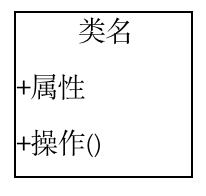
# 类图和对象图

## 类图和对象图概述

**1.1.1 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) o 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，如图所示



* 关于名称：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，其他组成部分可以省略。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。表示方法有以下两种。

1. 简单名:如Account,它只是一个单独的名称。
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

* 关于属性：属性描述了类在软件系统中代表的事物(即对象)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}]
* 类中属性的可见性主要包括公有(Public).私有(Private)和受保护 (Protected),分别由’+’, ‘-’， ‘#’表示。
* 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其余单词的首字母要大写。
* 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。可能的类型包括字符串(string).浮点型(float).整型(int)和布尔型(boolean)(以及其他的枚举类型)。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。
* 初始值：为了保护系统的完整性，防止漏掉取值或被非法的值破坏系统的完整性，可以设定属性的初始值。
* 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。
* 关于操作：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者根本没有操作。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：

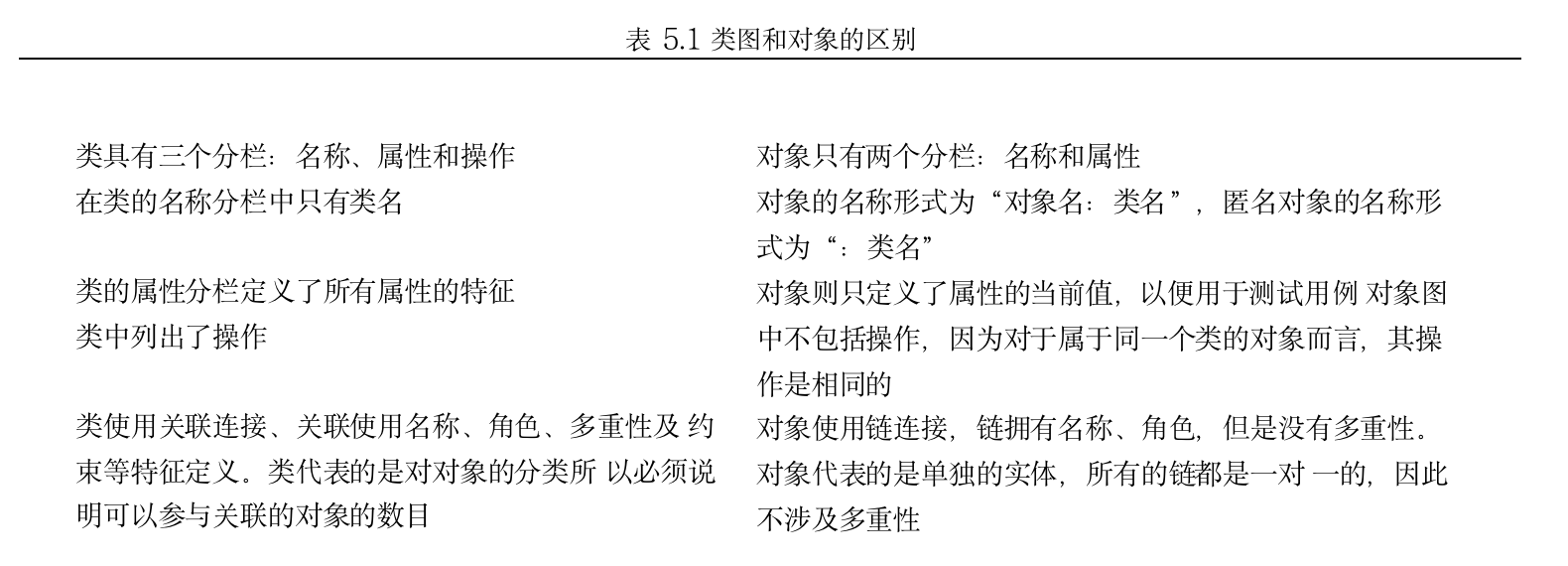
[可见性] 操作名 [{参数表}] [:返回类型] [{属性字符串}]

**1.1.2 对象图概述**

* 类图是描述类、接口、协作及它们之间关系的图，用来显示系统中各个类的静态结构。 类图是定义其他图的基础,在类图基础上，可以使用状态机图、通信图、构件图和配置图等进一步描述系统其他方面的特性。由此引出类和对象的对比。
* 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定义的角色。换句话说，对象是边界非常清楚的任何事物。一个对象 通常包含以下几部分。

1. 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”。
2. 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）。
3. 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釆取的行动和所做出的反应。

* 对象图概述：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。对象图可以被看作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部分：对象名、属性。
* 对象图和类图很相似但也有区别：



**1.1.3 接口**

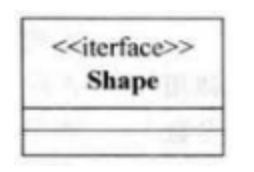
* 接口(Interface)是描述类的部分行为的一组操作，它也是一个类提供给另一个类的一组操作。通常接口被描述为抽象操作，也就是只用标识(返回值、操作名称、参数表)说明它 的行为，而真正实现部分放在使用该接口的对象中，也就是说接口只负责定义操作而不具体地实现。
* 接口的模型表示法和类大致相同，都是用一个矩形图标来代表。和类的不同之处在于，接口只是一组操作，没有属性。在UML图形上，接口的表示和类图的表示类似，只是在最上面的一层类名前加描述，或是简化表示，用一个圆圈表示，如下图。（书中的图其实把interface都打错了）

图5.5接口

**1.1.4 抽象类**

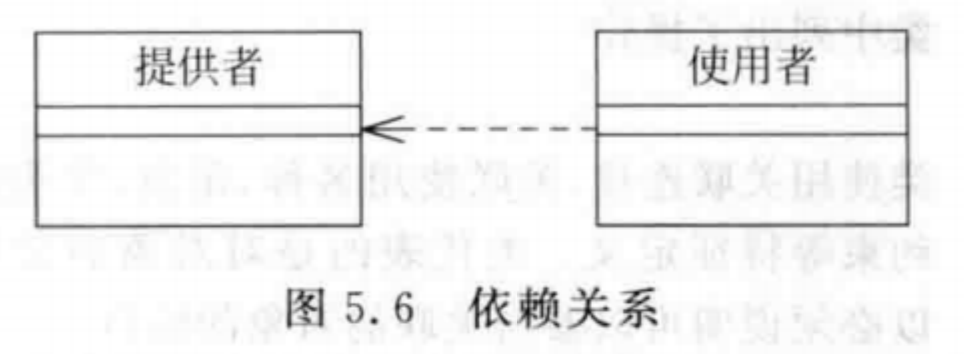
抽象类是包含一种或多种抽象方法的类，它本身不需要构造实例。定义抽象类后，其他类可以对它进行扩充并且通过实现其中的抽象方法，使抽象类具体化。在UML中抽象类的图形表示和类图一样，只是在最上面一层的类名前加描述<<abstract>＞或是在类的属性描述上设置该类为抽象类，抽象类的类名用斜体表示。

## 类之间的关系

关系是指事物之间的联系。在面向对象的建模中，类之间最常见的关系有：依赖关系、泛 化关系、关联关系和实现关系。在图形上，把关系画成一条线，并用不同的线来区别关系的种类。

#### 依赖关系

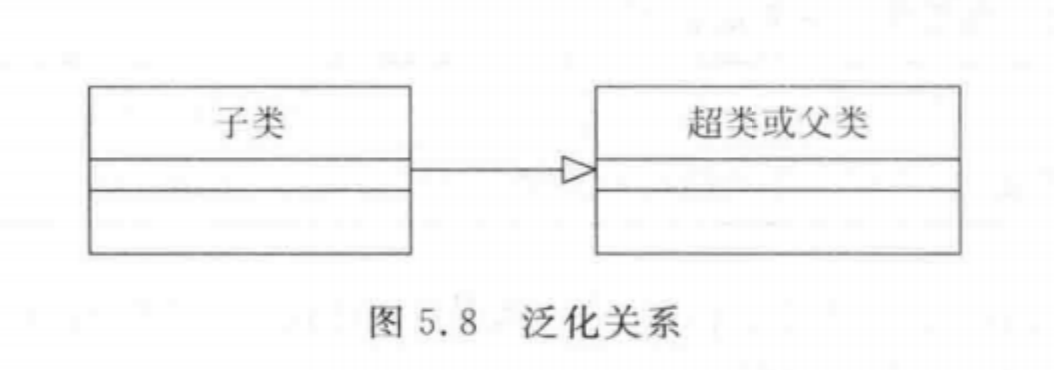
* 依赖关系（Dependency）表示两个或多个模型元素之间语义上的关系。它表示了这样一种情形，对于一个元素（服务提供者）的某些改变可能会影响或提供消息给其他元素（使用者），即使用者以某种形式依赖于其他类元。在UML图形上，把依赖画成一条有向的虚线，指向被依赖的事物。当要指明一个事物使用另一个事物时，就使用依赖。



* 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。

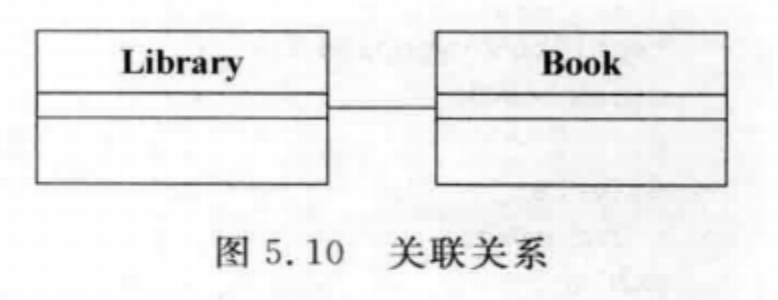
#### 泛化关系

* 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。



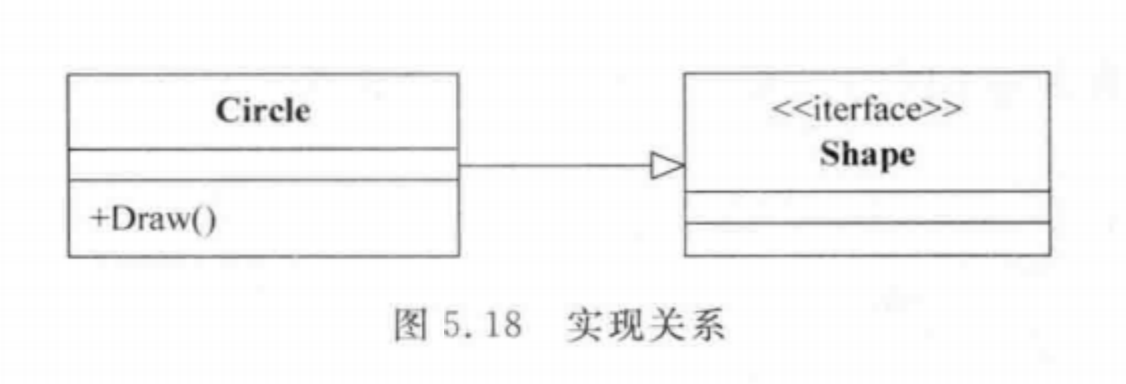
#### 关联关系

* 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也是合法的。在UML图形中，关联关系用一条连接两个类的实线表示。



**1.2.4 实现关系**

* 实现将一种模型元素与另一种模型元素连接起来，比如类和接口。泛化和实现关系都可以将一般描述与具体描述联系起来。泛化将同一语义层上的元素连接起来，并且通常在同一模型内。实现关系则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：在接口与实现该接口的类之间；在用例及实现该用例的协作之间。
* 在UML中，实现关系的符号与泛化关系的符号类似，用一条带指向接口的空心三角箭 头的虚线表示。



## **类图的建模技术**

在软件开发不同阶段使用具有不同的抽象层次的类图，即概念层、说明层和实现层。在UML中，从开始的需求分析到最终的设计类，类图也是围绕着这三个层次的观点来进行建模的。类图建模是先建立概念层到说明层，进而到实现层，随着抽象层次的逐步降低并逐步细化的过程。下面从层次的角度来说明建立类图的过程。

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 界的问题领域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。在概念层类图阶段很少考虑或者几乎不需要考虑类的实现问题。概念层类图中的类和类关系和最终的实现类并不一定有直接和明显的对应关系，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。因此，类名通常都是问题领域中实际事物的名称，并且独立于具体的编程语言。

1. **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。

1. **实现层类图**

* 真正需要考虑类的实现问题是在实现层类图阶段。提供实现的细节，在实现层阶段的类的概念才是真正的严格意义上的类。它揭示了软件 实体的构成情况。说明层的类有助于人们对软件的理解，而实现层的类是最常用的。

## 1.4 提问

1. 【多选】下列属于类之间的关系的是（）
2. 依赖关系 B.泛化关系 C.实现关系 D.关联关系

答案：ABCD

1. 【主观题】在UML类属性中的可见性中，public，private，protected分别用什么符号表示？

‘+’ ‘-’ ‘#’

## 参考资料

1. 杨弘平，《UML2基础、建模与设计教程》