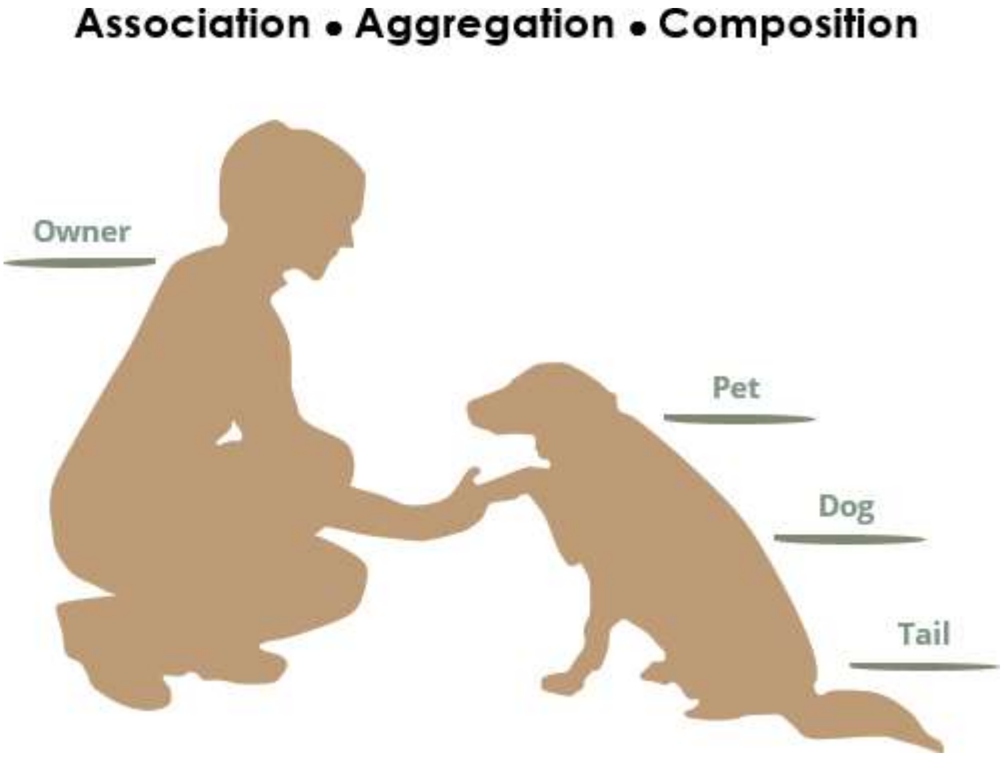
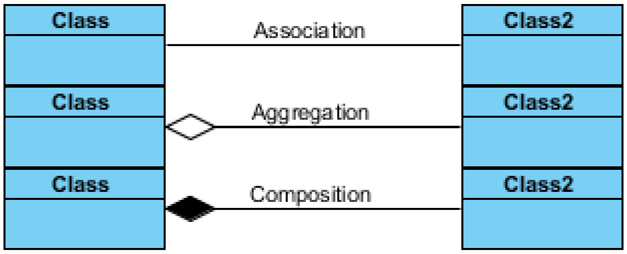
Consider the differences and similarities between the classes of the following objects: **pets**, **dogs**, **tails**, **owners**.



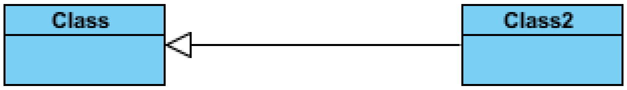
We see the following relationships:

- owners feed pets, pets please owners (**association**)
- a tail is a part of both dogs and cats (**aggregation** / **composition**)
- a cat is a kind of pet (**inheritance** / **generalization**)

The figure below shows the three types of association connectors: association, aggregation and composition. We will go-over them in this UML guide.



The figure below shows a generalization. We will talk about it later on in this UML guide.



Learn UML with Free UML Tool

Are you looking for a Free UML tool for learning UML faster, easier and quicker? Visual Paradigm Community Edition is a UML software that supports all UML diagram types. It is an international award-winning UML modeler, and yet it is easy-to-use, intuitive & completely free.

Free Download

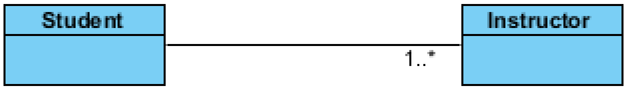
Association

If two classes in a model need to communicate with each other, there must be link between them, and that can be represented by an association (connector).

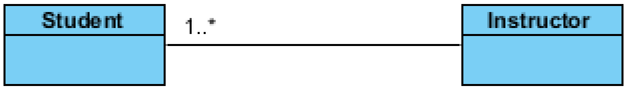
Association can be represented by a line between these classes **with an arrow indicating the navigation direction**. In case arrow is on the both sides, association has bidirectional association.

We can indicate the multiplicity of an association by adding multiplicity adornments to the line denoting the association. The example indicates that a Student has one or more Instructors:

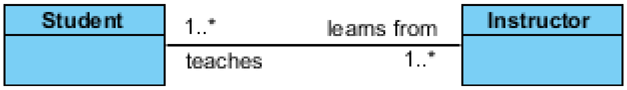
A single student can associate with multiple teachers:



The example indicates that every Instructor has one or more Students:



We can also indicate the behavior of an object in an association (i.e., the role of an object) using role names.



Association vs Aggregation vs Composition

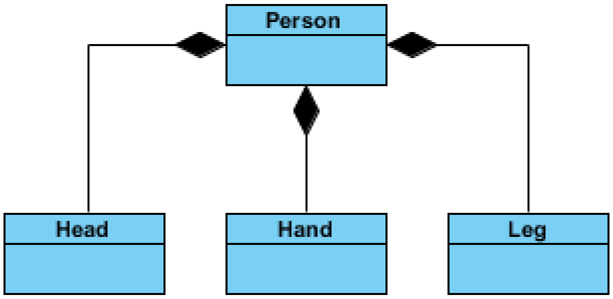
The question "What is the difference between association, aggregation and composition" has been frequently asked lately.

Actually, **Aggregation** and **Composition** are subsets of association meaning they are **specific cases of association**. In both aggregation and composition object of one class "owns" object of another class. But there is a subtle difference:

- **Aggregation** implies a relationship where the child can exist independently of the parent. Example: Class (parent) and Student (child). Delete the Class and the Students still exist.
- **Composition** implies a relationship where the child cannot exist independent of the parent. Example: House (parent) and Room (child). Rooms don't exist separate to a House.

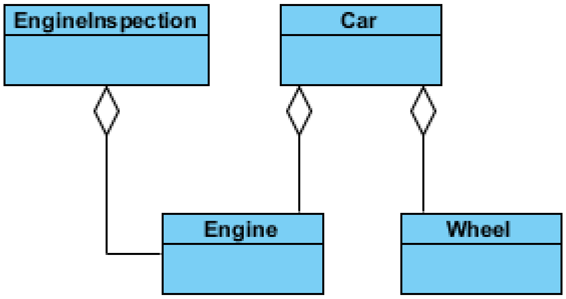
Composition Example:

We should be more specific and use the composition link in cases where in addition to the part-of relationship between Class A and Class B - there's a strong lifecycle dependency between the two, meaning that when Class A is deleted then Class B is also deleted as a result



Aggregation Example:

It's important to note that the **aggregation link doesn't state in any way that Class A owns Class B nor that there's a parent-child relationship** (when parent deleted all its child's are being deleted as a result) between the two. Actually, quite the opposite! The aggregation link is usually used to stress the point that Class A instance is not the exclusive container of Class B instance, as in fact the same Class B instance has another container/s.



Summing it up -

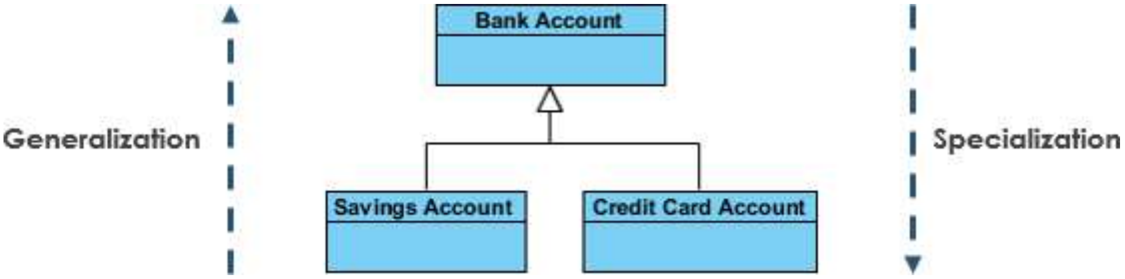
To sum it up association is a very generic term used to represent when on class used the functionalities provided by another class. We say it's a composition if one parent class object owns another child class object and that child class object cannot meaningfully exist without the parent class object. If it can then it is called Aggregation.

Generalization vs Specialization

Generalization is a mechanism for combining similar classes of objects into a single, more general class. Generalization identifies commonalities among a set of entities. The commonality may be of attributes, behavior, or both. In other words, a superclass has the most general attributes, operations, and relationships that may be shared with subclasses. A subclass may have more specialized attributes and operations.

Specialization is the reverse process of Generalization means creating new sub classes from an existing class.

For Example, a Bank Account is of two types - Savings Account and Credit Card Account. Savings Account and Credit Card Account inherit the common/ generalized properties like Account Number, Account Balance etc. from a Bank Account and also have their own specialized properties like unsettled payment etc.



Generalization vs Inheritance

Generalization is the term that we use to denote abstraction of common properties into a base class in UML. The **UML** diagram's **Generalization** association is also known as **Inheritance**. When we implement Generalization in a programming language, it is often called Inheritance instead. Generalization and inheritance are basically the same, the terminology just differs depending on the context where it is being used.