# 第2章

# UML语言体系

#### 本章要点

1.  UML语言组成、事物、关系、图。

2. 采用5种视图对系统体系结构建模。

#### 学习目标

1. 了解UML语言体系的组成和结构。

2. 理解系统体系结构建模的重要性。

## 2.1 UML语言组成

UML语言是一种绘制软件蓝图的设计语言。正如中文语言是由词、语法规则组成的一样，UML语言由构造块(相当于词)、规则(相当于语法规则)和公共机制3个部分构成。其结构如图2-1所示。



图2-1 UML语言的组成

#### 1．构造块

UML构造块又细分为3种：事物、关系和图。

(1) 事物：代表系统中的简单实体(如学生、老师、教师等)。

(2) 关系：代表实体间的联系(如同学关系、同事关系等)。

(3) 图：由多个边将多个顶点连接在一起构成的事物。这里的顶点指事物，边是指关系(如五边形图形由5个顶点和5条边组成)。如图2-2所示为构造块的3种类型。



图2-2 构造块的3种类型

#### 2．规则

规则是指每个构造块必须遵守的语法和表示法。比如，每个构造块必须有名称、范围、可见性、完整性和可执行等属性。构造块应该遵守的规则如图2-3所示。



图2-3 构造块应遵守的规则

(1) 名称：指每个构造块代表的事物应该有一个名字。

(2) 范围：每个构造块代表的事物的作用范围。

(3) 可见性：所有的构造块一定存在于某个包中，因此构造块必然存在访问权限或者级别。正如Java语言中的类、接口都存在可见性一样，UML中的构造块也存在可见性，即访问级别。

(4) 完整性：同一构造块代表的事物在不同模型中的语义必须是一致的。

(5) 可执行：构造块代表的事物在分析、设计、编码阶段都在演化，但是它代表的事物在整个演化过程中必须是合理、有效的。比如，分析阶段的类Dog，当演化到代码中的Dog时，这个Dog必须是合理、有效的。

#### 3．公共机制

公共机制是指每个事物必须遵守的通用规则。可以将公共机制进一步细分为详述、修饰、通用划分及扩展机制。公共机制的组成如图2-4所示。



图2-4 公共机制的组成

下面几节分别对事物、关系、图的概念、表示法进行详细介绍。

## 2.2 构造块：事物

事物是UML语言中最基本的构造块。根据构造块的性质，可将其划分为4类：结构事物、行为事物、分组事物和注释事物。

2.2.1 结构事物

结构事物是UML语言中的名词，通常用来描述系统中的概念或者物理实体，是模型的静态部分。结构事物进一步细分为7种，分别是类(及其对象)、接口、用例、协作、构件、结点和制品。下面分别介绍7种结构事物的概念和表示法。

#### 1．类和对象

类是对具有相同属性、相同操作及相同关系的一组对象的共同特征的抽象，即类是对一组对象共同特征的描述。类是对象的模板，而对象是类的一个实例。

1) 类的表示

在UML中，类用一个长方形框表示。图中把一般长方形框分为3栏，第1栏列出类名，第2栏列出类的属性，第3栏列出类的操作。类名不能省略，属性和操作可以不用列出。

图2-5是Flight类(航线)的UML表示法。第1栏列出类名Flight；第2栏列出类的3个属性，分别是flightNumber、departureTime和flightDuration；第3栏列出类的两个操作，分别是delayFlight()和getArrivalTime()。

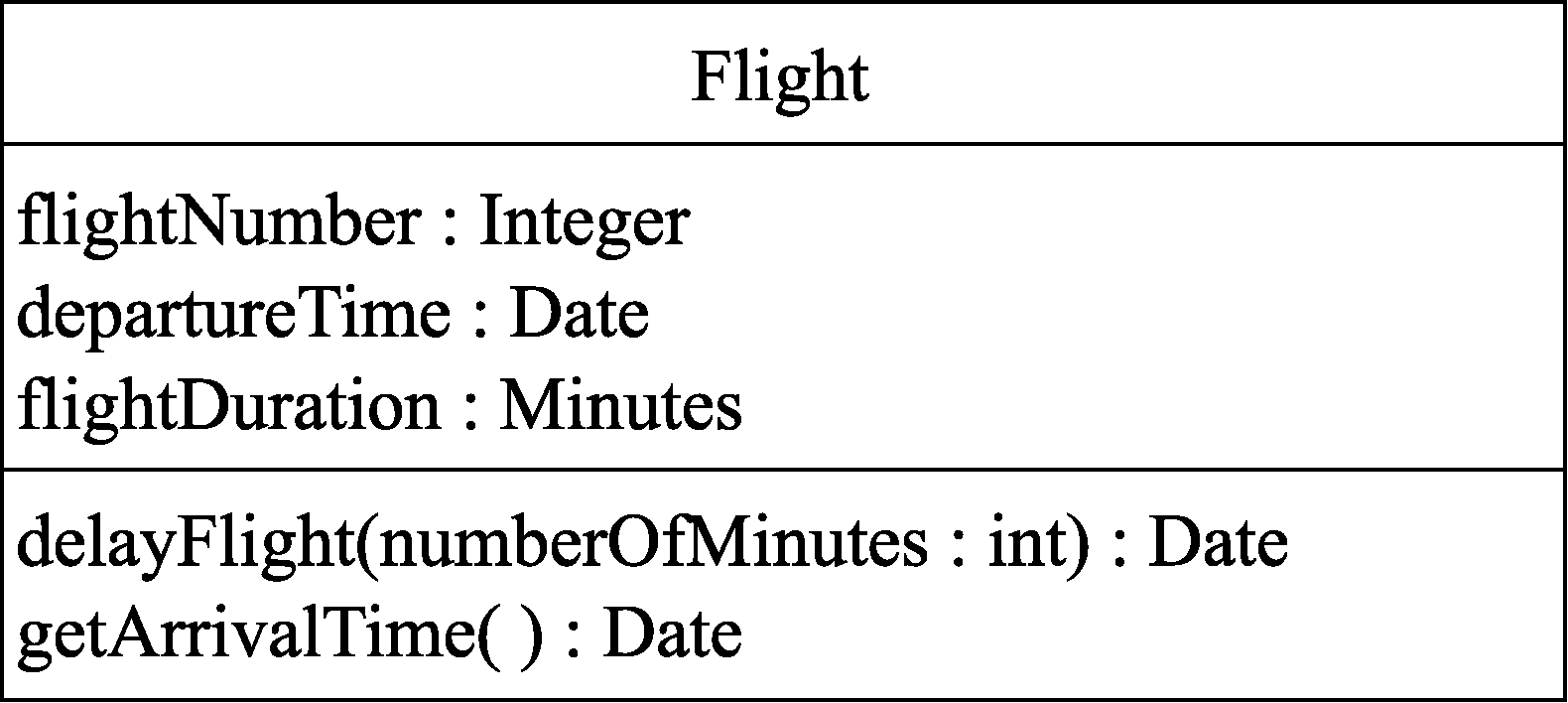


图2-5 Flight类的UML符号

2) 对象的表示

对象也用一个长方形框来表示，只是用“对象名：类名”的格式表示一个对象，并且对象名和类名下面必须带下画线。表示对象时，第2栏和第3栏可以省去。例如，图2-6所示为对象“李世民”的UML表示法。

李世民：People

图2-6 对象“李世民”的UML符号

3) 主动类及其表示

一个对象可以是主动的也可以是被动的。主动对象可以改变自身状态，被动对象只有在接收到消息后才会改变自身的状态。例如，定时器和时钟就是主动对象，它们可以在没有外部事件触发的情况下改变它们自身状态。银行账户就是被动对象，银行账户的属性不会发生变化，除非银行账户接收到一条设置余额(一种用于更新账户余额的操作)的消息，账户才改变状态。

用主动类创建的对象就是主动对象。主动类的表示与一般类相似，只是矩形框用粗线表示而已，如主动类(时钟)的表示方法如图2-7所示。

#### 2．接口

因为访问类、对象或构件是通过其方法来实现的，因此把类、对象、构件的方法集合称为接口。接口向外界声明了类(或构件)能提供的服务。

接口分为供给接口和需求接口两种，供给接口只能向其他类(或构件)提供服务，需求接口(本质上不存在需求接口，它只是一种形式)表示类(或构件)需要用到接口中声明的服务，这些服务由其他类(或构件)提供。比如，一个不会讲英语的人，需要“中英文翻译”接口，在形式上，就把“中英文翻译”这个由其他构件(中英文翻译机)提供的服务作为人的需求接口。本质上，人是没有“中英文翻译”接口的，它只是说明人需要能提供“中英文翻译”的构件。

上述两种接口的表示方法如图2-8所示。

|  |  |
| --- | --- |
| 图2-7 表示主动类(时钟)的符号 | 图2-8 表示接口的UML符号 |

#### 3．用例

把为实现某个目标而执行的一系列有序操作的集合称为场景。例如，客户小刘在柜员机上取款500元的一系列操作构成一个场景；客户小王在柜员机上取款300元的一系列操作也是一个场景。无论多少个客户，他们在柜员机上取款的一系列操作是相似的，即所有客户取款的场景是相似的，只是取款时，输入的密码、取款金额不同。

用例(用例定义，参考第13章)是对一组相似场景的共同操作的抽象。例如，可以用一个动作序列来描述所有取款客户的相同操作。因此，用例的每一次的具体执行就是一个场景，即场景是用例的一个实例，是用例的一次具体执行；用例是对所有相似场景的抽象和描述。用例与场景的关系正如类与对象的关系。

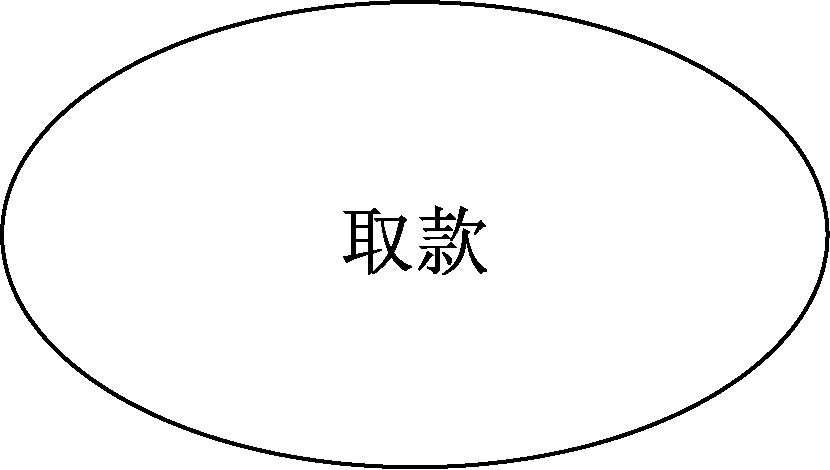


图2-9 用例“取款”的表示方法

在UML中，用例是用一个实线椭圆形来表示的，在椭圆中写上用例名称，如用例“取款”的表示方法如图2-9所示。

#### 4．协作

在系统中，把一组对象之间相互发送消息和接收消息的现象称为交互。把一组对象为了完成某个任务执行的交互现象称为协作。

用例的一次具体执行就是一个场景。在场景中，多个对象之间的相互协作实现了场景，即实现了用例描述的功能。本质上说，协作就是用例的实现。

协作用一个带两个分栏的虚线椭圆形来表示。例如，协作“销售”的表示方法如图2-10所示。

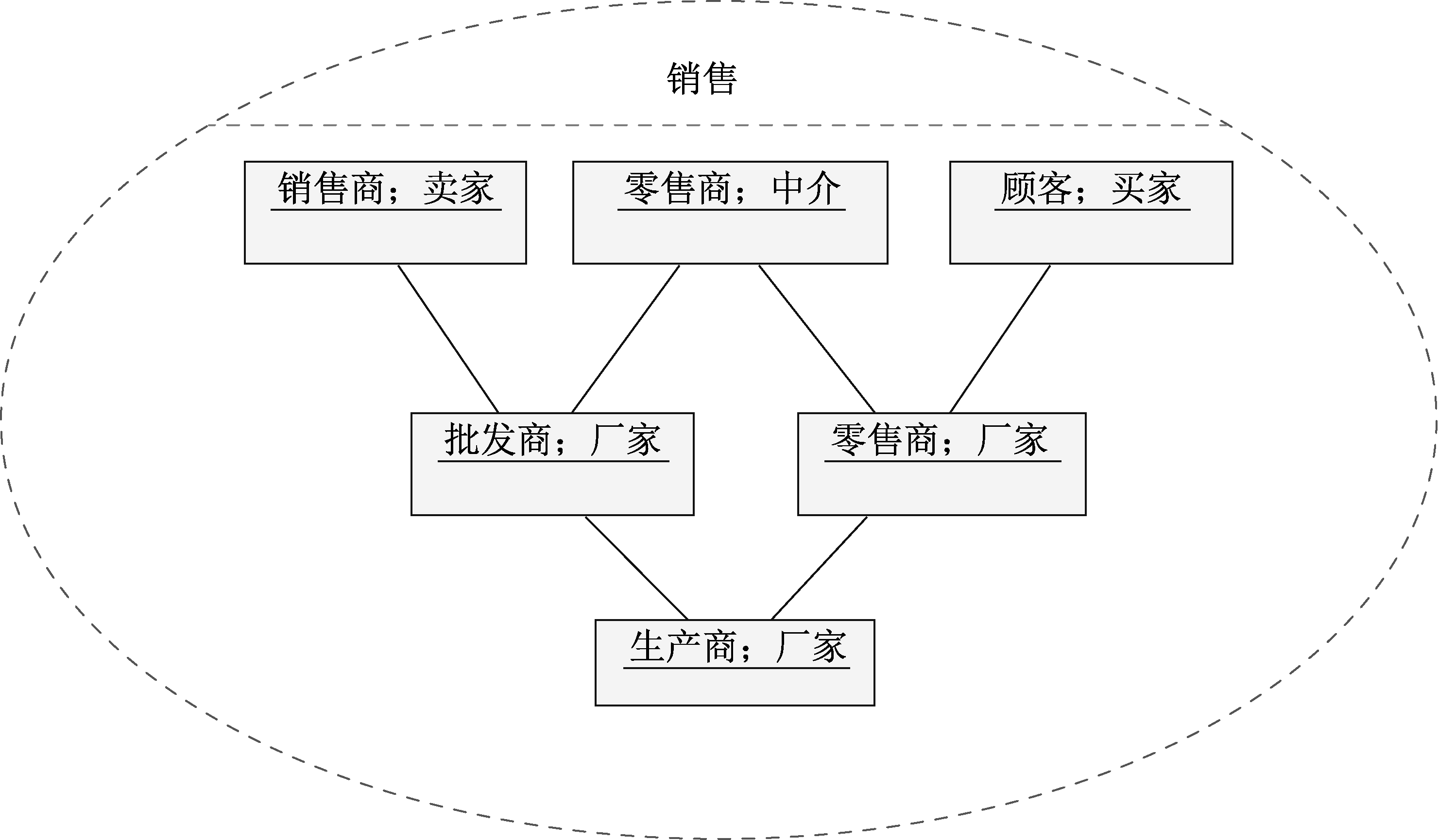


图2-10 协作“销售”的表示

图2-10表示协作“销售”由6个对象相互合作完成销售功能，其主要语义如下。

(1) 生产商生产出产品并以低价售给批发商和零售商，从中获得了利润。

(2) 批发商以比生产商较高的价格出售给销售商或中介，零售商在自己的商店得到 更高利润。

(3) 顾客以较高的价格买到自己想要的商品。

#### 5．构件

构件也称组件，它是指系统设计中的一个相对独立的软件部件，它把功能实现部分隐藏在内部，对外声明了一组接口(包括供给接口和需求接口)。因此，两个具有相同接口的构件可以相互替换。

构件是比“对象”更大的软件部件，如一个COM组件、一个DLL文件、一个JavaBeans及一个执行文件都可以是构件。构件通常采用带有两个小方框的矩形表示，将构件的名字写在方框中，如图2-11所示。

#### 6．结点

结点是指硬件系统中的物理部件，通常具有存储空间或处理能力，如PC、打印机、服务器、显示器等都是结点。在UML中，用一个立方体表示一个结点。例如，结点“显示器”的表示方法如图2-12所示。

#### 7．制品

制品是系统中可以替换的物理部件，包括物理信息(比特流)。制品通常代表对源码信息或者运行时信息的物理打包。一般将制品表示为一个矩形，在制品名称的上方写上关键字《artifact》。

|  |  |
| --- | --- |
| 2-11  图2-11 表示构件的UML符号 | 2-12  图2-12 结点“显示器”的UML符号 |

2.2.2 行为事物

行为事物描述了事物的动态特征，即描述事物之间的交互、交互引起的事物状态变化以及交互引起的活动执行轨迹。行为事物分3种：交互、状态、活动。

#### 1．交互

交互用来表示对象之间的相互作用，即发送和接收消息的现象。

一般用一条有向直线来表示对象间的交互，并在有向直线上方标注消息名称即可，如图2-13所示。

#### 2．状态

事物处于某个特定属性值时的现象称为状态(如某人年龄处在0～12岁时，认为是处于少年状态)。

在UML中，状态用一个圆角矩形表示，状态名称写在圆角矩形框中。例如，手机处在“正在通话”状态的表示方法如图2-14所示。

|  |  |
| --- | --- |
| 2-13  图2-13 表示交互的UML符号 | 图2-14 表示“正在通话”状态的UML符号 |

#### 3．活动

活动描述了事物执行的一系列步骤，它关注的是活动执行的流程。在第8章将详细讲解活动图。

交互强调一组对象之间的接收和发送消息的现象；状态强调的是在一个时间段内对象的生命周期；活动注重的是步骤之间的活动流程，并不关心哪个对象执行了哪个步骤。

2.2.3 分组事物

分组事物是UML语言提供的分割机制，它将系统中的事物分成多个部分进行管理。就像中文语言里，通过段标记把一篇文章分成多个段落一样。在开发大型软件系统时，通常会包含大量的类、接口及用例，为了能有效地对这些类、接口和用例进行分类和管理，就需要对其进行分组。在UML中可通过“包(Package)”来实现这一目标，即通过包对事物进行分组。

表示“包(Package)”的图形符号与Windows中表示文件夹的图形很相似，包的作用与文件夹的作用也很相似。例如，Java语言中的java.awt包，用UML符号表示则如图2-15所示。



图2-15 表示java.awt包的UML符号



图2-16 表示注释的UML符号

2.2.4 注释事物

注释就是对其他事物进行解释、说明。一般用文字进行注释。注释符号用一个右上角折起来的矩形表示，解释的文字就写在矩形框中，如图2-16所示。

## 2.3 构造块：关系

2.2节中介绍了代表事物的构造块，本节将介绍代表事物之间关系的符号。在UML中共定义了24种关系，相应的有24种关系符号，如表2-1所示。

表2-1 UML中的关系及其符号

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 关系 | 关系 细化 | UML中的关系 | UML符号 | 关系 | 关系 细化 | UML中的关系 | UML符号 |
| 抽象 | 派生 | 依赖关系 | 《derive》 | 导入 | 私有 | 依赖关系 | 《access》 |
| 显现 | 《manifest》 | 公有 | 《import》 |
| 实现 | 实现关系 | 虚线加空心三角形 | 信息流 |  | 《flow》 |
| 精化 | 依赖关系 | 《refine》 | 包含并 |  | 《merge》 |
| 跟踪 | 《trace》 | 许可 |  | 《permit》 |
| 关联 |  | 关联关系 | 实线 | 协议符合 |  |  | 未指定 |
| 绑定 |  | 依赖关系 | 《bind》  (参数表) | 替换 |  | 依赖关系 | 《substitu-te》 |
| 部署 |  | 《deploy》 | 使用 | 调用 | 《call》 |
| 扩展 | Extend | 《extend》  (扩展点) | 创建 | 《create》 |
| 扩展 | extension | 扩展关系 | 实线加实心三角形 | 实例化 | 《instanti-ate》 |
| 泛化 |  | 泛化关系 | 实线加空间三角形 | 职责 | 《responsi-bility》 |
| 包含 |  | 依赖关系 | 《include》 | 发送 | 《send》 |

上述有24种关系，在UML中，可以归纳为关联关系、泛化关系、实现关系、依赖关系和扩展关系5种，下面介绍这些关系的表示方法。

#### 1．关联关系

只要两个类之间存在联系，就认为这两个类之间存在关联关系。关联是人们赋予事物之间的联系。实现关系、泛化关系和依赖关系统称为关联关系，只是这些关系内涵更丰富，更明确，更具体。关联关系是对关系的最高层次的抽象，在所有关系中，关联的语义最弱。

在关联关系中有两种比较特殊的关系，它们是聚合关系和组合关系。聚合关系和组合关系能通过Java语言实现，关联关系不能通过Java语言实现，所以，在设计阶段，我们必须把分析阶段的关联关系细化为更具体的关系，如细化为聚合关系、组合关系或者依赖关系等。

1) 关联关系的表示

关联关系是比较抽象的关系，它包含的语义较少；聚合关系和组合关系是更具体的关联关系，它包含的语义更具体，内涵更丰富。在UML中，使用一条实线来表示关联关系，如图2-17所示。



图2-17 表示关联关系的UML符号

2) 聚合关系

聚合(Aggregation)是整体与部分的关系，是一种特殊形式的关联。聚合关系是一种松散的对象间关系——计算机与它的外围设备就是聚合关系。一台计算机(整体)和它的外设(部分)之间松散地结合在一起，这些外设可以与其他计算机共享，即部分可以离开整体而存在。

聚合的表示方法如图2-18(a)所示。其中菱形端表示事物的整体，另一端表示事物的部分。例如，计算机就是整体，外设就是部分。

3) 组合关系

如果发现“部分”的存在是完全依赖于“整体”，那么就应使用“组合”关系来描述。组合关系是一种非常强的对象间关系，就像树和树叶之间的关系，树和它的叶子紧密联系在一起，叶子完全依赖树，它们不能被其他的树所分享，并且当树死去时，叶子也会随之死去——这就是组合。在组合关系中，部分依赖于整体而存在。组合是一种强的聚合关系，它的表示方法如图2-18(b)所示。



(a) 聚合 (b) 组合

图2-18 表示聚合关系和组合关系的UML符号

#### 2．泛化关系

泛化关系描述从特殊事物到一般事物之间的关系，也就是子类到父类之间的关系，或者子接口到父接口的关系。表示泛化关系的符号是从子类指向父类的带空心箭头的实线，其表示方法如图2-19所示。而从父类到子类的关系则是特化关系。



图2-19 表示泛化关系的UML符号

#### 3．实现关系

实现关系是用来规定接口与实现接口的类之间的关系。接口是操作的集合，这些操作声明了类或组件所提供的服务。表示实现关系的符号是从类指向接口的带空心箭头的虚线，其表示方法如图2-20所示。

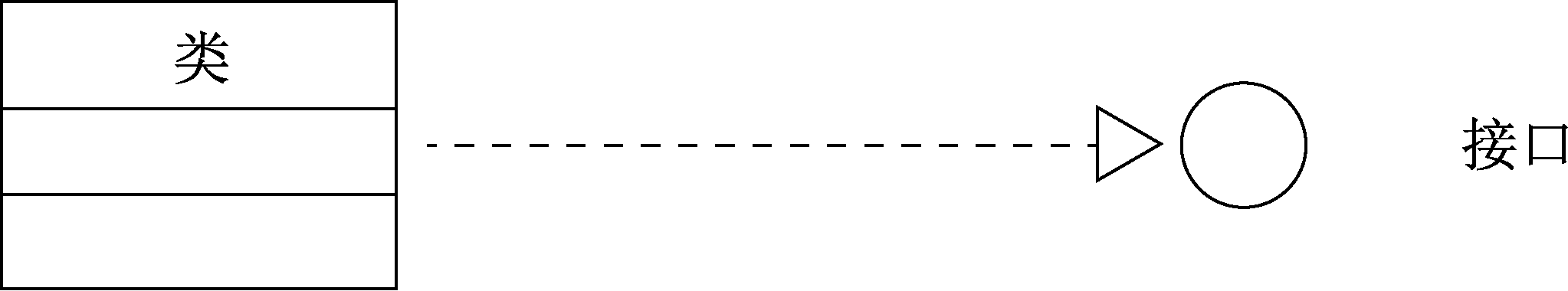


图2-20 表示实现关系的UML符号

#### 4．依赖关系

假设有两个元素X、Y，如果元素X的值发生变化时会引起元素Y值的变化，则称元素Y依赖于元素X。依赖关系的表示如图2-21所示。

2-21

图2-21 表示依赖关系的UML符号

如果两个元素是类，则类间的依赖现象有多种，如一个类向另一个类发送消息，一个类是另一个类的数据成员，一个类是另一个类的某个方法的参数。

从本质上说，聚合、组合、泛化及实现关系都属于依赖关系，但是它们有更特别的语义。

#### 5．扩展关系

在UML中，用一个带箭头的实线表示扩展关系，如图2-22所示。这里的扩展含义是指对一个元类的扩展，即通过扩展元类的语义，获得新的元类。

2-22

图2-22 表示扩展关系的UML符号

## 2.4 构造块：图

UML中的构造块包括3种：事物、关系和图。前面已经介绍了事物、关系，现在介绍图的概念。

图是由一组事物通过关系组织起来的图形，其中，图形的顶点代表事物，图形的弧代表关系。由此可见，图是一种更复杂的构造块。

2.4.1 UML图

UML中的图可分为两大类：结构图和行为图。结构图描绘系统中事物的组成及关系；行为图描述系统中事物间的交互行为。UML图的组成如图2-23所示。



图2-23 UML图的组成

#### 1．结构图

UML 的结构图可用于对系统的静态方面进行可视化、详述、构造和文档化，用于对系统相对稳定的骨架的表示。正如房子的静态方面是由墙、门、窗、管子、电线、通风孔等事物的布局组成一样，软件系统的静态方面是由类、接口、协作、构件和结点等事物的布局组成的。UML结构图是根据事物的主要组成部分来构建模型的。

结构图又分为6种，如图2-24所示。



图2-24 结构图组成

1) 类图

类图是建模时最常用的图，它展示了系统中的静态事物、结构及它们之间的关系。我们常用类图描述系统的逻辑设计和物理设计。

类图包含的主要UML元素：类、接口和协作。

2) 构件图

构件图用于说明软件系统中构件的组成、结构和如何协同工作。当构件实例化时，也实例化了其内部部件的副本。

构建图包含的主要UML元素：构件、端口和连接件。

3) 对象图

对象图可以展示系统中对象的组成和结构，是系统在某一时刻的快照。对象图是类图在某一时刻的快照。

对象图包含的主要UML元素：对象和关联。

4) 部署图

部署图可以展示系统中物理结点的组成、结构和运行时的体系结构。同时也展示了软件部件如何部署在硬件结点之上。部署图展示了硬件和软件系统的体系结构。

部署图包含的主要UML元素：构件和结点。

5) 组合结构图

组合结构图展示类或协作的内部结构及其内部关系。

6) 包图

包图可以展示系统的组成和包之间的依赖关系。包图常用来对系统中的元素进行分组，并用来表示软件的体系结构。

#### 2．行为图

UML的行为图可用于对系统的动态方面进行可视化、详述、构造和文档化，用于表示系统运行时的行为特征，正如房子建好以后，人在房子里走动、呼吸、睡觉，空气在房子里流动一样；软件系统的动态方面也包括了类似方面，如软件系统中流动的信息、信号、数据，软件各个组成部分的状态变化。软件系统中这些变化，是通过行为图来描述的。行为图又细分为8种，如图2-25所示。



图2-25 行为图分类

1) 活动图

活动图显示系统内部的活动控制流程。通常使用活动图描述不同的业务过程。

2) 状态图

状态图显示对象从一种状态迁移到其他状态的转换过程。例如，可以利用状态图描述电话系统中交换机的状态迁移过程。不同的事件触发交换机转移到不同的状态。在 UML 2.0中，状态图被称为状态机图。

3) 协作图

协作图(也称通信图)是一种交互图。协作图突出对象之间的合作，以及交互时每个对象承担的职责。

4) 顺序图

顺序图是一种交互图，它强调系统中对象相互作用时消息的先后顺序。

UML 2.0中又增加了下列几种行为图。

5) 定时图

定时图也是一种交互图，它描述了交互对象的状态转换或与条件变化有关的详细时间 信息。

6) 交互概观图

交互概观图是一种高层视图，用于从总体上显示交互序列之间的控制流。

 注意： 在实际进行系统建模时，几乎没有人会使用到UML标准中定义的所有图。

7) 用例图

用例图描述了系统外部参与者如何使用系统提供的服务，描述了系统中用例之间的关系。用例图把系统的行为组织在一起。

#### 3. 图的作用

在UML 2.0中共定义了13种图。表2-2中列出了这13种图的作用。

表2-2 UML 2.0中的图

|  |  |  |
| --- | --- | --- |
| 图 分 类 | 作 用 | 描 述 |
| 类图 | 描述系统中的类组成和类之间的关系 | 与UML 1.0相同 |
| 对象图 | 描述系统在某个时刻对象的组成和关系 | UML 1.0非正式图 |
| 组合结构图 | 描述复合对象的内部结构 | UML 2.0新增 |
| 构件图 | 描述构件的结构与组成 | 与UML 1.0相同 |
| 部署图 | 描述在系统中各个结点上的构件及其构件之间的关系 | 与UML 1.0相同 |
| 包图 | 描述系统的宏观结构，并用包来表示 | UML中非正式图 |
| 用例图 | 描述用户与系统如何交互及系统提供的服务 | 与UML 1.0相同 |
| 活动图 | 描述活动控制流程及活动结点的转换过程 | 与UML 1.0相同 |
| 状态机图 | 描述对象生命周期内，在外部事件的作用下，对象从一种状态如何转换到另一种状态 | 与UML 1.0相同 |
| 顺序图 | 描述对象之间的交互，重点在于强调消息发送的顺序 | 与UML 1.0相同 |
| 协作图 | 描述对象之间的交互，重点在于强调对象的职责 | UML 1.0中的协作图 |
| 定时图 | 描述对象之间的交互，重点在于描述时间信息 | UML 2.0 新增 |
| 交互概观图 | 是一种顺序图与活动图的混合嫁接 | UML 2.0新增 |

可以用13种图分别构造结构模型(也称为静态模型)和行为模型(也称为动态模型)。

2.4.2 UML视图

图是由事物和关系组成的，是一种大的构造块，它可以用来描述复杂事物。一个系统是由多个事物组成的，如果要从某个角度来描述系统，就要通过视图来实现。多个相关的图构成视图。视图用来描述系统某个方面的特征。

在UML参考手册第2版中，UML图可以构成9种视图，应用在4个领域，如表2-3所示。

表2-3 UML视图

|  |  |  |
| --- | --- | --- |
| 应用领域 | 视 图 | 组成要素(图) |
| 结构领域 | 静态视图 | 类图，对象图 |
| 设计视图 | 复合结构图、协作图、构件图、对象图 |
| 用例视图 | 用例图、顺序图 |
| 动态领域 | 状态视图 | 状态机图 |
| 活动视图 | 活动图 |
| 交互视图 | 顺序图、协作图、时间图、交互概况图 |
| 物理领域 | 部署视图 | 部署图 |
| 模型管理领域 | 模型管理视图 | 包图 |
| 特性描述 | 包图 |

其中，结构领域的视图描述了系统中的成员及其相互关系；动态领域的视图描述了系统随时间变化的行为；物理领域的视图描述了系统的硬件结构和部署在这些硬件上的系统软件；模型管理领域的视图说明了系统的分层组织结构。

## 2.5 UML语言中的公共机制

2.5.1 构造块描述机制

每种语言都有自己的表示风格，为了使模型的风格表示一致，UML规定了模型中构造块的表示风格，UML要求每种构造块的风格保持一致。UML语言对构造块本身的规范如下。

* 名称：代表事物和关系的每个构造块应该有一个名字，即事物、关系和图都应该有一个名字。和任何语言一样，名字即是一个标识符。
* 范围：每个事物都有它的作用范围，相当于程序设计语言中变量的“作用域”。
* 可见性：由于UML元素(构造块)在一个类或包中，因此，所有事物都存在访问级别，即可见性。在UML中，为元素定义了4种可见性，如表2-4所示。

表2-4 UML元素的可见性

|  |  |  |
| --- | --- | --- |
| 元素的可见性 | 规则(假设被访问的元素在包中) | 标准表示法 |
| public | 任一元素若能访问包，就可以访问包中的元素 | + |
| protected | 只有包中的元素或子包才能访问它 | # |
| private | 只有包中的元素才能访问它 | - |
| package | 只有声明在同一个包中的元素才能访问该元素 | ～ |

2.5.2 公共机制

在UML语言中，有4种机制贯穿于整个UML语言，这4种机制是规格描述、修饰、通用划分和扩展机制。

#### 1. 规格描述

在UML语言中，每个元素(构造块)都有一个对应的图形符号，同时，对每个图形符号的语义有一个详细的文字描述，这种对图形符号的语义进行的文字描述称为规格描述，也称为详述。

如图2-26所示，在左边的方框中有3个用图形符号表示的用例，分别是“存款”“取款”“转账”。在右边的方框中，分别对每个图形符号表示的用例进行了详细的文字描述，即规格描述。

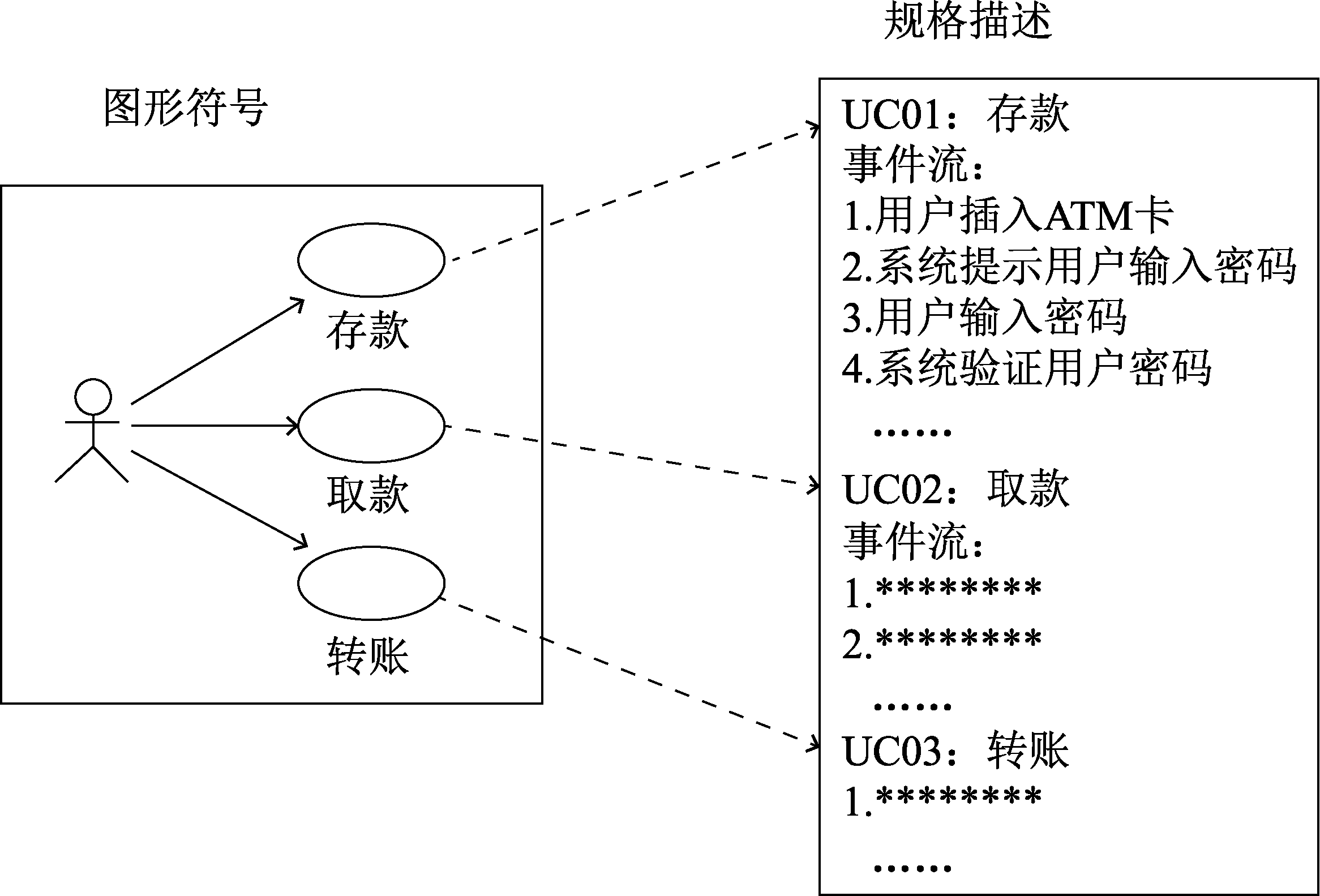


图2-26 图形符号与对应的规格描述

#### 2．修饰

在UML中，每个元素符号对事物的主要方面提供了可视化表示(图形符号)，若想将事物的细节表示出来，则必须对元素符号加以修饰。例如，用斜体字表示抽象类，用“+”“-”符号表示元素的访问级别，这些都是通过修饰符号来表示事物的细节。所谓修饰，就是增加元素符号的内涵，为被修饰的元素提供更多的信息。

#### 3．通用划分

对于面向对象的系统建模来说，通常有3种划分方式，它们是类与对象的划分、接口与实现的分离、类型与角色的分离。

(1) 类与对象的划分：类是对一组对象共同特征的抽象，对象是类的一个实例。UML中，每一个构造块几乎都存在“类-对象”这样的二分法。例如，用例和用例的实例(场景)、构件和构件的实例、结点和结点的实例。

(2) 接口与实现的分离：接口是一种声明、一个合同，是一组方法的集合，而实现则是对一个合同的实施，它完成了合同要求的职责。UML中，每一个构造块几乎都存在“接口-实现”这样的二分法。例如，用例和协作(实现用例的协作)、操作和实现操作的方法。

在UML中，用例就是对事物功能的抽象描述；而协作则是实现用例的功能；操作名是声明服务的，而方法体则是实现服务的。因此，用例与协作、操作名与方法之间的关系就是接口和实现的关系。

(3) 类型与角色的分离：类型声明了实体的种类，而角色描述了实体在语境中承担的职责。UML中，每一个构造块几乎都存在“类型-角色”这样的二分法。每一个实体同时具有两重性，既属于一种类型，又具有角色含义。

#### 4．扩展机制

由于UML中定义的构造块无法将现实世界中所有事物的特征表示出来，因此需要通过一些方法扩展构造块。UML提供的扩展机制有3种：构造型、标记值和约束。

1) 构造型

构造型就是指分析师自己创建新的UML元素(构造块)，给这种新的元素赋予特别的含义。例如，分析师创建一个元素《Interrupt》，赋予元素《Interrupt》的语义是“中断”。

如果分析师自定义一种新的UML元素，如《Interrupt》，则该元素的语义是中断，如图2-27所示，就是用自定义的UML元素《Interrupt》表示中断的3种不同格式。



图2-27 构造型的3种表示方法

① 第一种表示方法：创建一种新的UML元素符号《Interrupt》，其语义表示“中断”，然后，在《Interrupt》符号右边放置一个图标。这样，构造符号“《Interrupt》”与图标一起代表“中断”。

② 第二种表示方法：创建一种新的UML元素符号《Interrupt》，其语义表示“中断”。这是一种标准表示方法。

③ 第三种表示方法：直接用一个图标表示新的构造元素符号，该图标表示的语义是“中断”。

2) 标记值

标记值是为事物添加新特征，即为事物增加一个属性。其格式是“{标记名=标记值}”，标记名代表事物的属性，标记值表示事物的属性值。例如，{name="李小平"}。其中，标记名是name，分隔符是“=”，标记值是“李小平”。

3) 约束

约束扩展了UML构造块的语义，是通过在元素之间添加约束条件、增加新的语义或改变已存在规则的一种机制。可以通过文本和OCL两种方法表示约束。OCL(Object Constraint Language)是一种对象约束语言。约束的表示方法和标记值的表示方法类似，都是使用花括号括起来的字符串。例如，对people类中人名(name)的约束，可以表示为：{name like "李\*"}，这个约束表示人的姓氏为"李"的所有人。

## 2.6 对系统体系结构建模

当用RUP统一过程开发软件系统时，对体系结构建模最重要。在项目开发的各个阶段，都是把系统的体系结构模型作为焦点来控制系统的迭代和增量式开发。

在开发过程中，为了清晰理解未来软件系统的体系风格、组织架构、组成元素、接口、行为和协作方式，有必要从5个角度(5种视图)对软件系统建模，这5个视图分别是用例视图、设计视图、构件视图、并发视图和部署视图，即从5个角度来描述系统的5个方面。在开发软件系统过程中，要以用例视图为中心去构造其他4种视图。

#### 1．用例视图

描述了系统的功能和参与者。用例视图由多个用例图组成。

#### 2．设计视图

又称逻辑视图，描述软件系统的组成、结构和行为，是软件系统的蓝图。设计视图常由类图、交互图、状态图和活动图组成。

#### 3．构件视图

描述了软件系统的组成和结构。构件视图描述系统包含的软件构件和文件，该视图常由一组构件图组成。

#### 4．并发视图

描述系统各部分之间的同步和异步执行情况，该视图由状态图和活动图来描述。

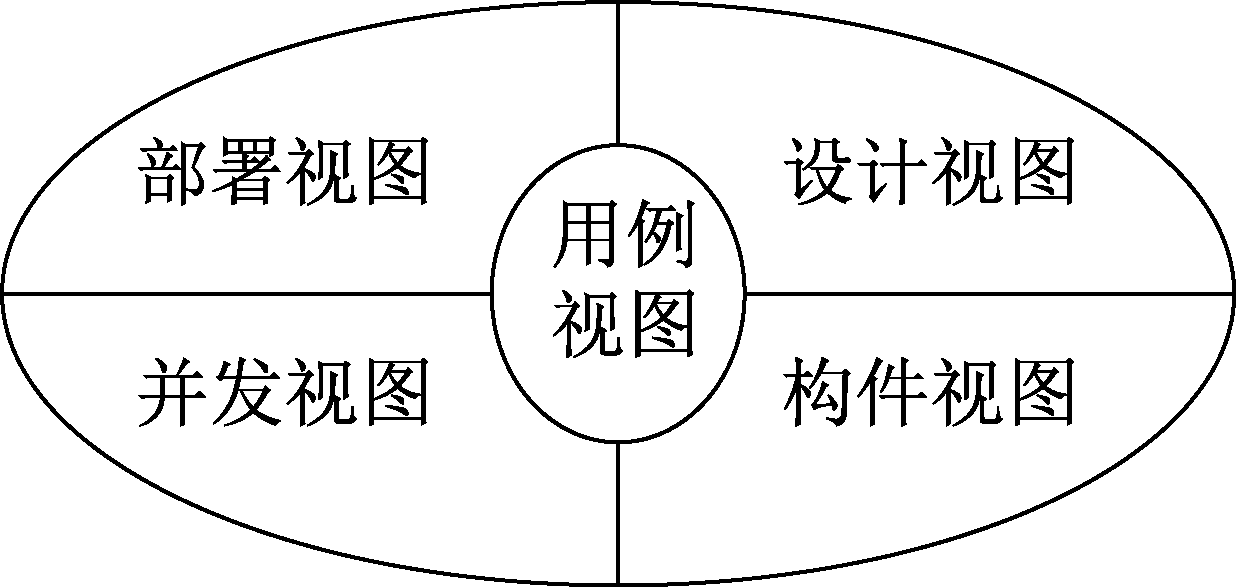


图2-28 软件系统

#### 5．部署视图

描述软件系统的各部分如何部署到各硬件结点上。

如图2-28描述了5种视图的逻辑关系。其中，用例视图是其他4种视图的中心和焦点，也是系统开发的目标。

## 2.7 本 章 小 结

本章首先讲了UML是由构造块、规则和公共机制3个部分组成，然后对这3个方面展开了进一步说明。

首先，阐述了事物构造块和关系构造块，它们是UML建模元素的主体。其中，事物构造块又包括结构事物、行为事物、分组事物和注释事物4种类型；关系构造块详细地描述了关联、泛化、依赖和实现4种主要关系。

接着简要阐述了UML中共同的规则，并以命名规则、范围规则和可见性规则为例说明了它们对UML模型的影响。

然后系统地介绍了规格说明、修饰、通用划分和扩展机制。用户可以通过构造块添加新的事物，通过标记值添加新的特性，通过约束更好地体现模型，通过扩展机制为UML建模能力添加新的功能。

在本章的最后又将读者的目光聚焦到“图”这个最重要的构造块中，简要地阐述了 UML 2.0中定义的13种图，以及不同图的划分和类别，同时结合RUP中的“4+1”视图说明了系统体系结构的表示方法。

## 2.8 习 题

#### 1. 填空题

(1) UML中主要包含4种关系，分别是\_\_\_\_\_、\_\_\_\_\_\_\_、\_\_\_\_\_\_\_\_和\_\_\_\_\_\_\_。

(2) 常用的UML扩展机制分别是\_\_\_\_\_\_、\_\_\_\_\_\_\_和\_\_\_\_\_\_\_\_。

(3) UML的通用机制分别是\_\_\_\_\_\_、\_\_\_\_\_\_和\_\_\_\_\_\_。

#### 2. 选择题

(1) UML中的事物包括结构事物、分组事物、注释事物和\_\_\_\_\_\_。

(A)实体事物 (B)边界事物 (C)控制事物 (D)动作事物

(2) UML中的4种关系是依赖、泛化、关联和\_\_\_\_\_\_。

(A)继承 (B)合作 (C)实现 (D)抽象

(3) 下面不是UML中的静态视图的是\_\_\_\_\_\_\_\_\_。

(A)构件图 (B)用例图 (C)对象图 (D)类图

#### 3. 问答题

(1) 在UML中，可见性有哪几种？

(2) 图形符号与规格描述有何区别？

(3) 约束有哪两种表示法？