



Class modelling (part 2)

Fabrizio Maria Maggi

Institute of Computer Science

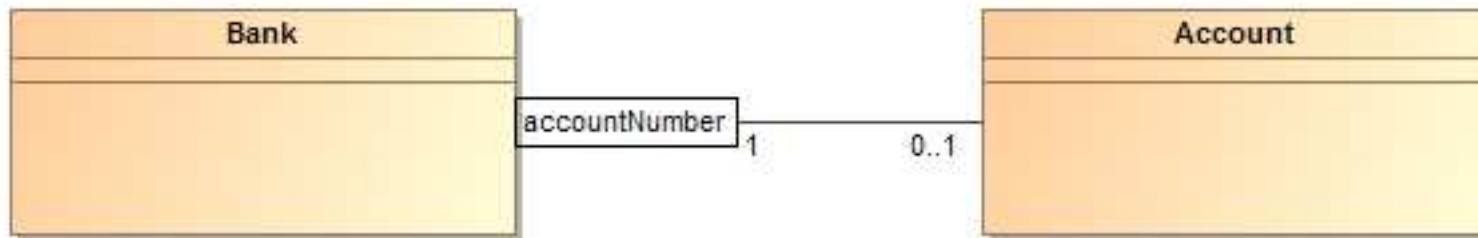
(these slides are derived from the book "Object-oriented modeling and design with UML")

Qualified Associations



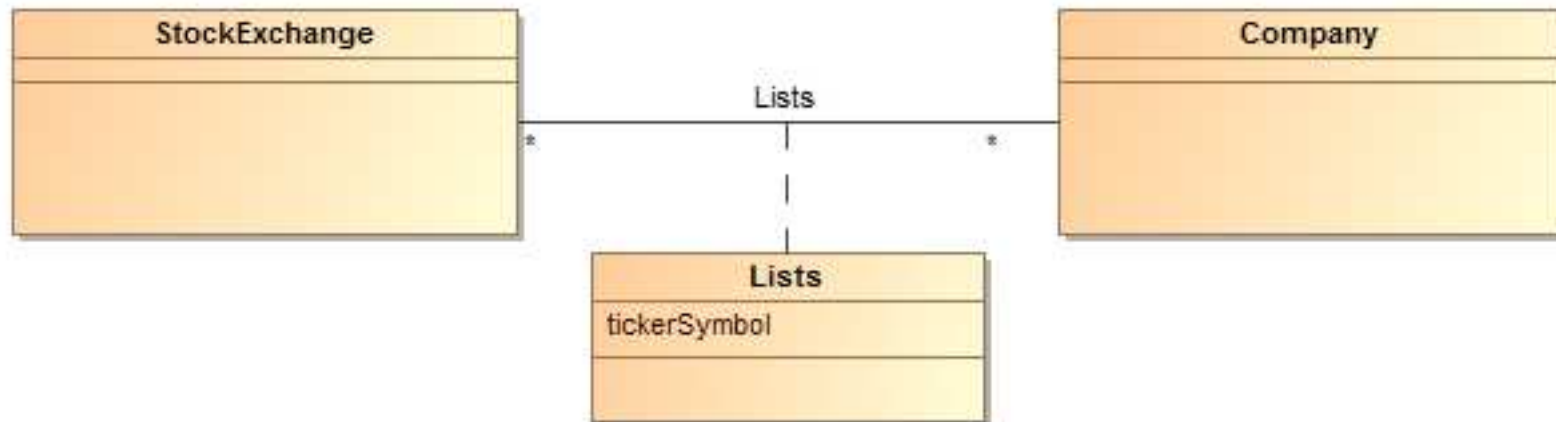
- ▶ What is the meaning of this association?
- ▶ How can we implement it?
- ▶ Is this a realistic representation?

Qualified Associations



- ▶ How can we implement a qualified association?

Qualified Associations



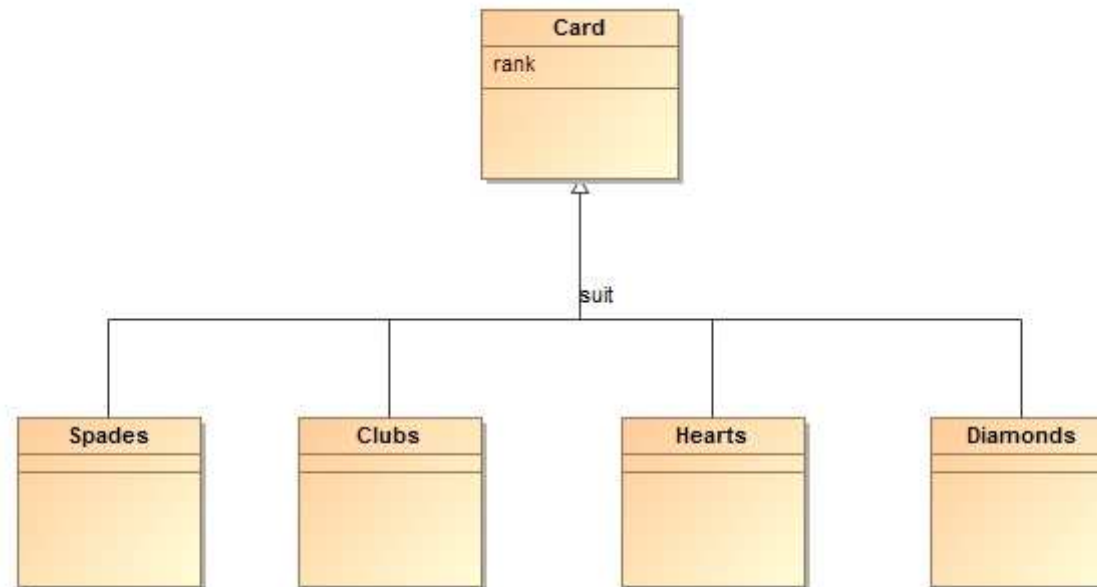
- ▶ What is the meaning of this association?
- ▶ Given a Stock Exchange, can it list different Companies with different ticker symbols?
- ▶ Given a Stock Exchange, can it list different Companies with the same ticker symbol?

Enumerations

- ▶ An enumeration is a data type that has a finite set of values.
- ▶ Enumeration is a data type: you can declare an enumeration by listing the keyword enumeration in angle quotes (<< >>) above the enumeration name in the top section of a box. The second section lists the enumeration values.
- ▶ Do not use generalization to capture the values of an enumerated attribute:
 - ▶ An enumeration is a list of values.
 - ▶ Introduce generalization only when at least one subclass has significant attributes, operations, or associations that do not apply to the superclass.

Enumerations

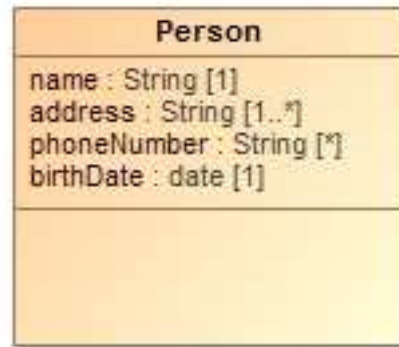
wrong



correct



Multiplicity for attributes

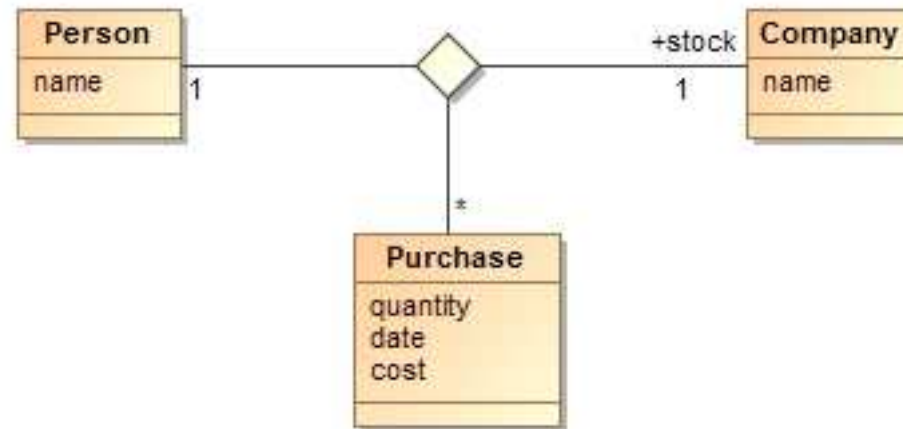


- ▶ You can specify if an attribute is single or multivalued, mandatory or optional.

N-ary associations

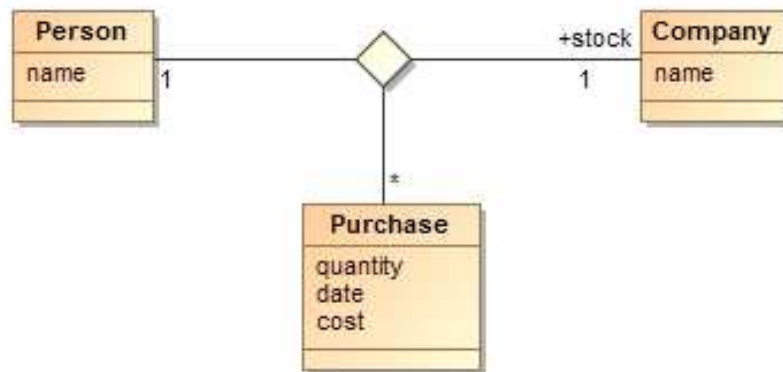
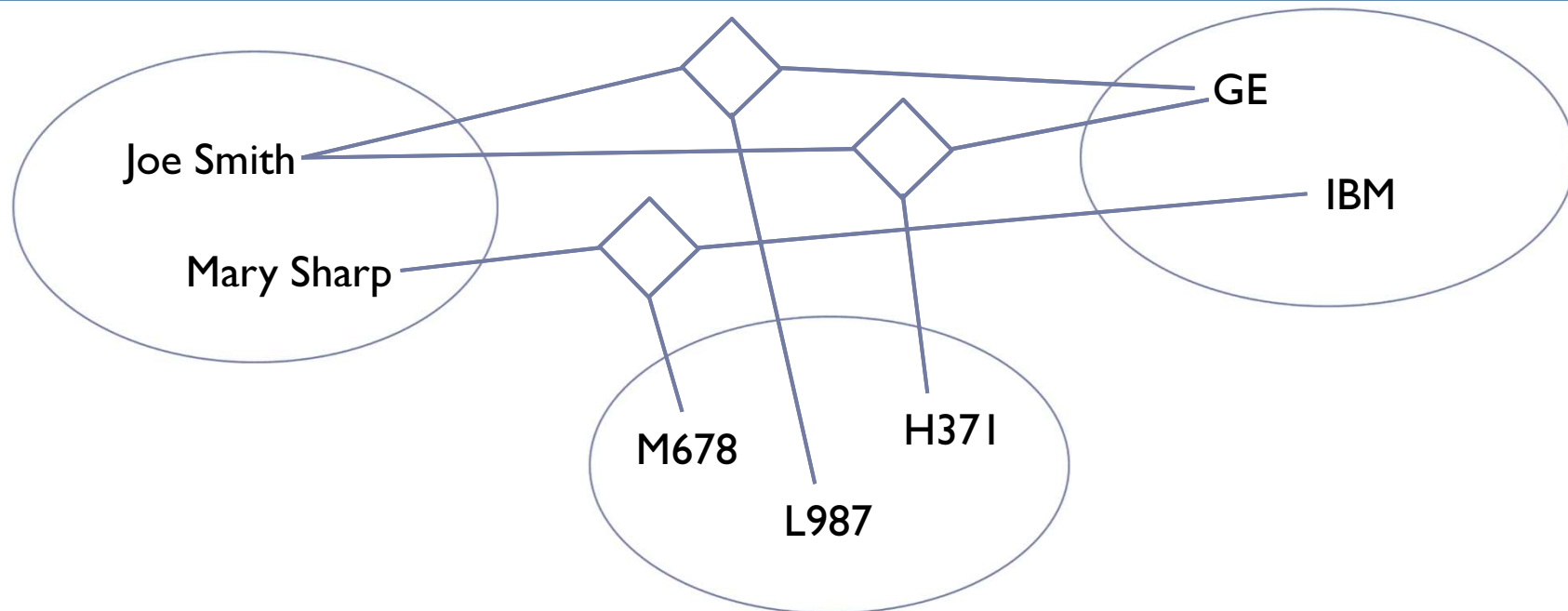
- ▶ Many relationships involve just two things and can be modeled with the simple binary association.
 - ▶ It is not however uncommon for three or more things to be involved in a relationship.
- ▶ An n-ary association can be used in these circumstances and allows any or "n" number of things to be related in a single cohesive group.
- ▶ An n-ary association is used when the three or more things are all related to each other in a structural or behavioral way.
 - ▶ It does not replace the use of two binary associations where a classifier is related to two other classifiers, but the latter two classifiers aren't related to each other.
 - ▶ Think of two people being married by a celebrant or minister; all three are involved and have an association with each other.

N-ary associations



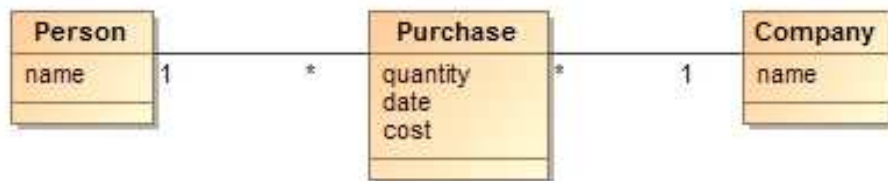
- ▶ Example: a person makes a purchase of stock in a company.
- ▶ Read the multiplicities.

N-ary associations



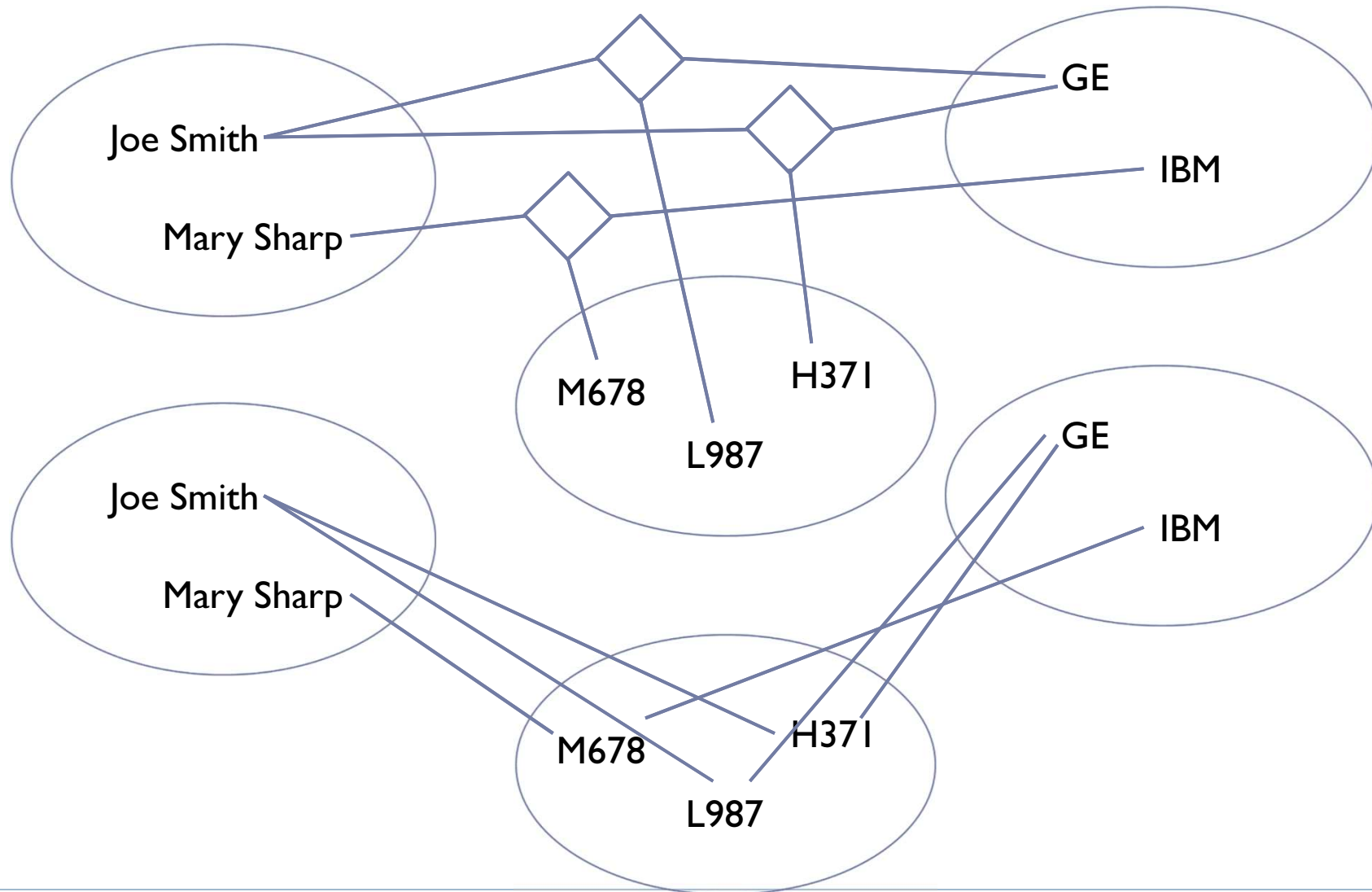
► Is this a genuine ternary association?

N-ary associations

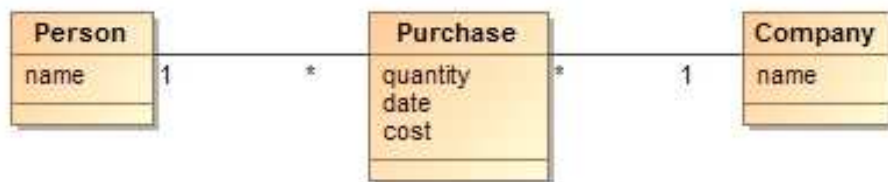


► Are we losing information?

N-ary associations

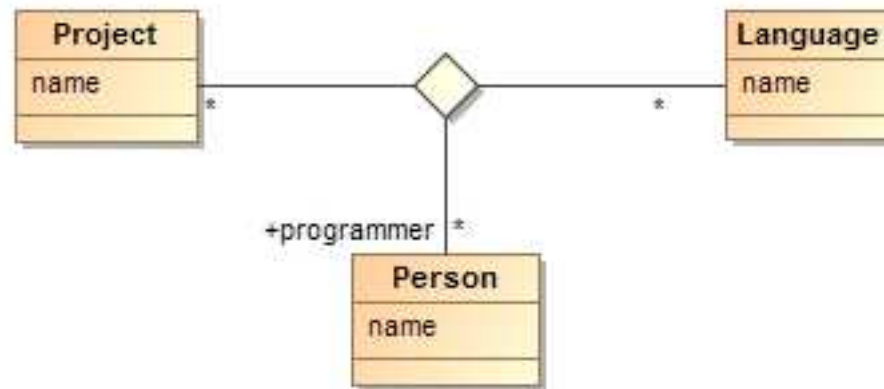


N-ary associations



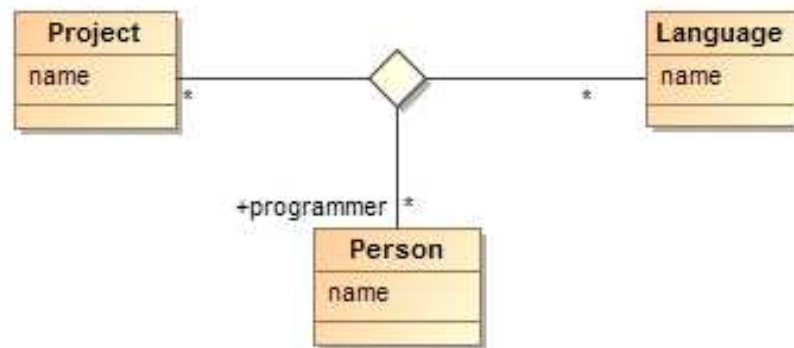
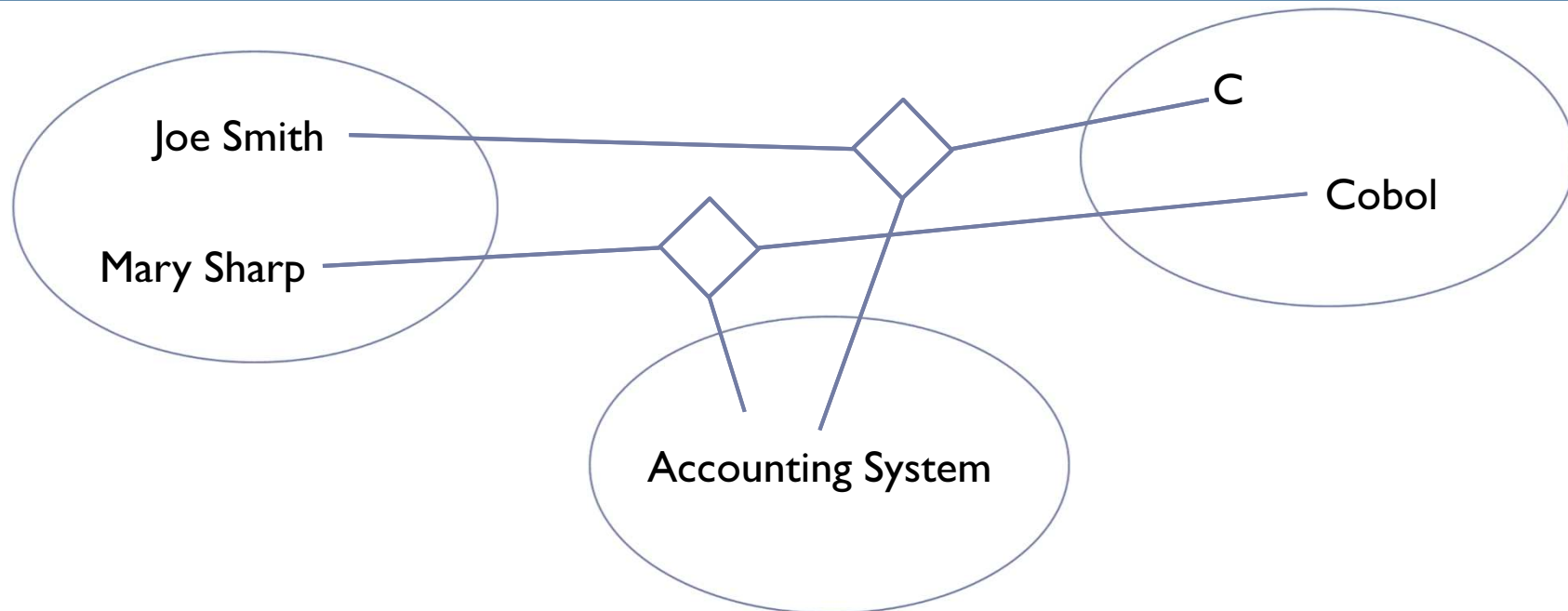
- ▶ You can decompose most n-ary associations into binary associations.

N-ary associations



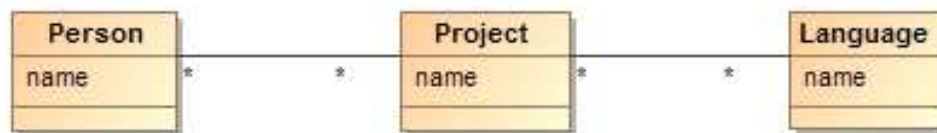
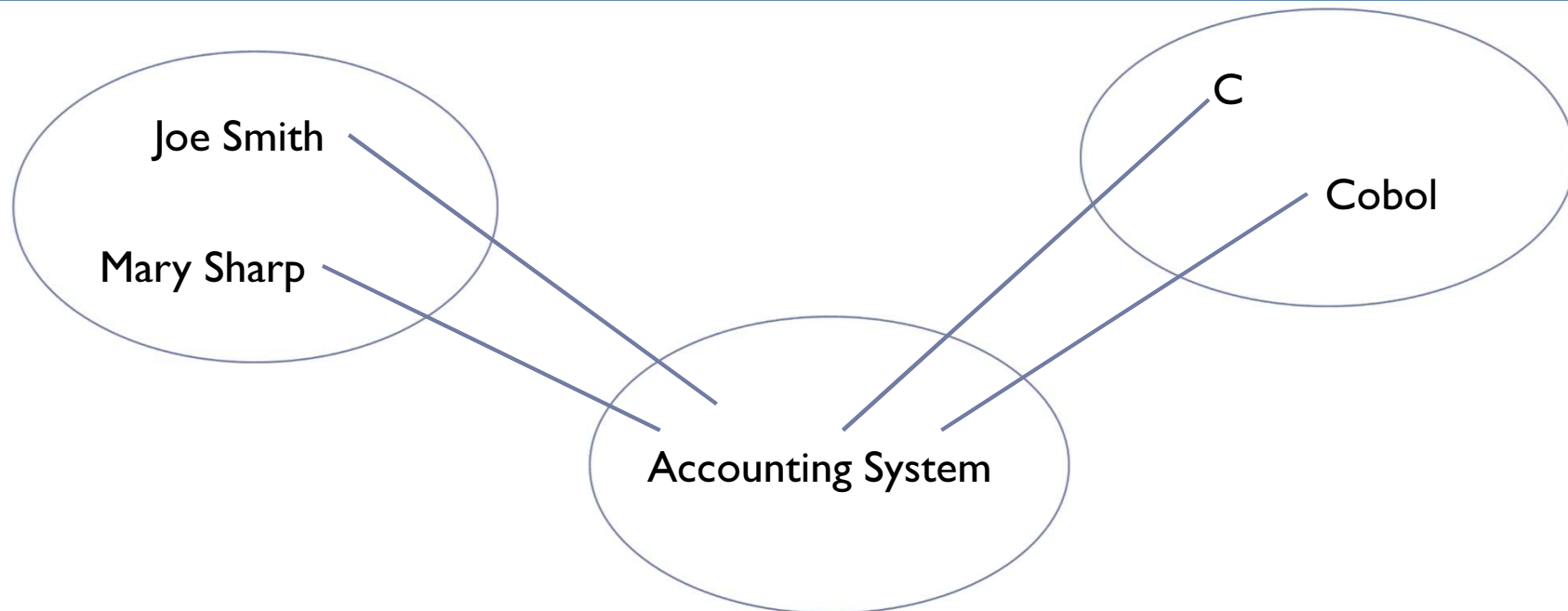
- ▶ Example: programmers use computer languages on projects.

N-ary associations



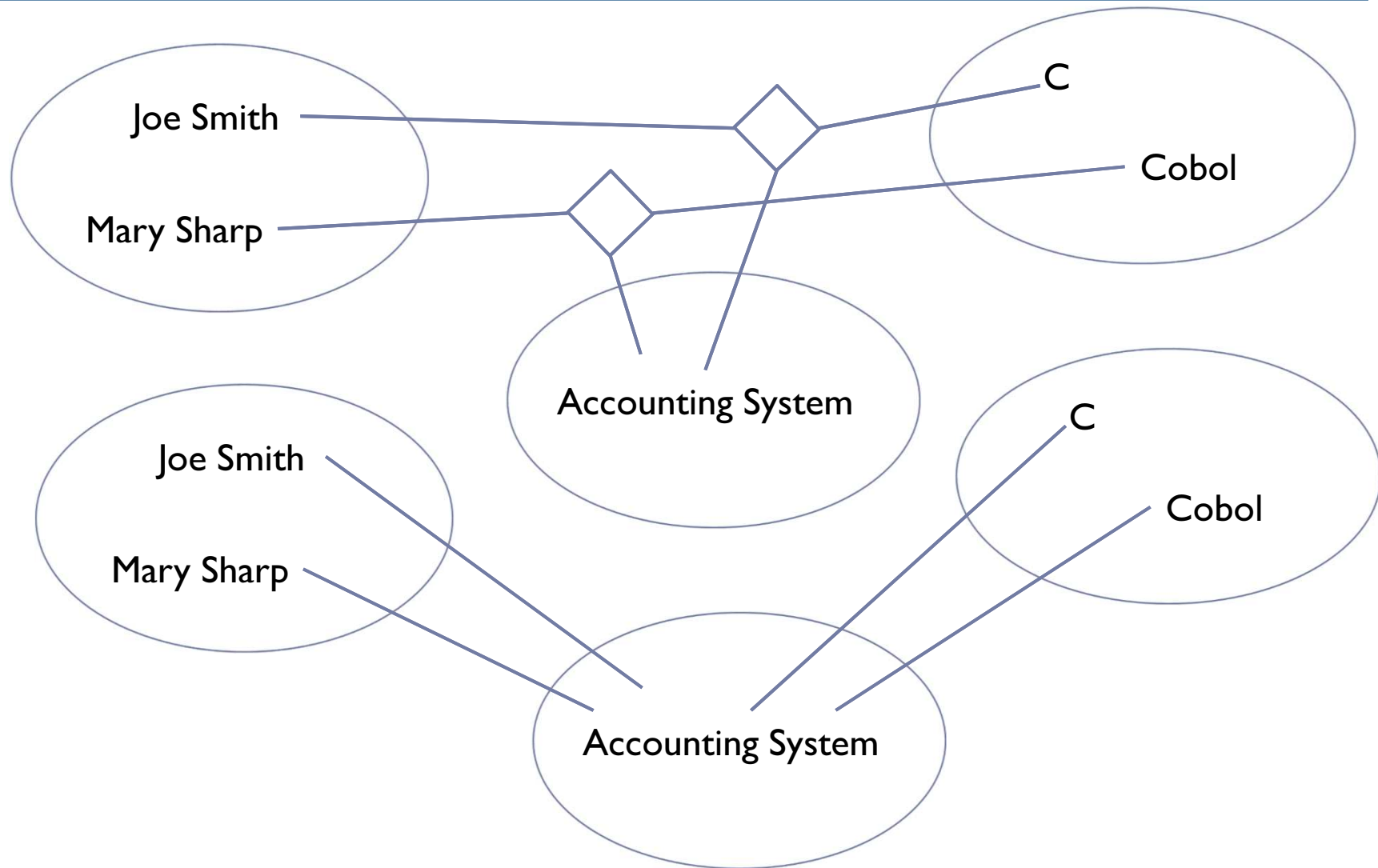
► Is this a genuine ternary association?

N-ary associations

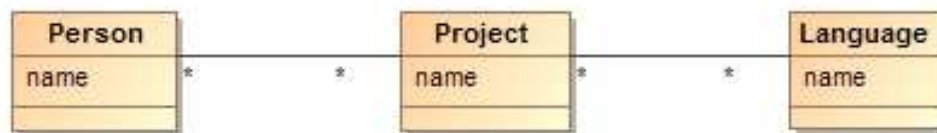
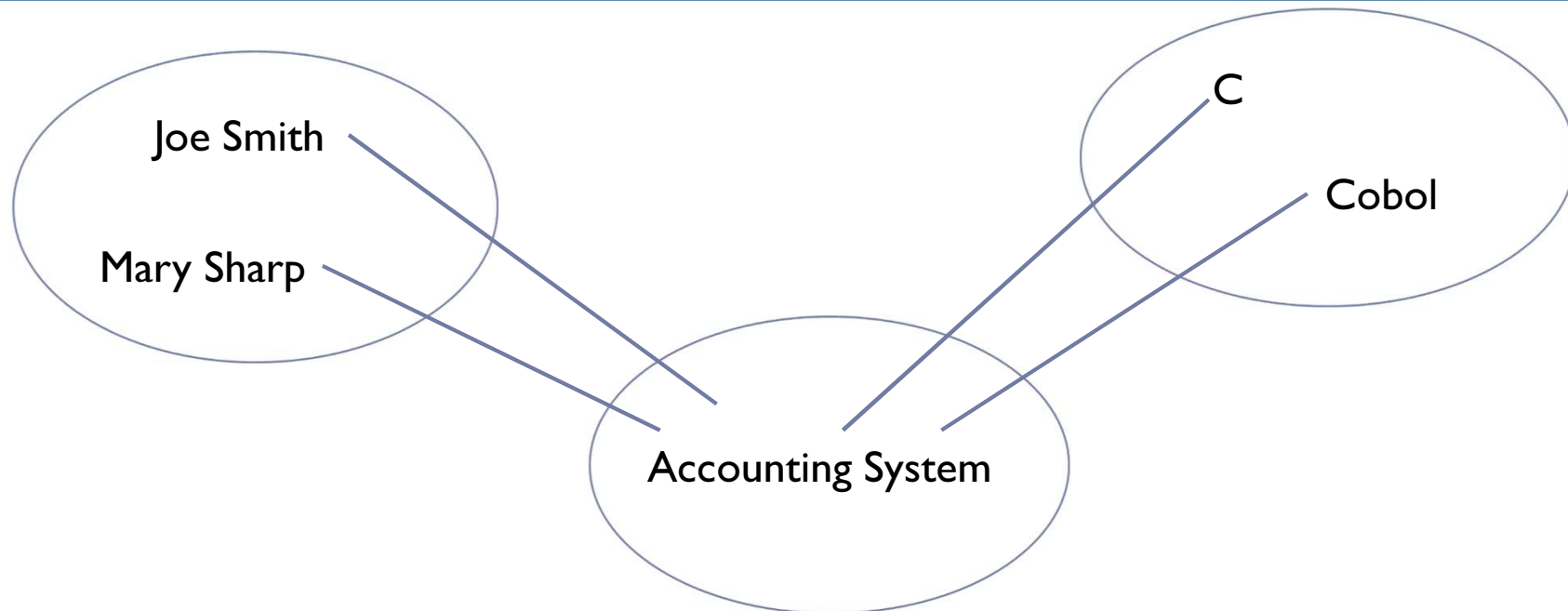


► Are we losing information?

N-ary associations

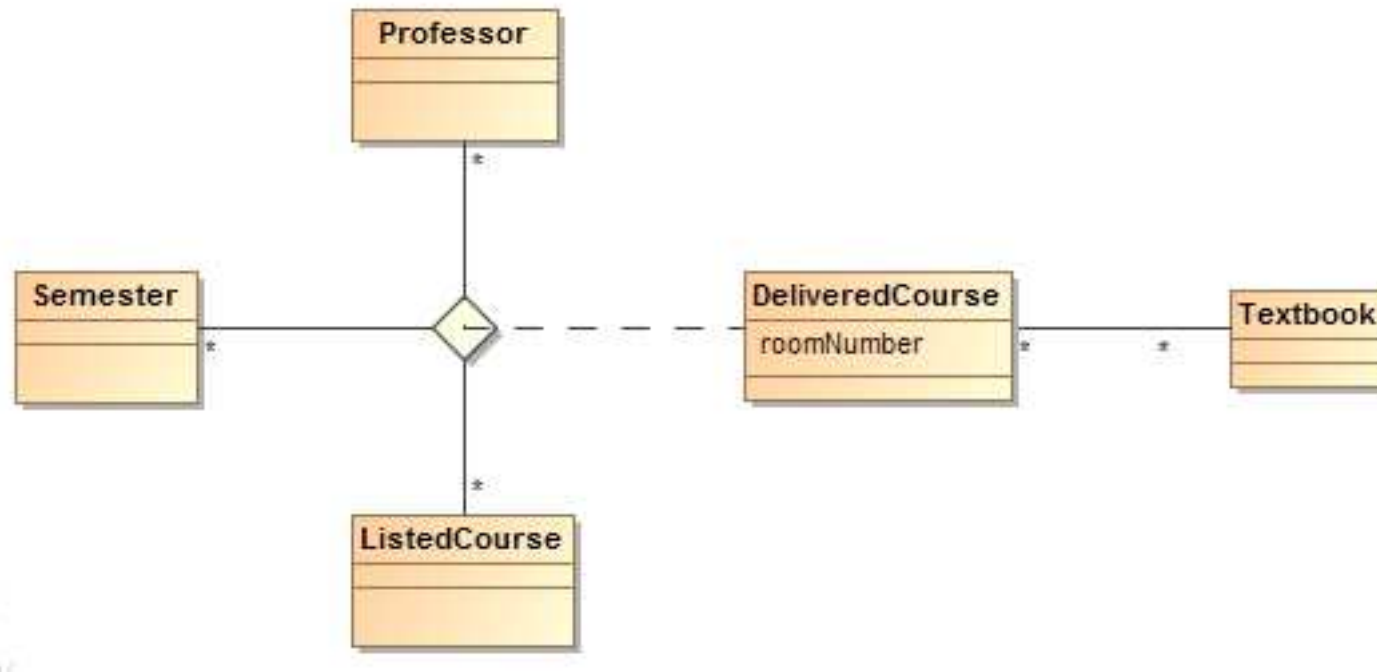


N-ary associations



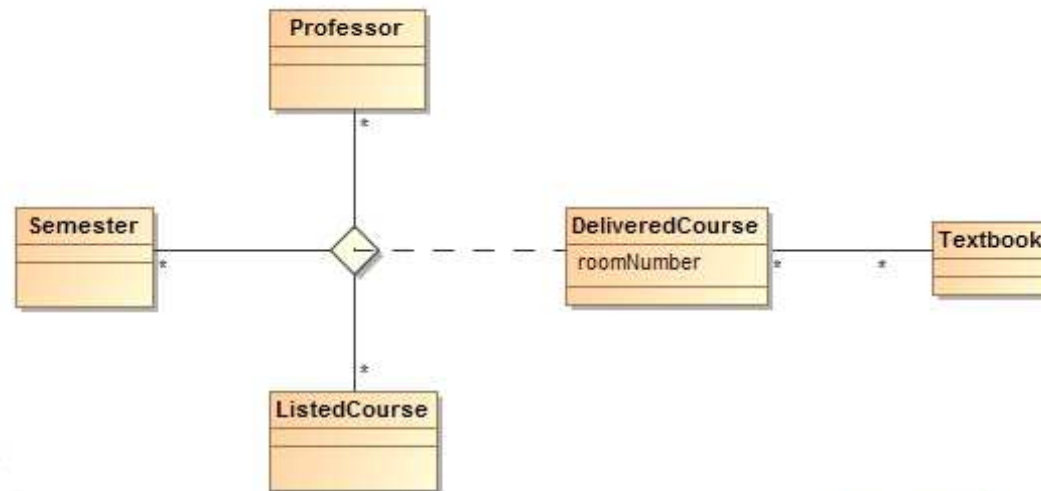
► Are we losing information?

N-ary association classes

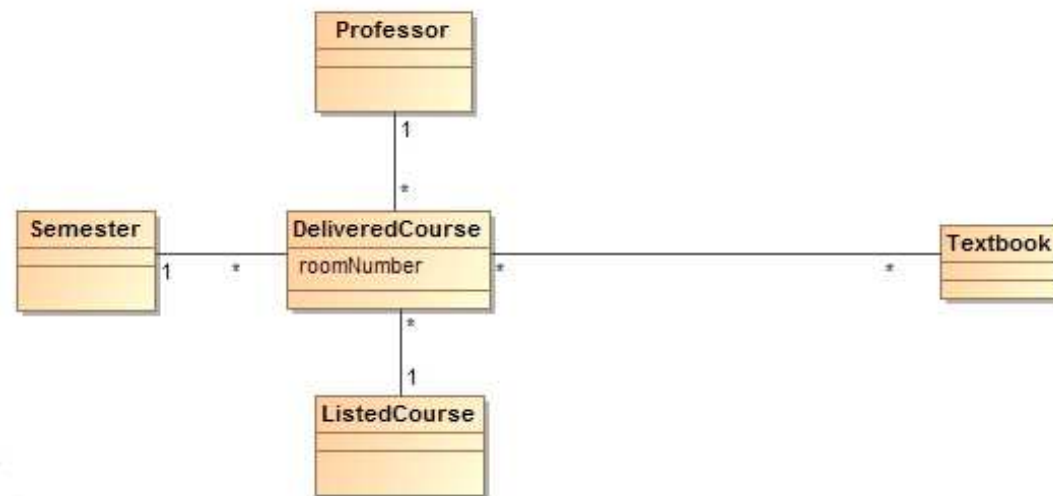


- ▶ Professors teach listed courses during semesters. Each delivered course has a room number and any number of textbooks.

N-ary association classes

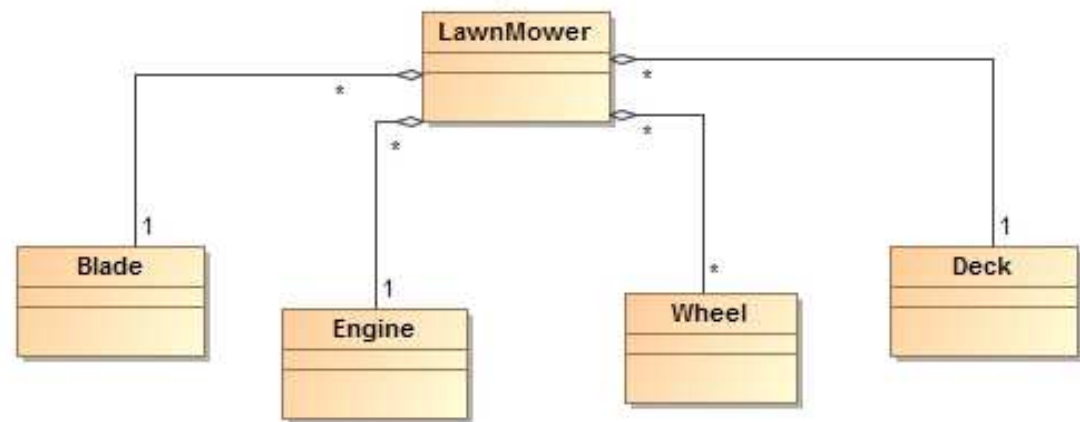


DIFFERENCE?



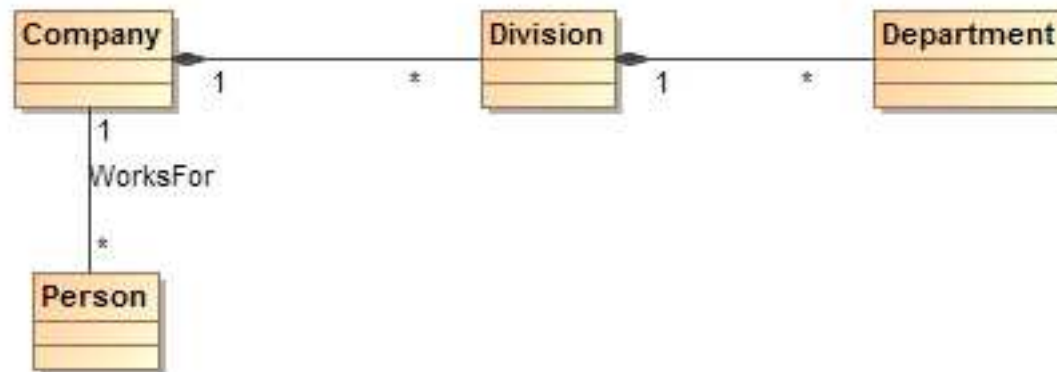
Aggregation

- ▶ Aggregation is a special form of association.
 - ▶ Underlines the fact that an object is made of constituent parts.
- ▶ The UML notation for aggregation is like the one for association with a small diamond indicating the assembly end.



Composition

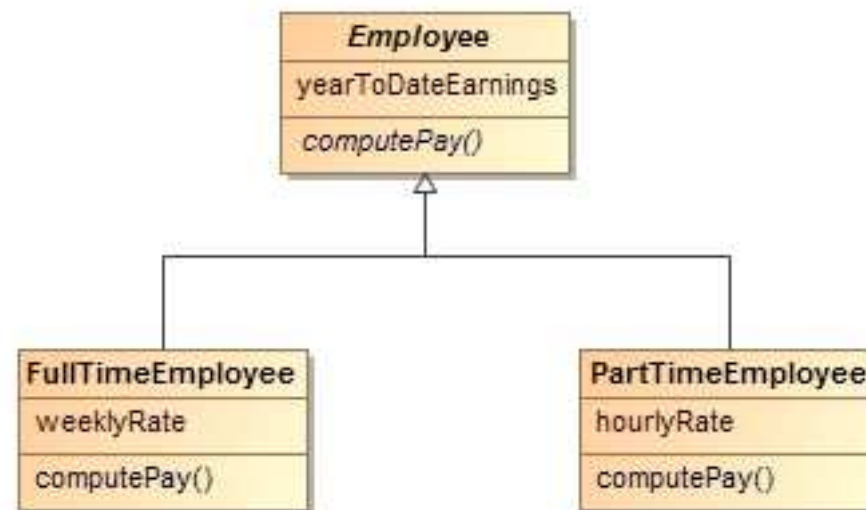
- ▶ Composition is a more restrictive form of aggregation.
 - ▶ Two additional constraints:
 - ▶ A constituent part can belong to at most one assembly.
 - ▶ The part has a coincident lifetime as the assembly.
- ▶ The UML notation for composition is a small solid diamond next to the assembly class.



Abstract classes

- ▶ An abstract class is a class that has no direct instances but whose descendants classes have direct instances.
- ▶ A concrete class is a class that is instantiable.
- ▶ A concrete class may have abstract subclasses, but they in turn must have concrete descendants: only concrete classes can be leaf classes in an inheritance tree.
- ▶ In the UML notation an abstract class name is listed in an italic font (or using {abstract} near the class name) .

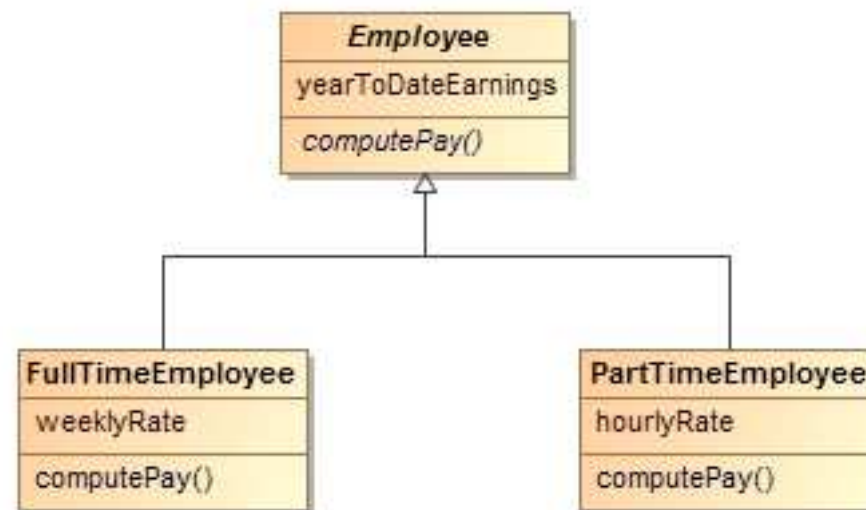
Abstract classes



Abstract classes

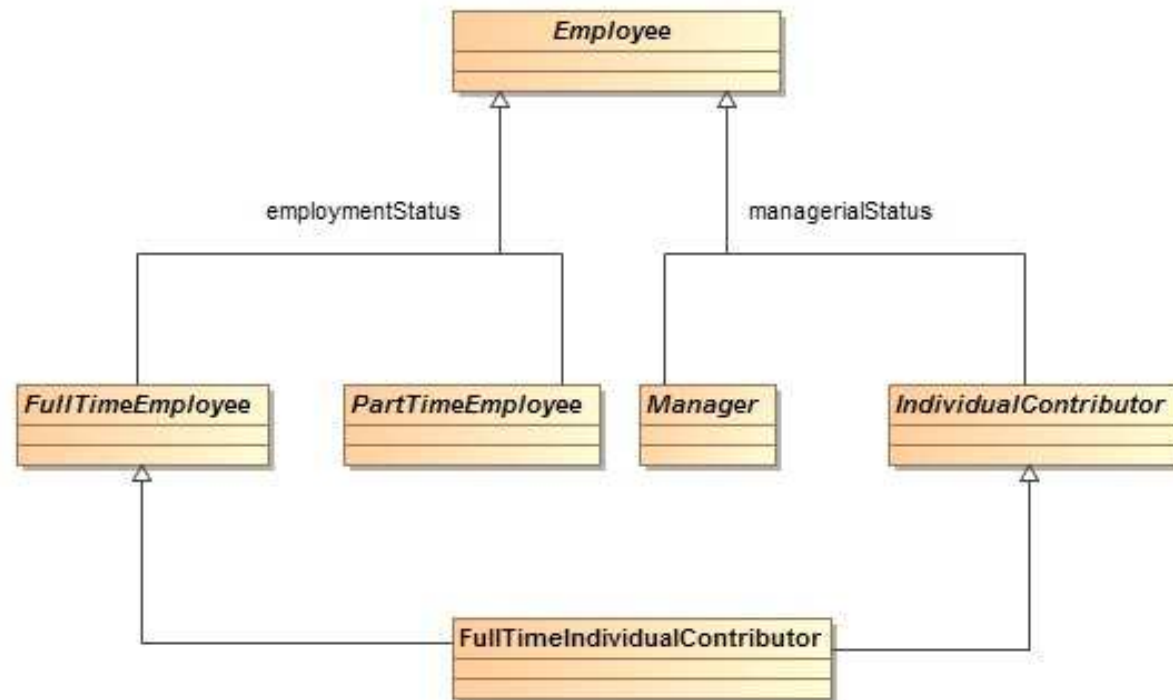
- ▶ Abstract classes can be used to define methods that can be inherited by subclasses.
- ▶ Abstract classes can define the signature of an operation without supplying a corresponding method.
- ▶ Abstract operations:
 - ▶ An abstract operation defines the signature of an operation for which each concrete subclass must provide its own implementation.
 - ▶ An abstract operation is designated by italics or the keyword `{abstract}`.

Abstract classes



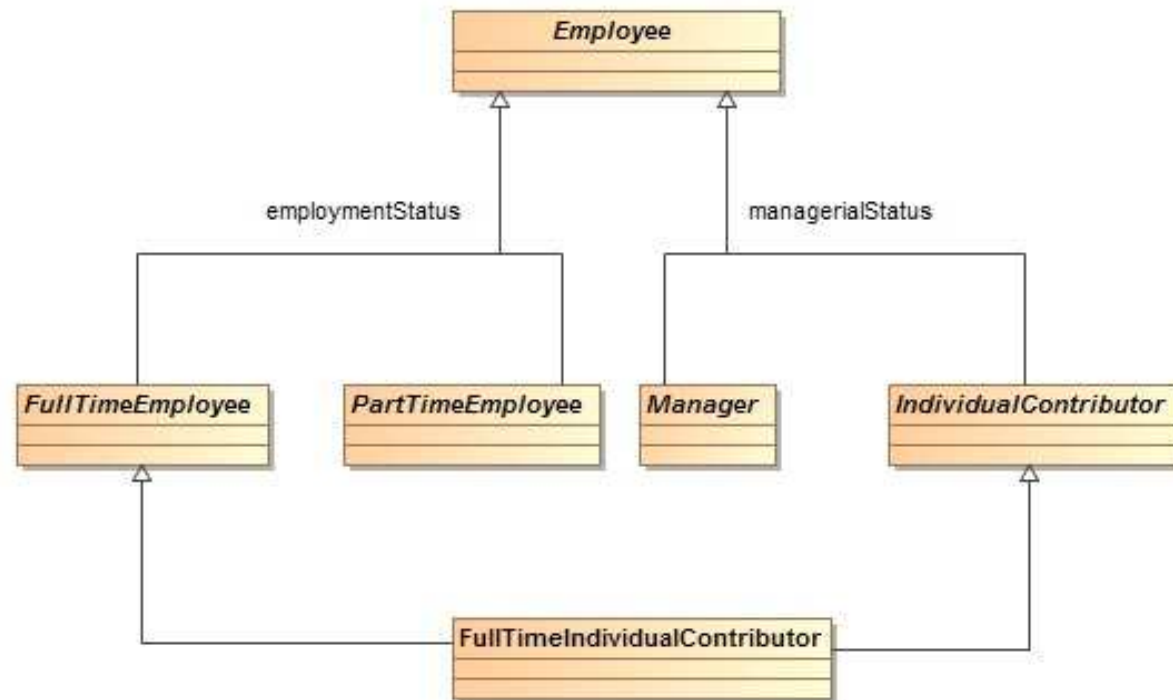
Multiple Inheritance

- ▶ Multiple inheritance permits a class to have more than one superclass and to inherit features from all parents.
- ▶ The most common form of multiple inheritance is from sets of disjoint classes.



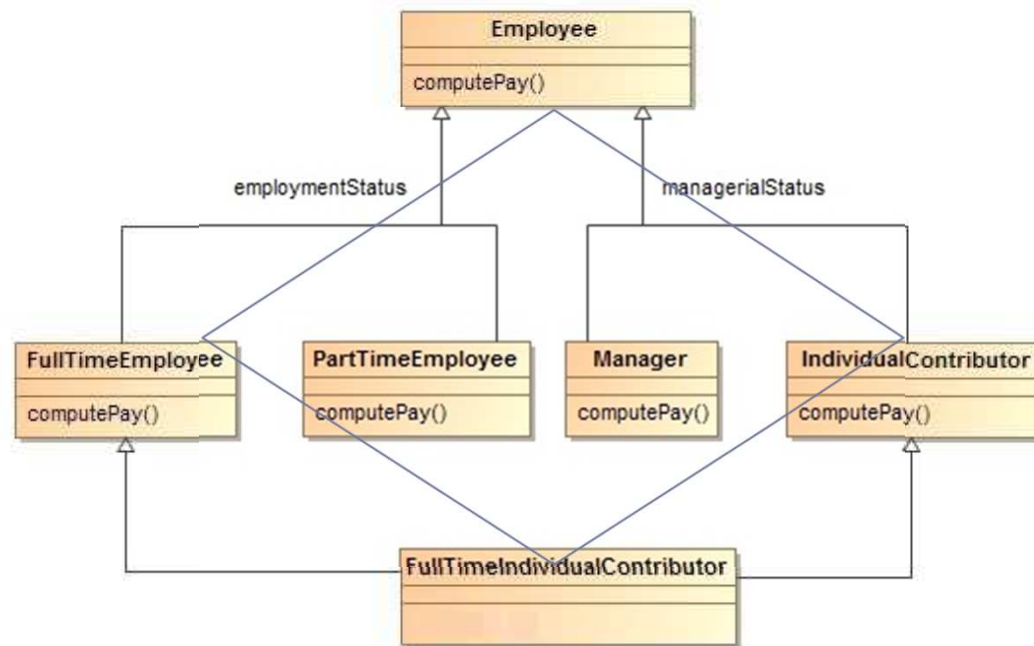
Multiple Inheritance

- ▶ A subclass inherits a feature from the same ancestor class found along more than one path only once.
- ▶ *FullTimeIndividualContributor* inherits *Employee* features along two paths but it has only a single copy of them.



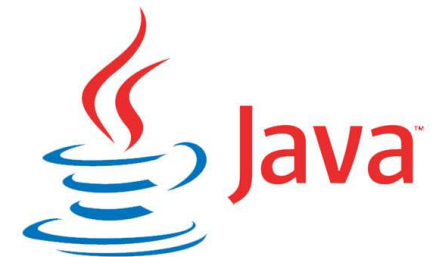
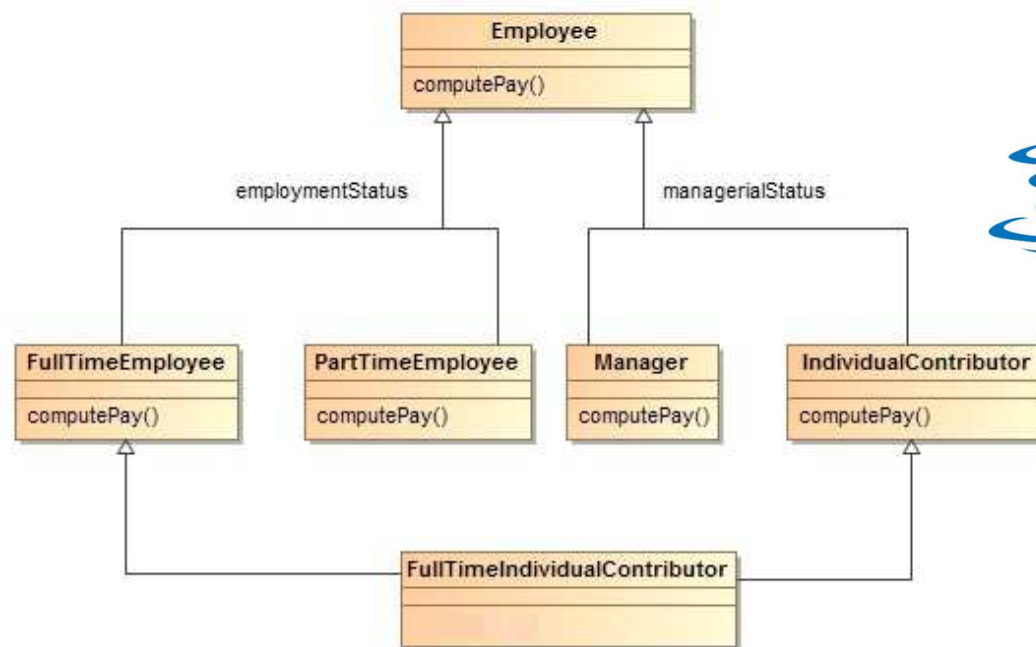
Multiple Inheritance

- ▶ Conflicts among parallel definitions create ambiguities that implementations must resolve.
 - ▶ Diamond problem: which version of *computePay()* should be used in *FullTimeIndividualContributor*?



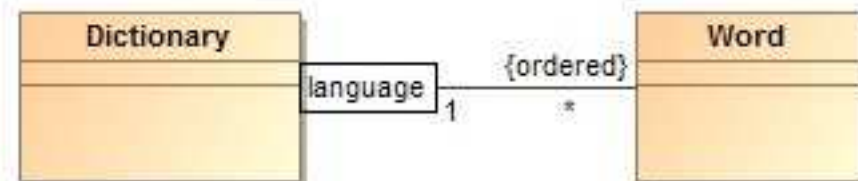
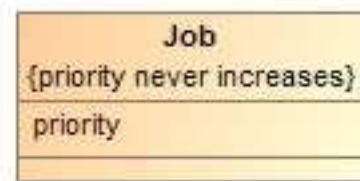
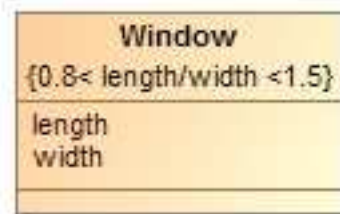
Multiple Inheritance

- ▶ Conflicts among parallel definitions create ambiguities that implementations must resolve.
 - ▶ Diamond problem: which version of *computePay()* should be used in *FullTimeIndividualContributor*?



Constraints

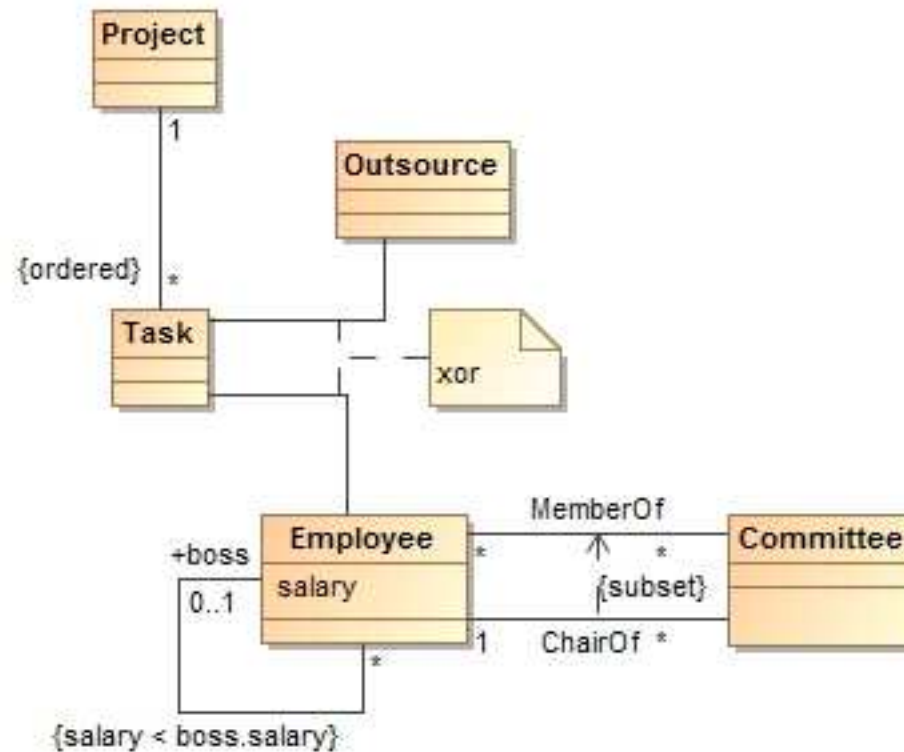
- ▶ A constraint is a condition involving model elements, such as objects, classes, attributes, links, associations and generalization sets.
 - ▶ A constraint restricts the values that elements can assume.
 - ▶ A constraint specifies limitations that implementers need to satisfy.



Constraints

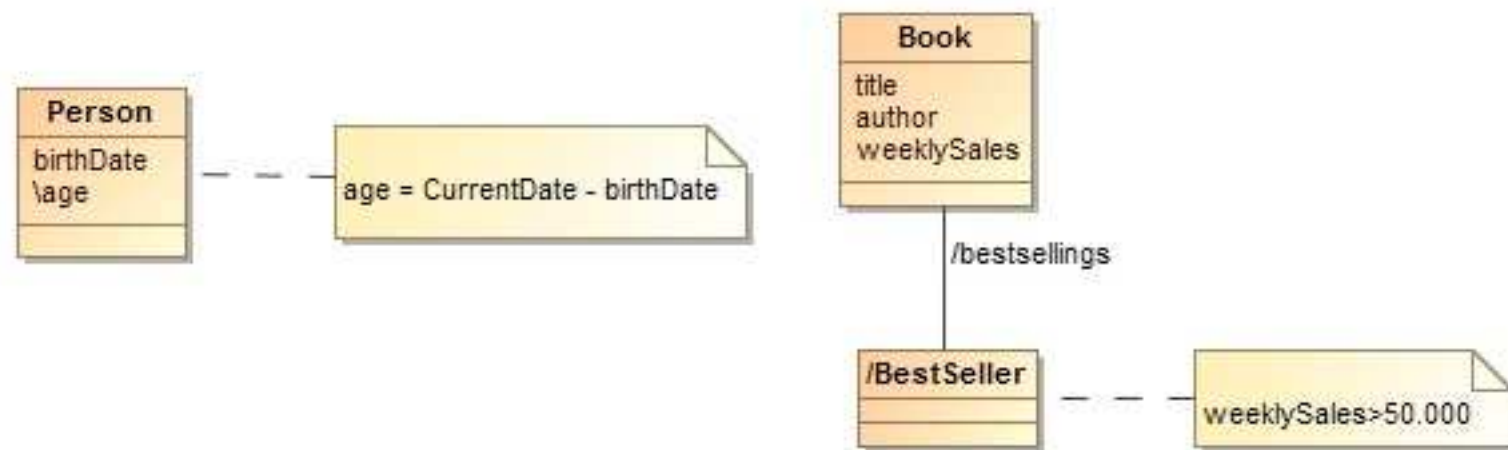
- ▶ Multiplicity is a constraint:
 - ▶ Multiplicity for an association restricts the number of objects associated to a given object.
 - ▶ Multiplicity for an attribute specifies the number of values that are possible for each instantiation of an attribute.
- ▶ Qualification is a constraint:
 - ▶ A qualifier attribute is significant in resolving the “many” objects at an association end.
- ▶ There are several UML notations for constraints:
 - ▶ Delimit constraints with braces.
 - ▶ Place a constraint in a “dog-eared” comment box.
 - ▶ Use dashed lines to connect constrained elements.
 - ▶ Use a dashed arrow to connect a constrained element to the element on which it depends.

Examples of constraints



Derived data

- ▶ A derived element is a function of one or more elements, which in turn can be derived.
- ▶ Ultimately the derivation tree terminates with base elements (elements that cannot be derived).
- ▶ Classes, associations and attributes may be derived.
- ▶ The notation for a derived element is a slash in front of the element name.
- ▶ The constraint that determines the derivation must be shown.



Packages

- ▶ You can fit a class model on a single page for many small and medium-sized problems
 - ▶ However it is often difficult to grasp the entirety of a large model.
- ▶ A package is a group of elements (classes, associations, generalizations and nested packages) with a common theme.
 - ▶ A package partitions a model making it easier to understand and manage,
- ▶ The UML notation for a package is a box with a tab:
 - ▶ The tab suggests the enclosed content, like a tabbed folder.

