# UML Basics

1)      **Realization**: If a concrete class implements another class then it is known as Realization. For e.g. relationship between an Interface Runnable and concrete class MyRunnable or lets say there is an interface IFileReadUtils and there are its implementations like CsvFileReader, XmlFileReader , JsonReader etc.

2)      **Generalization**: If a concrete class extends a class(abstract/concrete) then it is knows an Generalization. It’s a simple IS-A Relationship. For e.g. there is an abstract class Animal and the specialized classes like Dog, Cat etc. Common behavior of all animal will come as concrete methods in Abstract class, for e.g. all animals have same organ system. While distinguished behaviors will come as an abstract method in abstract class, for e.g. each animal will make different sound. So each specialized class will override makeSound() of abstract class. This kind of relationship is known as Generalization.

3)      **Composition** : It’s a has-a relationship but a very strong one. Here one class is owner of another class. When owner dies so does the contained object. For e.g. Bowl has water. When bowl break water will also be gone.

Public Class Car {

Private final Engine e;

Car(Engine e){

this.e =e;

}

}

Public class Engine{}

So as you can see Car has Engine. When reference to Car is gone then there is no way you can reach Engine (from garbage collection paradigm, when an object cannot be reached then contained references cannot be reached). Important part is you have specified final keyword as this enforces us to

Instantiate Engine at Car construction time so that Engine is created at time when Car was made.

4)      **Aggregation**: It’s  an association relationship but not strong one. Here one class is not owner of another class. For e.g. EETeam has members. But Impetus employee can exist without EE team. See code below:

Public class EETeam{

private List<ImpetusEmployee> memberList;

//populate List in a method.

}

Public class ImpetusEmployee{

//name, experience,project etc….

}

5) **Dependency**: If you see Composition, Aggregation and Dependency are all a kind of Association. How a class use another class decides what type of association it would be.

It’s not dependency as in this case the class use another class in some point of time say in a method as a local Variable. For e.g.

Class A{

Public void do(){

B b = new B();

b.execute();

}

}

Above is an example of dependency in UML terms. Do go through this url to understand OOAAD principles.

<http://javarevisited.blogspot.in/2012/03/10-object-oriented-design-principles.html>