Association, Aggregation, Composition, Abstraction, Generalization, Realization, Dependency

Last modified on September 7th, 2014 by Joe.

These terms signify the relationships between classes. These are the building blocks of object oriented programming and very basic stuff. But still for some, these terms look like Latin and Greek. Just wanted to refresh these terms and explain in simpler terms.

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.

Example: A Student and a Faculty are having 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.

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.

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

Difference between aggregation and composition

Composition is more restrictive. When there is a composition between two objects, the composed object cannot exist without the other object. This restriction is not there in aggregation. Though one object can contain the other object, there is no condition that the composed object must exist. The existence of the composed object is entirely optional. In both aggregation and composition, direction is must. The direction specifies, which object contains the other object.

Example: A Library contains students and books. Relationship between library and student is aggregation. Relationship between library and book is composition. A student can exist without a library and therefore it is aggregation. A book cannot exist without a library and therefore its a composition. For easy understanding I am picking this example. Don’t go deeper into example and justify relationships!

Abstraction

Abstraction is specifying the framework and hiding the implementation level information. Concreteness will be built on top of the abstraction. It gives you a blueprint to follow to while implementing the details. Abstraction reduces the complexity by hiding low level details.

Example: A wire frame model of a car.

Generalization

Generalization uses a “is-a” relationship from a specialization to the generalization class. Common structure and behaviour are used from the specializtion to the generalized class. At a very broader level you can understand this as inheritance. Why I take the term inheritance is, you can relate this term very well. Generalization is also called a “Is-a” relationship.

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.

Realization

Realization is a relationship between the blueprint class and the object containing its respective implementation level details. This object is said to realize the blueprint class. In other words, you can understand this as the relationship between the interface and the implementing class.

Example: A particular model of a car ‘GTB Fiorano’ that implements the blueprint of a car realizes the abstraction.

Dependency

Change in structure or behaviour of a class affects the other related class, then there is a dependency between those two classes. It need not be the same vice-versa. When one class contains the other class it this happens.

Example: Relationship between shape and circle is dependency.