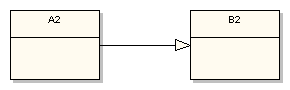
1. Dependency
2. Association:
   1. Aggregation
   2. Composition
3. **Generalization:**

UML generalization symbolizes what is known as inheritance in the world of object-oriented programming. It is sometimes also called specialization.



More specifically UML generalization corresponds to class extension in the Java language. The above diagram fragment would be implemented in Java as follows:

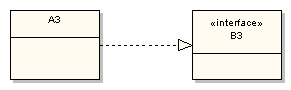
import B2;

public class A2 extends B2 { // . . . }

B2 is the super class and A2 is the subclass in the relationship. Just remember that the generalization symbol forms a line from the subclass to its super class with the clear triangular arrowhead pointing at the super class.

1. **Realization:**

This relationship is somewhat related to generalization, but a bit different. In object-oriented programming parlance realization represents the implementation of an interface by a class. So it represents how some characteristics of a class are defined, but says nothing about the implementation details:



This diagram fragment states that class A3 implements or realizes the interface defined by B3. In the Java language the above realization relationship would be programmed as follows:

import B3;

public class A3 implements B3 {

// . . .

}

Realization is very important when designing object-oriented subsystems and frameworks. The interface being realized in a class diagram represents a contract between the subsystem or framework and its consumer. The interface publisher guarantees that any consumer implementing one or more of its public interfaces properly will have some level of consistent integration with the interface-defining subsystem or framework.