# UML 建模简介

#### 类、接口和类图

1. 类

类（Class）是指具有相同属性，方法和关系的对象的抽象，它封装了数据和行为，是面向对象程序设计（OOP）的基础，具有封装性、继承性和多态性等三大特性。在UML 中，类:使用包含类名，属性和操作且带有分隔线的矩形来表示。

1. 类名（Name）是一串字符串，例如Student。
2. 属性（Attribute）是指类的特性，即类的成员变量。UML按下格式表示:

**[可见性]属性名:类名[=默认值]**

例如:-name:String 🡪 private String name;

注意:可见性表示属性对类的元素是否可见，包括公有（Public）、私有（Private）、受保护的（Protected）和朋友（Friendly）4种，在类图中分别用+、-、#、~ 表示。

在Java 中权限修饰符的访问域如下:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | UML符号 | 本类中 | 子类中 | 同包类中 | 其他类中 |
| public | + | 可以 | 可以 | 可以 | 可以 |
| protected | # | 可以 | 可以 | 可以 | 不可以 |
| 默认（default） | ~ | 可以 | 同包子类可以 | 可以 | 不可以 |
| private | - | 可以 | 不可以 | 不可以 | 不可以 |

1. 操作（Operations）是类的任意一个实例对象都可以使用的行为，是类的成员方法。UML如下图所示:

**[可见性]名称(参数列表)[:返回类型]**

例如: +display():void. 🡪 public void display(){}

学生

Student

- no:long

-name:String

-school:String

-totalScore:float

+display():void

public class Student{

private long no;

private String name;

private String school;

private totalScore float;

public void display(){}

}

1. 接口

接口（Interface）是一种特殊的类，它具有类的结构但不可被实例化，只可被子类实现。它包含抽象操作，但不包含属性。他描述了类或组件对外可见的动作。在UML 中，接口使用一个带有名称的小圈来进行表示。

如:

Graph

+getArea():double

+getPerimeter():double

1. 类图

类图（ClassDiagram）是用来描述系统中的类、接口协作以及它们之间的静态结构模型。它主要用于描述软件系统的结构化设计，帮助人们简化对软件系统的理解，它是系统分析与设计阶段的重要产物，也是系统编码与测试的重要模型依据。

类图中的类可以通过某种编程语言直接实现。类图在软件系统开发的整个生命周期都是有效的，它是面向对象系统的建模中最常见的图。下图所示是计算长方形和圆心的周长与面积的类图，图形接口有计算面积和周长的抽象方法，长方形和圆形实现这两个方法供访问类调用。

访问对象

+calculate(Graph tx):void

Graph

+getAre():double

+getPerimeter():double

原形

Circular

-radius:double

+getAre():double

+getPerimeter():double

长方形

Rectangle

-length:double

-width:double

+getArea():double

+getPerimeter():double

#### 类之间的关系

在软件系统中，类不是孤立存在的。类与类之间存在各种关系。根据类与类之间的耦合度从弱到强排列，UML 中的类图有以下几种关系: 依赖关系、关联关系、聚合关系、组合关系、泛化关系和实现关系。其中泛化和实现的耦合度相等，它们是最前的。

1. 依赖关系

依赖关系（Dependency）关系是一种使用关系，它是对象之间耦合度最弱的种关联。在代码中，某个类的方法通过局部变量、方法的参数或者对静态方法的调用来访问另一个类（被依赖类）中的某些方法来完成一些职责。

在 UML 类图中，依赖关系使用带箭头的虚线来表示，箭头从使用类指向被依赖的类。如图下:人与手机的关系图，人通过手机的语音传送方法打电话。

手机

MobilePhone

+transfer():void

人

Person

-name:String

+call(MobilePhone mp):void

public void call (MobilePhone map) {

mp.transfer();

}

1. 关联关系

关联（Assoclation） 关系是对象之间的一种引用关系，用于表示一类对象与另一类对象之间的联系，如老师和学生、师傅和徒弟、丈夫和妻子等。关联关系是类与类之间最常用的一种关系，分为一般关联关系、聚合关系 和 组合关联关系。

1. 关联是双向的，也可以是单项的。在UML 类图中，双向的关联关系可以用带两个箭头或者没有箭头的实线来表示，单向的关联用带一个箭头实线来表示，箭头从来使用类指向被关联的类。也可以在关联线的两端标注角色，代表两种不同的角色。

在代码中通常将一个类的对象，作为另一个类的成员变量来实现联系。如下图是老师和学生的关联关系。每个老师可以教多个学生，每个学生可以向多个老师学习，他们是双关联关系。

教师

Teacher

-name:String

+teaching():void

老师

Teacher

-name:String

-stus:List<Student>

+teaching():void

teaches to🡪

教学者 学习者

1. 聚合关系

聚合（Aggregation）关系是关联关系的一种，是强关联关系，是整体和部分之间的关系, 是has-a 的关系。

聚合关系也是通过成员对象来实现的, 其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而独立存在的，例如学校和老师的关系，学校包含老师，但如果学校停办了老师依然存在。

在UML 类图中，聚合关系可以用带空心的菱形的实线来表示，菱形指向整体，如下图:

教师

Teacher

-name:String

+teaching():void

大学

University

-teachers:List<Teacher>

1. 组合关系

组合（Composition）关系也是关联关系的的一种，也表示类之间的整体与部分的关系，但它是一种更强烈的聚合关系，是cxmtains-a 关系。在组合关系中，整体对象可以控制部分对象的生命周期，一旦整体对象不存在，部分对象不能脱离整体对象存在。例如头和嘴的关系，没有了头，嘴也就不存在了。

在UML 在类图中，组合关系用带实心菱形的实线来表示，菱形指向整体。

嘴

Mouth

+eat():void

头

Head

-mouth:Mouth

黑色

1. 泛化关系

泛化关系（Generalization）关系是对象之间耦合度最大的一种关系，表示一般特殊的关系，是父类与子类之间的关系，是一种继承关系，is-a 的关系。

在UML 中 类图中，泛化关系用带空心的三角箭头的实线来表示，箭头从子类指向父类，在代码是实线时，使用面向对象的继承机制来实现泛化关系。

例如，Student 类 和 Teacher 类都是Person 类的子类，如图下:

人

Person

-name:String

-age:int

+speak():void

老师

Teacher

-teacherNo:long

+teaching():void

学生

Student

-studentNo:long

+study():void

1. 实现关系

实现（Realiztion） 关系是接口与实现类之间的关系。在这种关系中类实现了接口，类中的操作实现了接口中的所有抽象操作。在UML 类图中，实现关系使用带空心三角头虚线来表示，箭头从实现类指向接口。

列如:

车辆

Vehicle

+move():void

船

Ship

+move():void

汽车

Car

+move(): void