Things in the UML

- Structural Things; Behavioral Things; Grouping Things; Annotational Things;
- Structural things are mostly static parts of a mode representing elements that are either conceptual or physical. Often called “Classifiers”;
- Class: a description of a set of objects that share the same attributes, operations, relationships and semantics;

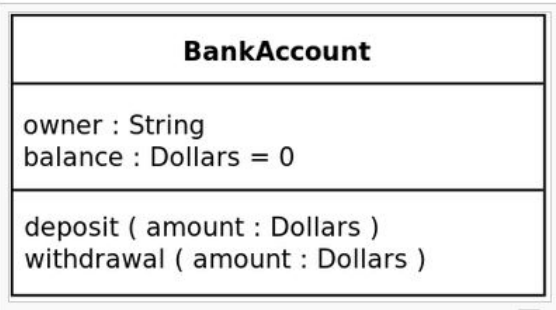
Class Diagram

- A main building block of OO-modeling;
- A type of static structure diagram describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among objects;
- Used for – 1. general conceptual modeling of the systematics of the application, 2. detailed modeling translating the models into programming code, 3. data modeling;
- Represents the main object, interactions in the application and the classes to be programmed;

Class with three sections

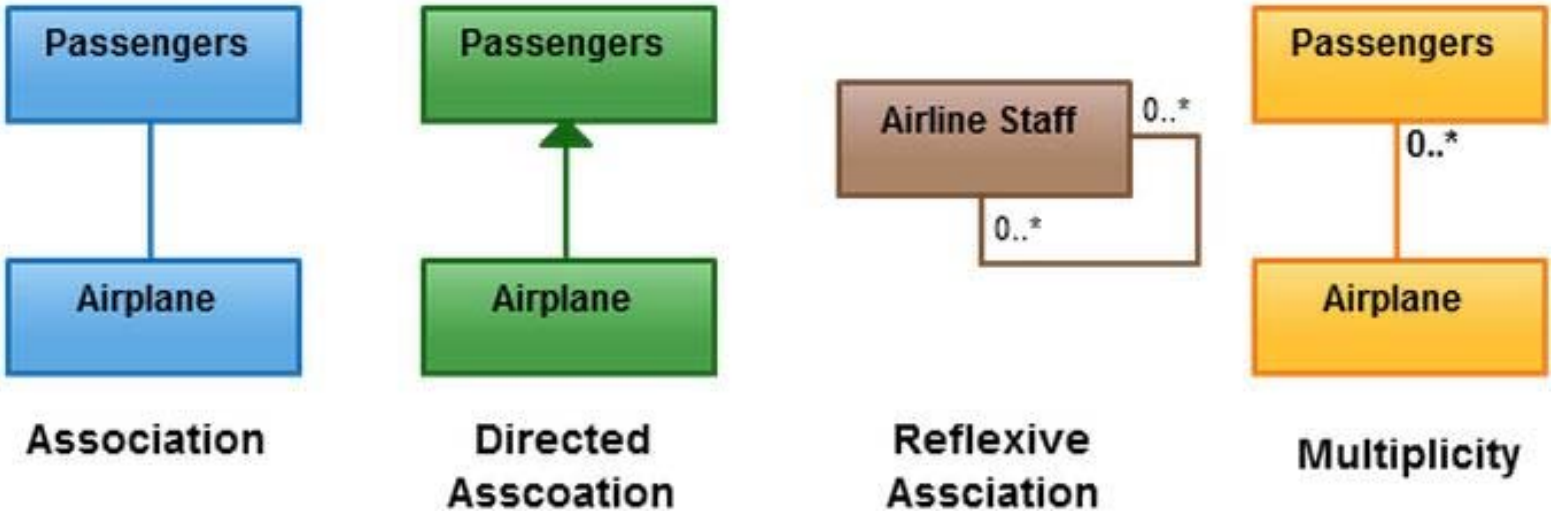
Classes are represented with boxes containing three parts:

- Top part: the name of the class
- Middle part: the attributes of the class
- Bottom part: the methods the class can execute



In the design of a system, a number of classes are identified and grouped together in a class diagram that helps to determine the static relations between those objects.

Relationships in Class Diagrams



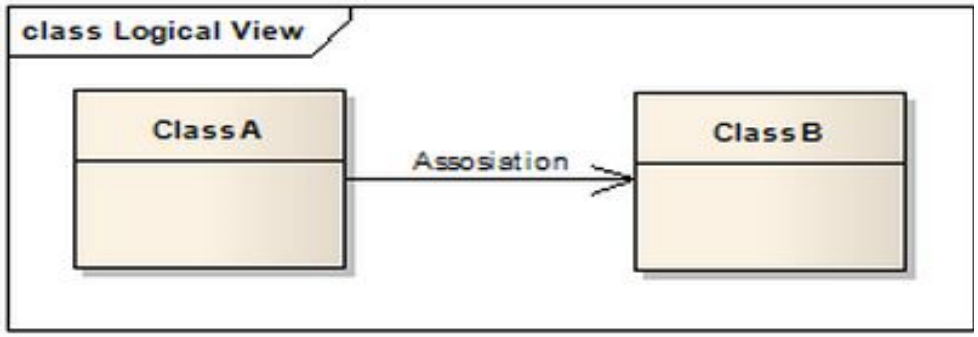
Association: broad term for any logical connection or relationship between classes;

Direct association: direct relationship represented by a line with an arrowhead;

Reflexive association: when a class may have multiple functions or responsibilities;

Multiplicity: active logical association when the cardinality of a class in relation to another is being depicted;

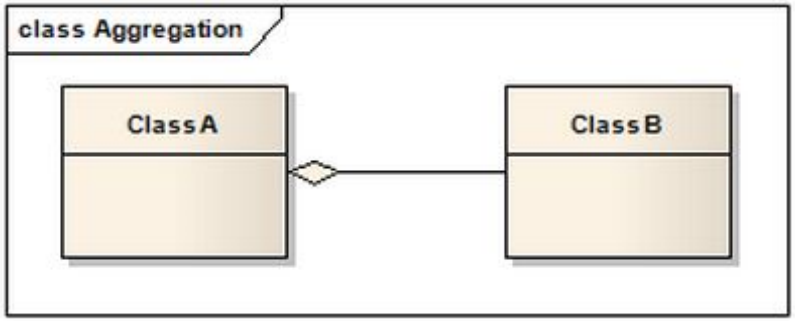
Association



Professor *"playing the role" of* ***author*** *is associated with* ***textbook*** *end typed as* ***Book***.

- The most abstract way to describe static relationship between classes is using the “association” link;
- States that there is some kind of a link or a dependency between two classes or more;

Aggregation (Shared Association)



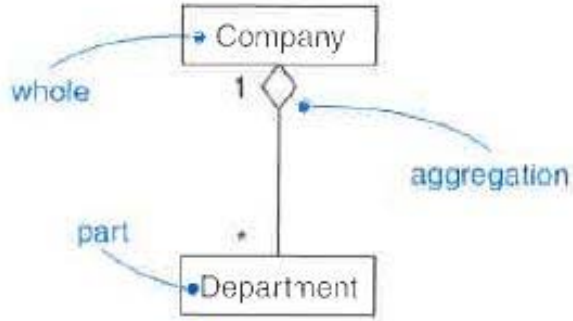
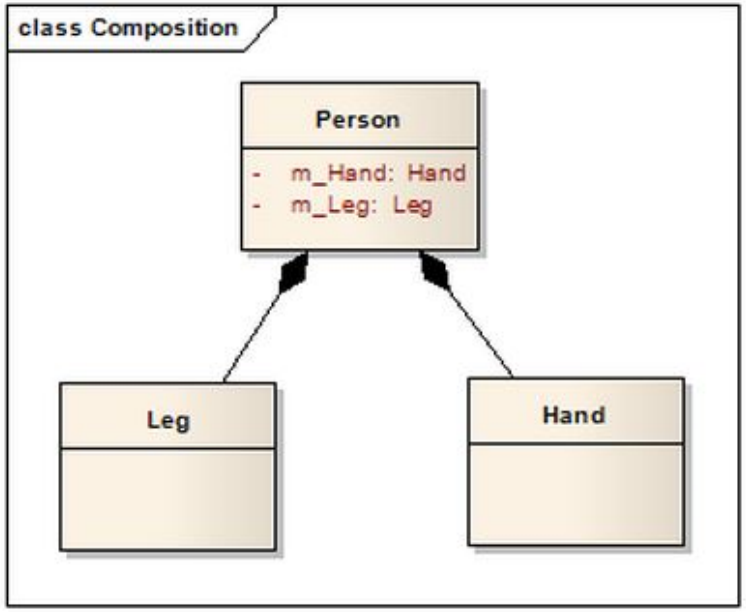


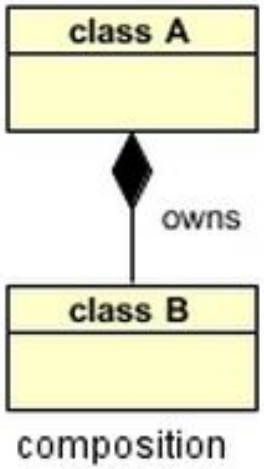
Figure 5-7: Aggregation

- "whole/part" relationship, in which one class represents a larger thing (the "whole") consisting of smaller things (the "parts") ➔ Class A (whole) "has" a Class B (part) relationship, i.e., an object of the whole has objects of the part;

- Class B (department) can be aggregated by other classes (company) in the application, aka. Shared association.

Composition





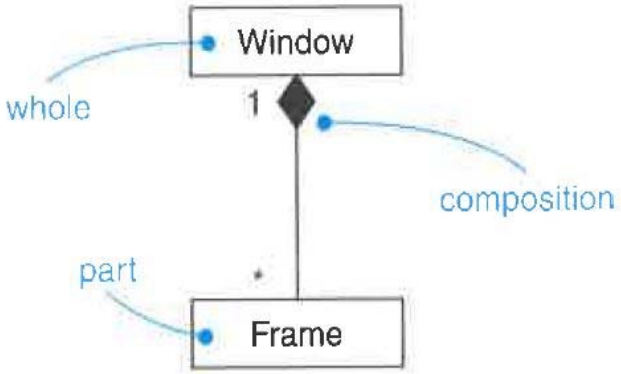


Figure 10-6: Composition

- Form of aggregation with strong ownership and there is a strong life cycle dependency between the two:

- Class A owns a Class B relationship;
- Class A is deleted then Class B is also deleted as a result;
- The whole is responsible for the disposition of its parts, i.e., the composite must manage the creation and destruction of its parts;

Aggregation vs Composition

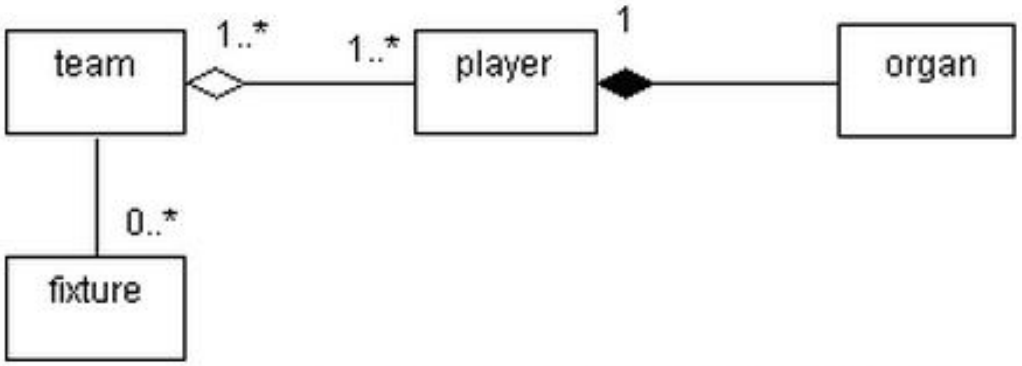


Figure 1. association, aggregation and composition.

- A team being aggregate of its players.
  - A village of its inhabitants, a company of its workforce, a society and its membership.
- A person being compose of heart, brain, etc.
  - A stadium and its pitch & seats, a wall and the bricks from which it is built.

Generalization/Inheritance

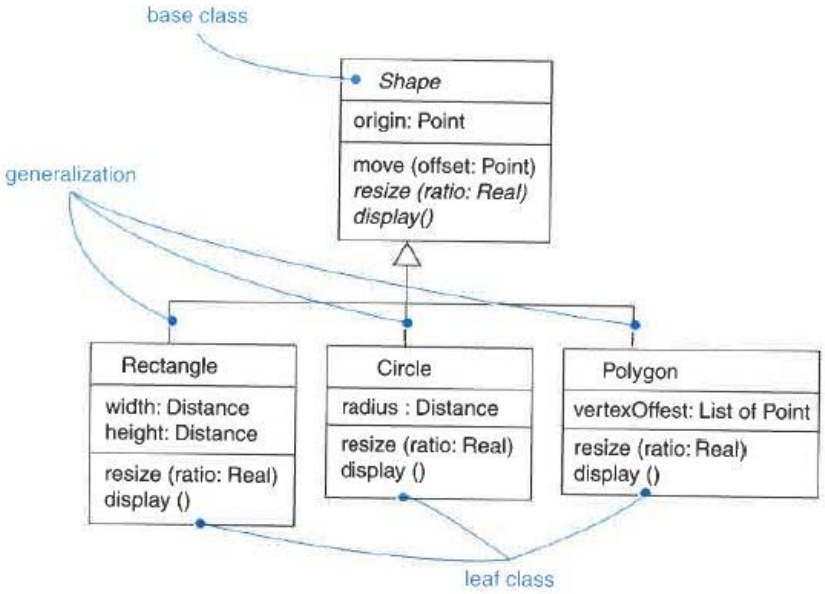
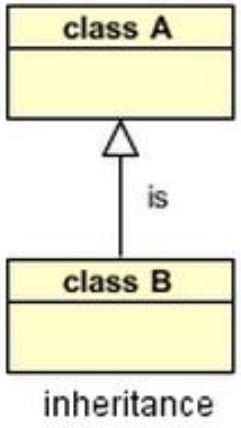
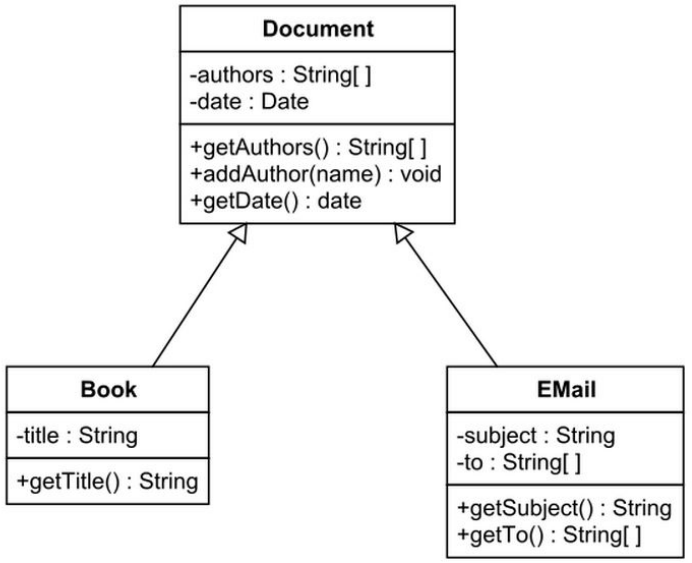


Figure 5-3: Generalization



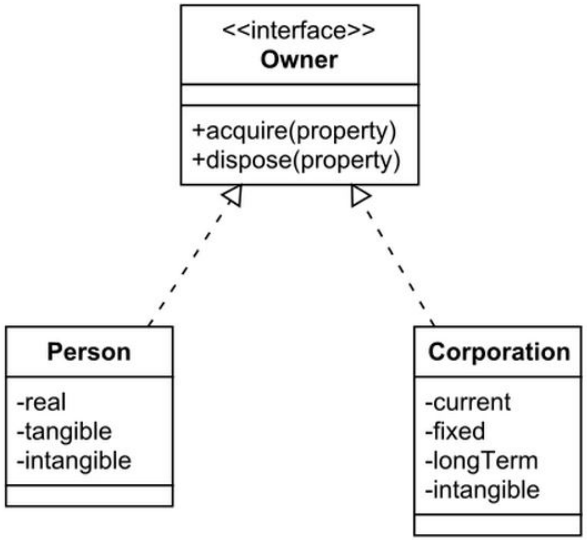


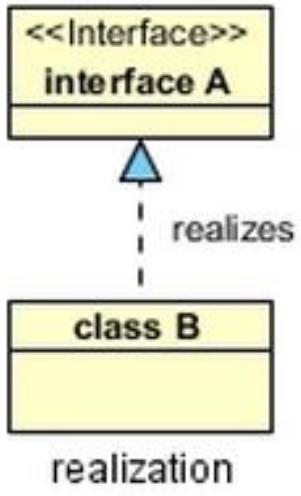
Generalization/Inheritance: a relationship between a general kind of things (superclass/parent) and a more specific kind of things (subclass/child):

- Class B is a Class A (or Class A is extended by Class B);

- Book extends the document class, which might include the Email class. Book and Email classes inherit the variables and methods of the Document class (or modifying the methods), but might add additional variable and methods;

Realization





- The implementation of the functionality defined in one class by another class;
- A class implements an interface; Class ‘B’ realizes a Class ‘A’ (A is realized by B);
- A semantic relationship between classifiers in which one specifies a contract that another guarantees to carry out; a dashed directed line with an open arrowhead pointing to the classifier that specifies the contract;
- Ex: the owner interface specifies method for acquiring and disposing of property. The Person and Corporation classes need to implement these methods, possibly in very different ways;