UML: Introduction

Unified Modeling Language (UML)

Tool for Lab: StarUML
https://docs.staruml.io/

Topics covered in these Sessions

1. Introducing UML.
2. What constitutes the UML.
3. Relationship in UML.
4. Class diagrams in UML

What is UML?

- Is a *language*: capturing knowledge(semantics) about a subject and expressing knowledge(syntax) regarding the subject for the purpose of communication.
- Applies to *modeling* and systems: a focus on a subject (system) and capturing and being able to communicated this knowledge.

Unified Modeling Language (UML)

- Version 1.1 was adopted in November 1997 by the Object Management Group (OMG) as a standard language for object-oriented analysis and design
- Initially based on a combination of the Booch, OMT (Object Modeling Technique): Jim Rumbaugh and OOSE (Object-Oriented Software Engineering): Ivar Jacobson methods
- UML was refined and extended by a consortium of several companies, and is undergoing minor revisions by the OMG Revision Task Force.

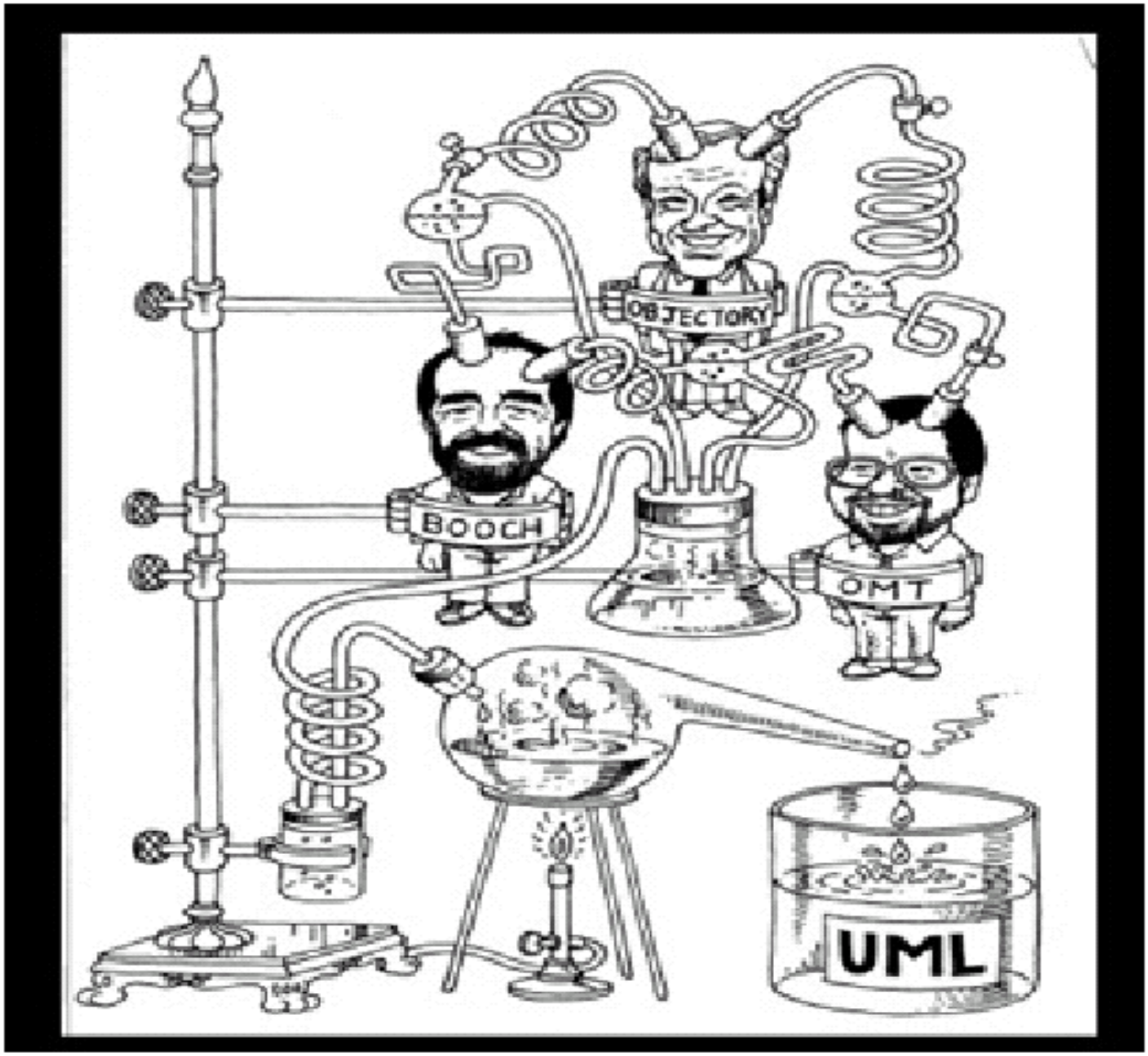
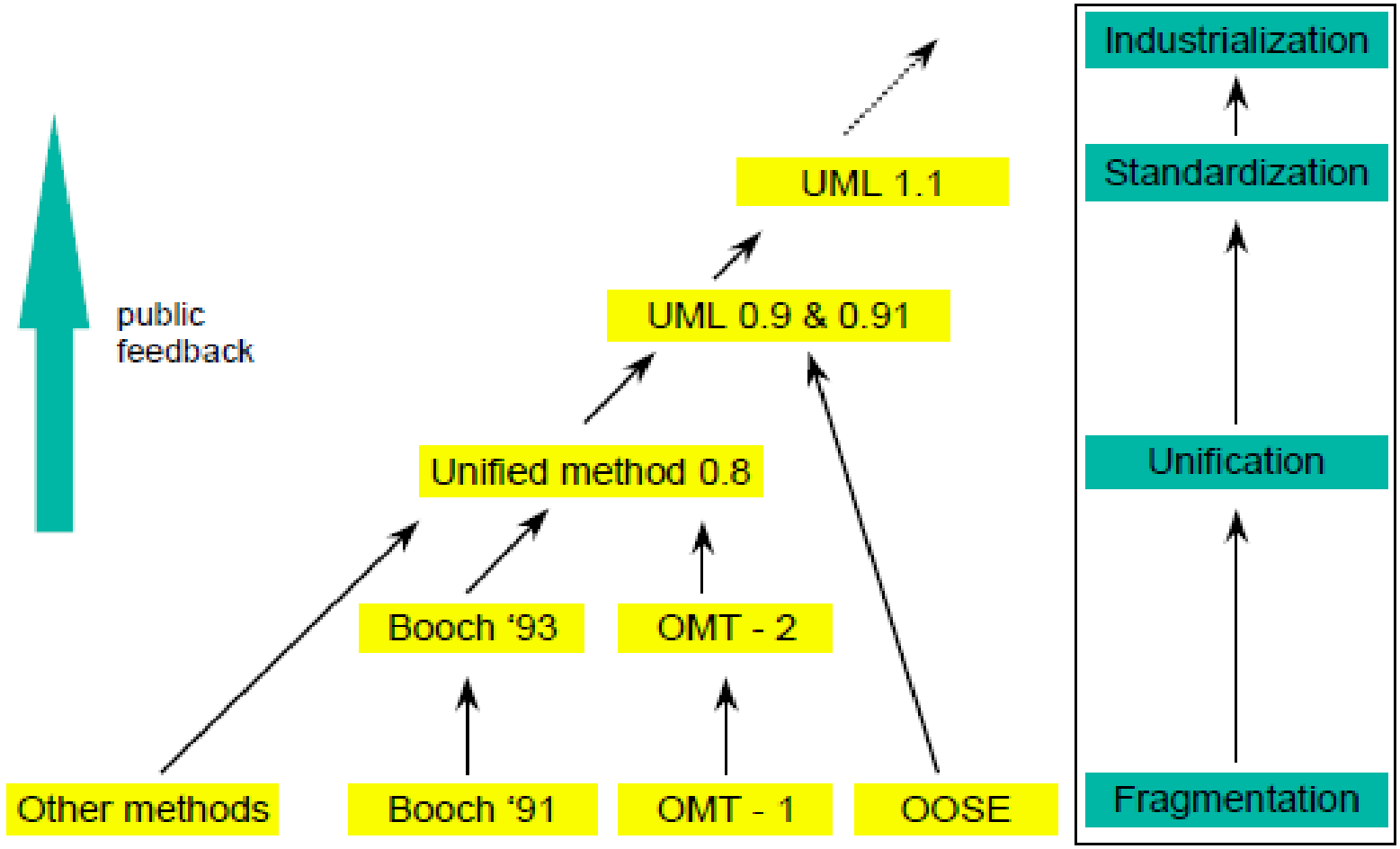


Figure 1: How it all began

The Unified Modeling Language.

History of UML



UML Diagrams

UML includes diagrams for

- use cases
- static structures (class and object diagrams)
- behavior (state-chart, activity, sequence and collaboration diagrams)
- implementation (component and deployment diagrams).

The UML process

There are four kinds of things in the UML.

1. Structural Things.
2. Behavioral Things.
3. Grouping Things.
4. Annotational Things.

Things in UML

Structural Things

1. Class
2. Interface
3. Collaboration
4. Use Case
5. Active Class
6. Components
7. Nodes

Behavioral Things

1. Interaction
2. State Mechanism

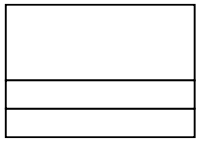
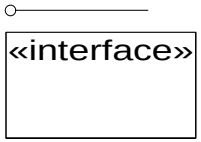
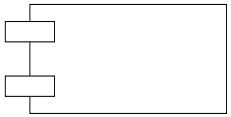
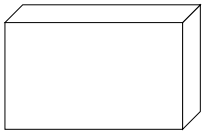
Grouping Things

1. Packages

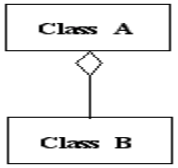
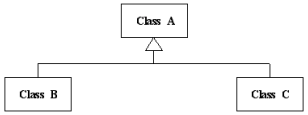
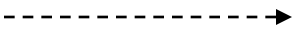
Annotational Things

1. Notes

Structural Modeling: Core Elements

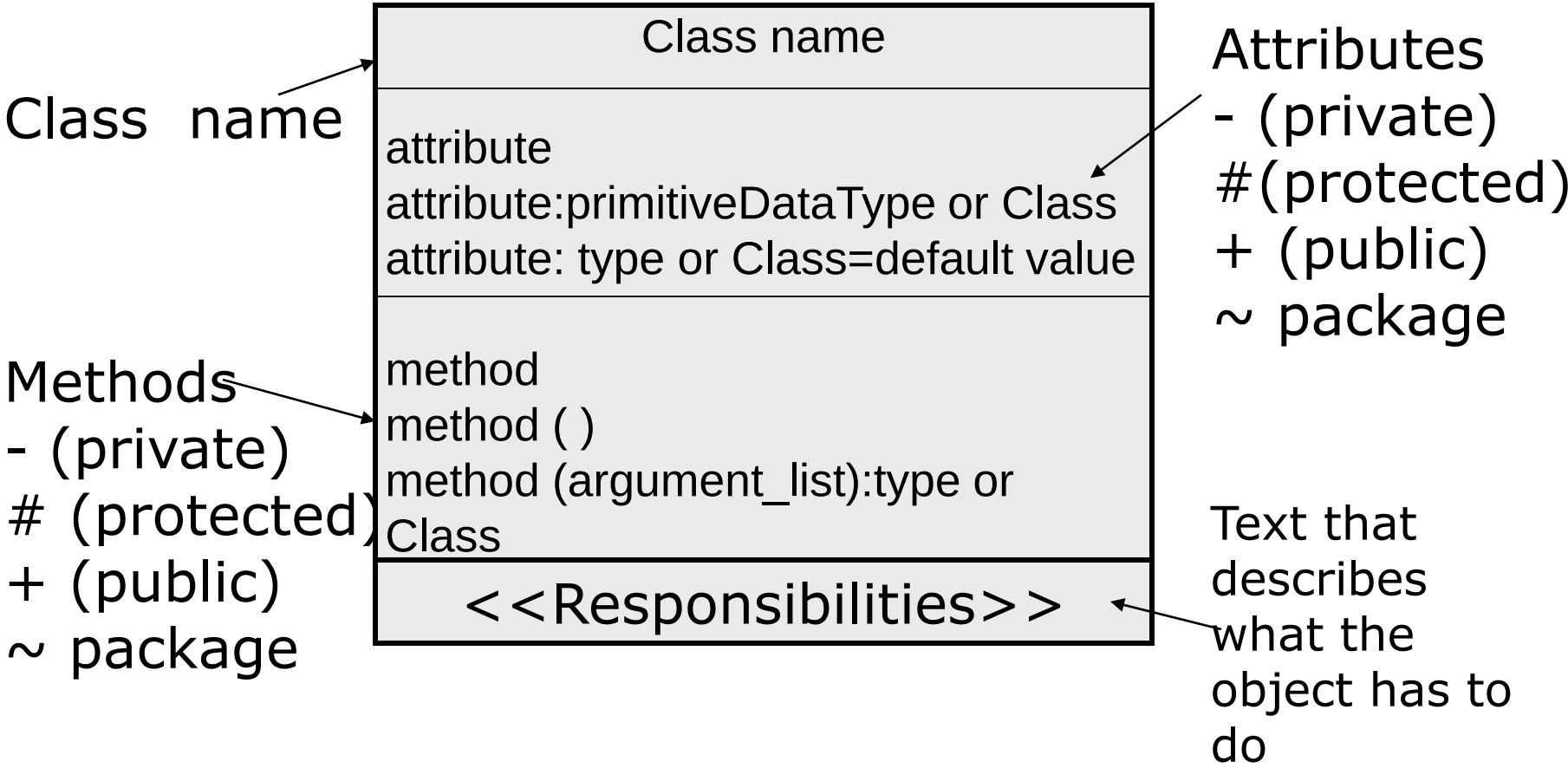
Construct	Description	Syntax
class	a description of a set of objects that share the same attributes, operations, methods, relationships and semantics.	
interface	a named set of operations that characterize the behavior of an element.	
component	a physical, replaceable part of a system that packages implementation and provides the realization of a set of interfaces.	
node	a run-time physical object that represents a computational resource.	

Structural Modeling: Core Relationships

Construct	Description	Syntax
association	a relationship between two or more classifiers that involves connections among their instances.	
aggregation	A special form of association that specifies a whole-part relationship between the aggregate (whole) and the component part.	
generalization	a taxonomic relationship between a more general and a more specific element.	
dependency	a relationship between two modeling elements, in which a change to one modeling element (the independent element) will affect the other modeling element (the dependent element).	

Static Structure: Class

Basic UML notation for classes



Static Structure

Operations / methods

visibility name(parameters) : return_type

visibility:

+ public

protected

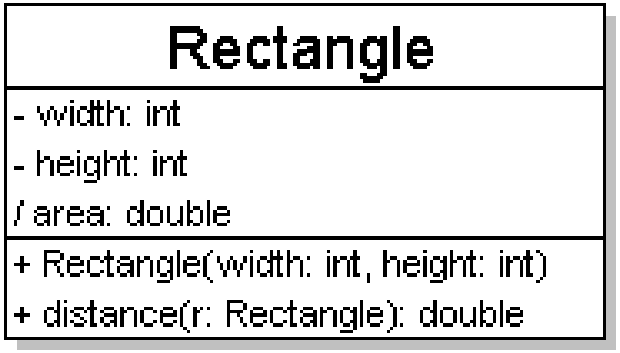
- Private

~ package (default)

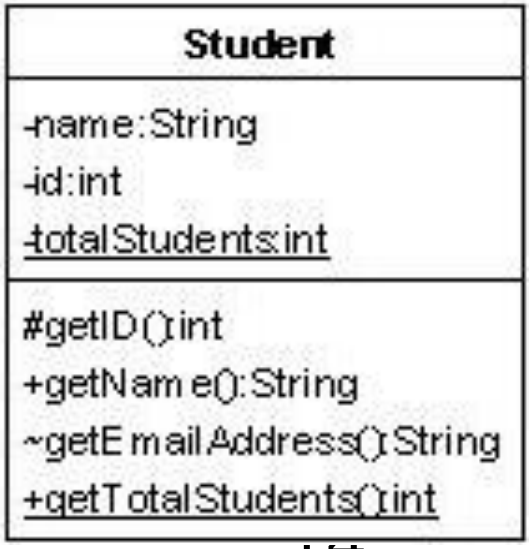
- underline static methods
- parameter types listed as (name: type)
- omit return_type on constructors and when return type is void

Diagram of one class

- class name in top of box
  - write <<interface>> on top of interfaces' names
  - use *italics* for an *abstract class* name



- attributes (optional)
  - should include all fields of the object
  - / Derived



- operations / methods (optional)
  - may omit trivial (get/set) methods
    - but don't omit any methods from an interface!
  - should not include inherited methods

Class Relationships in UML

Class Relationships in UML

- Generalization
- Dependency
- Association

-Role1 -Role2

* 0..1

- realization (interface)
- instanceOf

<<instanceOf>>

Association

- Structural relationship between peer classes (or objects).
- Association can have a name and direction, or be bi-directional
- Role names for each end of the association
- Multiplicity of the relationship

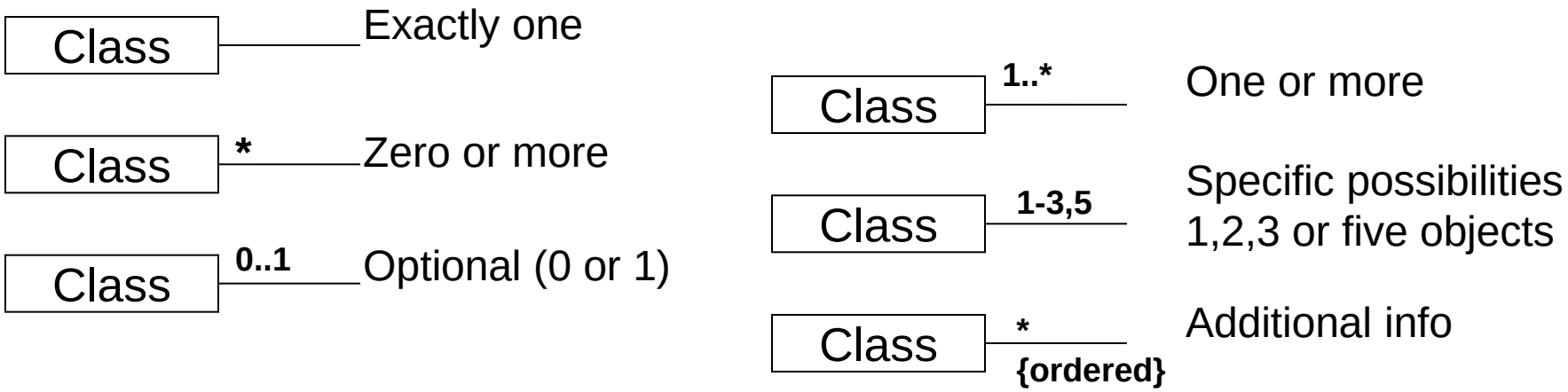
Associations

Associations document relations between two or more classes

Multiplicity of Associations

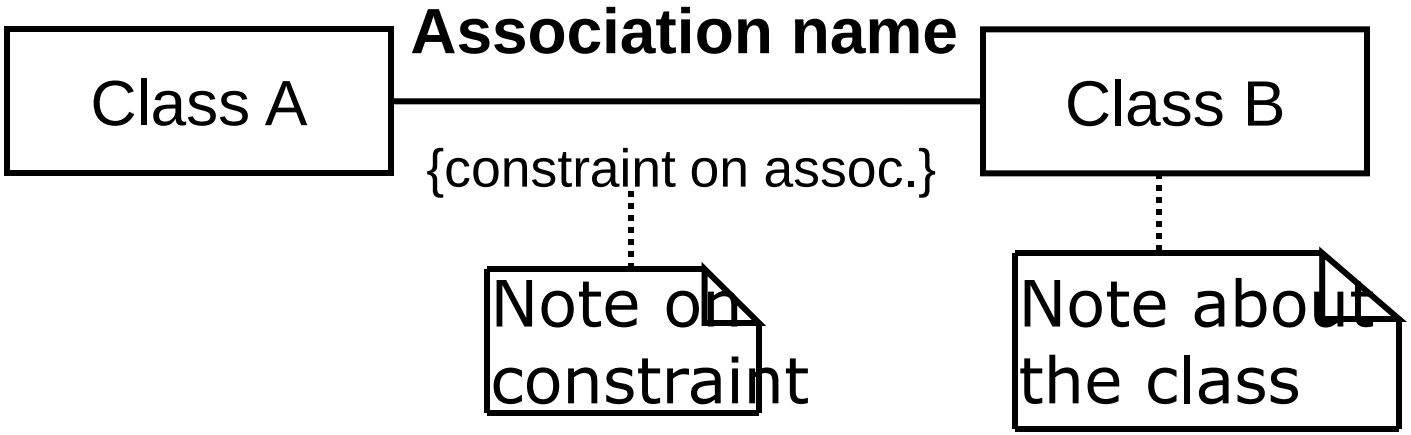
- Multiplicity: refers to how many objects of one class can relate to each object of another class.

  Symbols used to indicate multiplicity in associations:



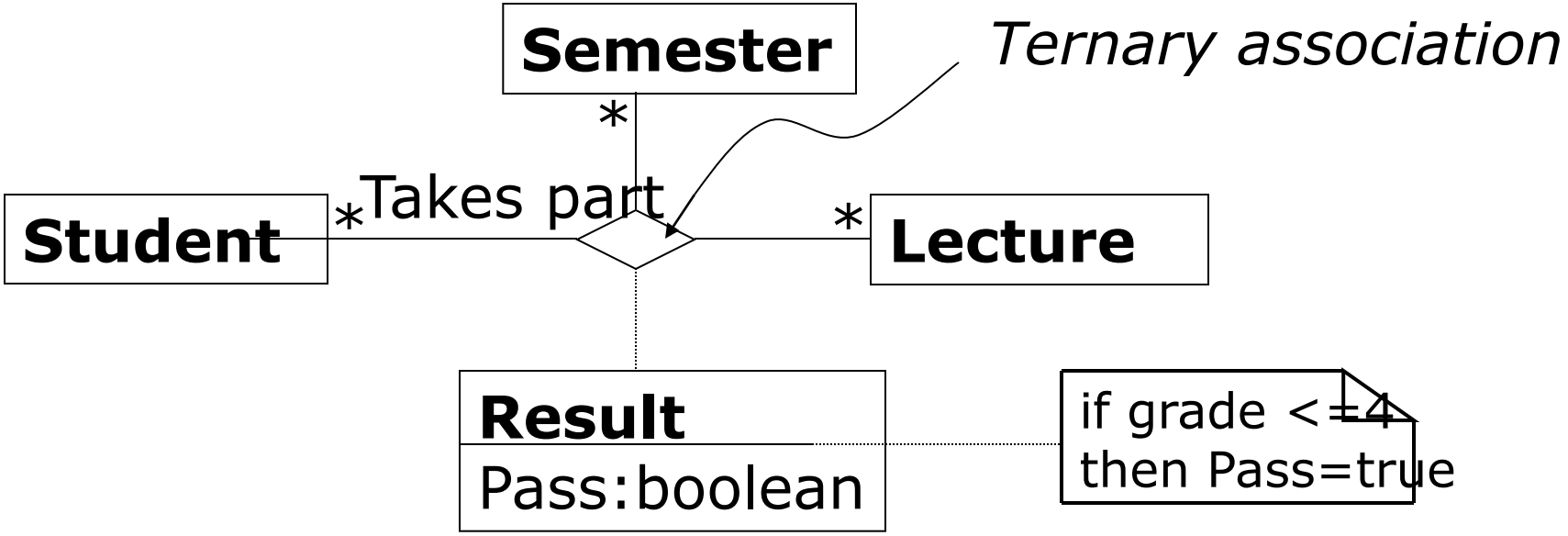
Constraints & Notes

- Constraints are fuctional relationships between elements of a class, for example, limits on the values an attribute can take, limit the number of objects that can exist at any point of time, etc.
- Notes: explanation or more complex constraints



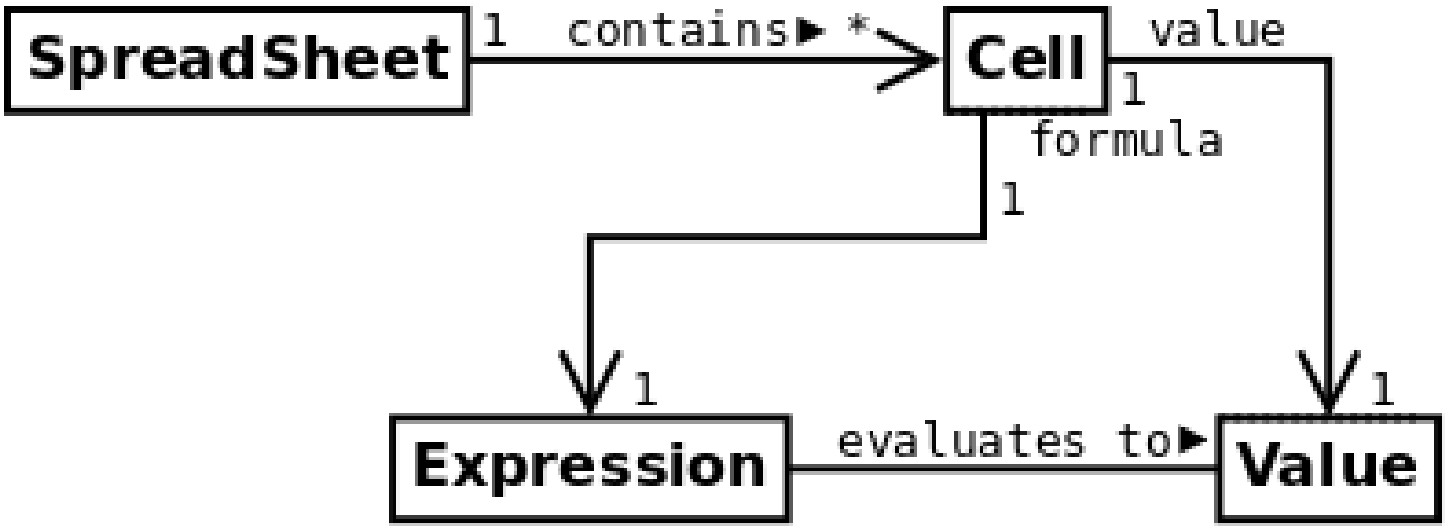
Associations

- Association Class: An association that has the properties of a class.



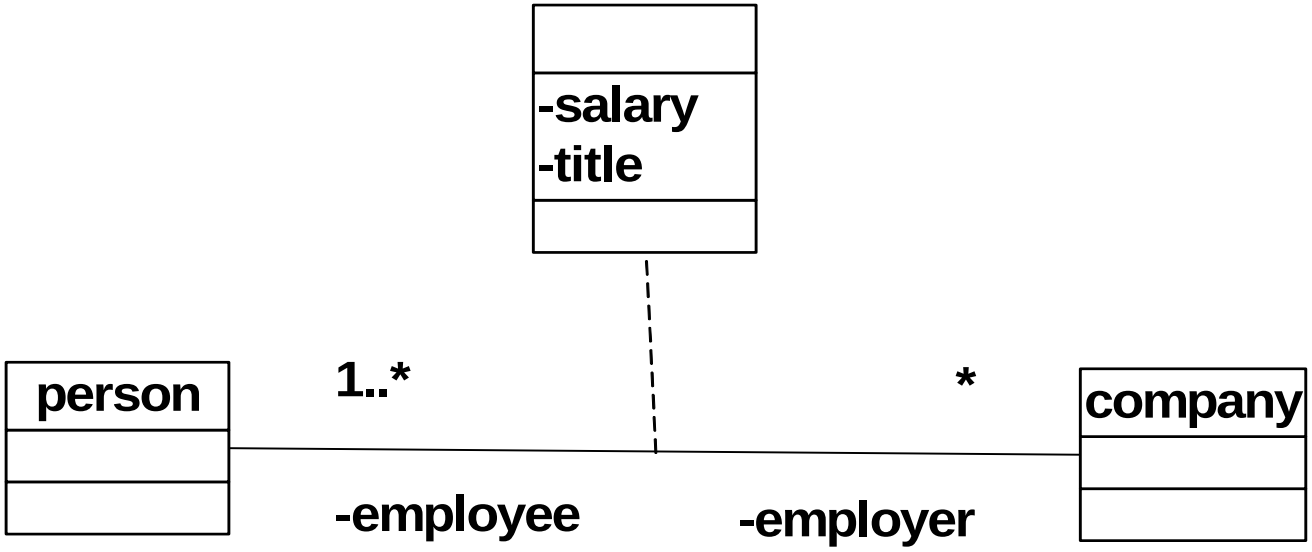
Examples of Association

Association: Decorations



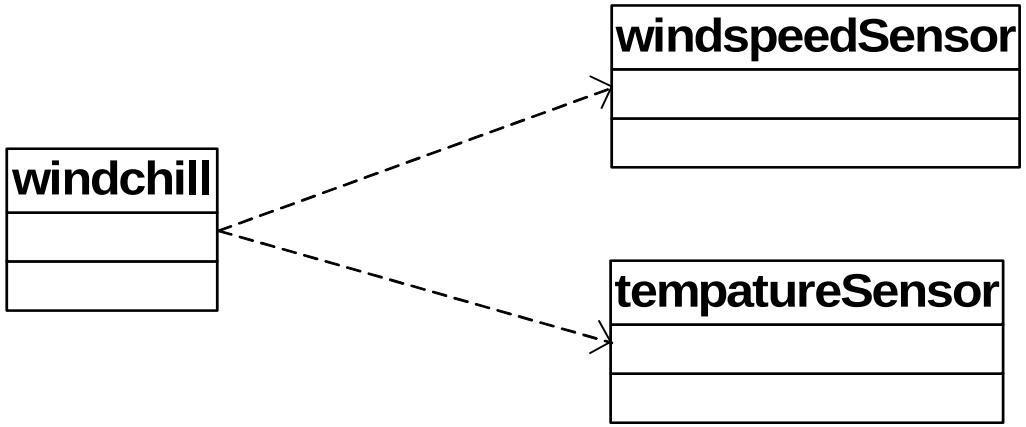
Link Attributes

- Associations may have properties in the same manner as objects/classes.



Dependency

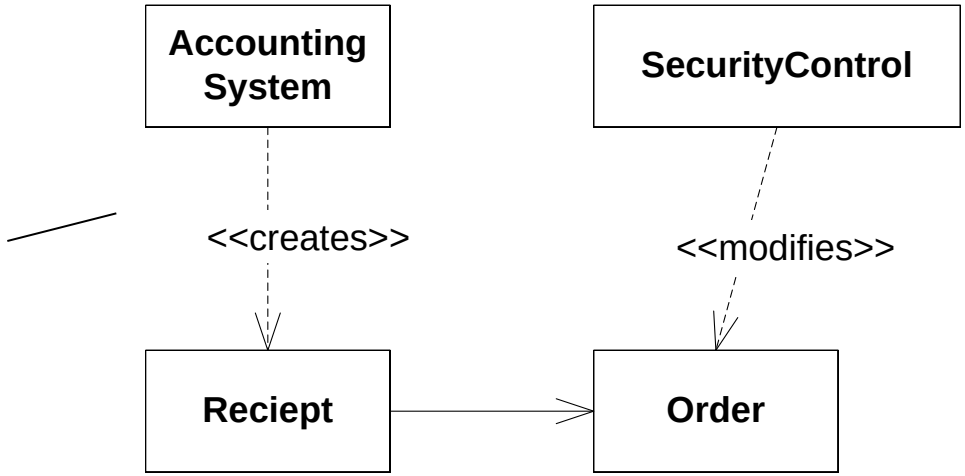
- Change in specification in one class effects another class (but not the other way around)
- Represents a *using* relationship



Dependency

Dependency

- Notated by a dotted line
- The most general relation between classes
- Indicates that an object affects another object



AccountingSystem creates a **Receipt** object

Dependency

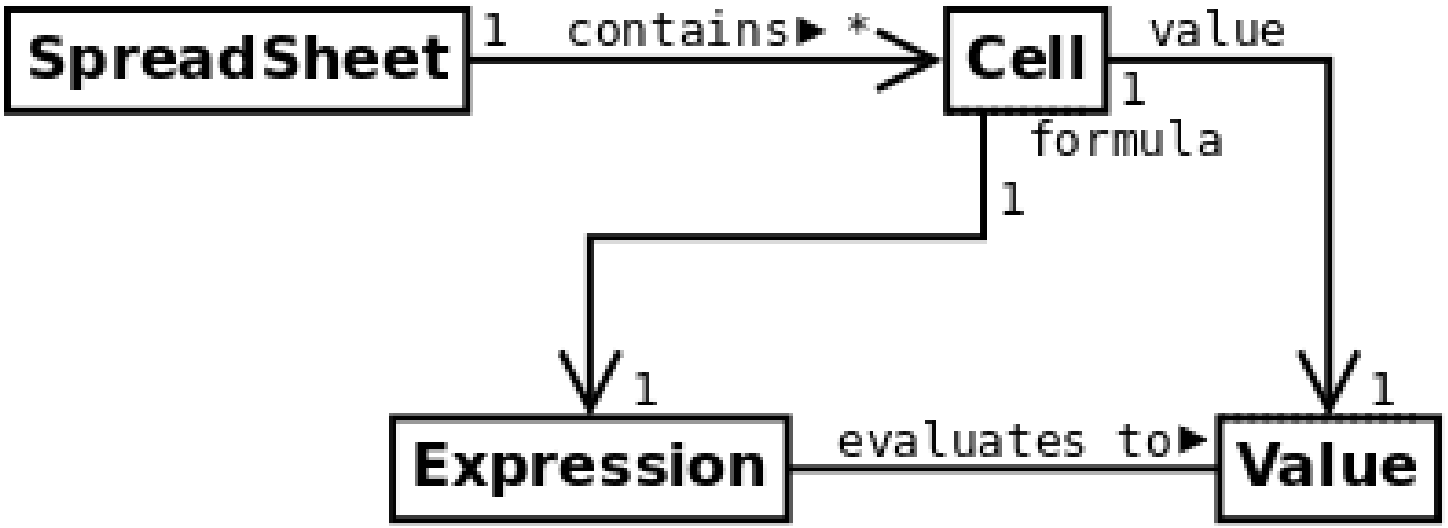
- Properties:
  - Dependencies are always directed
  - Dependencies do not have cardinality.
- If instances of two classes send messages to each other, but are not tied to each other, then dependency is appropriate.
- Types:
  - «call»
  - «create»

Navigability

- Navigability arrows indicate whether, given one instance participating in a relationship, it is possible to determine the instances of the other class that are related to it.

- Given a spreadsheet, we can locate all of the cells that it contains, but that
  - we cannot determine from a cell in what spreadsheet it is contained.
- Given a cell, we can obtain the related expression and value, but
  - given a value (or expression) we cannot find the cell of which those are attributes.

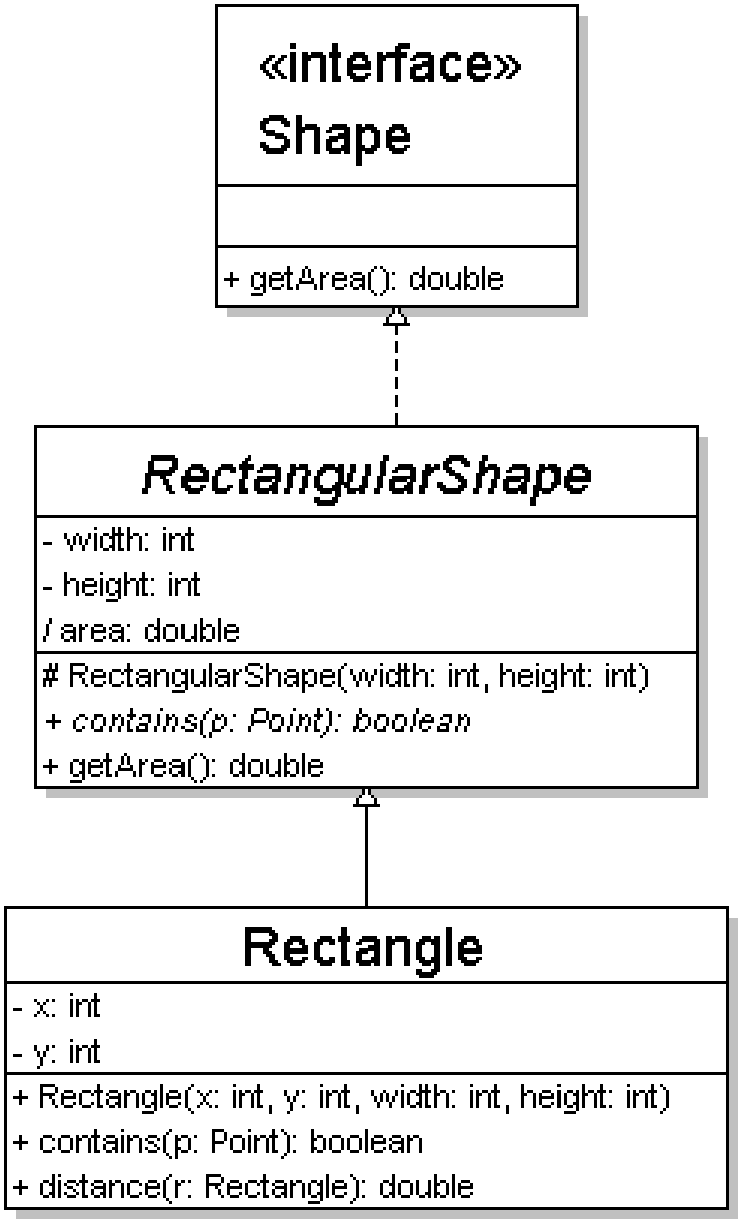
Association: Decorations



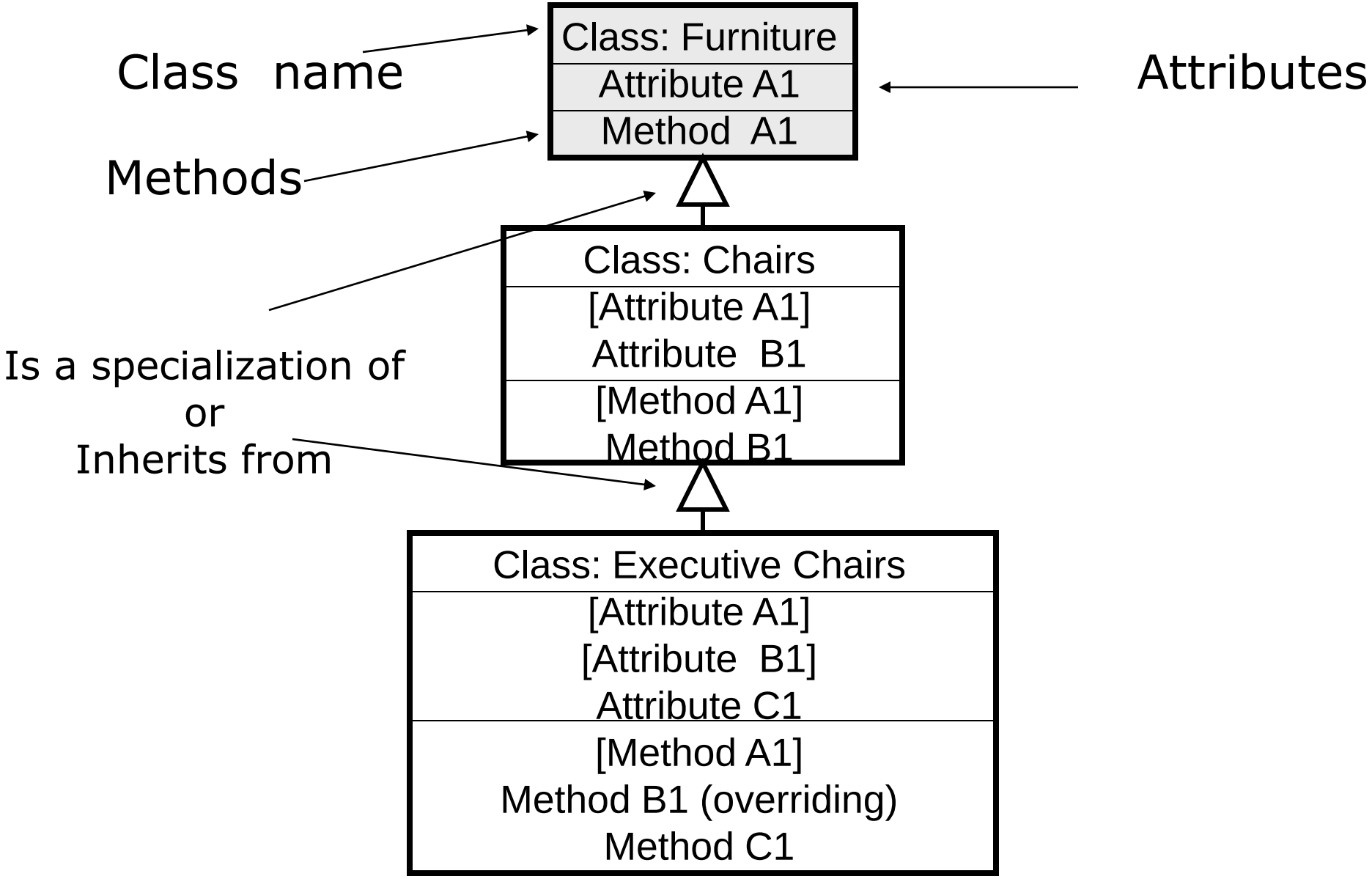
Class Inheritance & Specialization

Generalization relationships

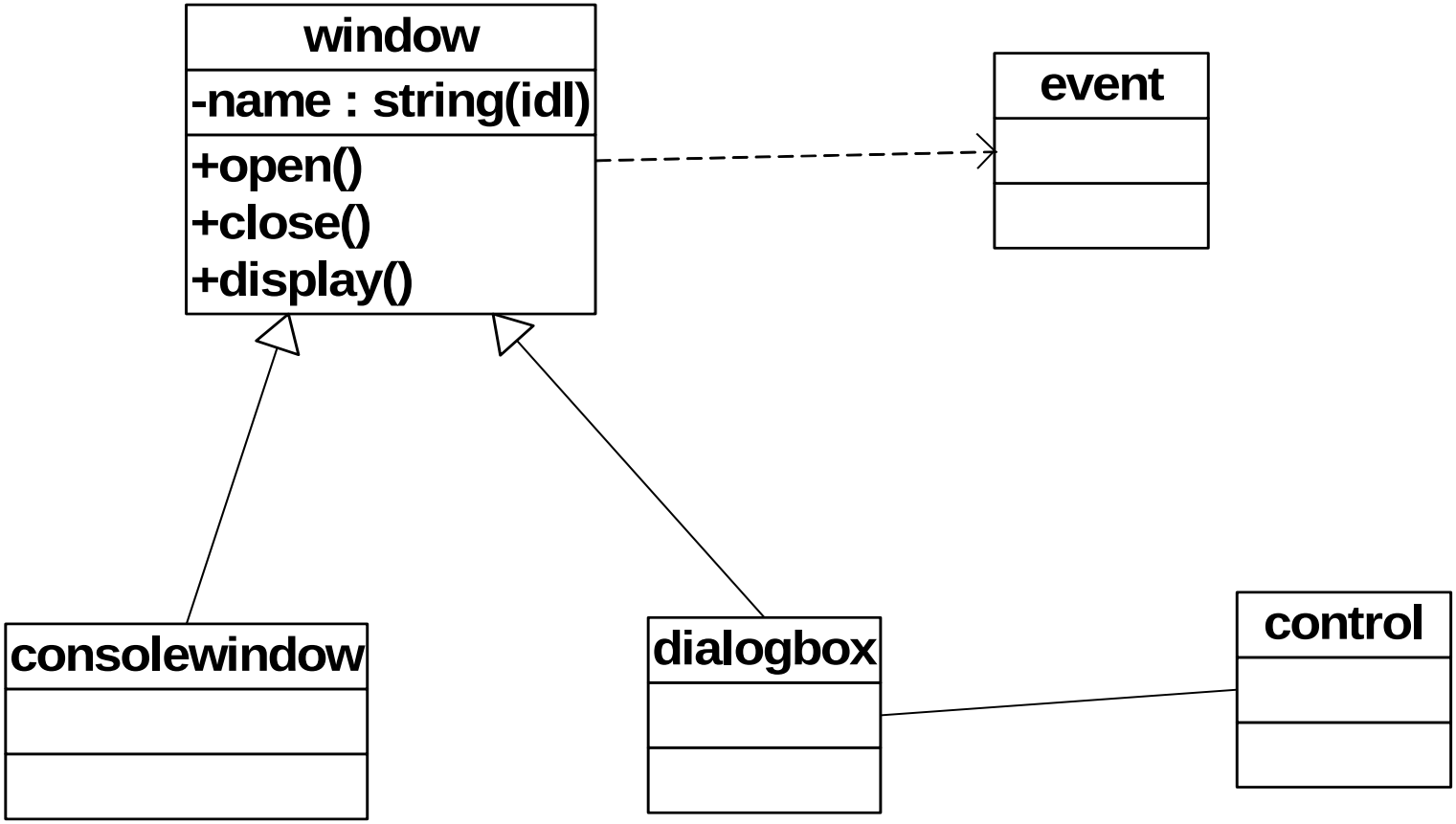
- generalization (inheritance) relationships
  - hierarchies drawn top-down with arrows pointing upward to parent
  - don't draw trivial relationships, such as drawing the Object class as a parent



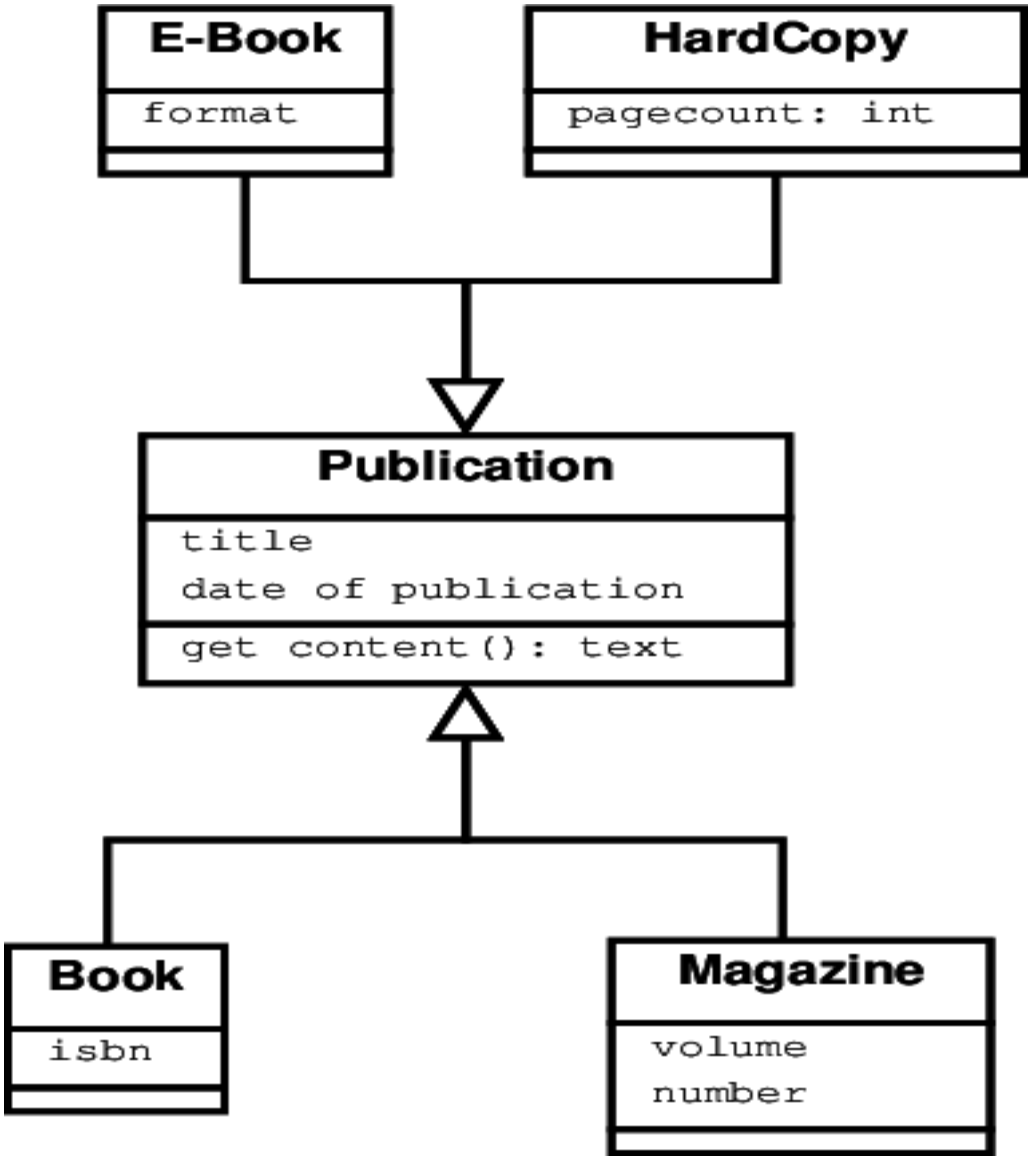
Class Inheritance & Specialization



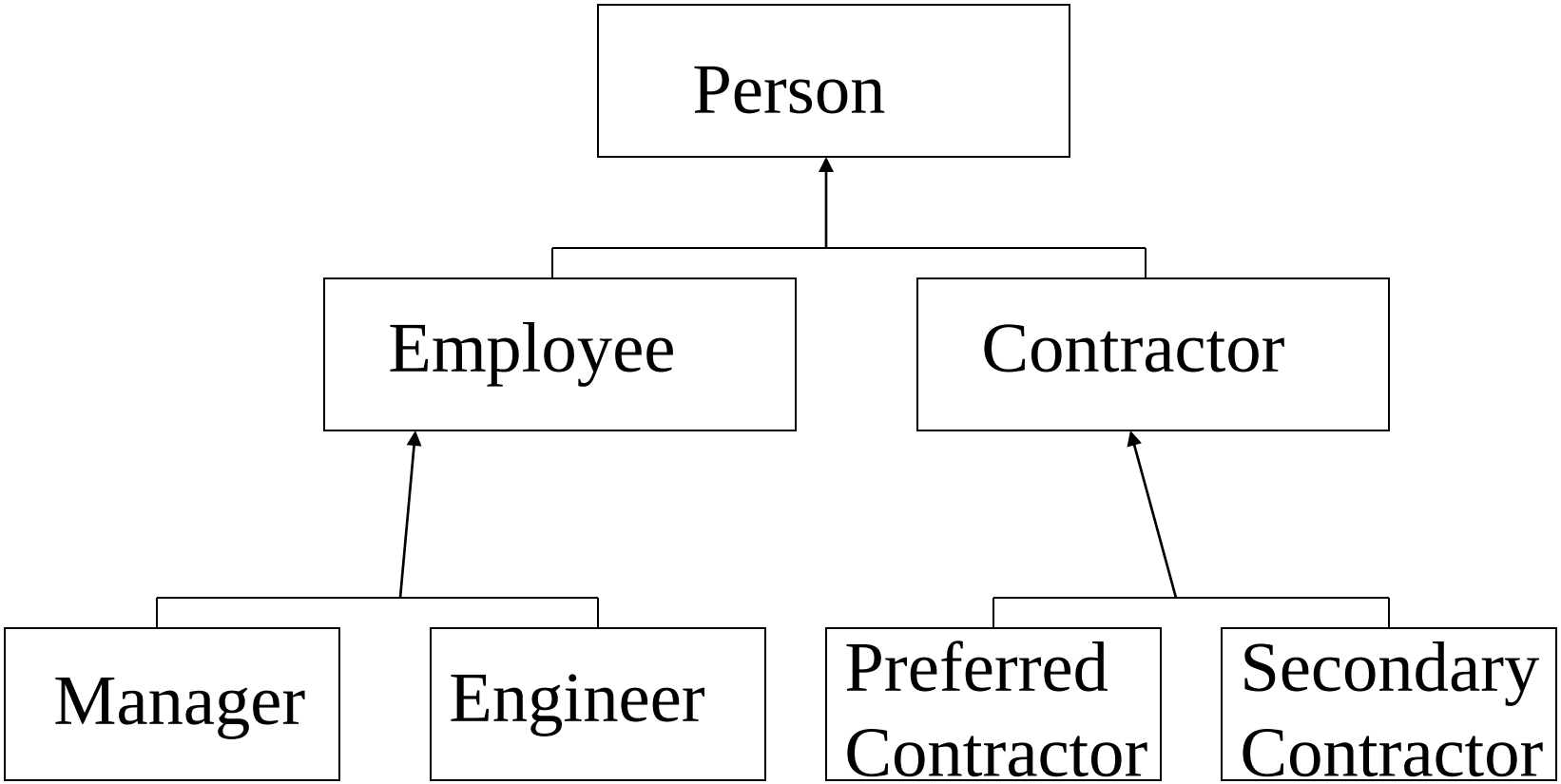
Example class diagram



Multiple Specializations



Generalization Relationship



Aggregation & Composition

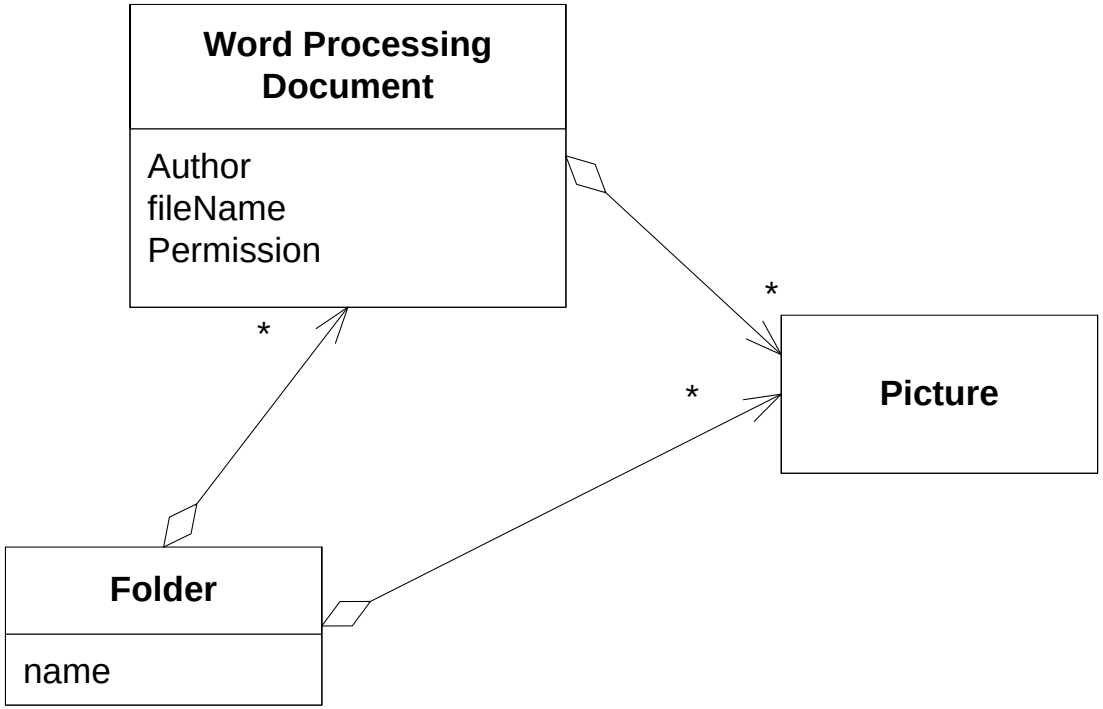
Aggregation & Composition

- Aggregation is a special type of association: a part-whole relationship - a class has an attribute which is an object of another class.

- Composition is a special case of aggregation where
  1. The class is the only one that has objects of that class as attributes
  2. This class is the only one that can create or destroy these objects.

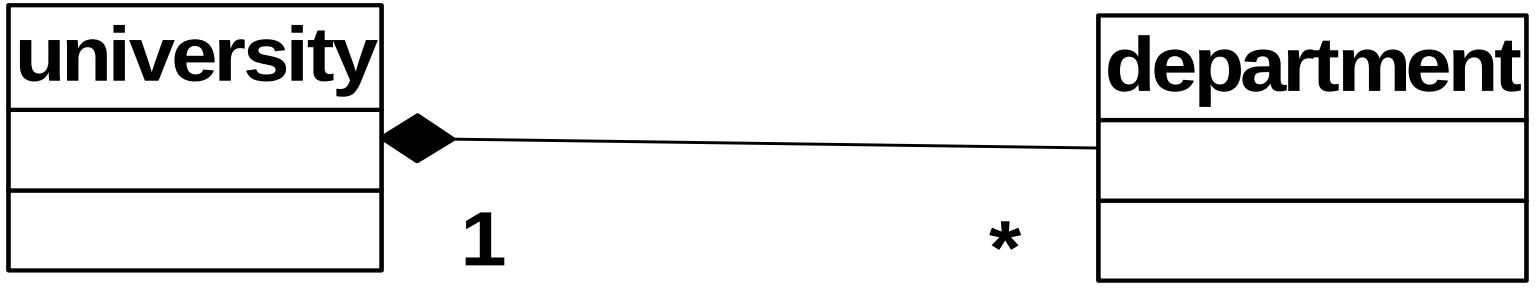
Aggregation

- "Whole-part" relationship between classes
- Assemble a class from other classes

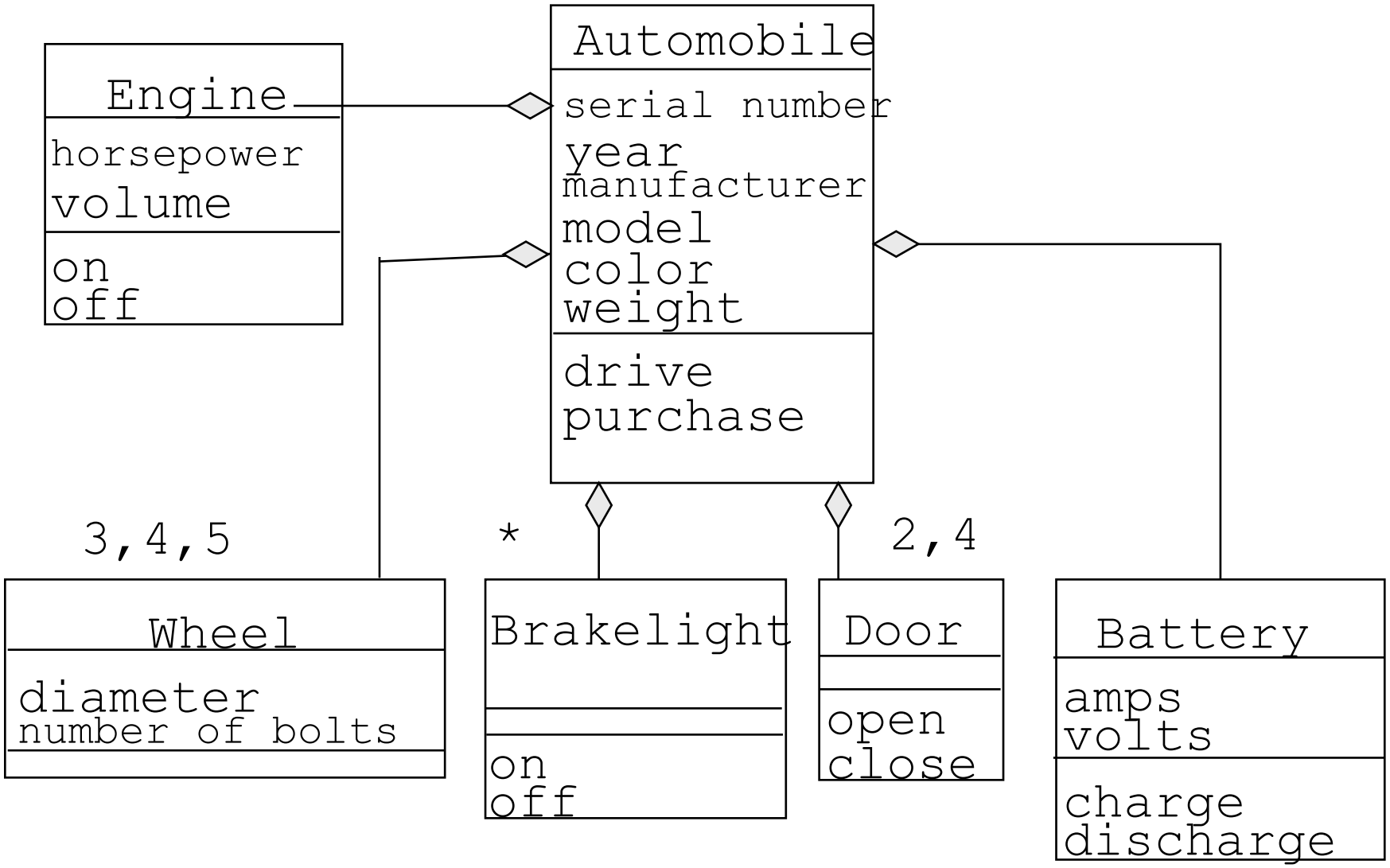


Compostition

- Composition is a special type of association
- *part-of* relationship
- Can use roles and multiplicity

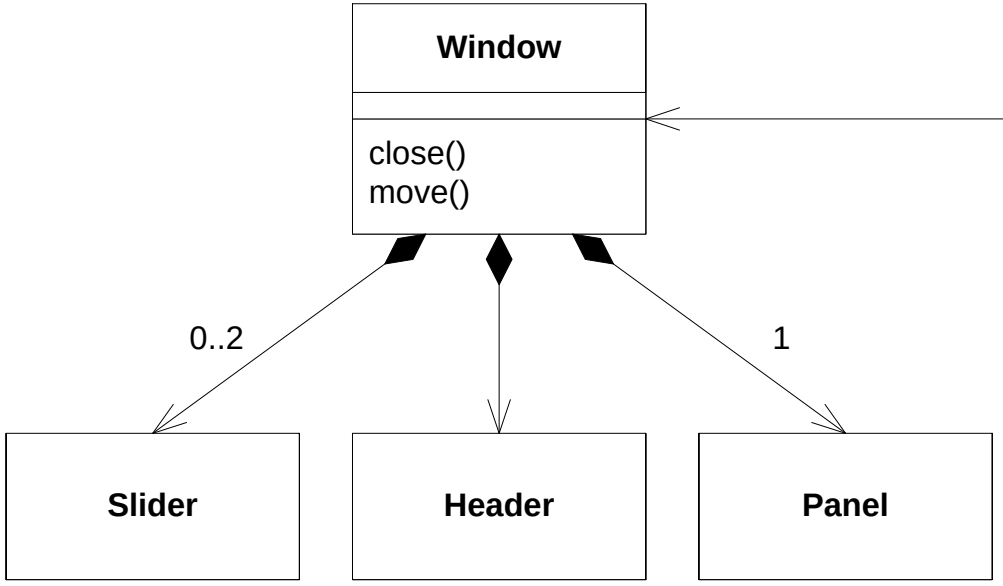


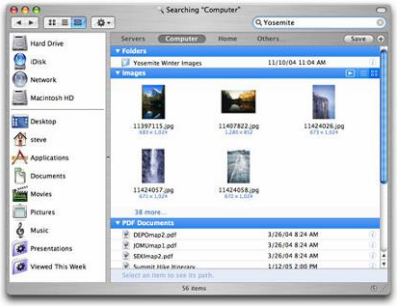
Composition



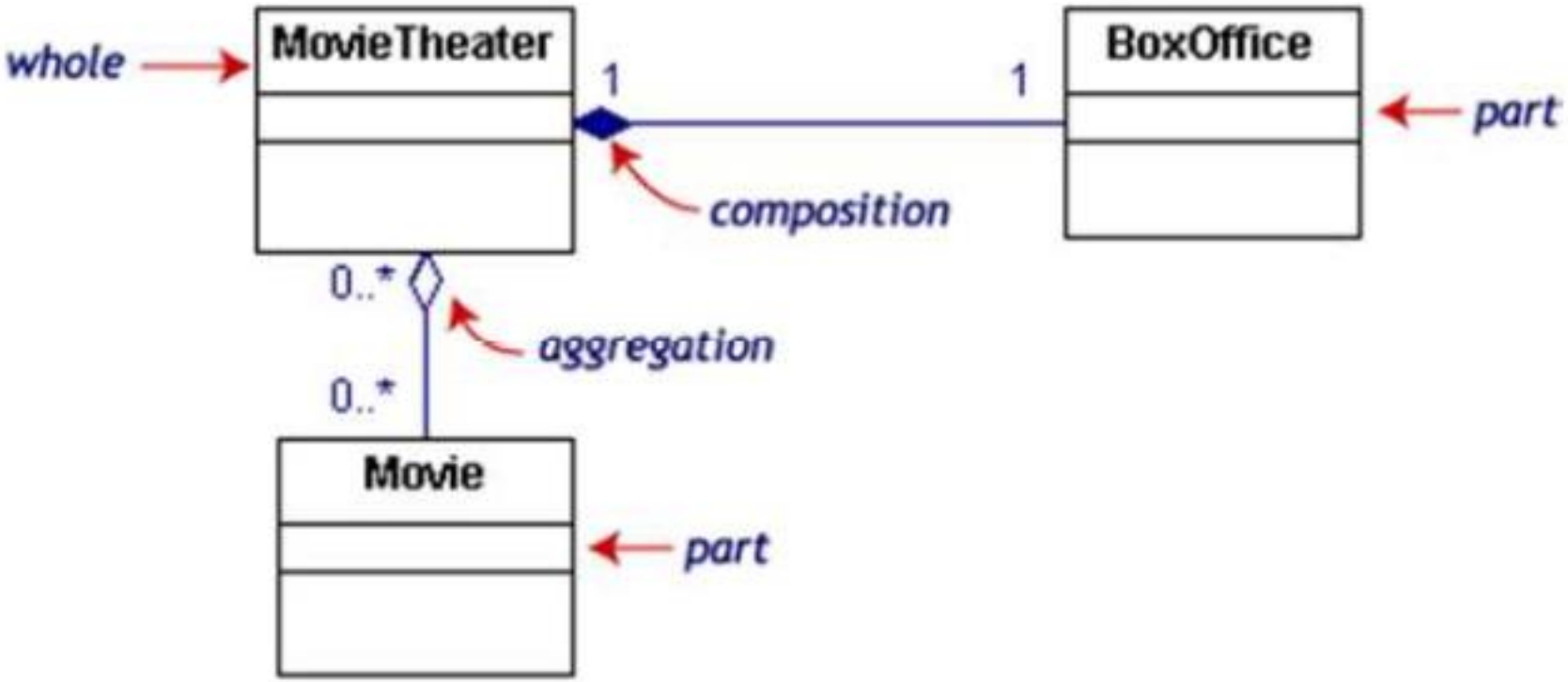
Composition

- Composition is a stronger form of aggregation
- Contained objects that live and die with the container
- Container creates and destroys the contained objects

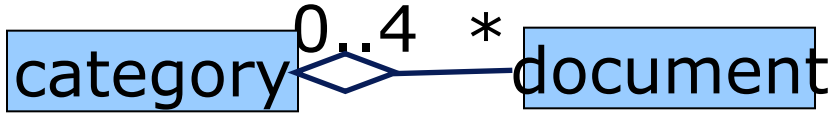




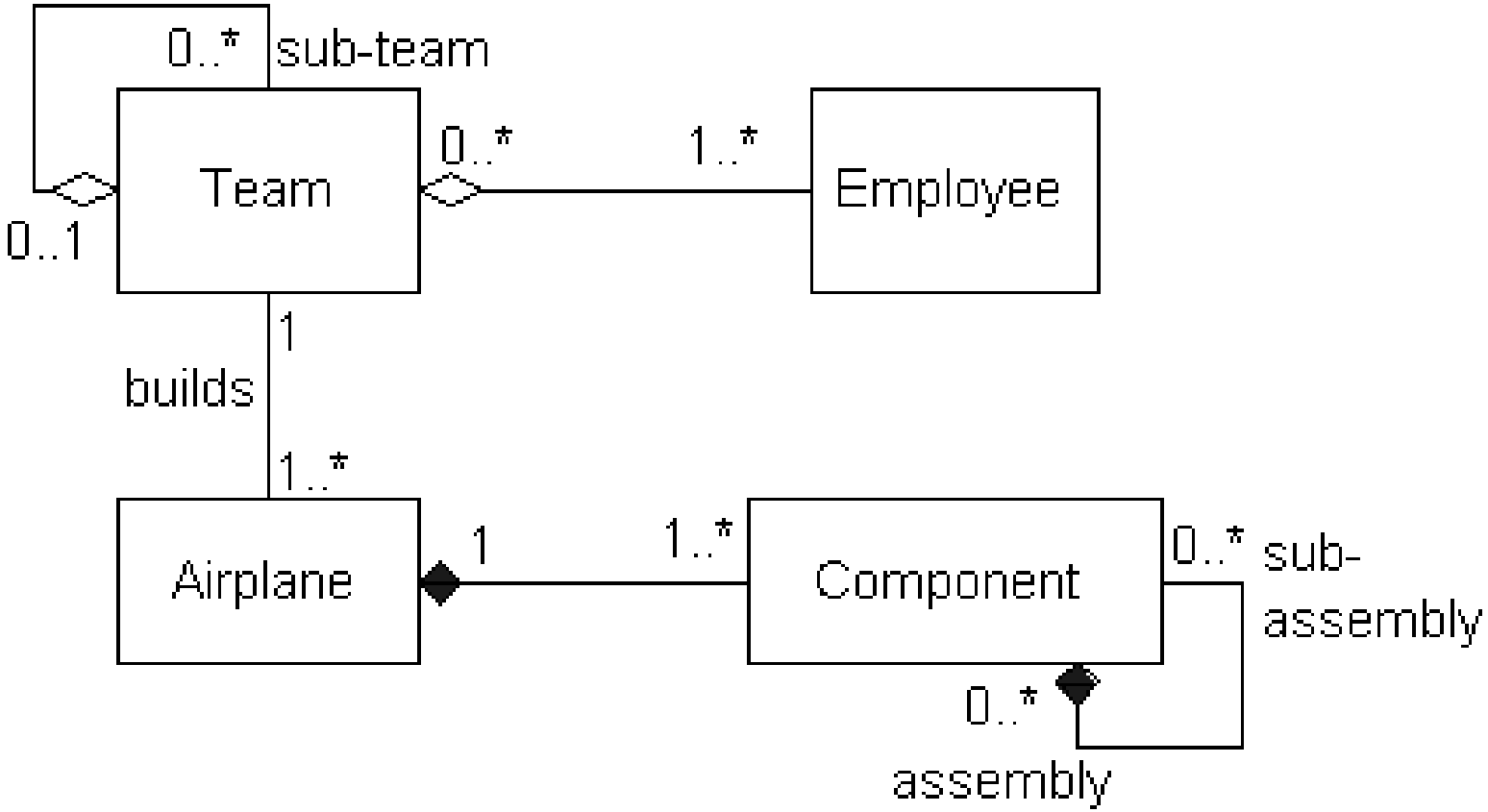
Composition and Aggregation



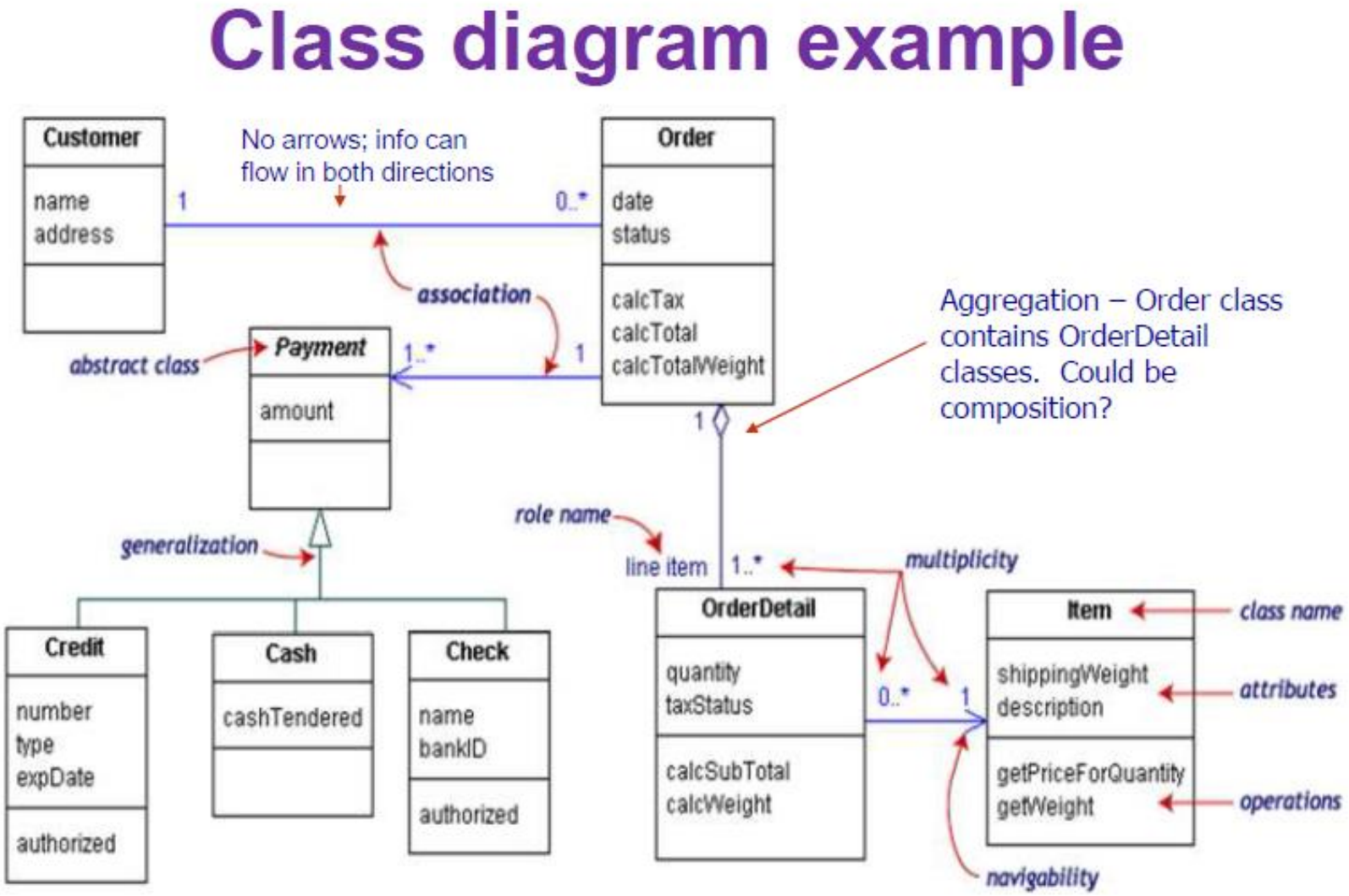
Composition vs. Aggregation

Aggregation	**Composition**
Part can be shared by several wholes 	Part is always a part of a single whole 
Parts can live independently (i.e., whole cardinality can be 0..*)	Parts exist only as part of the whole. When the wall is destroyed, they are destroyed
Whole is not solely responsible for the object	Whole is responsible and should create/destroy the objects

Reflexive associations



Relationships



Notes

A note is a graphical symbol containing text and/or graphics that offer(s) some comment or detail about an element within a model.