**依赖和关联的区别**

对于依赖和关联，网上大部分都是流行下面这篇文章的说法，自己对于它们之间的区别还是不太了解，同时系统结构中各个层之间是相互依赖还是相互关联，还是分不清。打算先看看写写代码，然后再体会体会。

转：依赖和关联的区别

首先类之间的关系**从宏观上**可以分为：关联、依赖、继承（泛化），而其中关联又有两种特例：聚合和组合，不过在UML2.X中聚合和组合合成一种关系了。

 **从使用的频率来看**，关联（包括聚合和组合）关系是使用最为广泛的；其次是依赖和继承

 关联和依赖的区别：

对于依赖和关联，网上大部分都是流行下面这篇文章的说法，自己对于它们之间的区别还是不太了解，同时系统结构中各个层之间是相互依赖还是相互关联，还是分不清。打算先看看写写代码，然后再体会体会。

转：依赖和关联的区别

首先类之间的关系**从宏观上**可以分为：关联、依赖、继承（泛化），而其中关联又有两种特例：聚合和组合，不过在UML2.X中聚合和组合合成一种关系了。

 **从使用的频率来看**，关联（包括聚合和组合）关系是使用最为广泛的；其次是依赖和继承

 关联和依赖的区别：

1. **从类之间关系的强弱程度来分**，关联表示类之间的很强的关系；依赖表示类之间的较弱的关系；
2. **从类之间关系的时间角度来分**，关联表示类之间的“持久”关系，这种关系一般表示一种重要的业务之间的关系，需要保存的，或者说需要“持久化”的，或者说需要保存到数据库中的。比如学生管理系统中的Student类和Class（班级）类，一个Student对象属于哪个Class是一个重要的业务关系，如果这种关系不保存，系统就无法管理。另外，依赖表示类之间的是一种“临时、短暂”关系，这种关系是不需要保存的，比如Student类和StuEditScreen（学生登录界面）类之间就是一种依赖关系，StuEditScreen类依赖Student类，依赖Student对象的信息来显示编辑学生信息。
3. **设计类之间的关系是遵循的原则：**首先判断类之间是否是一种“关联”关系，若不是再判断是否是“依赖关系”，一般情况下若不是关联，就是依赖关系
4. **依赖一般情况下是以下几种情况之一**：a、ClassA中某个方法的参数类型是ClassB；这种情况成为耦合；b、ClassA中某个方法的参数类型是ClassB的一个属性；这种情况成为紧耦合；c、ClassA中某个方法的实现实例化ClassB；d、ClassA中某个方法的返回值的类型是ClassB；如果出现了上述四种情况之一，两个类很有可能就是“依赖”关系。
5. 依赖关系（Dependency）：是类与类之间的连接，**依赖总是单向的**。依赖关系代表一个类依赖于另一个类的定义。下面的例子中class A 依赖与class B、C、D。

java 代码

* 1. public class A{
  2. public B getB(C c, D d){
  3. E e = new E();
  4. B b = new B(c, d, e);
  5. }
  6. }

1. 关联是一种结构关系，说明一个事物的对象与另一个事物的对象相联系。给定一个连接两各类的关联，可以从一个类的对象导航到另一个类的对象。  
   关联可以有方向，即导航。一般不作说明的时候，导航是双向的，不需要在线上标出箭头。大部分情况下导航是单向的，可以加一个箭头表示。**关联在代码中一般表示为属性（成员变量）**，例如下面例子中 class A与B关联

java 代码

* 1. public class A{
  2. private B b;
  3. }

如果B也关联到A，那么它们就是双向的关联。

java 代码

* 1. public class B{
  2. private A a;
  3. }

1. 聚合关系(Aggregation)：是关联关系的一种，是强的关联关系。聚合是整体与个体之间的关系。如汽车类与引挚类，轮胎类之间的关系就是整体与个体的关系。

与关联关系一样，聚合关系也是通过实例变量来实现的。

\*\*\*关联关系所涉及的两个类是处于同一层次上的，而在聚合关系中，两个类处在不平等的层次上的，一个代表整体，一个代表部分。（**关联与聚合仅仅从语法上是区分不开的，需要察所涉及的类之间的逻辑关系。**）

1. 组成关系(Composition)：是关联关系的一种，是比聚合关系强的关系。它要求普通的聚合关系中代表的对象负责代表部分的对象的生命周期，合成关系是不能共享的。

代表整体的对象需要负责保持对象的存活，在一些情况下负责将代表部分的对象湮灭掉。代表整体的对象可以将代表部分的对象传递给另一个对象，由后者负责此对象的生命周期。换言之，

\*\*\*\*\*\*\*代表部分的对象在每一个时刻只能与一个对象发生合成关系，由后者排它的负责其生命周期。