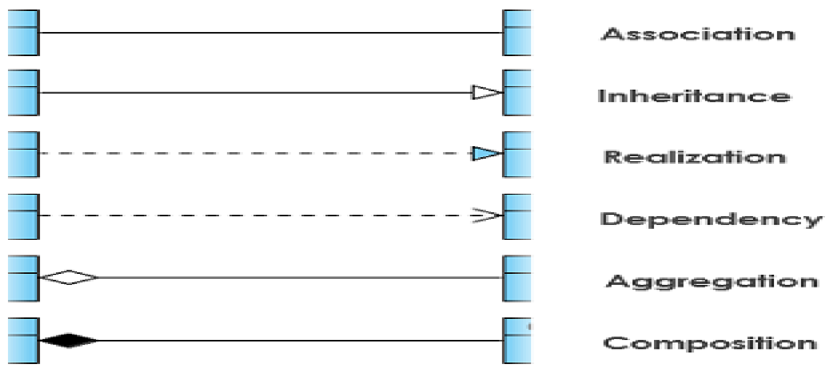
Module 1: UML Class Diagram and Relationships

Question:

- How can we determine classes while developing a software system?

  - **When**: in which phase in software process?
  - **Where**: Class name/attributes/methods will define from where?
  - **Who**: who takes part in these activities?

- How many relationships among classes in UML are there?
- Given two classes A, B, how can we define a relationship between these two classes?
- Is there more than one relationship between two classes? If more than one, which one should be selected? Selection depends what?
- Class relationship depends on **business process**?
- Using Design pattern DAO (Data Access Object) why?



There are six relationships between classes in a UML class diagram :

Dependency

Association

Aggregation

Composition

Inheritance

Realization

The above relationships are read as follows:

- Dependency : class A uses class B
- Association: class A associates with class B
- Aggregation : class A has a class B
- Composition : class A owns a class B
- Inheritance : class B is a Class A (or class A is extended by class B)
- Realization : class B realizes Class A (or class A is realized by class B)

Dependency/association is represented when a reference to one class is passed in as a method parameter to another class. For example, an instance of class B is passed in to a method of class A:

```
public class A {
public void doSomething(B b) {   }
}
```

Aggregation: If class A stored the reference to class B for later use we would have a different relationship called Aggregation. A more common and more obvious example of Aggregation would be via setter injection:

```
public class A {
   private B bb;
   public void setB(B b) { bb = b; }
   }
```

Composition: Aggregation is the weaker form of object containment (one object contains other objects). The stronger form is called Composition. In Composition the containing object is responsible for the creation and life cycle of the contained object. Implementation of composition is of the following forms:

- First, via member initialization:

```
public class A {
   private B b = new B();
}
```

- Second, via constructor initialization:

```
public class A {
   private B bb;
   public A() {
    bb = new B();
  }
}// default constructor
```

- Third, via lazy init:

```
public class A {
  private B _b;
  public B getB() {
    if (null == _b) {
      _b = new B();
    }
    return _b;
  } // getB()
```

Inheritance is a fairly straightforward relationship to depict in Java:

```
public  class A {
      ...
  } // class A

public  class B extends A {
      ....
  } // class B
```

Realization is also straighforward in Java and deals with implementing an interface:

```
public interface A {
      ...
  } // interface A
public class B implements A {
      ...
  } // class B
```

We will present three examples which are investigated in all modules of the subject A&D

Example 1: Library

Example 2: Book Store Online

Example 3: Subject Registering System at University