在软件系统中，类并不是孤立存在的，类与类之间存在各种关系，对于不同类型的关系，UML提供了不同的表示方式。

1.关联关系

关联(Association)关系是类与类之间最常用的一种关系，它是一种结构化关系，用于表示一类对象与另一类对象之间有联系，如汽车和轮胎、师傅和徒弟、班级和学生等等。在UML类图中，用实线连接有关联关系的对象所对应的类，在使用Java、C#和C++等编程语言实现关联关系时，通常将一个类的对象作为另一个类的成员变量。在使用类图表示关联关系时可以在关联线上标注角色名，一般使用一个表示两者之间关系的动词或者名词表示角色名（有时该名词为实例对象名），关系的两端代表两种不同的角色，因此在一个关联关系中可以包含两个角色名，角色名不是必须的，可以根据需要增加，其目的是使类之间的关系更加明确。

如在一个登录界面类LoginForm中包含一个JButton类型的注册按钮loginButton，它们之间可以表示为关联关系，代码实现时可以在LoginForm中定义一个名为loginButton的属性对象，其类型为JButton。如图1所示：

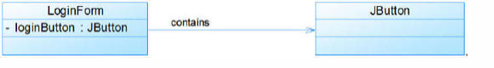


图1 关联关系实例

图1对应的Java代码片段如下：

publicclass LoginForm **{** private **JButton loginButton;** //定义为成员变量

**……**

**}**

publicclass JButton **{**

**……**

**}**

在UML中，关联关系通常又包含如下几种形式：

### **(1) 双向关联**

默认情况下，关联是双向的。例如：顾客(Customer)购买商品(Product)并拥有商品，反之，卖出的商品总有某个顾客与之相关联。因此，Customer类和Product类之间具有双向关联关系，如图2所示：



图2 双向关联实例

图2对应的Java代码片段如下：

publicclass Customer **{** private **Product[] products;**

**……**

**}**

publicclass Product **{** private **Customer customer;**

**……**

**}**

### **(2) 单向关联**

类的关联关系也可以是单向的，单向关联用带箭头的实线表示。例如：顾客(Customer)拥有地址(Address)，则Customer类与Address类具有单向关联关系，如图3所示：



图3 单向关联实例

对应的Java代码片段如下：

publicclass Customer **{** private **Address address;**

**……**

**}**

publicclass Address **{**

**……**

**}**

### **(3) 自关联**

在系统中可能会存在一些类的属性对象类型为该类本身，这种特殊的关联关系称为自关联。例如：一个节点类(Node)

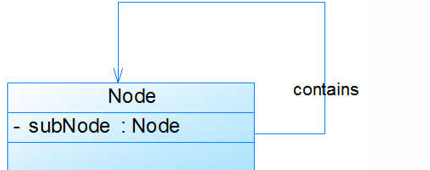
的成员又是节点Node类型的对象，如图4所示：

图4 自关联实例

图4对应的Java代码片段如下：

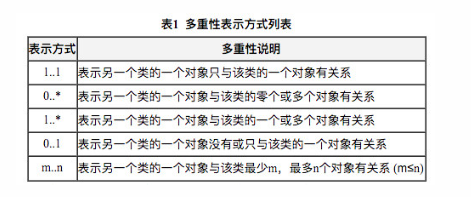
publicclass Node **{** private **Node subNode;**

**……**

**}**

### **(4) 多重性关联**

多重性关联关系又称为重数性(Multiplicity)关联关系，表示两个关联对象在数量上的对应关系。在UML中，对象之间的多重性可以直接在关联直线上用一个数字或一个数字范围表示。

对象之间可以存在多种多重性关联关系，常见的多重性表示方式如表1所示：

例如：一个界面(Form)可以拥有零个或多个按钮(Button)，但是一个按钮只能属于一个界面，因此，一个Form类的对象可以与零个或多个Button类的对象相关联，但一个Button类的对象只能与一个Form类的对象关联，如图5所示：

图5 多重性关联实例

图5对应的Java代码片段如下：

publicclass Form **{** private **Button[] buttons;** //定义一个集合对象

**……**

**}**

publicclass Button **{**

**……**

**}**

### **(5) 聚合关系**

聚合(Aggregation)关系表示整体与部分的关系。在聚合关系中，成员对象是整体对象的一部分，但是成员对象可以脱离整体对象独立存在。在UML中，聚合关系用带空心菱形的直线表示。例如：汽车发动机(Engine)是汽车(Car)的组成部分，但是汽车发动机可以独立存在，因此，汽车和发动机是聚合关系，如图6所示：

### **6) 组合关系**

组合(Composition)关系也表示类之间整体和部分的关系，但是在组合关系中整体对象可以控制成员对象的生命周期，一旦整体对象不存在，成员对象也将不存在，成员对象与整体对象之间具有同生共死的关系。在UML中，组合关系用带实心菱形的直线表示。例如：人的头(Head)与嘴巴(Mouth)，嘴巴是头的组成部分之一，而且如果头没了，嘴巴也就没了，因此头和嘴巴是组合关系，如图7所示：

图7 组合关系实例

在代码实现组合关系时，通常在整体类的构造方法中直接实例化成员类，图7对应的Java代码片段如下：

publicclass Head **{**

private **Mouth mouth;**

public Head() **{**

**mouth =** new **Mouth();** //实例化成员类

**}**

**……**

**}**

publicclass Mouth **{**

**……**

**}**

V7VZ%@DMCVO`$C_RGWJYY_0