UML and EA (Enterprise Architecture) Tools

UML stands for **Unified Modelling Language**.

Following are some fundamental concepts of the object-oriented world −

**Objects** − Objects represent an entity and the basic building block.

**Class** − Class is the blue print of an object.

**Abstraction** − Abstraction represents the behavior of an real world entity.

**Encapsulation** − Encapsulation is the mechanism of binding the data together and hiding them from the outside world.

**Inheritance** − Inheritance is the mechanism of making new classes from existing ones.

**Polymorphism** − It defines the mechanism to exists in different forms.

The building blocks of UML can be defined as −

* Things: It can be Structural, Behavioral, Grouping, Annotational
* Relationships
* Diagrams

Structural Things

* **Structural things** define the static part of the model. They represent the physical and conceptual elements. Following are the brief descriptions of the structural things.
* **Class −** Class represents a set of objects having similar responsibilities.

class

* **Interface −** Interface defines a set of operations, which specify the responsibility of a class.

Interface

* **Collaboration −**Collaboration defines an interaction between elements.

Collaboration

* **Use case −**Use case represents a set of actions performed by a system for a specific goal.

Use case

* **Component −**Component describes the physical part of a system.

Component

* **Node −** A node can be defined as a physical element that exists at run time.

## **Relationship**

**Relationship** is another most important building block of UML. It shows how the elements are associated with each other and this association describes the functionality of an application.

There are four kinds of relationships available.

### Dependency

Dependency is a relationship between two things in which change in one element also affects the other.

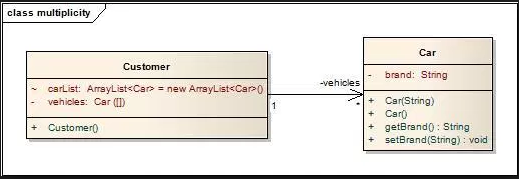
Dependency

## **Association**

Association is a relationship between two objects. In other words, association defines the multiplicity between objects. You may be aware of one-to-one, one-to-many, many-to-one, many-to-many all these words define an association between objects. Aggregation is a special form of association. Composition is a special form of aggregation.

http://javapapers.com/wp-content/uploads/2010/06/association.jpg

**Example:**A Student and a Faculty are having (has-a) an association.



**Aggregation:**

Aggregation is a special case of association. A directional association between objects. When an object ‘has-a’ another object, then you have got an aggregation between them. Direction between them specified which object contains the other object. Aggregation is also called a “Has-a” relationship.

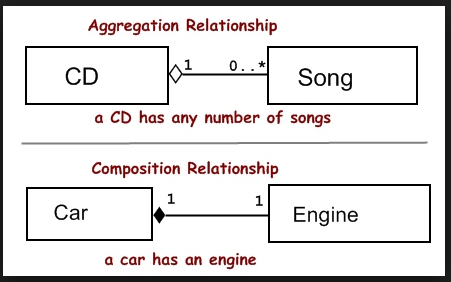
http://javapapers.com/wp-content/uploads/2010/06/aggregation.jpg

## **Composition**

Composition is a special case of aggregation. In a more specific manner, a restricted aggregation is called composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition.

http://javapapers.com/wp-content/uploads/2010/06/composition.jpg

**Example:**A class contains students. A student cannot exist without a class. There exists composition between class and students.

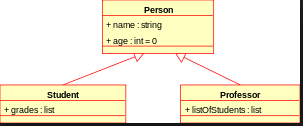
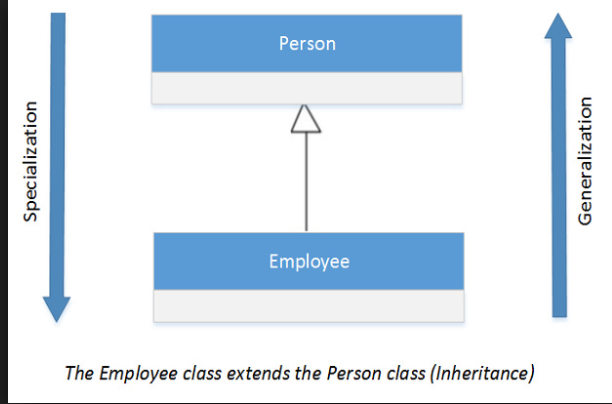


### Generalization

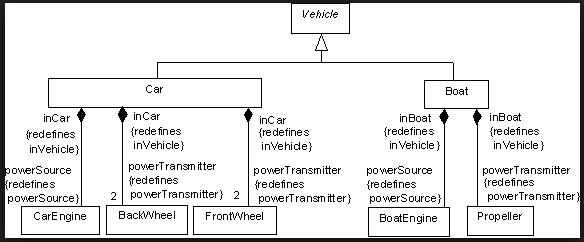
Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes the inheritance relationship in the world of objects.

Generalization

***Example:*** Consider there exists a class named Person. A student is a person. A faculty is a person. Therefore, here the relationship between student and person, similarly faculty and person is generalization.

**A sample relationship:**



**UML Diagrams:**

UML includes the following nine diagrams,

* Class diagram
* Object diagram
* Use case diagram
* Sequence diagram
* Collaboration diagram
* Activity diagram
* Statechart diagram
* Deployment diagram
* Component diagram

These are divided into three important types of Modelling:

1. **Structural Modeling**

Structural modeling captures the static features of a system. They consist of the following −

* Classes diagrams
* Objects diagrams
* Deployment diagrams
* Package diagrams
* Composite structure diagram
* Component diagram

1. **Behavioral Modeling**

Behavioral model describes the interaction in the system. It represents the interaction among the structural diagrams. Behavioral modeling shows the dynamic nature of the system. They consist of the following −

* Activity diagrams
* Interaction diagrams
* Use case diagrams

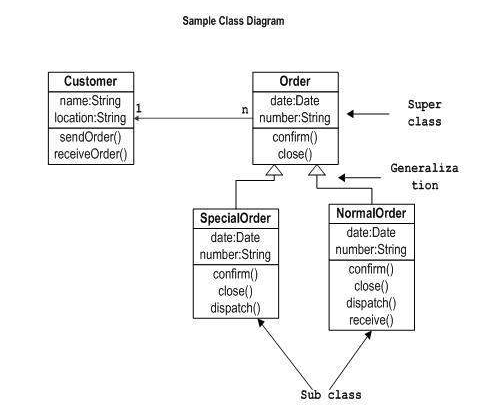
1. **Architectural Modeling**

Architectural model represents the overall framework of the system. It contains both structural and behavioral elements of the system. Architectural model can be defined as the blueprint of the entire system. Package diagram comes under architectural modeling.

**UML- Class Diagram**

The purpose of the class diagram can be summarized as −

* Analysis and design of the static view of an application.
* Describe responsibilities of a system.
* Base for component and deployment diagrams.
* Forward and reverse engineering.



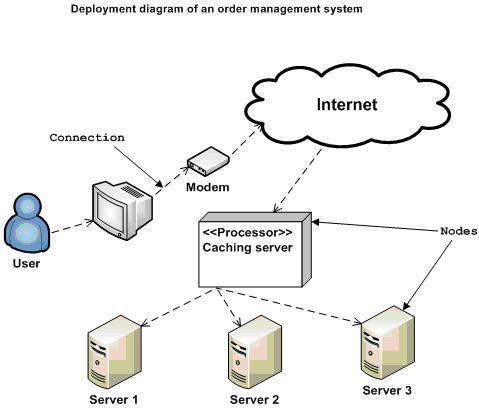
class diagrams are used for −

* Describing the static view of the system.
* Showing the collaboration among the elements of the static view.
* Describing the functionalities performed by the system.
* Construction of software applications using object oriented languages.

**UML-Deployment Diagram**

Deployment diagrams are useful for system engineers. An efficient deployment diagram is very important as it controls the following parameters −

* Performance
* Scalability
* Maintainability
* Portability



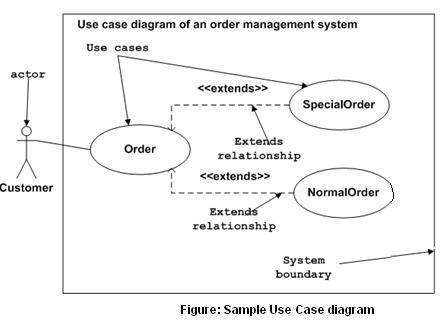
Deployment diagrams can be used −

* To model the hardware topology of a system.
* To model the embedded system.
* To model the hardware details for a client/server system.
* To model the hardware details of a distributed application.
* For Forward and Reverse engineering.

**UML- Use Case Diagram**

In brief, the purposes of use case diagrams can be said to be as follows −

* Used to gather the requirements of a system.
* Used to get an outside view of a system.
* Identify the external and internal factors influencing the system.
* Show the interaction among the requirements are actors.



Use case diagrams can be used for −

* Requirement analysis and high level design.
* Model the context of a system.
* Reverse engineering.
* Forward engineering.

UML- Interaction Diagrams

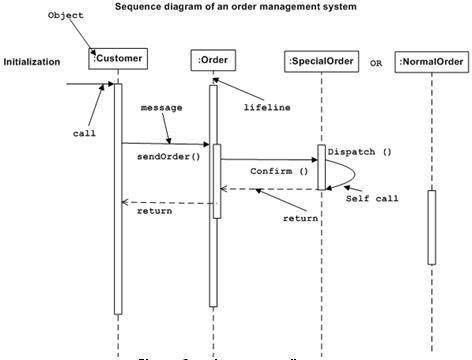
This interactive behavior is represented in UML by two diagrams known as **Sequence diagram** and **Collaboration diagram**. The basic purpose of both the diagrams are similar.

Sequence diagram emphasizes on time sequence of messages and collaboration diagram emphasizes on the structural organization of the objects that send and receive messages.

Following things are to be identified clearly before drawing the interaction diagram

* Objects taking part in the interaction.
* Message flows among the objects.
* The sequence in which the messages are flowing.
* Object organization.

**UML- Sequence Diagrams**



Interaction diagrams can be used −

* To model the flow of control by time sequence.
* To model the flow of control by structural organizations.
* For forward engineering.
* For reverse engineering.