据作者的个人理解，关系的强度：依赖
依赖 意在说明2个类存在关系，一般Java语言中体现为局域变量、方法的形参，或者对静态方法的调用；
关联一般是一个类持有另一个类作成员变量来体现，只说明类与类之前是有联系的，至于他们的关系是聚合还是组合，还要看业务和表现形式。Both aggregation and composition are special kinds of associations.
聚合，聚合关系是关联关系的一种，聚合是整体和个体之间的关系。Aggregation is used to represent ownership or a whole/part relationship。一般的表现形式是一个类持有另一个类作成员变量来体现，但绝对不可以New它。聚合关系当一个类destory的时候，持有的另一个类不会受到影响。聚合是"has a"，有一个，但它不一定只是你自己的，也可以被另的类有。
组合，组合关系是关联关系的一种，是比聚合关系强的关系。它要求普通的聚合关系中代表整体的对象负责代表部分对象的生命周期，组合关系是不能共享的。composition is used to represent an even stronger form of ownership.With composition, we get coincident lifetime of part with the whole. The composite object has sole responsibility for
the disposition of its parts in terms of creation and destruction.In implementation terms, the composite is responsible for memory allocation and deallocation。If the composite is destroyed, it must either destroy all its parts or else give responsibility for
them to some other object. 聚合关系当一个类destory的时候，持有的另一个类一定也会被destory。是个体的生活周期被整体所决定。组合是"contain a"，有一个，它一定只是你自己的，它跟随你的创建而创建，跟随你的消亡而消亡。
In fact, composition was originally called aggregation-by-value in an earlier UML draft, with “normal” aggregation being thought of as aggregation-by-reference.对这句话的理解，我认为聚合只是引用，组合而是去New对象了，赋值了。The
distinction between aggregation and composition is more of a design concept and is not usually relevant during analysis. 聚合和组合是在设计时的一种关系说明.
个人理解，如有错误，敬请指正。
---------- 参考爱的分割线 ----------
aggregation:聚合关系 composition组合关系
Both aggregation and composition are special kinds of associations. Aggregation is used to represent ownership or a whole/part relationship, and composition is used to represent an even stronger form of ownership. With composition, we get coincident lifetime of part with the whole. The composite object has sole responsibility for the disposition of its parts in terms of creation and destruction. In implementation terms, the composite
is responsible for memory allocation and deallocation
Moreover, the multiplicity of the aggregate end may not exceed one; i.e., it is unshared. An object may be part of only one composite at a time. If the composite is destroyed, it must either destroy all its parts or else give responsibility for them to some

other object. A composite object can be designed with the knowledge that no other object will destroy its parts.
Composition can be used to model by-value aggregation, which is semantically equivalent to an attribute. In fact, composition was originally called aggregation-by-value in an earlier UML draft, with “normal” aggregation being thought of as aggregation-by-reference.
The definitions have changed slightly, but the general ideas still apply. The distinction between aggregation and composition is more of a design concept and is not usually relevant during analysis.
Finally, a word of warning on terminology. While UML uses the terms association, aggregation, andcomposition with specific meanings, some object-oriented authors use one or more of these terms with
slightly different interpretations. For example, it is fairly common to see all three UML relationships grouped under a single term, say composition, and then to discuss object-oriented relationships as being either inheritance (generalization) or composition.
依赖(Dependency)关系是类与类之间的联接。依赖关系表示一个类依赖于另一个类的定义。例如，一个人(Person)可以买车(car)和房子(House)，Person类依赖于Car类和House类的定义，因为Person类引用了Car和House。与关联不同的是，Person类里并没有Car和House类型的属性，Car和House的实例是以参量的方式传入到buy()方法中去的。一般而言，依赖关系在Java语言中体现为局域变量、方法的形参，或者对静态方法的调用。
关联(Association)关系是类与类之间的联接，它使一个类知道另一个类的属性和方法。关联可以是双向的，也可以是单向的。在Java语言中，关联关系一般使用成员变量来实现。
聚合(Aggregation) 关系是关联关系的一种，是强的关联关系。聚合是整体和个体之间的关系。例如，汽车类与引擎类、轮胎类，以及其它的零件类之间的关系便整体和个体的关系。与关联关系一样，聚合关系也是通过实例变量实现的。但是关联关系所涉及的两个类是处在同一层次上的，而在聚合关系中，两个类是处在不平等层次上的，一个代表整体，另一个代表部分。
组合(Composition) 关系是关联关系的一种，是比聚合关系强的关系。它要求普通的聚合关系中代表整体的对象负责代表部分对象的生命周期，组合关系是不能共享的。代表整体的对象需要负责保持部分对象和存活，在一些情况下将负责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组合关系，由后者排他地负责生命周期。部分和整体的生命周期一样。
——摘自《Java面向对象编程》，作者：孙卫琴
以上关系的耦合度依次增强(关于耦合度的概念将在以后具体讨论，这里可以暂时理解为当一个类发生变更时，对其他类造成的影响程度，影响越小则耦合度越弱，影响越大耦合度越强)。由定义我们已经知道，依赖关系实际上是一种比较弱的关联，聚合是一种比较强的关联，而组合则是一种更强的关联，所以笼统的来区分的话，实际上这四种关系、都是关联关系。
其关系强弱为: 依赖

依赖关系比较好区分，它是耦合度最弱的一种，在java中表现为局域变量、方法的形参，或者对静态方法的调用，如下面的例子：Driver类依赖于Car类，Driver的三个方法分别演示了依赖关系的三种不同形式。

```
class Car {undefined
public static void run(){undefined
System.out.println("汽车在奔跑");
}
}
class Driver {undefined
//使用形参方式发生依赖关系
public void drive1(Car car){undefined
car.run();
}
//使用局部变量发生依赖关系
public void drive2(){undefined
Car car = new Car();
car.run();
}
//使用静态变量发生依赖关系
public void drive3(){undefined
Car.run();
}
}
```

关联关系在java中一般使用成员变量来实现，有时也用方法形参的形式实现。依然使用Driver和Car的例子，使用方法参数形式可以表示依赖关系，也可以表示关联关系，毕竟我们无法在程序中太准确的表达语义。在本例中，使用成员变量表达这个意思：车是我自己的车，我“拥有”这个车。使用方法参数表达：车不是我的，我只是个司机，别人给我什么车我就开什么车，我使用这个车。

```
class Driver {undefined
//使用成员变量形式实现关联
Car mycar;
public void drive(){undefined
mycar.run();
}
...
//使用方法参数形式实现关联
public void drive(Car car){undefined
car.run();
```

}
}
聚合关系是是一种比较强的关联关系，java中一般使用成员变量形式实现。对象之间存在着整体与部分的关系。例如上例中
class Driver {undefined
//使用成员变量形式实现聚合关系
Car mycar;
public void drive(){undefined
mycar.run();
}
}
假如给上面代码赋予如下语义：车是一辆私家车，是司机财产的一部分。则相同的代码即表示聚合关系了。聚合关系一般使用setter方法给成员变量赋值。
假如赋予如下语义：车是司机的必须有的财产，要想成为一个司机必须要先有辆车，车要是没了，司机也不想活了。而且司机要是不干司机了，这个车就砸了，别人谁也别想用。那就表示组合关系了。一般来说，为了表示组合关系，常常会使用构造方法来达到初始化的目的，例如上例中，加上一个以Car为参数的构造方法
public Driver(Car car){undefined
mycar = car;
}
所以，关联、聚合、组合只能配合语义，结合上下文才能够判断出来，而只给出一段代码让我们判断是关联，聚合，还是组合关系，则是无法判断的。
————————————————