Chapter 6:

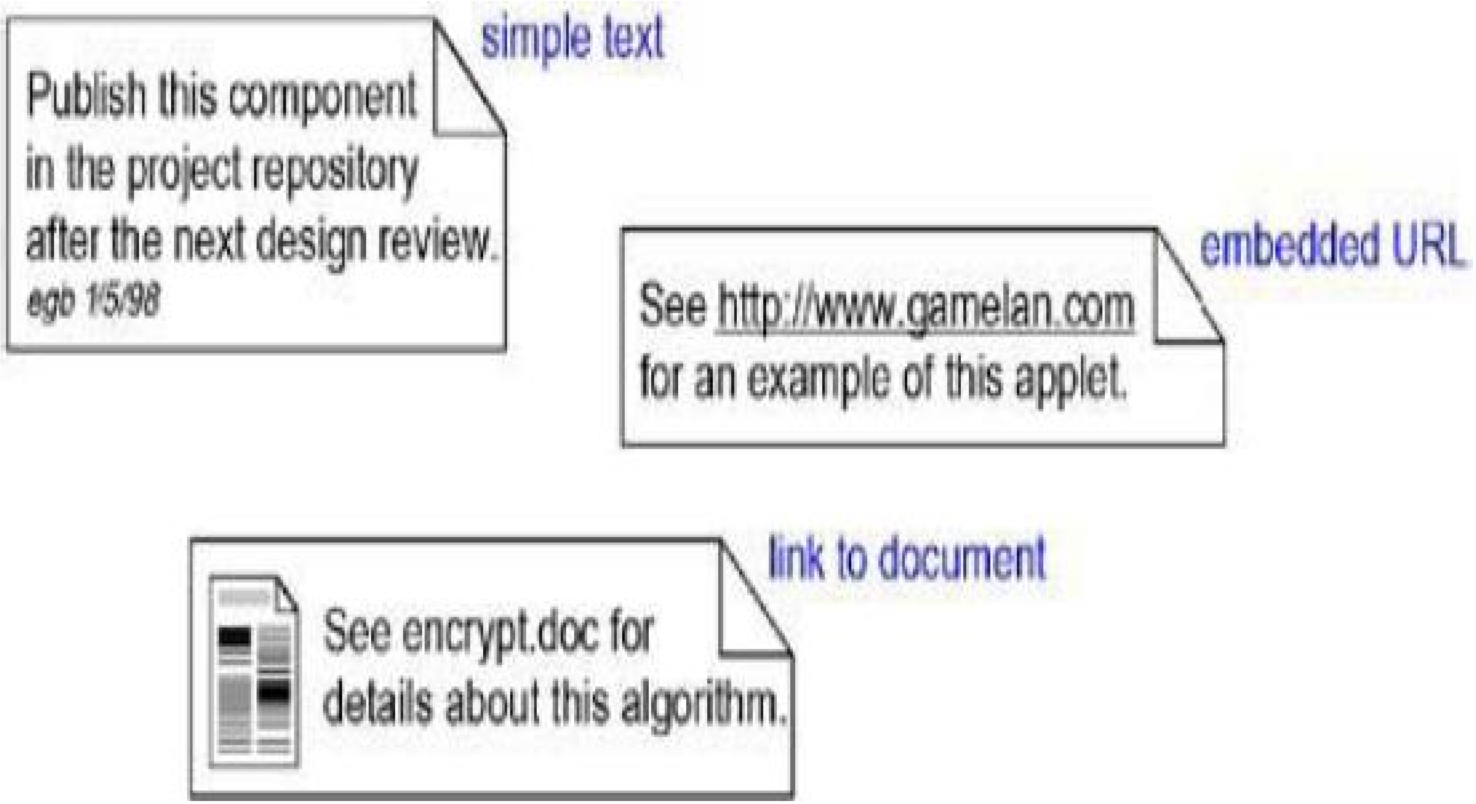
Common Mechanism for UML diagram

Prof. Kirtankumar Rathod
Department of Computer Science
Indus University

Notes:-

- Graphical symbol for rendering constraints or comments attached to an element or collection of elements.
- No Semantic Impact: does not alter the meaning of the model.
- Specify things like: Requirements, Observations, Reviews, Explanations, Constraints

example of notes



Publish this component in the project repository after the next design review.
egb 1/5/98

See http://www.gamelan.com for an example of this applet.

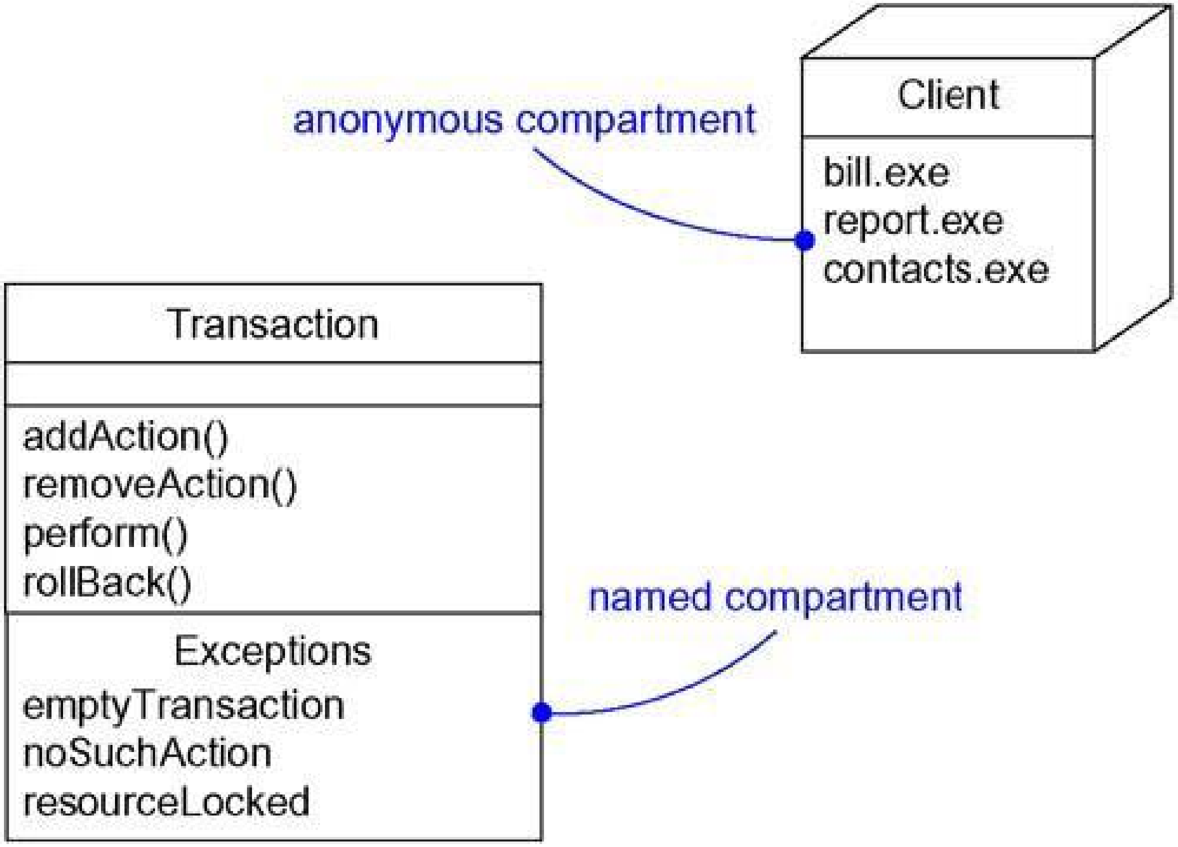
See encrypt.doc for details about this algorithm.

Notes

Adornments:-

Adornments are textual or graphical items that are added to an elements basic notation and are used to visualize details from the elements specification.

- Placed near the element as
  - Text
  - Graphic
- Special compartments for adornments in
  - Classes
  - Components
  - Nodes



Extra Compartments

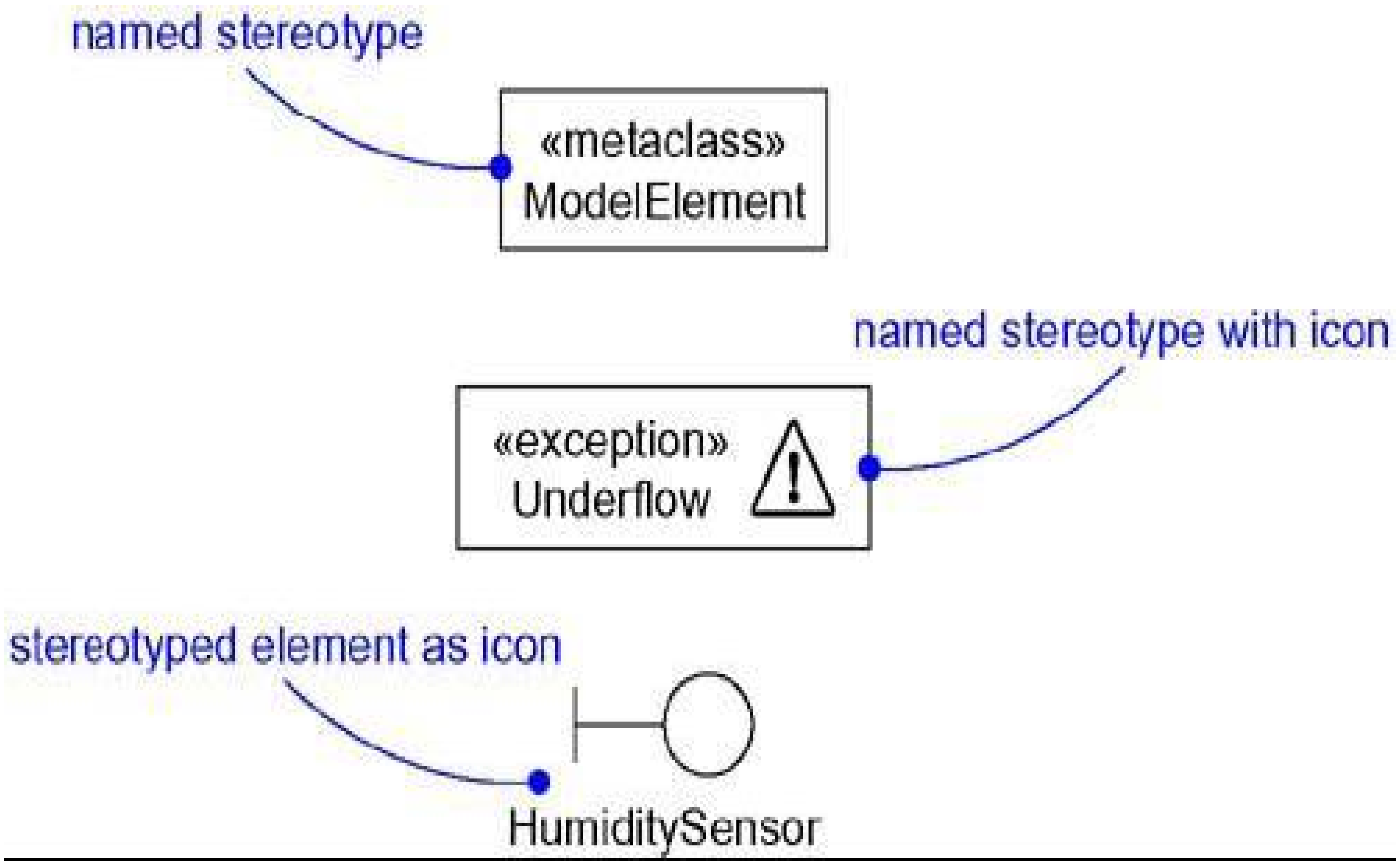
Stereotypes:-

A Stereotype is a metatype, because each one creates the equivalent of a new class in the UML's methodology.

Named stereotype

Named stereotype with Icon

Stereotype element as Icon

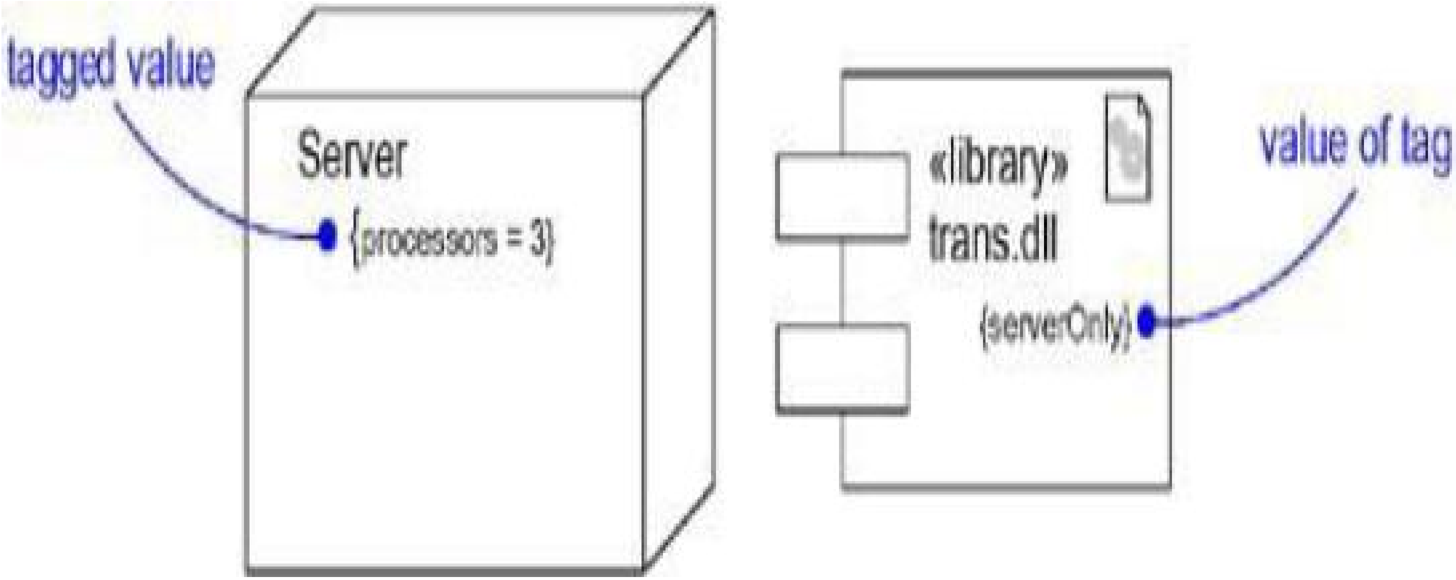


Stereotypes

Tagged Values:-

Tagged values add new properties.

A tagged value is not the same as a class attribute. Rather, you can think of a tagged value as metadata because its value applies to the element itself, not its instances.

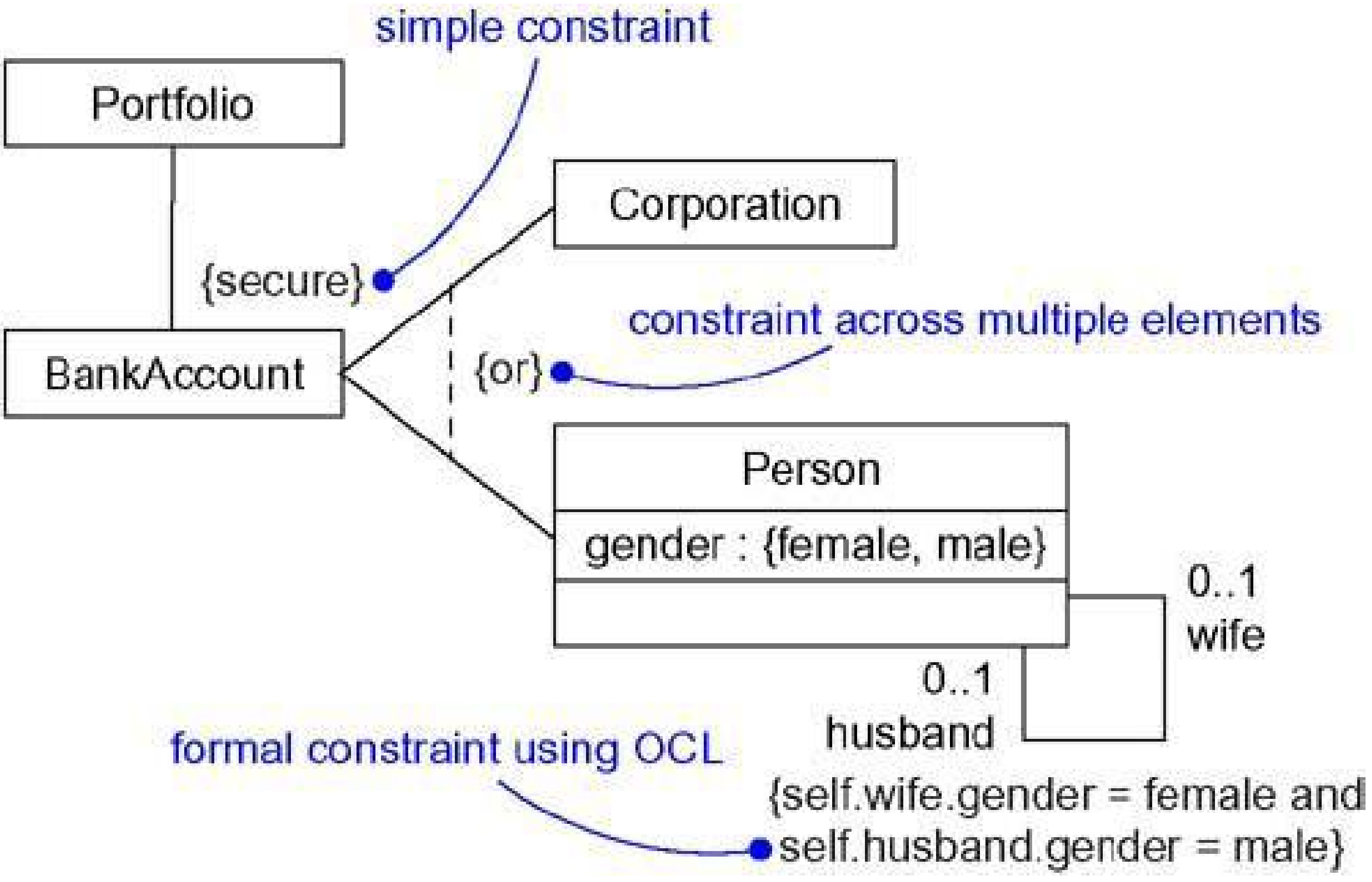


Tagged Values

Constraints:-

Constraints add new semantics or change existing rules.

A constraint specifies conditions that must be held true for the model to be well-formed.


simple constraint
Portfolio
Corporation
{secure}
constraint across multiple elements
BankAccount
{or}
Person
gender : {female, male}
0..1
wife
0..1
husband
formal constraint using OCL
{self.wife.gender = female and
self.husband.gender = male}

Constraint

Class Diagrams: Basic Concepts

Prof. Kirtankumar Rathod
Dept. of Computer Science
Indus University

Objects

- **The purpose of class modeling is to describe objects.**
- **An object is a concept, abstraction or thing that has meaning for a domain/application.**
- **Some objects have real world counterparts while others are conceptual entities.**
- **The choice of objects depends on judgment and the nature of problem.**
- **◌All objects have identity and are distinguishable.**

Classes

- An **object** is an **instance** – occurrence – of **a class**
- ◌ **A class describes a group of objects with the same properties (attributes), behavior (operations), kinds of relationships, and semantics.**
- **The objects in a class share a common semantic purpose, above and beyond the requirement of common attributes and behavior.**
- **By grouping objects onto classes we abstract a problem.**

UML representation of classes/objects:

- **UML: Unified Modeling Language (OMG Standard): O.O Visual Modeling language**

- **Class/object representation**

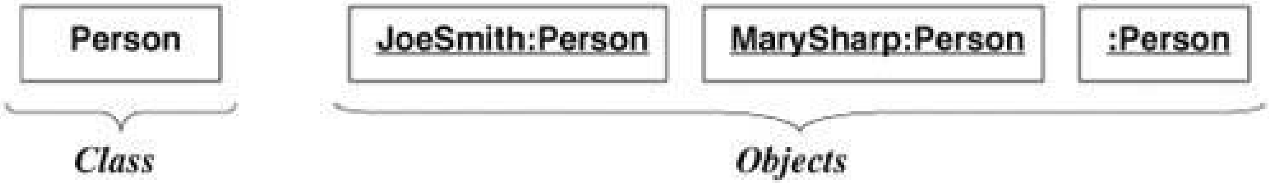


Figure 3.1 A class and objects. Objects and classes are the focus of class modeling.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha
and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc.,
Upper Saddle River, NJ. All rights reserved.

Values and attributes

- **Value : piece of data.**
- **Attribute: a named property of a class that describes a value held by each object of the class.**
  - **Attributes may be discovered by looking for adjectives or by abstracting typical values.**
  - **Don’t confuse values with objects:**
    - **An attribute should describe values, not objects.**
    - **Unlike objects, values lac identity**

UML representation

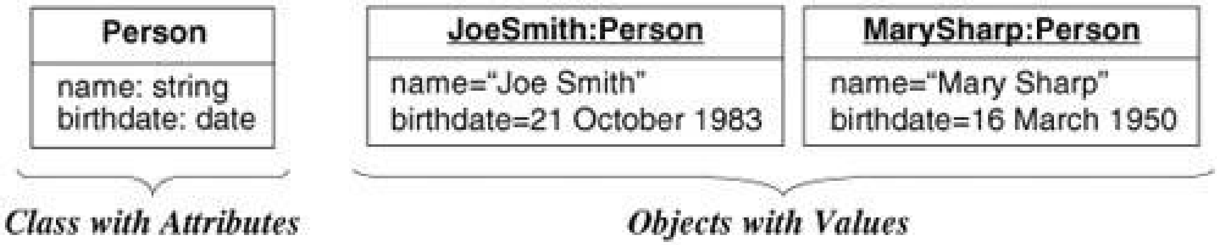


Figure 3.2 Attributes and values. Attributes elaborate classes.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Object identifiers

- Objects identifiers are implicit.
- Objects belonging to the same and having the same attributes values may be different individual objects.

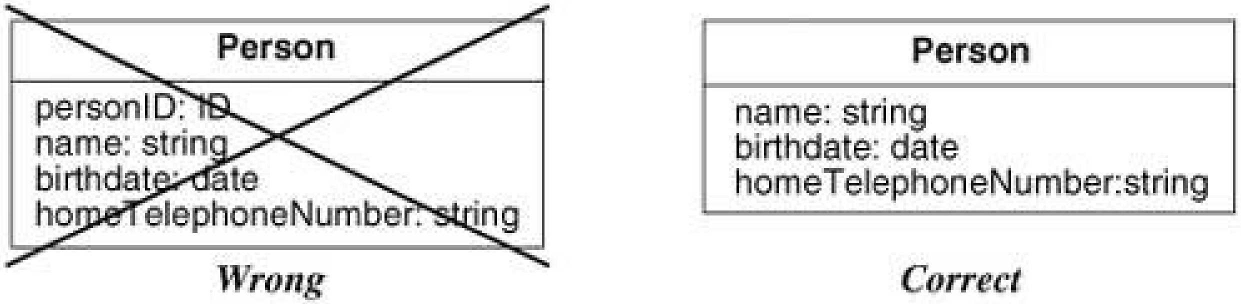


Figure 3.3 Object identifiers. Do not list object identifiers; they are implicit in models.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh, ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

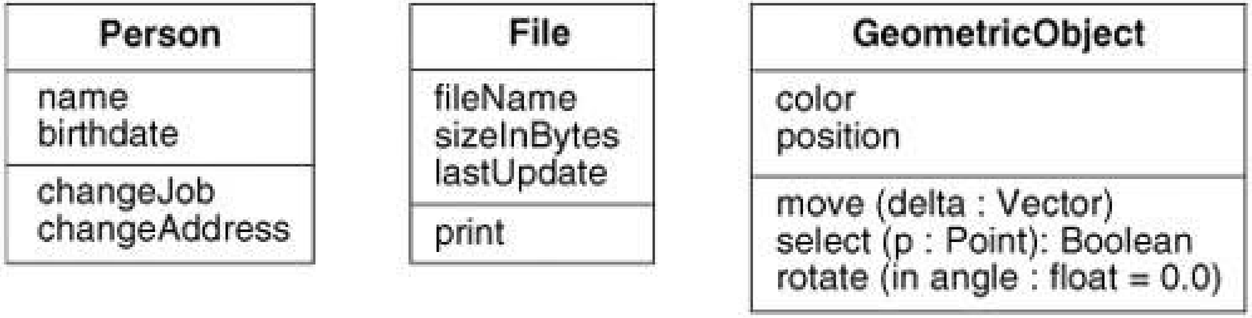
Operations and Methods

- **An operation is a function or procedure that may be applied to or by objects in a class.**
- **Each operation has a target object as an implicit parameter.**
- **All objects in a class share the same operations.**
- **The behavior of the operation depends on the class of its target.**
- **The same operation may apply to many different classes. Such an operation is POLYMORPHIC.**

Operations and Methods

- **A method is the implementation of an operation for a class.**
- **A different piece of code may implement each method.**
- **An operation may have arguments in addition to its target object. These arguments may be placeholders for values and/or for objects.**
- **When an operation has methods on several classes these methods must have the same SIGNATURE: number and types of arguments, type of result value.**

UML representation



Person

name
birthdate

changeJob
changeAddress

File

fileName
sizeInBytes
lastUpdate

print

GeometricObject

color
position

move (delta : Vector)
select (p : Point): Boolean
rotate (in angle : float = 0.0)

Figure 3.4 Operations. An operation is a function or procedure that may be applied to or by objects in a class.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Links and Association concepts

- A link is a physical or conceptual connection among objects.
- Most links relate two objects, but some links relate three or more objects.
- A link is defined as a tuple, that is a list of objects.
- A link is an instance of an association.
- An association is a description of a group of links with common structure and semantics.
- Association is denoted by a line. Its name is optional if the model is unambigious.

Examples

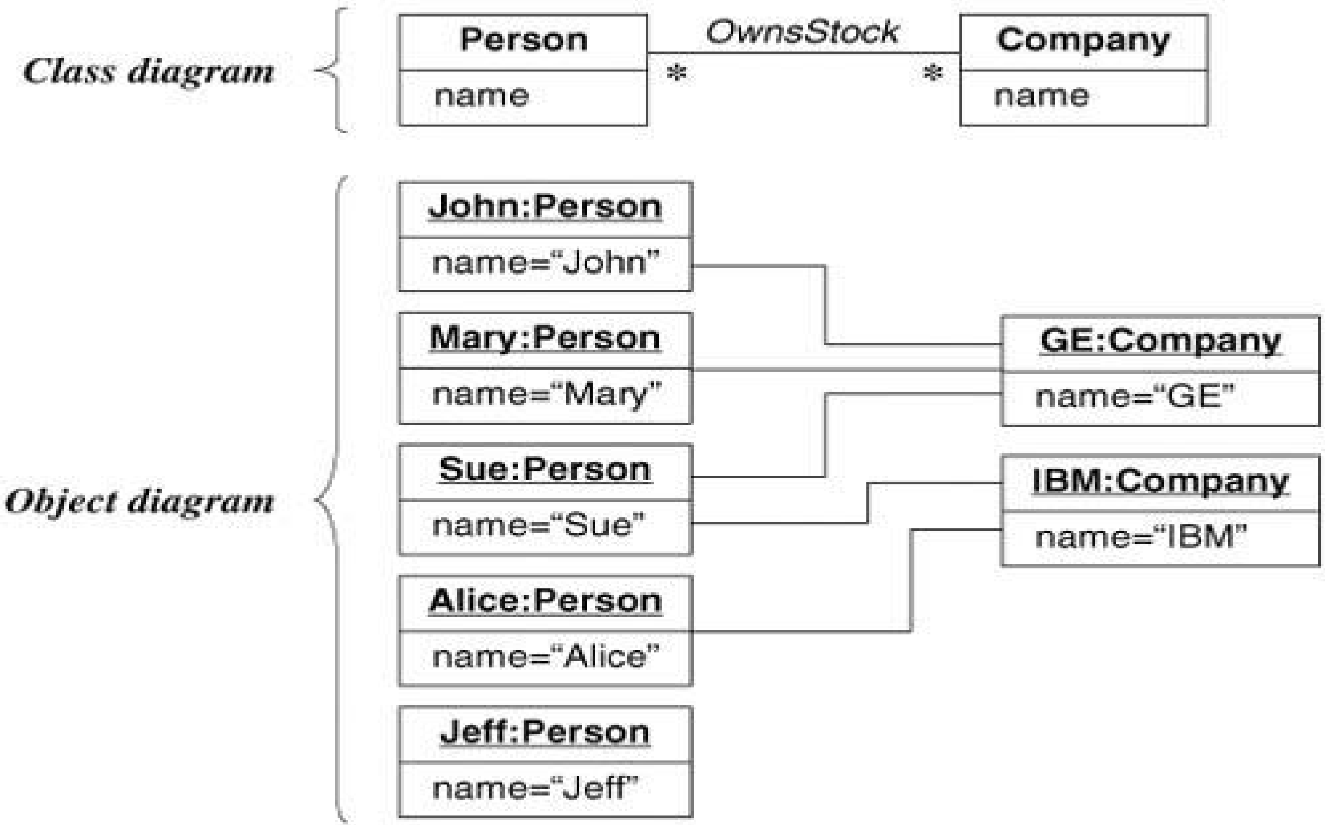


Figure 3.7 Many-to-many association. An association describes a set of potential links in the same way that a class describes a set of potential objects.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

- Associations are inherently bi-directional. The association name is usually read in a particular direction but the binary association may be traversed in either direction.

- ◌ A reference is an attribute in one object that refers to another object.

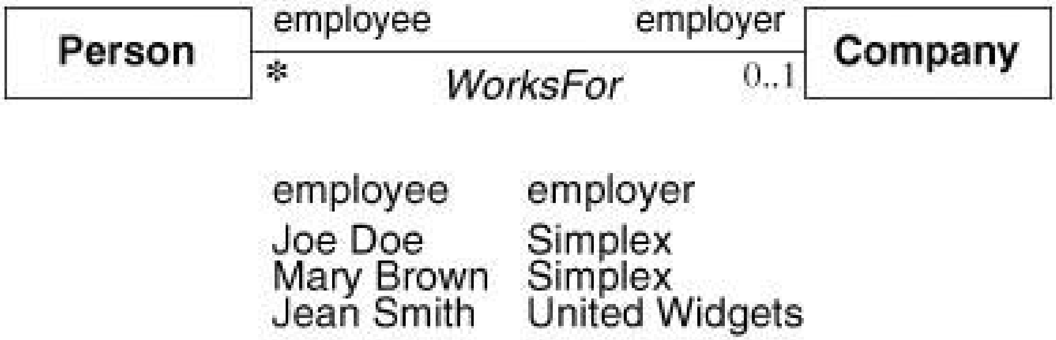
Multiplicity

- It specifies the number of instances of one class that may relate to a single instance of the associated class.

- UML diagrams explicitly list multiplicity at the end of association lines.
- Intervals are used to express multiplicity:
  - 1 (exactly one)
  - 0..1 (zero or one)
  - 1..* (one or more)
  - 0..* (zero or more)
  - 3..5 (three to five inclusive)

Association Ends

- Associations have ends. They are called ‘Association Ends’.
- They may have names (which often appear in problem descriptions).



employee	employer
Joe Doe	Simplex
Mary Brown	Simplex
Jean Smith	United Widgets

Figure 3.12 Association end names. Each end of an association can have a name.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Association Ends

- Use of association end names is optional.
- But association end names are useful for traversing associations.
- ◌Association end names are necessary for for associations between obje

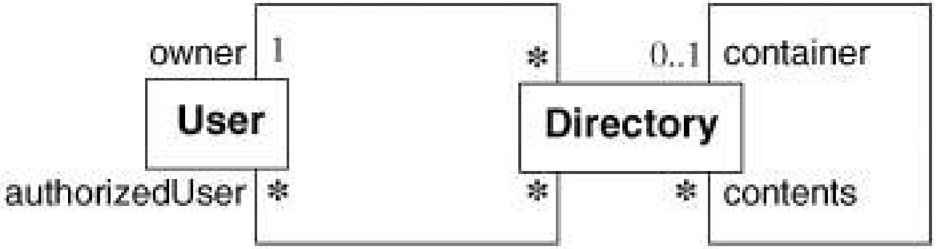


Figure 3.13 Association end names. Association end names are necessary for associations between two objects of the same class. They can also distinguish multiple associations between a pair of classes.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4.
© 2005 Pearson Education, Inc., Upper Saddle River, NJ.
All rights reserved.

Example of association ends use

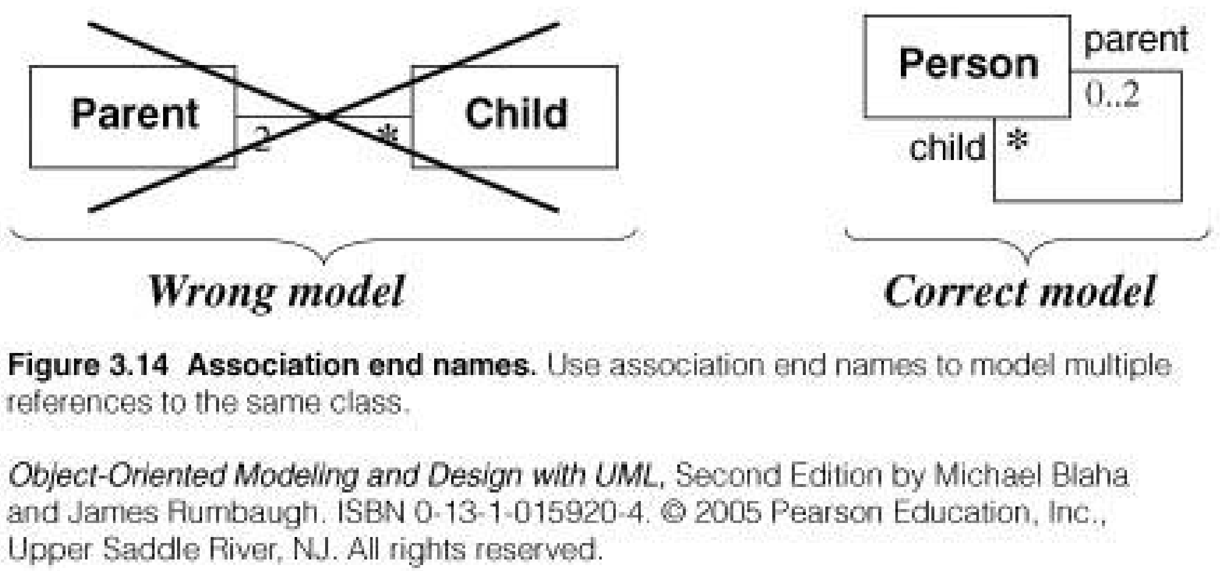


Figure 3.14 Association end names. Use association end names to model multiple references to the same class.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Association: ordering, bag, sequence

- On a ‘many” association end, sometimes, it is required that objects have an explicit order.
- In this case the ordering is an inherent part of the association
- Example:

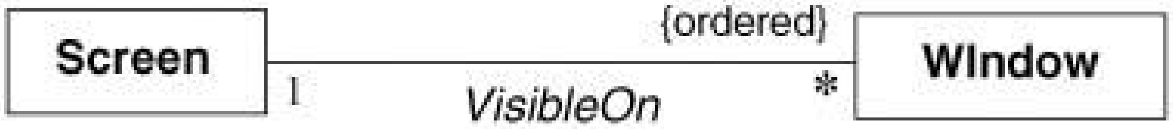


Figure 3.15 Ordering the objects for an association end. Ordering sometimes occurs for "many" multiplicity.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Association: ordering, bag, sequence

- Ordinary a binary association has at most one link for a pair of objects
- However we can permit multiple links for a pair of objects by annotating an association end with {bag} or {sequence}
- A **bag** is a collection of elements with duplicates allowed.
- A **sequence** is ar duplicates allowe

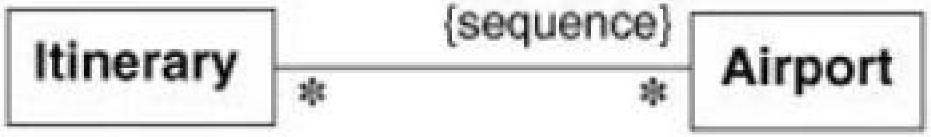
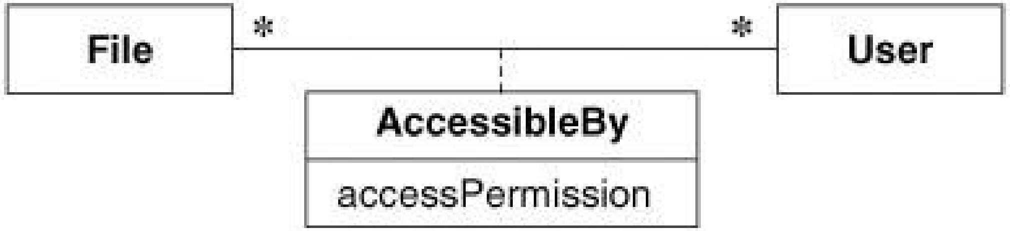


Figure 3.16 An example of a sequence. An itinerary may visit multiple airports,
so you should use {sequence} and not {ordered}.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Association class

- UML offers the ability to describe links of association with attributes like any class.
- An association class is an association that is also a class.



/etc/termcap	read	John Doe
/etc/termcap	read-write	Mary Brown
/usr/doe/.login	read-write	John Doe

Figure 3.17 An association class. The links of an association can have attributes.

Object-Oriented Moldeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Association class

- Examples:

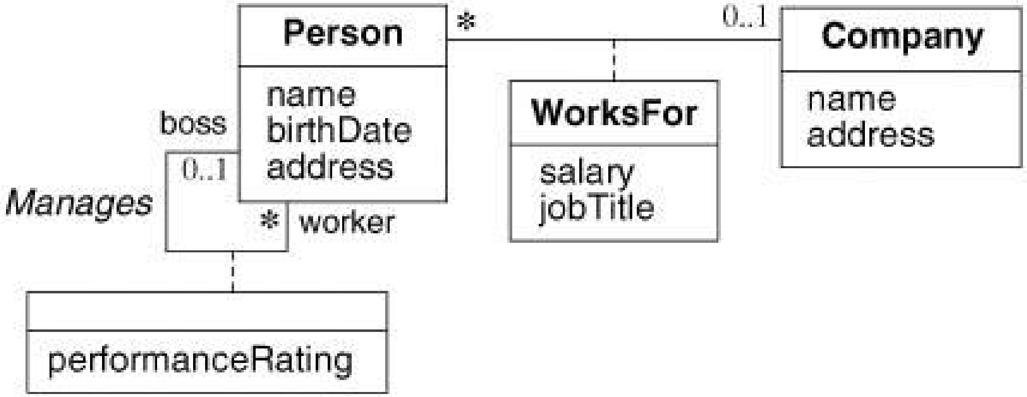


Figure 3.18 Association classes. Attributes may also occur for one-to-many and one-to-one associations.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Association class

- Example

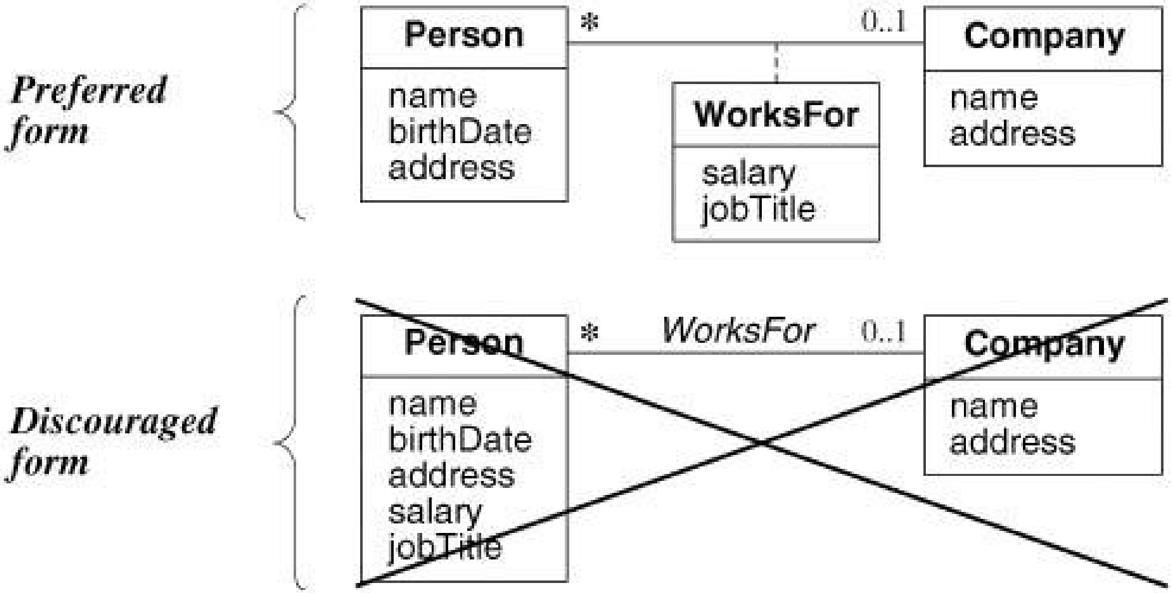


Figure 3.19 Proper use of association classes. Do not fold attributes of an association into a class.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha
and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc.,
Upper Saddle River, NJ. All rights reserved.

Qualified Association

- A qualified association is an association in which an attribute called Qualifier disambiguates the objects for a ‘many’ association’ end.
- A qualifier selects among the target objects, reducing the effective multiplicity fro ‘many’ to ‘one’.
- Both below models are acceptable but the qualified model adds information.

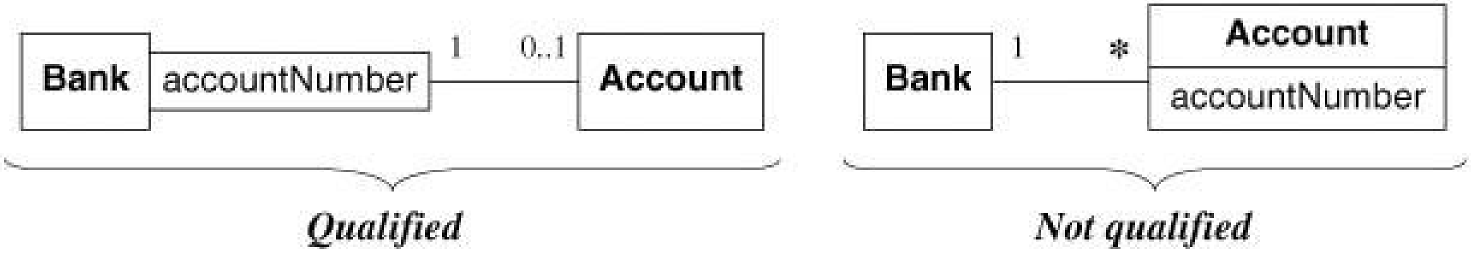


Figure 3.22 Qualified association. Qualification increases the precision of a model.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh.
ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Qualified Association

- Example:

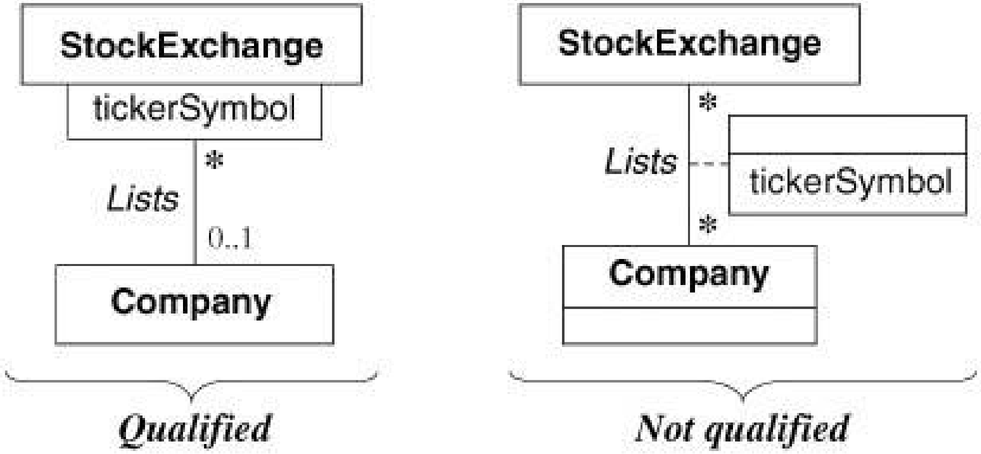


Figure 3.23 Qualified association. Qualification also facilitates traversal of class models.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Generalization/Inheritance

- Generalization is the relationship between a class (**superclass**) and one or more **variations** of the class (**subclasses**).
- Generalization organizes classes by their **similarities** and their **differences, structuring** the descriptions of objects.
- A superclass holds **common** attributes, attributes and associations.
- The subclasses **adds specific** attributes, operations, and associations. They i**nherit** the features of their superclass.
- Often **Generalization** is called a “**IS A”** relationship
- **Simple generalization** organizes classes into a **hierarchy**.
- A subclass may **override** a superclass **feature (**attribute default values, operation) by **redefining a feature with the same name**.
- Never override the signature of methods.

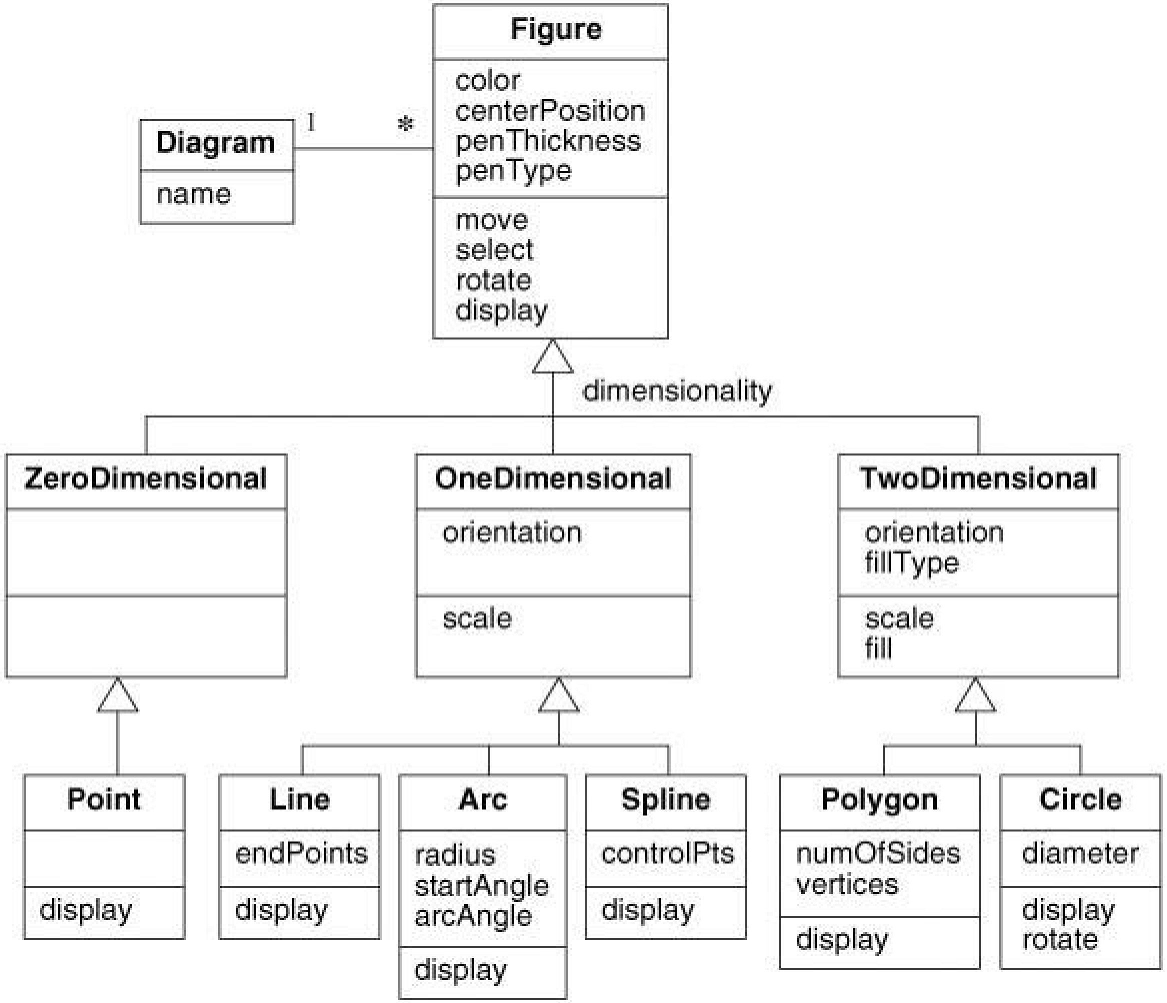


Figure 3.25 Inheritance for graphic figures. Each subclass inherits the attributes, operations, and associations of its superclasses.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha and James Rumbaugh.
ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc., Upper Saddle River, NJ. All rights reserved.

Use of generalization

- **Used for three purposes:**
  - Support of polymorphism:
    - polymorphism increases the flexibility of software.
    - Adding a new subclass and automatically inheriting superclass behavior.
  - Structuring the description of objects:
    - Forming a taxonomy (classification), organizing objects according to their similarities. It is much more profound than modeling each class individually and in isolation of other similar classes.
  - Enabling code reuse:
    - Reuse is more productive than repeatedly writing code from scratch.

Aggregation

- **Aggregation** is a **strong** form of **association** in which an **aggregate object** is made of **constituent parts.**
- The **aggregate** is semantically an extended object that is treated as a **UNIT in many operations, although physically it is made of several lesser objects.**

- **Aggregation is a transitive relation:**
  - if A is a part od B and B is a part of C then A is also a part of C
- **Aggregation is an antisymmetric relation:**
  - If A is a part of B then B is not a part of A.

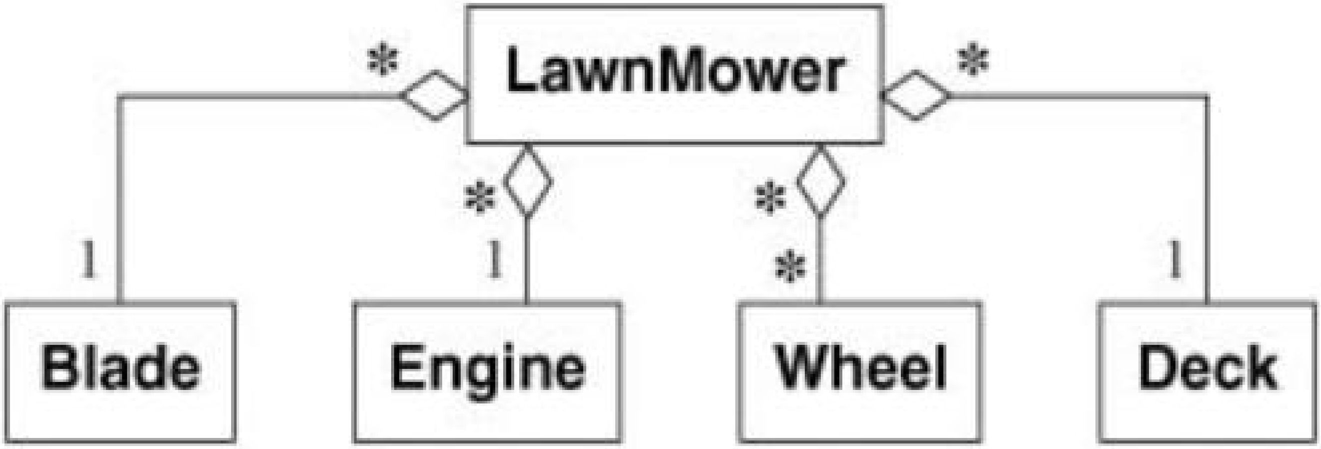


Figure 4.9 Aggregation. Aggregation is a kind of association in which an aggregate object is made of constituent parts.

Object-Oriented Modeling and Design with UML, Second Edition
by Michael Blaha and James Rumbaugh.
ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc.,
Upper Saddle River, NJ. All rights reserved.

Aggregation versus Association

- Aggregation is a special form of association, not an independent concept.
- Aggregation adds semantic connotations:
  - If two objects are tightly bound by a part-whole relation it is an aggregation.
  - If the two objects are usually considered as independent, even though they may often be linked, it is an association. ◌ˏ
- Discovering aggregation
  - Would you use the phrase **part of** ?
  - Do some operations on the whole automatically apply to its parts?
  - Do some attributes values propagates from the whole to all or some parts?
  - Is there an asymmetry to the association, where one class is subordinate to the other?

Aggregation versus Composition

- **Composition** is a form of aggregation with additional constraints:
  - A constituent part can belong to **at most one** assembly (whole).
    - it has a coincident lifetime with the assembly.
    - Deletion of an assembly object triggers automatically a deletion of all constituent objects via composition.
  - Composition implies ownership of the parts by the whole.
    - Parts cannot be shared by different wholes.

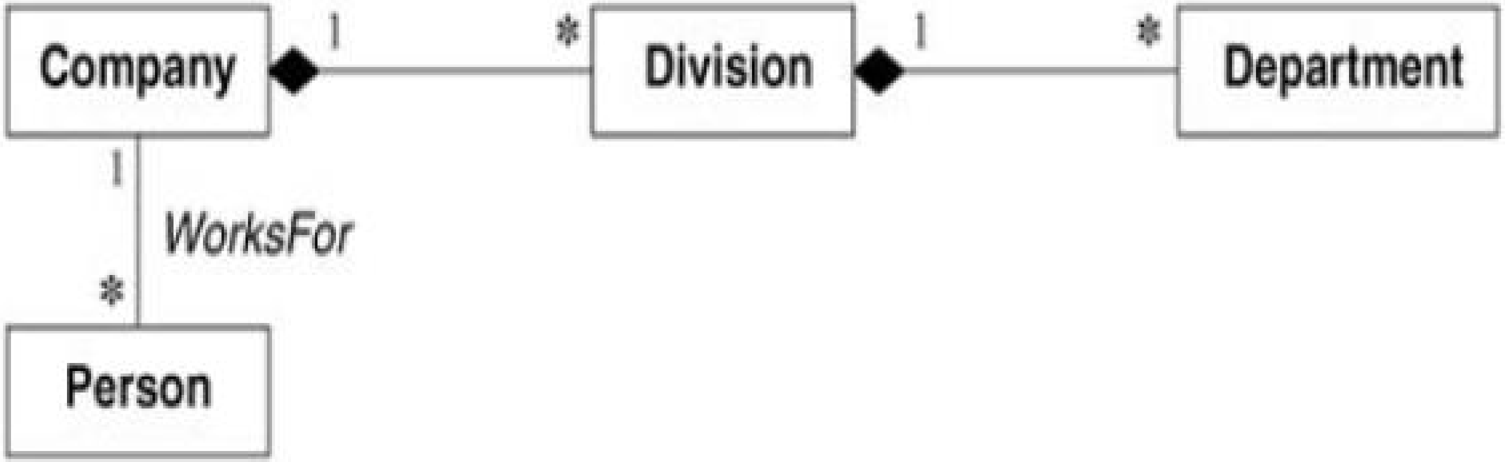


Figure 4.10 Composition. With composition a constituent part belongs to at most one assembly and has a coincident lifetime with the assembly.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha
and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc.,
Upper Saddle River, NJ. All rights reserved.

Propagation of operations

- Propagation is the automatic application of an operation to a network of objects when the operation is applied to some starting object.
- Propagation of operations to parts is often a good indicator of propagation.

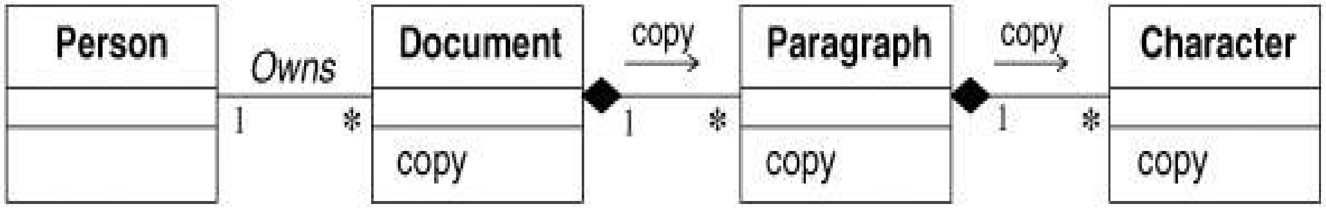


Figure 4.11 Propagation. You can propagate operations across aggregations and compositions.

Object-Oriented Modeling and Design with UML, Second Edition by Michael Blaha
and James Rumbaugh. ISBN 0-13-1-015920-4. © 2005 Pearson Education, Inc.,
Upper Saddle River, NJ. All rights reserved.