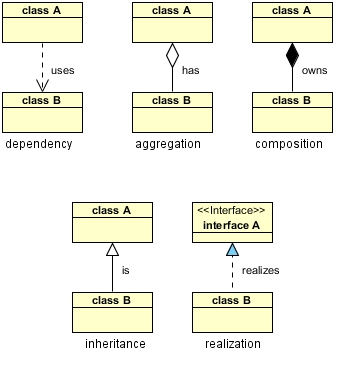


Systems Development: Object Oriented Analysis and Design

(H172 35)

Class Diagram Associations



Dependency: class A uses 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 interface A (or interface A is realized by class B)

**Dependency** 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)**

**{**

Now, 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 \_b;**

**public void setB(B b)**

**{**

**\_b = b;**

**}**

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 (either directly or indirectly)

Following are a few examples of Composition

First, via member initialization:

**public class A**

**{**

**private B \_b = new B();**

Second, via constructor initialization:

**public class A { private B \_b; public A() { \_b = new B(); } // default constructor**

Third, via lazy init (completely hide reference

to B):

**public class A { private B \_b; public void doSomethingUniqueToB() { if (\_b == null) { \_b = new B(); } return \_b.doSomething(); } // doSomethingUniqueToB()**

**Inheritance** is a fairly straightforward relationship:

**public class Album:Product {... } // create a class called Album which inherits from Product class**

**Realization** is also straighforward and deals with implementing an

interface:

**class MyClass implements Printable {…} // MyClass implements the Printable interface and will provide implementation for the methods described in the interface**