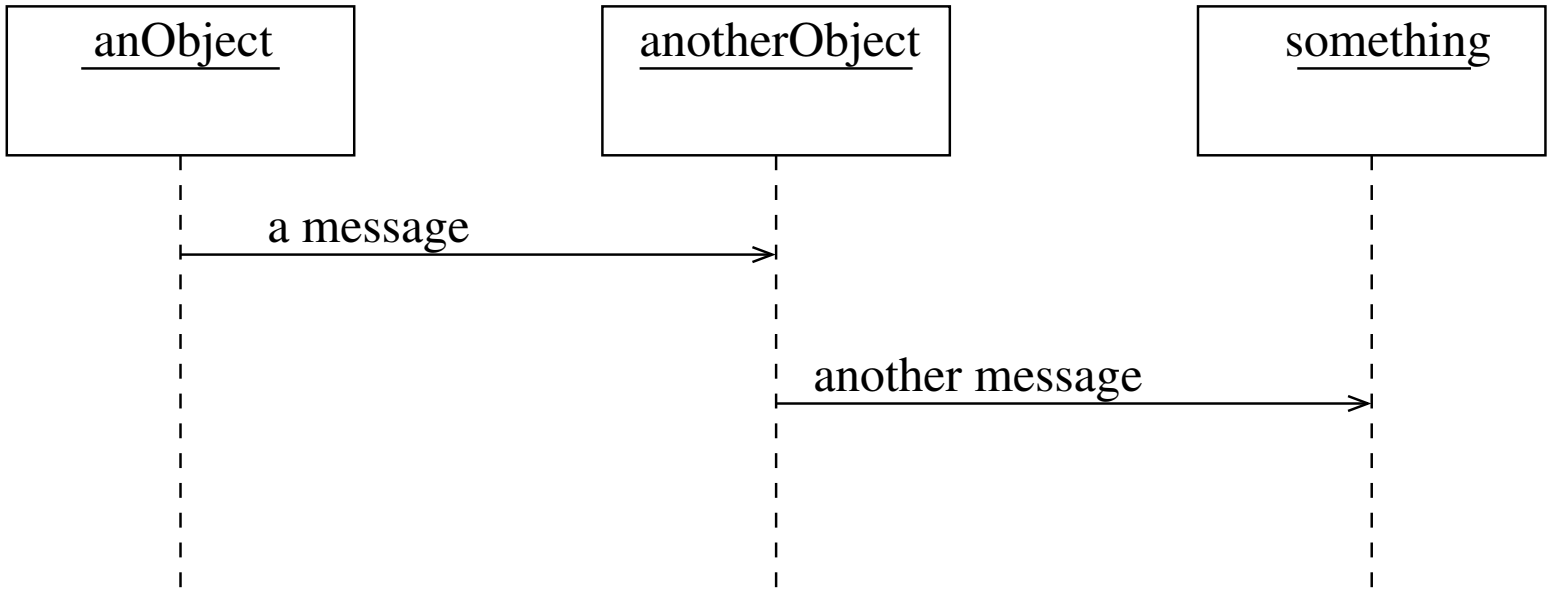
- Show interactions between object from the time point of view.
- Notation [1] :
  - An objet is materialized by a rectangle and a vertical line called *the life line*



[1] Object Message Sequence Chart, Siemens Pattern Group. Wiley 1996 *Pattern-riented Software*

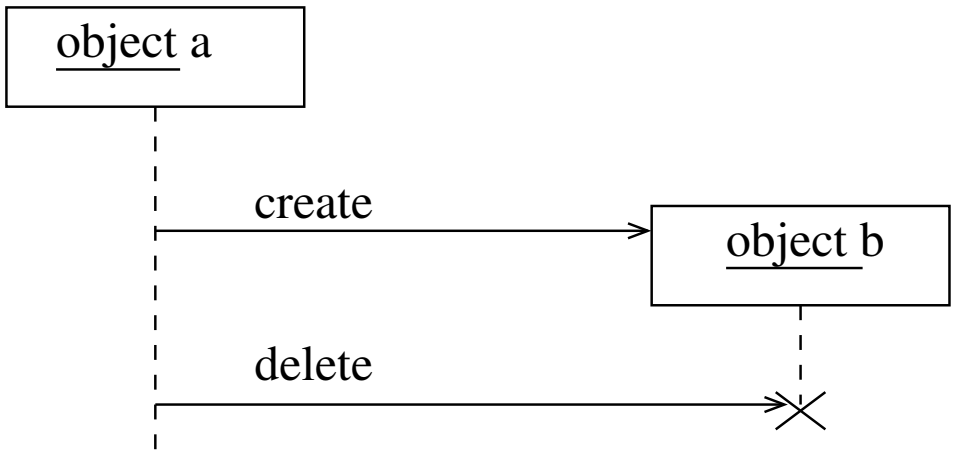
Sequence diagram

- The rank of the message sending is done by the position on the vertical axe.



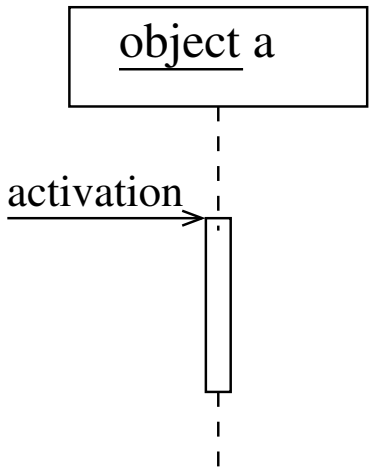
Sequence Diagram

- Creation / destruction of object



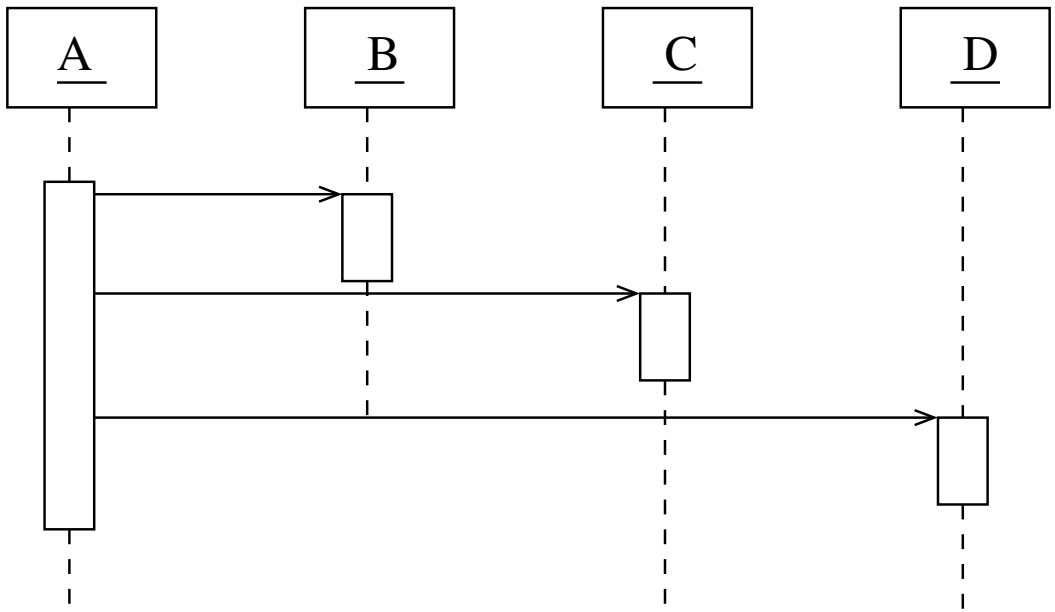
Sequence diagram

- Representation of activity periods for objects = working time for this object.
- Beginning and the end of the vertical band correspond to the beginning and the end of the activiy period.



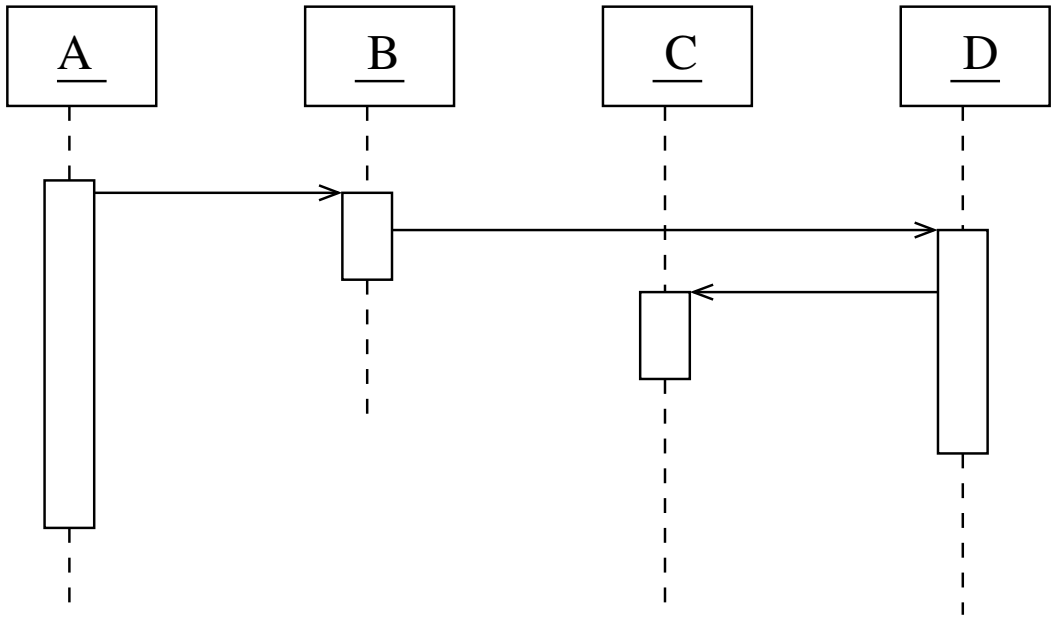
Centralized - decentralized mode

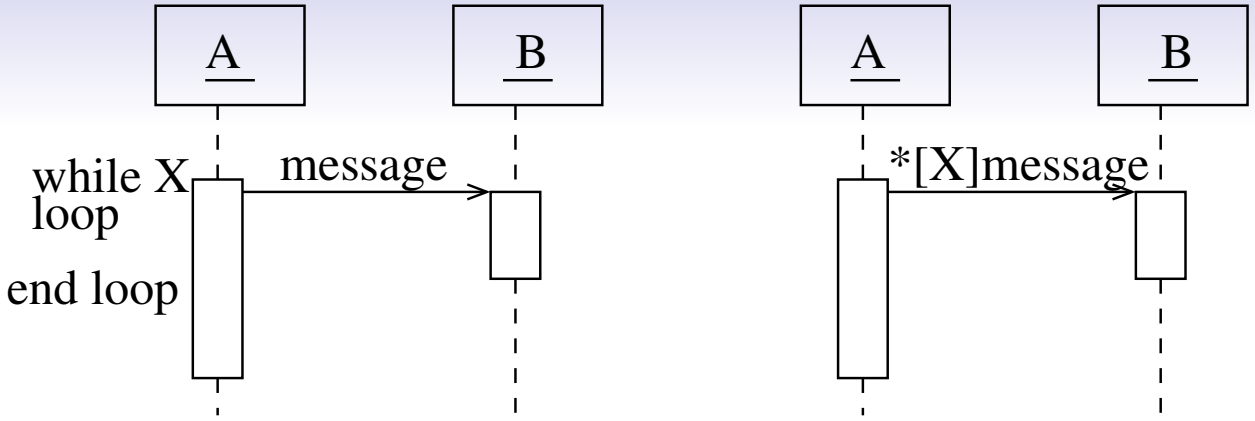
- Sequence diagrammes can show the control structure choice.



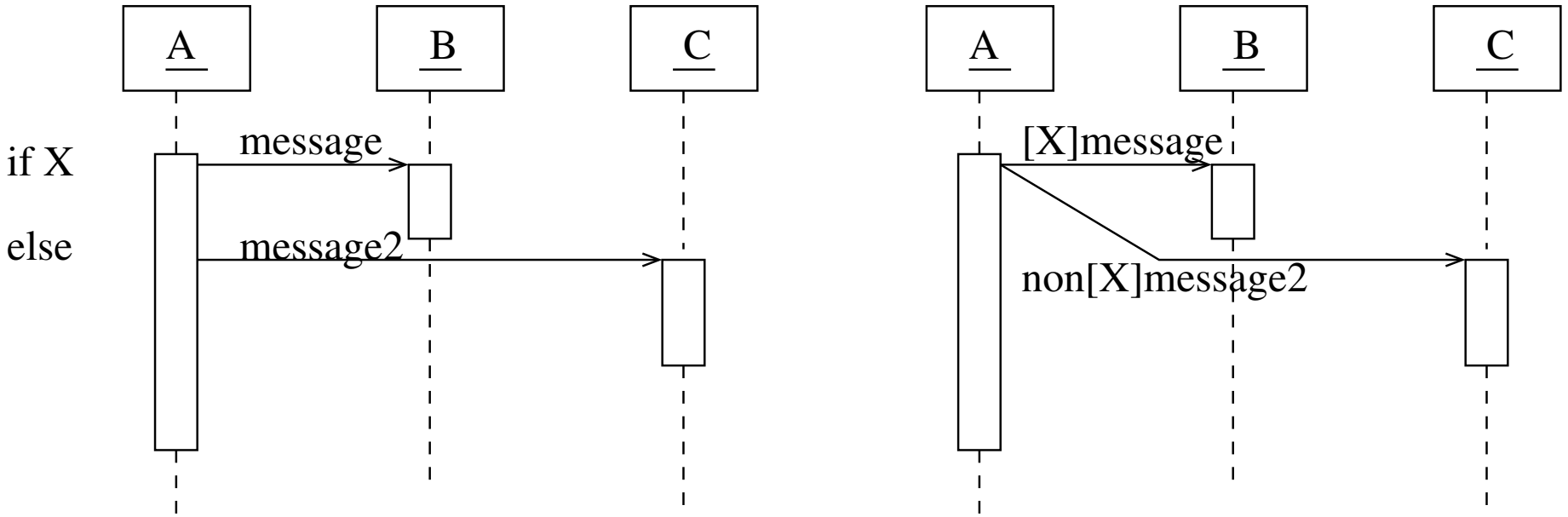
Centralised - decentralised mode

- Decentralised sending of messages.



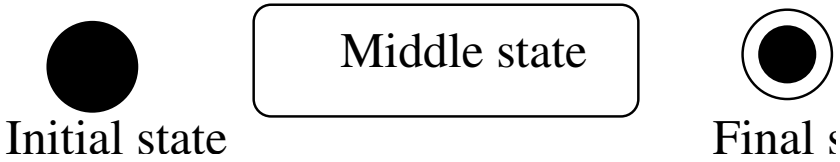

A
B
while X
loop
message
end loop
A
B
*[X]message

Conditional branching



State-transitions diagram

- An object is in a given state at each step.
- The state of an object is given by the values of its attributes



State-transitions diagram

- The object change from a state to another one by using transition.
- Fired by event, transitions allows the change immediatly.