类与类之间的关系对于理解面向对象具有很重要的作用，以前在面试的时候也经常被问到这个问题，在这里我就介绍一下。

类与类之间存在以下关系: (1)泛化(Generalization) (2)关联(Association) (3)依赖(Dependency) (4)聚合(Aggregation)

UML 图与应用代码例子:

1.泛化(Generalization)

[泛化]

表示类与类之间的继承关系，接口与接口之间的继承关系，或类对接口的实现关系。一般化的关系是从子类指向父类的，与继承或实现的方法相反。

[具体表现]

父类 父类实例＝new 子类()

[UML 图](图 1.1)

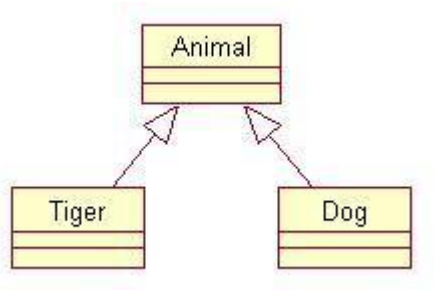

图 1.1 Animal 类与 Tiger 类,Dog 类的泛化关系

[代码表现]

```
1. class Animal{}
2. class Tiger extends Animal{}
3. public class Test
4. {
5.          public void test()
6.           {
7.                    Animal a=new Tiger();
```

```
8.          }
9.  }
```

2.依赖(Dependency)

[依赖]

对于两个相对独立的对象，当一个对象负责构造另一个对象的实例，或者依赖另一个对象的服务时，这两个对象之间主要体现为依赖关系。

[具体表现]

依赖关系表现在局部变量，方法的参数，以及对静态方法的调用

[现实例子]

比如说你要去拧螺丝，你是不是要借助(也就是依赖)螺丝刀(Screwdriver)来帮助你完成拧螺丝(screw)的工作

[UML 表现](图 1.2)

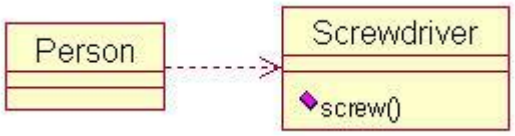

图 1.2 Person 类与 Screwdriver 类的依赖关系

[代码表现]

```
1.  public class Person{
2.          /** 拧螺丝 */
3.          public void screw(Screwdriver screwdriver){
4.                  screwdriver.screw();
5.           }
6.  }
```

3.关联(Association)

[关联]

对于两个相对独立的对象，当一个对象的实例与另一个对象的一些特定实例存在固定的对应关系时，这两个对象之间为关联关系。

[具体表现]

关联关系是使用实例变量来实现

[现实例子]

比如客户和订单，每个订单对应特定的客户，每个客户对应一些特定的订单；再例如公司和员工，每个公司对应一些特定的员工，每个员工对应一特定的公司

[UML 图] (图 1.3)

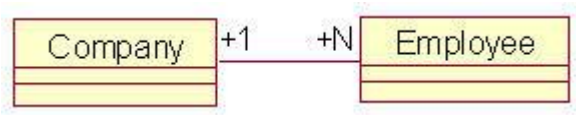

图 1.3 公司和员工的关联关系

[代码表现]

```
1.  public class Company{
2.          private Employee employee;
3.          public Employee getEmployee(){
4.                  return employee;
5.           }
6.          public void setEmployee(Employee employee){
7.                  this.employee=employee;
8.           }
9.          //公司运作
10.         public void run(){
11.                  employee.startWorking();
12.          }
13.}
```

(4)聚合（Aggregation）

[聚合]

当对象 A 被加入到对象 B 中，成为对象 B 的组成部分时，对象 B 和对象 A 之间为聚集关系。聚合是关联关系的一种，是较强的关联关系，强调的是整体与部分之间的关系。

[具体表现]

与关联关系一样，聚合关系也是通过实例变量来实现这样关系的。关联关系和聚合关系来语法上是没办法区分的，从语义上才能更好的区分两者的区别。

[关联与聚合的区别]

(1)关联关系所涉及的两个对象是处在同一个层次上的。比如人和自行车就是一种关联关系，而不是聚合关系，因为人不是由自行车组成的。

聚合关系涉及的两个对象处于不平等的层次上，一个代表整体，一个代表部分。比如电脑和它的显示器、键盘、主板以及内存就是聚集关系，因为主板是电脑的组成部分。

(2)对于具有聚集关系（尤其是强聚集关系）的两个对象，整体对象会制约它的组成对象的生命周期。部分类的对象不能单独存在，它的生命周期依赖于整体类的对象的生命周期，当整体消失，部分也就随之消失。比如张三的电脑被偷了，那么电脑的所有组件也不存在了，除非张三事先把一些电脑的组件（比如硬盘和内存）拆了下来。

[UML 图](图 1.4)

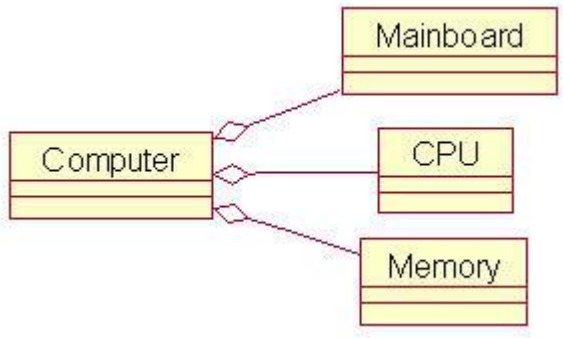

图 1.3 电脑和组件的聚合关系

[代码表现]

```
1.  public class Computer{
2.          private CPU cpu;
3.          public CPU getCPU(){
```

```
4.                  return cpu;
5.           }
6.          public void setCPU(CPU cpu){
7.                  this.cpu=cpu;
8.           }
9.          //开启电脑
10.         public void start(){
11.                 //cpu运作
12.                  cpu.run();
13.          }
```