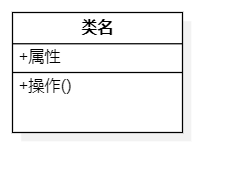
# 第5章 类图和对象图

## 5.1 类图和对象图概述

### 5.1.1 类图概述

类是对一组具有相同属性、操作、关系和语义的对象的抽象。主要包括名称部分（Name）、属性部分（Attribute）和操作部分（Operation）。在UML中类用一个矩形框表示，它包含三个区域，最上面是类名、中间是类的属性、最下面是类的方法。



1. 名称

每个类都必须有一个能和其他类进行区分的名称，类的名称部分是不能省略的，其他部分可以省略。名称（Name）是一个文本串，类的命名要求为由字符、数字、下划线组成的唯一的字符串即可。表示方法有两种：（1）简单名：一个单独的名称。（2）全名：也称为路径名，就是在类名前面加上包的名称，例如Business：Account。

2. 属性

属性描述了类在软件系统中代表的事物（对象）所具备的特性。类可以有任意数目的属性，也可以没有属性。类如果有属性，则每一个属性都必须有一个名字，另外还可以有其他的描述信息，如可见性、数据类型、默认值等。

在UML中，类属性的语法为：

【可见性】属性名【：类型】【=初始值】【{属性字符串}】

（1）可见性：类中属性的可见性主要包括公有（Public）、私有（Private）和受保护(Protected)。在UML中，用“+“表达公用类型，用”-“表达私有类型，而用”#“表示受保护类型。UML的类中不存在默认的可见性，如果没有显示任何一种符号，就表示没有定义该属性的可见性。

（2）属性名：每个属性都必须有一个名字以区别于类中的其他属性，是类的一个特性。属性名由描述所属类的特性的名词或名词短语组成。按照UML的约定，单字属性名小写。如果属性名包含多个单词，这些单词要合并，且除了第一个单词外的其余单词首字母均大写。

（3）类型：说明属性的数据类型。在类的图标里，可以指定每个属性值的类型。指明类型时，需要在属性值后面加上类型名，中间用冒号隔开。还可以为属性指定一个默认值。

（4）初始值：为了保护系统的完整性，防止漏掉取值或非法的值破坏系统的完整性，可以设定属性的初始值。

（5）属性字符串：属性字符串用来指定关于属性的其他信息，例如某个属性应该是永久的。任何希望添加在属性定义字符串值但又没有合适地方可以加入的规则，都可以放在属性字符串里。

3. 操作

操作是对类的对象所能做的事务的一个抽象。一个类可以有任意数量的操作或根本没有操作。类如果有操作，则每一个操作也都有一个名字，其他可选的信息包括可见性、参数的名字、参数的类型、参数默认值和操作的返回值的类型等。

在UML中，类操作的语法为：

【可见性】操作名【（参数表）】【：返回类型】【{属性字符串}】

（1）可见性：类中操作的可见性主要包括公有（Public）、私有（Private）、受保护(Protected)和包内公有（Package）。在UML中，用“+“表达公用类型，用”-“表达私有类型，而用”#“表示受保护类型，包内公有类用”~“表示。

（2）操作名：用来描述所属类的行为的动词或动词短语。

（3）参数表：一些按顺序排列的属性定义了操作的输入，是可选的。参数的定义方式：”名称:类型“。若存在多个参数，将各个参数用逗号隔开。参数可以具有默认值。

（4）返回类型：可选的

（5）属性字符串：在操作的定义中加入一些除了预定义元素之外的信息。

4. 职责

在操作列表框下面的区域，可以用来说明类的职责，类用来做什么或说明另一个类的信息。类的职责可以是一个短语或一个句子。在UML中，把职责列在类底部的分隔栏中。

5. 约束

说明类的职责是消除二义性的一种非形式化的方法，形式化的方法是使用约束。约束制定了该类所要满足的一个或多个规则。在UML中，约束是用{}的格式写在类的边上，指定个别属性的取值范围。

### 5.1.2 对象图概述

1.什么是对象

对象指的是一个单独的、可确认的物体、单元或实体，它可以是具体的也可以是抽象的。在问题领域里有确切的定义的角色。即对象是边界非常清楚的任何事物。一个对象通常包括以下几个部分：

标识（名字）：为了将一个对象与其他的对象区分开，通常会给对象起一个“标识”，也就是对象名。

状态（属性）：对象的状态包括对象的所有属性（通常是静态的）和这些属性的当前值（通常是动态的）。

行为（方法，事件）：没有一个对象是孤立存在的，对象可以被操作，也可以操作别的对象。而行为就是一个对象根据它的状态改变和消息传送所采取的行动和所做出的反应。

对象和类的区别如下：

（1）对象是一个存在于时间和空间中的具体的实体，而类仅代表一个抽象，抽象出对象的本质。

（2）类是共享一个公共结构和一个公共行为的对象的集合。

（3）类是静态的，对象是动态的。类是一般化，对象是个性化。类是定义，对象是实例。类是抽象，对象是具体。

2. 对象图

对象图（Object Diagram）描述的是参与交互的各个对象在交互过程中某一时刻的状态。对象图可以被看做是类图在某一时刻的实例。在UML中，对象图使用的是与类图相同的符号和关系，因为对象就是类的实例。

对象图主要包括以下几个部分：

对象名：由于对象是一个类的实例，因此其名称的格式是“对象名：类名”，这两个部分是可选的，但如果是包含类名，则必须加上“：”，为了与类名区别，还必须加上下划线。

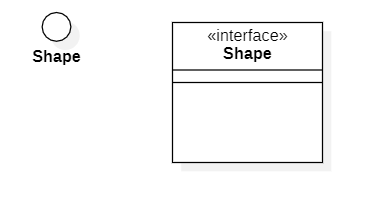
属性：由于对象是一个具体的事物，因此所有的属性值都已经确定，因此通常在属性的后面列出其值。



### 5.1.3 接口

接口（Interface）是描述类的部分行为的一组操作，它也是一个类提供给另一个类的一组操作。通常接口被描述为抽象操作，也就是只用标识（返回值、操作名称、参数表）说明它的行为，而真正实现部分放在使用该接口的对象中，也就是该接口只负责定义操作而不具体地实现。

接口的模型表示法和类大致相同，都是用一个矩形图标来代表。和类的不同之处在于，接口只是一组操作，没有属性。在UML图形上，接口的表示和类图的表示类似，只是在最上面的一层类名前加描述<<interface>>，或是简化表示，用一个圆圈表示。



### 5.1.4 抽象类

抽象类是包含一种或多种抽象方法的类，它本身不需要构造实例。定义抽象类后，其他类可以对它进行扩充并且通过实现其中的抽象方法，使抽象类具体化。在UML中抽象类的图形表示和类图一样，只是在最上面一层的类名前加描述<<abstract>>或是在类的属性描述上设置该类为抽象类，抽象类的类名用斜体表示。

抽象类和接口的区别：

（1）抽象类可以包含某些实现代码，但接口没有任何实现部分。

（2）抽象类可以包含属性而接口没有。

（3）接口可以被结构继承，但抽象类不行。

（4）抽象类可以有构造函数和析构函数，而接口没有。

（5）抽象类可以继承其他类和接口而接口仅能继承接口。

（6）接口支持多继承而抽象类只能支持单继承。

## 5.2 类之间的关系

### 5.2.1 依赖关系

依赖关系（Dependency）表示两个或多个模型元素之间语义上的关系。它表示了这样的一种情形，对于一个元素（服务提供者）的某些改变可能会影响或提供消息给其他元素（使用者），即使用者以某种形式依赖于其他类元。在UML上，把依赖画成一条有向的虚线，指向被依赖的事物。当要指明一个事物使用另一个事物时，就使用依赖。

UML定义了4种基本依赖，分别是使用依赖、抽象依赖、授权依赖和绑定依赖。

（1）使用依赖。使用依赖是一种非常直接的关系，它通常表示使用者使用服务提供者所提供服务实现它的行为。有五种类型的使用依赖：使用<<use>>，调用<<call>>，参数<<parameter>>，发送<<send>>,实例化<<instantiate>>。

（2）抽象依赖。抽象依赖建模表示使用者和提供者之间的关系。它依赖于在不同抽象层次上的事物。共有三种类型的抽象依赖：追踪<<trace>>,精化<<refine>>，派生<<derive>>。

（3）授权依赖。授权依赖表达了一个事物访问另一个事物的能力。提供者可以规定使用者的权限，这是提供者控制和限制对其内容访问的方法。主要有三种类型的授权依赖：访问<<access>>，导入<<import>>，友元<<friend>>。

（4）绑定依赖。它表明对目标模板使用给定的实例参数进行实例化。当对模板类的细节模板时，要使用绑定<<bind>>。

### 5.2.2 泛化关系

泛化关系（Generalization）是一种存在于一般元素和特殊元素之间的分类关系，它只使用在类型上，而不是实例上。在类中，一般元素被称为超类或父类，而特殊元素被称为子类。在UML中，泛化关系用一条从子类指向父类的空心三角箭头表示。

### 5.2.3 关联关系

关联关系（Association）是一种结构关系，它指明一个事物的对象与另一个事物的对象之间的联系。也就是说，关联描述了系统中对象或实例之间的离散连接。给定一个连接两个类的关联，可以从一个类的对象联系到另一个类的对象。关联的两端都连接到一个类在理论上是合法的。在UML图形中，关联关系用一条连接两个类的实线表示。

在UML中，有可应用到关联的基本修饰。

1.关联名即名称

名称用来描述关联的性质，通常使用一个动词或动词短语来命名关联，因为它表面源对象正在目标对象上执行的动作。为了消除名称含义的歧义，UML提供了一个指引读者名称方向的三角形，并给名称一个方向。

2.角色

当一个类处于关联的某一端时，该类就在这个关系中扮演了一个特定的角色。它呈现的是对另一端的职责。可以显示地命名一个类在关联中所承担的角色。关联端点承担的角色称为端点名或角色名，端点名称是名词或名词短语，以解释对象是如何参与关联的。

3. 多重性

关联表示了对象间的结构关系。有时在建模时需要说明一个关联的实例中有多少互连接的对象。即多重性是指有多少对象可以参与该关联。可以表达一个取值范围、特定值、无限定的范围或一组离散值。格式：“minimum .. maximum”均为int型。

UML使用星号\*来代表许多和多个。在一种语义中，两点代表Or（或）关系。在另一种语义中，Or用逗号表示。

4. 聚合

聚合关系表示整体和部分关系的关联。聚合描述了“has – a“的关系，意思是整体对象拥有部分对象。实质上聚合就是一种特殊的关联。

在UML中，聚合被表示为在整体的一端用一个空心菱形修饰的简单关联。

5. 组合关系

组合关系是聚合关系中的一种特殊情况，是更强形式的聚合，又被称为强聚合。在组合中，成员对象的生命周期取决于聚合的生命周期，聚合不仅控制着成员对象的行为，而且控制成员对象的创建和撤销。在组合式聚合中，一个对象在一个时间内只能是一个组合的一部分。

6. 导航性

导航性描述的是一个对象通过链（关联的实例）进行导航访问另一个对象，即对一个关联端点设置导航意味着本端的对象可以被另一端的对象访问。可以在箭头上加上导航方向。

导航共分为以下两类：

（1）单向关联（Unidirection Association）：只在一个方向上可以导航的关联，用一条带箭头的实线来表示。

（2）双向关联（Bidirection Association）：在两个方向上都可以导航的关联，用一条没有箭头的实线来表示。

7. 关联类

关联类是同时具有类和关系特征的模型元素，一个关联类可以看作是一个拥有类特性的关联，也可以看作是一个拥有关联特性的类。关联类的可视化表示方式与一般的类相同，但是要用一条虚线把关联类和对应的关联类连接起来。关联类可以和其他类关联。

8. 约束

由于两个类之间的一个关联可能应对有一个规则。可以通过关联线附近加注一个约束来说明这个规则。

### 5.2.4 实现关系

实现关系将不同语义层内的元素连接起来，通常建立在不同的模型内。

实现关系通常在两种情况下被使用：在接口和实现该接口的类之间。在用例及实现该用例之间的协作之间。

在UML中，实现关系符号与泛化关系的符号类似，用一个带指向接口的空心三角箭头的虚线表示。

## 5.3 类图的建模技术及应用

1.概念层类图

概念层的类图描述的是现实世界中对问题领域的概念理解，类图中表达的类与现实世界的问题有着明显的对应关系，类之间的关系也与问题领域中实际事物的关系有着明显的对应关系。

在概念层上，类图着重于对问题领域的概念化理解，而不是实现。

2.说明层类图

在说明层阶段主要考虑的是类的接口部分，而不是实现部分。这个接口可能因为实现环境、运行特性等有多种不同的实现。

3. 实现层类图

实现层类图是真正要考虑类的实现问题的阶段。提供实现的细节，在实现层阶段的类的概念才是真正严格意义上的类。它揭示了软件实体的构成情况。

建立类图的步骤如下：

（1）研究分析问题领域，对系统进行需求分析，确定系统需求。

（2）确定系统中的类，明确类的含义和职责以及确定类的属性和操作。

（3）最后确定类之间的关系。

对系统进行建模时，对类的识别是一个需要大量技巧的工作，寻找类的一些方法包括：名词识别法；根据用例描述确定类；使用CRC分析法；对领域进行分析或利用已有领域分析结果得到类；利用RUP中如何在分析和设计中寻找类的步骤。

使用类图对系统进行建模时需注意以下几点：

（1）应该从简单概念开始创建类图，比如类的关系等。

（2）在项目的不同开发阶段，使用不同的观点来创建类图。

（3）不要为每一个事物都画一个模型，应该把精力放在关键的领域。使用类图的最大风险是过早地着眼于实现的细节。

问题：

（1）简述题：简述对象和类的区别。

答案：1）对象是一个存在于时间和空间中的具体的实体，而类仅代表一个抽象，抽象出对象的本质。

2）类是共享一个公共结构和一个公共行为的对象的集合。

3）类是静态的，对象是动态的。类是一般化，对象是个性化。类是定义，对象是实例。类是抽象，对象是具体。

（2）选择题：UML中定义了4种基本依赖，分别是使用依赖、抽象依赖、授权依赖和（）依赖。

A.参数 B.派生 C.访问 D.绑定

答案：D