Unified Modeling Language统一建模语言

UML类图中的关系分为四种：泛化、依赖、关联（细分为聚合、组合）、实现

## 一、泛化（Generalization）

泛化是父类和子类之间的关系，子类继承父类的所有结构和行为。在子类中可以增加新的结构和行为，也可覆写父类的行为。

一般用一个带空心箭头的实线表示泛化关系，UML图如下：

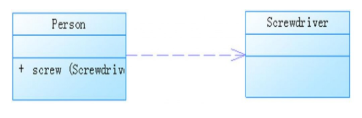


泛化对应Java中继承关系，即子类继承父类中除private修饰外的所有东西（变量、方法等）

## 二、依赖（Dependency）

依赖关系是一种使用关系，特定事物的改变有可能会影响到使用该事物的事物，反之不成立。在你想显示一个事物使用另一个事物时使用。

一般用一条指向被依赖事物的虚线表示，UML图如下：



通常情况下，依赖关系体现在某个类的方法使用另一个类作为参数。示例如下：

public class Screwdriver { //螺丝刀，作为人类的工具，是用来被人类使用的

}

public class Person{

public void screw(Screwdriver src){ //拧螺丝，需使用螺丝刀

}

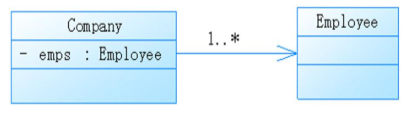
}

Person类的screw()方法在使用时就得传入一个Screwdriver类型的参数，这样Screwdriver的改变就会影响到Person，因此Person与Screwdriver之间就是依赖关系（Person依赖于Screwdriver）

## 三、关联（Association）

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。给定有关联的两个类，可以从一个类的对象得到另一个类的对象。关联有两元关系和多元关系。两元关系是指一种一对一的关系，多元关系是一对多或多对一的关系。两个类之间的简单关联表示了两个同等地位类之间的结构关系。当你想要表示结构化关系时使用关联。（可以想想Hibernate的关联关系）

一般用实线连接有关联的同一个类或不同的两个类。UML图如下：



通常情况下，关联关系是通过类的成员变量来实现的。代码示例：

public class Company { //公司

private Employee emp;//一个公司雇员，公司与雇员之间就是一种关联关系。

}

public class Employee{

}

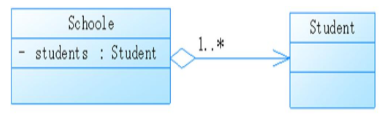
雇员作为公司的属性，不同于上面的依赖。依赖的话，雇员的改变会影响公司，显然不是。在这里雇员仅仅作为公司的一个属性而存在。因此Employee与Company之间是关联关系。关联关系还可以细分为聚合和组合两种。

### 3.1聚合（Aggregation）

聚合是一种特殊的关联。它描述了“has a”关系，表示整体对象拥有部分对象。

关联关系和聚合关系在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。聚合是较强的关联关系，强调的是整体与部分之间的关系。例如，学校和学生的关系。聚合的整体和部分之间在生命周期上没有什么必然的联系，部分对象可以在整体对象创建之前创建，也可以在整体对象销毁之后销毁。

一般用带一个空心菱形（整体的一端——学校）的实线表示。UML图如下：



与关联关系一样，聚合关系也是通过类的成员变量来实现的。示例代码：

public class Student{

}

public class School{

private List<Student> students ; //学校与学生是聚合关系

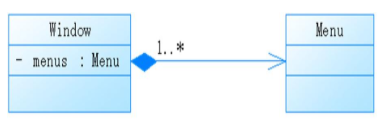
}

学校是整体，而学生是部分。学校与学生都是可以独立存在的，之间没有什么必然的联系。因此学校与学生就是聚合关系。

### 3.2组合（Composition）

组合是聚合的一种形式，它具有更强的拥有关系，强调整体与部分的生命周期是一致的，整体负责部分的生命周期的管理。生命周期一致指的是部分必须在组合创建的同时或者之后创建，在组合销毁之前或者同时销毁，部分的生命周期不会超出组合的生命周期。例如Windows的窗口和窗口上的菜单就是组合关系。如果整体被销毁，部分也必须跟着一起被销毁，如果所有者被复制，部分也必须一起被复制。

一般用带实心菱形（整体的一端）的实线来表示。UML图如下：



与关联关系一样，组合关系也是通过类的成员变量来实现的。示例代码：

public class Menu{

}

public class Window{

private List<Menu> menus ;

}

菜单的存在前提是窗口的存在，两者之间存在很强的拥有关系。且窗口对菜单的生命周期负责，只有在窗口创建之后，菜单才能够创建，菜单必须在窗口销毁之前销毁，因此Window和Menu之间是组合关系。

### 聚合和组合的区别在于：

聚合关系是“has-a”关系，组合关系是“contains-a”关系；聚合关系表示整体与部分的关系比较弱，而组合比较强；聚合关系中代表部分事物的对象与代表聚合事物的对象的生存期无关，一旦删除了聚合对象不一定就删除了代表部分事物的对象。组合中一旦删除了组合对象，同时也就删除了代表部分事物的对象。

另外有一个差别是组合中的一个对象在同一时刻只能属于一个组合对象，而聚合的一个部分对象可以被多个整体对象聚合，例如一个学生可以在多个学校就读，而一个菜单在同一时刻只能是某个窗口内的对象。

## 四、实现（Realization）

实现关系指定两个实体之间的一个合约。换言之，一个实体定义一个合约，而另一个实体保证履行该合约。对类来说，就是一个类实现了一个接口。

一般用一条指向接口的虚线表示，UML图如下：



实现对应Java中的实现接口（implements）。示例代码：

public interface Person{

}

public class Student implements Person{

}

## 五、总结

类间关系有很多种，在大的类别上可以分为两种：纵向关系、横向关系。纵向关系就是继承关系。横向关系较为微妙，按照UML的建议大体上可以分为四种：依赖（Dependency）、关联（Association）、聚合（Aggregation）、组合（Composition）

关于关联，聚合，组合在实现上没有显著区别，想区别他们只有通过判断关系双方之间的实际关系，如关系强弱、创建与销毁之间有无必要关联等。

他们的强弱关系是没有异议的：依赖 < 关联 < 聚合 < 组合 < 泛化（继承）

实现方式区别：

依赖：关系对象出现在局部变量或者方法的参数里，或者关系类的静态方法被调用

关联：关系对象出现在实例变量中

聚合：关系对象出现在实例变量中

组合：关系对象出现在实例变量中

泛化：extends

实现：implements