

# 第五章 面向对象方法-概念与表示

## 一、引言

### 1、构造模块的四种基本观点

- 1) 以“过程”或“函数”为基点，使每一模块实现一项功能；
- 2) 以一个“数据结构”为基点，使每一模块实现该数据结构上的操作功能；
- 3) 以“事件驱动”为基点，使每一模块识别一个事件并对该事件作出响应；
- 4) 以“问题域中的一个成分”为基点，使每一模块对应现实世界中的一个事物。





## 2、OO方法基于的“世界观”：

- 世界是由对象构成的；
- 对象有其自己的属性和内部运动规律；
- 对象之间的相互作用，构成了大千世界的各式各样的不同系统。






### 3、OO方法的两种学派:

一种：以“方法（method）”驱动的方法学。

- 基本思想：在给出符号体系的基础上，明确规定进行的“步骤”，并在每一步中给出“实现策略”。
- 代表：P.Coad的“OOA（1990）”，“OOD（91）”
- 优缺点分析：
  - 优点：容易学习和掌握。
  - 缺点：不够灵活，可能对出现的新问题就没有办法处理。





### 3、OO方法的两种学派：

第二种：以“模型（model）”驱动的方法学。

- 基本思想：给出模型化概念，即符号体系以及目标模型；而不明确规定实现目标的“步骤”但给出一些必要的指导。
- 代表：Rumbaugh 的“OMT（1991）”等
- 优缺点分析：
  - 优点：比较灵活；
  - 缺点：与OOA相比，不易学习和掌握。





# 面向对象建模技术

如以前所述,建模方法主要包括:建模语言和过程.

建模语言(**Modeling Language**)是用以表述设计方法的表示法(主要是图形的);

过程(**Process**)是对设计中所应采取的步骤的意见.

在建模语言方面,“UML已成为一种绘制面向对象设计图的标准工具,并已传播到非面向对象领域.面向对象以前的主要方法已经消逝.UML登场了,并且稳居宝座.”  
——摘自<UML精粹>序,徐家福译







# 1、类图

## 1) 类

根据OO的世界观，对象有其自己的属性和内部运动规律，并与其他对象相互作用。因此，可以在结构上把对象抽象为两个主要部分：

### 数据和操作

其中，数据用于表达对象的状态，而操作表达在特定状态对外提供的服务（功能）。





于是，现在可以从二个层面来理解“对象”：

概念层：对象是可标识的、可触摸的和可感知的一切事物。

结构层：对象是数据和操作(或谓属性和行为)的封装通信单位.数据表示对象的属性状态，操作(或称方法)决定了对象的行为和与其它对象进行通信的接口。





例如：

| 库 帐  |       |
|------|-------|
| 属性 { | 品名：   |
|      | ..... |
| 操作 { | 入库    |
|      | 出库    |
|      | 日结    |
|      | 查询    |
|      | 年报    |







为了控制信息组织的复杂性，引入了类的概念：

## (2) 语义

类是具有相同结构、行为和关系的一组对象的描述符。

表示正被建模系统的一个概念。

就描述问题的方法而言，一般采用“二分法”；例如：《类，实例》

《类型，值》

**type array[1..5] of integer;**

《主语，特定的名词或名词短语》

.....





- 例如：
- 抽象类
  - 基类，超类
  - 聚合类：

**collection(to hold component objects of a single type and its subtypes.[Firesmith] )**

**container(to hold unrelated component objects of multiple unrelated types.[Booch,Firesmith,Rumbaugh] )**


**structure(class)(component objects of which are interrelated.[Firesmith ])**

**•composite class:any class that composed of other classes.**

**[Wirfs-Brock]**



**北京大学软件工程国家工程研究中心**  
NATIONAL ENGINEERING RESEARCH CENTER FOR  
SOFTWARE ENGINEERING OF PEKING UNIVERSITY



## 2) 属性 语义

表示对象状态的一组值。

**Any named property of an object that takes a literal as its value, defines the abstract state of its object, and appear within the interface rather than the implementation.**

**[ODMG]**

**Any named property used as a data abstraction to describe its enclosing object, class, or extent.**

**[Firesmith, Rumbaugh]**



北京大学软件工程国家工程研究中心  
NATIONAL ENGINEERING RESEARCH CENTER FOR  
SOFTWARE ENGINEERING OF PEKING UNIVERSITY



3) 操作：为其它对象提供的服务。

语义

**-any service that can be requested.**


**[Jacobson,Martin,OMG]**

**-Any discrete activity,action,or behavior  
that is performed by (I.e.,belongs to)  
an object or class.**

**[Ada95,Firesmith,Hender-Sellers]**

同义词：**function,method,service.**





对象：（1）语义

对象是类的一个实例。

（2）表示

|               |
|---------------|
| <u>对象名：类名</u> |
| 属性列表          |

|                 |
|-----------------|
| <u>triangle</u> |
|-----------------|

|                         |
|-------------------------|
| <u>Triangle:Polygon</u> |
|-------------------------|

|                 |
|-----------------|
| <u>:Polygon</u> |
|-----------------|

名字的其他情况：**display-window：WindowSystem::**

**GragicWindows::Window**

**a:b,c,d(规则：在一个时刻只属于一个实现类)**







## 5) 接口

### (1)语义(仅以类为例)

接口描述类、构件或者子系统的外部可见操作，并不描述内部结构。

通常，一个接口仅描述一个特定类的有限行为。

接口没有实现，接口也没有属性、状态或者关联，接口只有操作。

接口只可以被其它类目使用，而其本身不能访问其它类目。

可以对接口使用泛化关系。

接口在形式上等价于一个没有属性、没有方法而只有抽象操作的抽象类。





## (2) 表示法

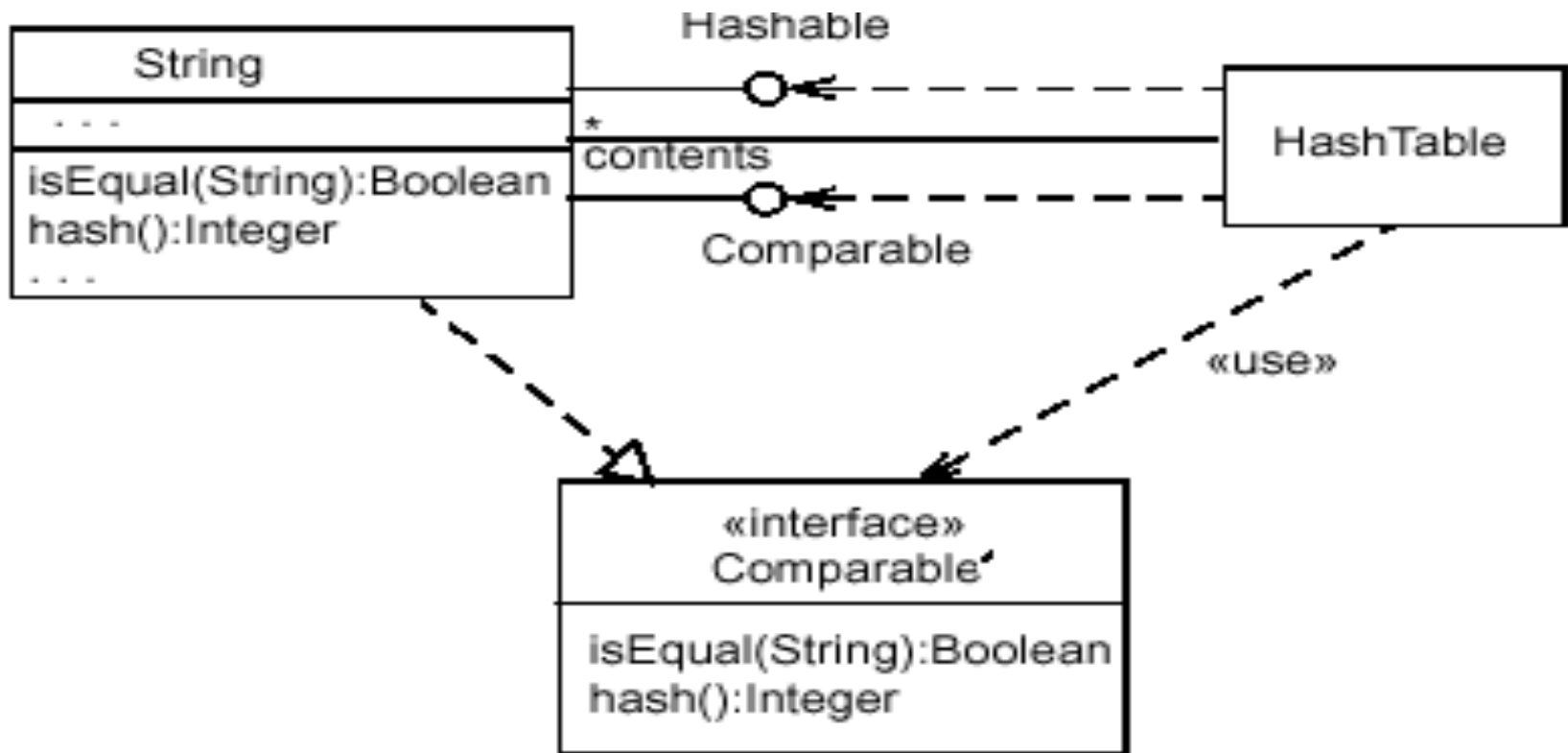
可以用带有分栏和关键字<<interface>>的矩形符号表示接口。在操作分栏中给出接口支持的操作列表。接口的属性分栏总是空的。把从类目到它支持的接口的实现关系显示为带有实三角箭头的虚线。用带有《use》标记的虚线箭头表示类目（在箭头尾部）使用或者需要接口提供的操作。

也可以把接口表示成小圆圈。接口名放在圆圈的下面，并用实线把圆圈连接到支持它的类目上。这意味着这个类目要提供在接口中的所有操作，其实类目提供的操作可能要更多。要显示接口的操作列表的话，就不能使用圆圈表示法，而应该使用矩形表示法。可以用指向圆圈的虚线箭头，把使用或者需要接口提供的操作的类目连到圆圈。

上述的虚线箭头还意味着使用接口的类目并不需要使用接口给出的全部操作。



### (3) 例子



该图表明，类String支持接口Hashable、Comparable，

而类HashTable使用接口Hashable、Comparable。





## 以上内容小结

(知识点 和 解决问题的基本思想和途径)

### 1、知识点

(1) 给出了表达客观事物基本成分（对描述客观事物而

言不可再分的）的概念： 对象，类

语义及其表示。

(2) 给出了这些成分的基本构造： 属性，操作

语义及其表示

(3) 给出了支持功能抽象的机制： 接口

语义及其表示，以及







## (4) 接口与其他成分之间的关系

### ①接口：

一组操作（没有实现的），这组操作在一个特定类中实现。

作为一种机制，支持功能抽象。

### ②操作：一个类对外提供的服务。

可以是抽象的。

其声明可以作为接口的成分。

### ③方法：是操作的一个实现。







至此可以说：

这五个概念围绕一个问题，即如何描述客观事物-“对象”展开的！

如何抽象对象的“结构”：属性，操作

如何描述一组具有相似性质的对象：类

如何抽象并描述在特定环境中对象的功能：

接口。





## 2、解决问题的基本思想和途径

(1) 大千世界是有对象构成的，对象有其自己的属性与运动规律。

### (2) 基本途径

①数据和操作（功能）的局部化-支持事物语义的表达；

②数据和操作的封装性--支持交互；

③功能模块化（接口）--支持在一个层次上的功能抽象；

④描述的“二分法”（《元数据，数据》）--支持复杂性控制（抽象）

《类，对象》    《接口，操作》    《操作，方法》

提高    模型的稳定性，可维护性





## 6) 关联

对象之间存在的引用关系-元组，称为链。

关联是对一组具有相同结构特性、行为特性和语义的链的描述。

**association: any semantic relationship between two or more class or types [Booch,...]**

关联不仅适用于类，对接口、用况和子系统也适用。

围绕“关联”，涉及的内容有：

- (1) 二元关联 及 (2) N元关系；
- (3) 关联端点：用于给出关联语义的描述；
- (4) 限定符：确定关系的条件；
- (5) 关联类：定义关系的属性及实现；

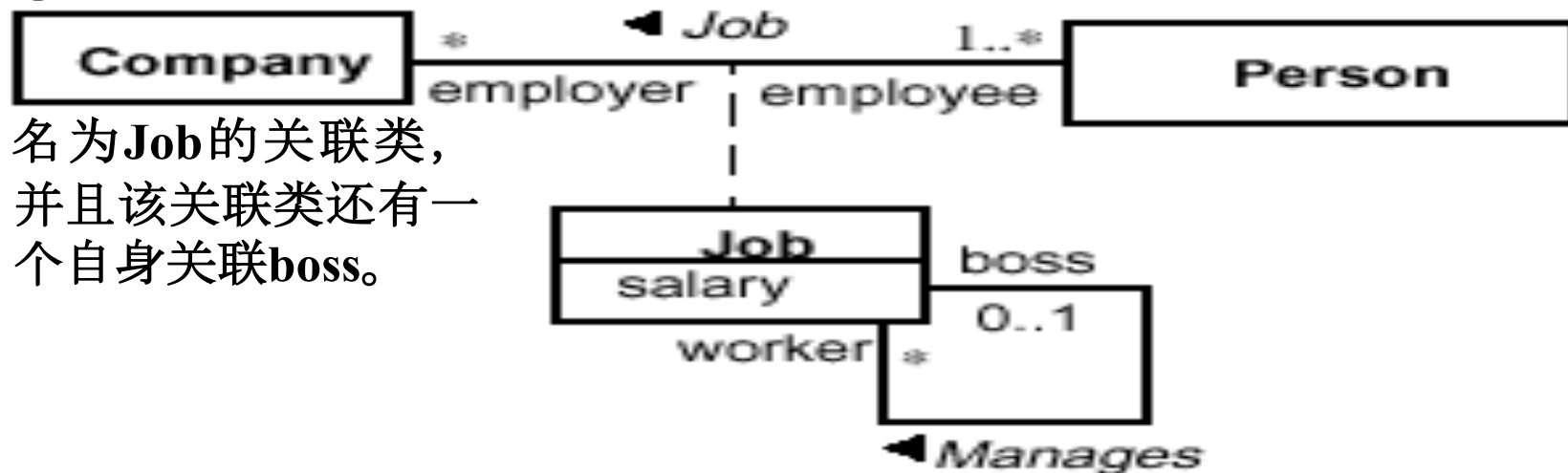


## (1)二元关联(仅以类为例)

### ①语义

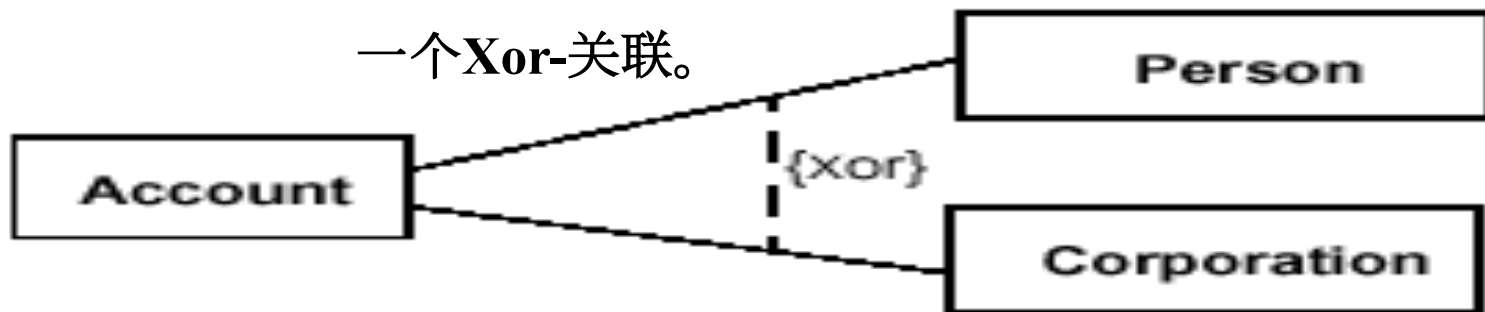
二元关联是两个类之间的关联。其中，相关联的两个类可以是同一个类。

### ②表示法



名为**Job**的关联类，  
并且该关联类还有一个  
自身关联**boss**。

一个Xor-关联。





## (2) 关联端点

### ①语义

关联端点是关联的组成部分，而不是独立的模型元素。  
每个关联有两个或多个端点。

### ②表示法

关联路径通过端点与类符号相连。

在关联路径的每个端点上，都可以附属有该关联的描述信息，以表明该关联的性质。

附属的信息属于关联符号，而不属于类符号。

在关联端点上，可以给出如下类型的信息：





## (A) 语义

多重性是对集合可能采取的基数范围的说明。

可以为关联中的角色、组合中的组成部分、某些元素的重复以及其它目的，给出相应的多重性规约。

多重性规约是非负整数开集的一个子集。

## (B) 表示法

多重性表示为：由用逗号分开的整数间隔序列组成的字符串，间隔代表整数的范围（可能无限），其格式为：

下限..上限

其中，下限和上限都是整型值，说明从下限到上限的整数闭区间。星号（\*）可以用于上限，表明不限制上限。





## ②排序

如果没有指明多重性，相应的元素形成集合就是无序的。

如果多重性大于1，相应的元素集合可以是有序的，也可以是无序的。

在关联端点上，可以通过关键字“`{ordered}`”来指定对元素的排序。但这并没有指定排序是怎样建立和保持的，仅仅表明元素是有序的，在实现时可以按各种方式排序。

## ③导航性

箭头附属到关联路径的端点，表明朝着连接到箭头的类的方向支持导航。

箭头可以附属0个、1个或2个路径端点。

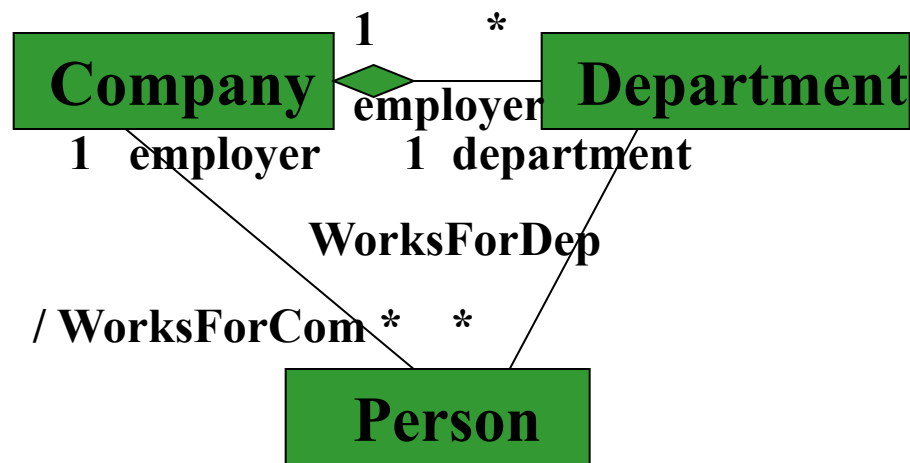
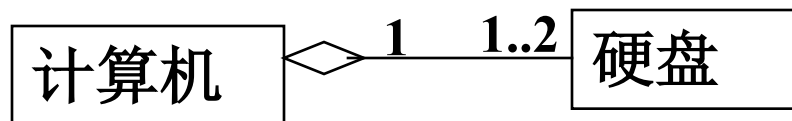




#### ④ 聚合符

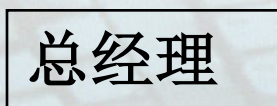
聚合是一种特殊的关联，表示整体类和部分类之间的“整体-部分”关系，其中的整体类称之为聚集。

聚合符是一个空心菱形，它附属在关联路径表示聚合的那一端，但不能附属在路径的两端。



#### ⑤ 角色名

角色名靠近关联路径的名称串。它表示在其附近的与关联路径端点相连接的类所扮演的角色。



服务业务

副总经理



北京大学软件工程国家工程研究中心  
NATIONAL ENGINEERING RESEARCH CENTER FOR  
SOFTWARE ENGINEERING OF PEKING UNIVERSITY

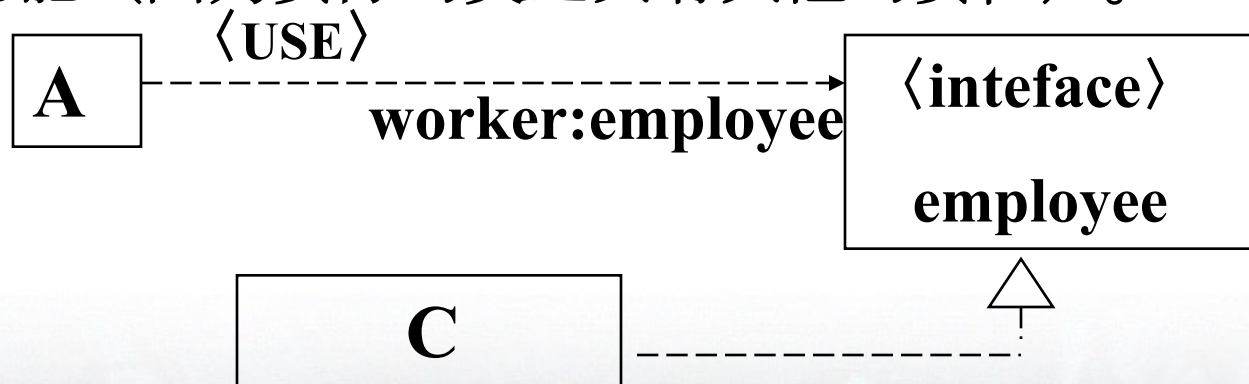
## ⑥接口说明符

接口说明符的语法为：

角色：接口名， ...

例如：**worker:employee**. 表明关联中的一个实例**worker**期待接口**employee**提供的操作。

既该关联的实现，需要接口**employee**为角色**worker**提供相应的操作。通常实现接口操作的类具有为实现这一特定关联更多的功能（因为实际的类还具有其他的责任）。



注：如果省略一个接口说明符，可用关联获取对被关联的类的完全访问。







## ⑦可变性

性质串{frozen}表明：

在对象被产生和初始化后，不能加入、产生或移走从该对象出发的链。

性质串{addOnly}表明：

可以加入另外的链（设想多重性是可变的），然而不能调整或删除链。如果链是可变的（能加入、删除和移走），就不需要标示信息。

## ⑧可见性

在角色名前面加可见性说明符（‘+’、‘#’、‘-’或{public}、{protected}、{private}）。

指明了朝着给定角色名方向的关联的可见性。

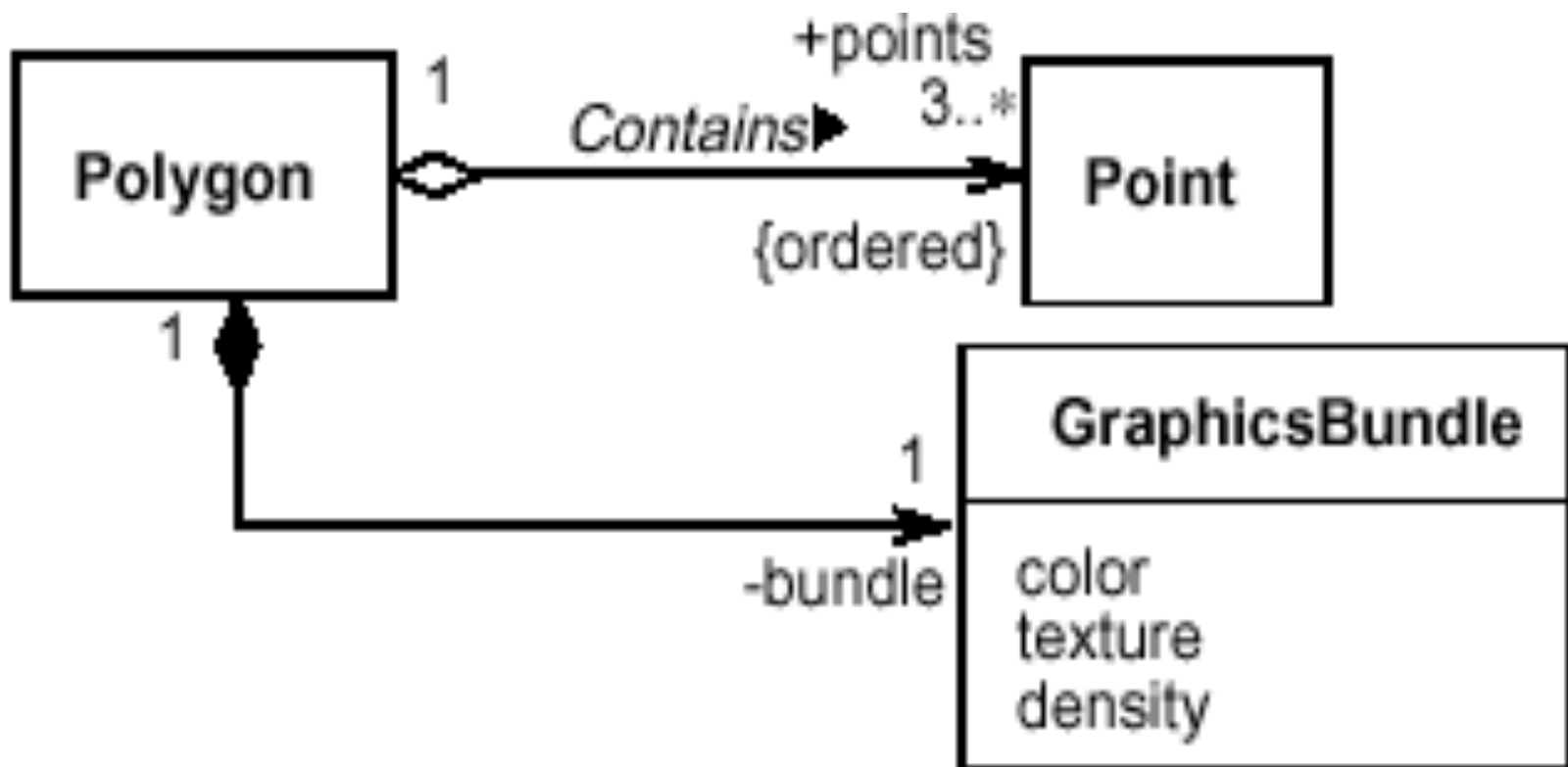




### ③风格指南

如果在一个关联端点上有多个信息，给出它们的顺序为：限定符、聚合符号、导航箭头。

角色名和多重性可以放在同一关联端点的两侧，或是放在一起（例如：“\*employee”）。



关联角色上的多种信息



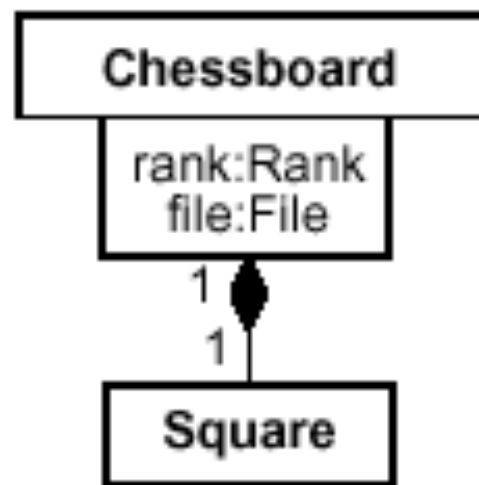
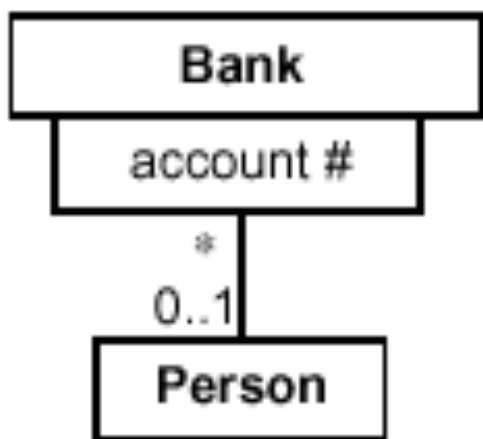
### (3) 限定符

#### ① 语义

一个限定符是一个属性或是属性表，这些属性的值将与一个相关联的对象集做了一个划分。

限定符是关联的属性。

#### ② 表示法



左图的限定符有一个属性account#，表明：在一个银行中，一个帐户对应一个用户，或没有对应人员。

右图的限定符有两个属性，它们与Chessboard一起确定了Square，且Square是其组成部分。





## (4) N元关联

### ①语义

一个N元关联是三个或多个类之间的一个关联，其中的一个类可能在一个N元关联中多次出现。

该关联的每一个实例是一个N元组，即它由N个分别来自相关的类的对象组成。

可以规约N元关联的多重性：在一个角色上的多重性是指，当该N元关联中的其它N-1个类的对象被确定时，该关联潜在的实例元组的数目。

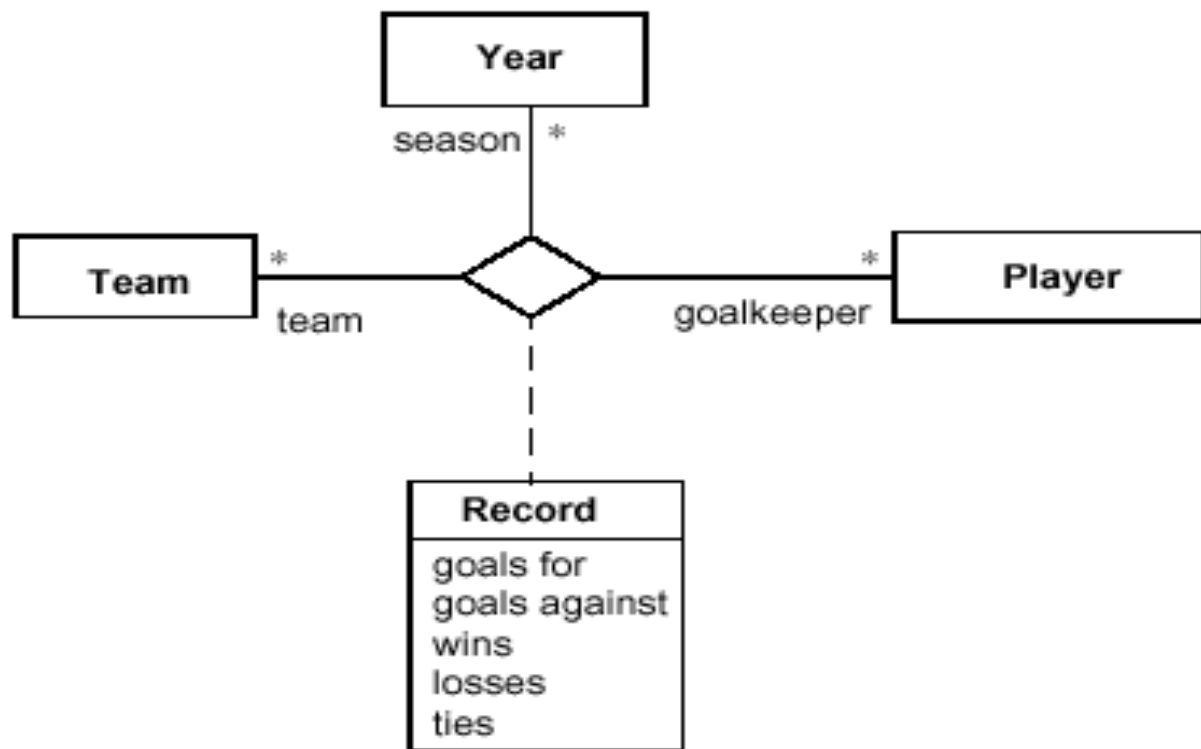
对于一个N元关联，在任意角色上都不能有聚合标志。



## ②表示法

本例描述了一个具有一个特殊守门员的球队在每一个赛季的记录。

假设在本赛季可出售守门员，且该守门员可出现在多个球队。



通过一个大的菱形表示一个N元关联。  
如果关联有名字，就显示在菱形附近。  
同二元关联一样，角色可以显示在每一条路径上。  
可以指出多重性；然而不允许有限定符和聚合符。  
可以用虚线把关联类符号与菱形连接起来，表示具有属性、操作或关联的N元关联。





## 7) 组合

### (1) 语义

组合是一种关联，是聚合的一种形式。其部分和整体之间具有很强的“属于”关系，并且它们的生存期是一致的。

显然，❶这种聚集（也称为组成）末端的多重性不能超过1；

❷在一个组合中，由一个链所连接的对象而构成的任何元组，必须都属于同一个容器对象。

在一个组合中，其部分可以包含一些类和关联；根据需要，

也可以把它们规约为关联类。

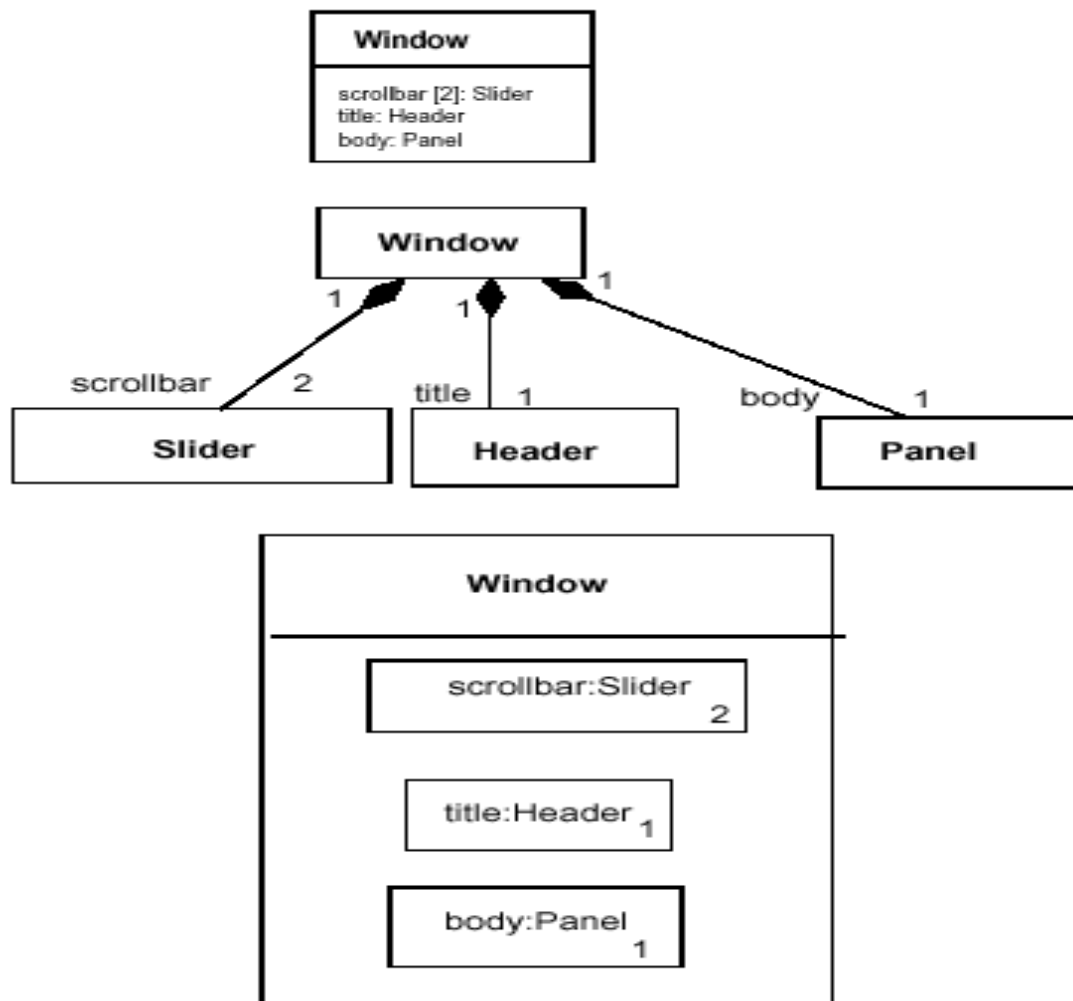






(2) 表示

表示组合的  
不同方法



该例给出了三种表示组合的方法。

其中，类**Window**由类**Silder**（角色为**Scrollbar**）、**Header**（角色为**title**）和**Panel**（角色为**body**）组成。





## 8) 链

### (1) 语义

链是对象引用的一个元组。

链是关联的一个实例。

### (2) 表示法

二元链表示为两个实例之间的路径。

可以把一个实例与它自身之间的链表示为一个具有单一实例的环。

在链的各端可以表示角色名。

链没有实例名，从与它们相关的关联中得到其标识。

多重性不能显示在链上，因为链是实例。

其它的关联附属物（组合、聚合和导航）可以表示在链端点上。

限定符可以表示在链上，限定符的值也表示在它的框中。

用带有连向各个参与实例的路径的菱形表示N元链。





## 9 ) 泛化

### (1) 语义

泛化是一般元素（父亲）和特殊元素（儿子）之间的一种分类关系，其中特殊元素的结构完全与一般元素一致，并附加了一些信息。

泛化可用于类、包、用况、关联和其它元素。

泛化支持继承 (inheritance): 复用机制

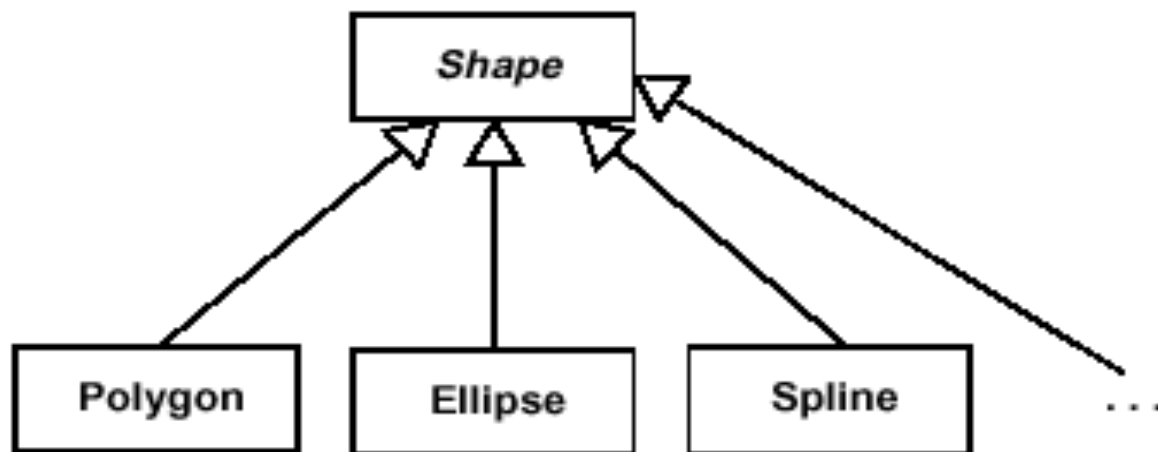
**the incremental construction of a new definition in term  
of existing definitions without disturbing the original  
constructions and their client .[Ada95,firesmith]**



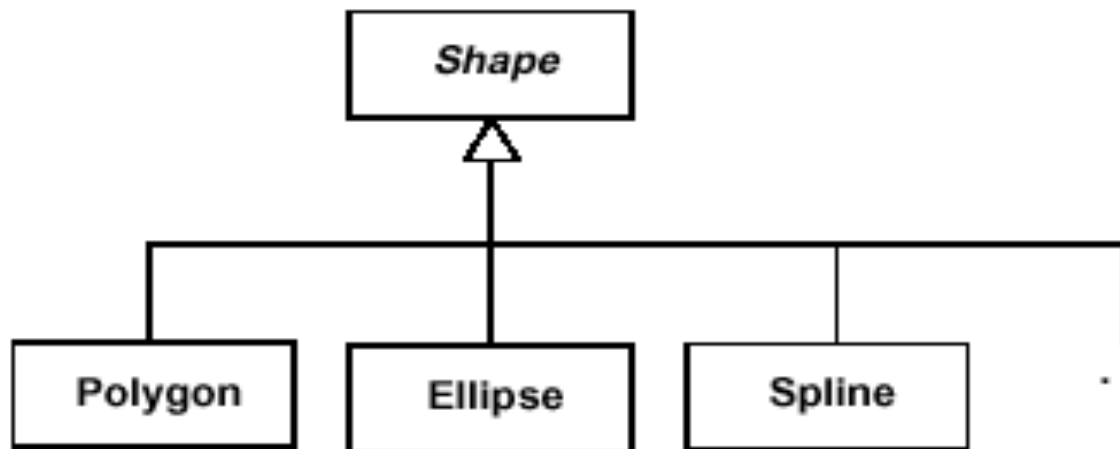
## (2) 表示法

把泛化表示成从子类（特殊类）到父类（一般类）的实线路径，在较一般的元素的路径端有一个大的空心三角。

### 分离表示法



### 共享表示法

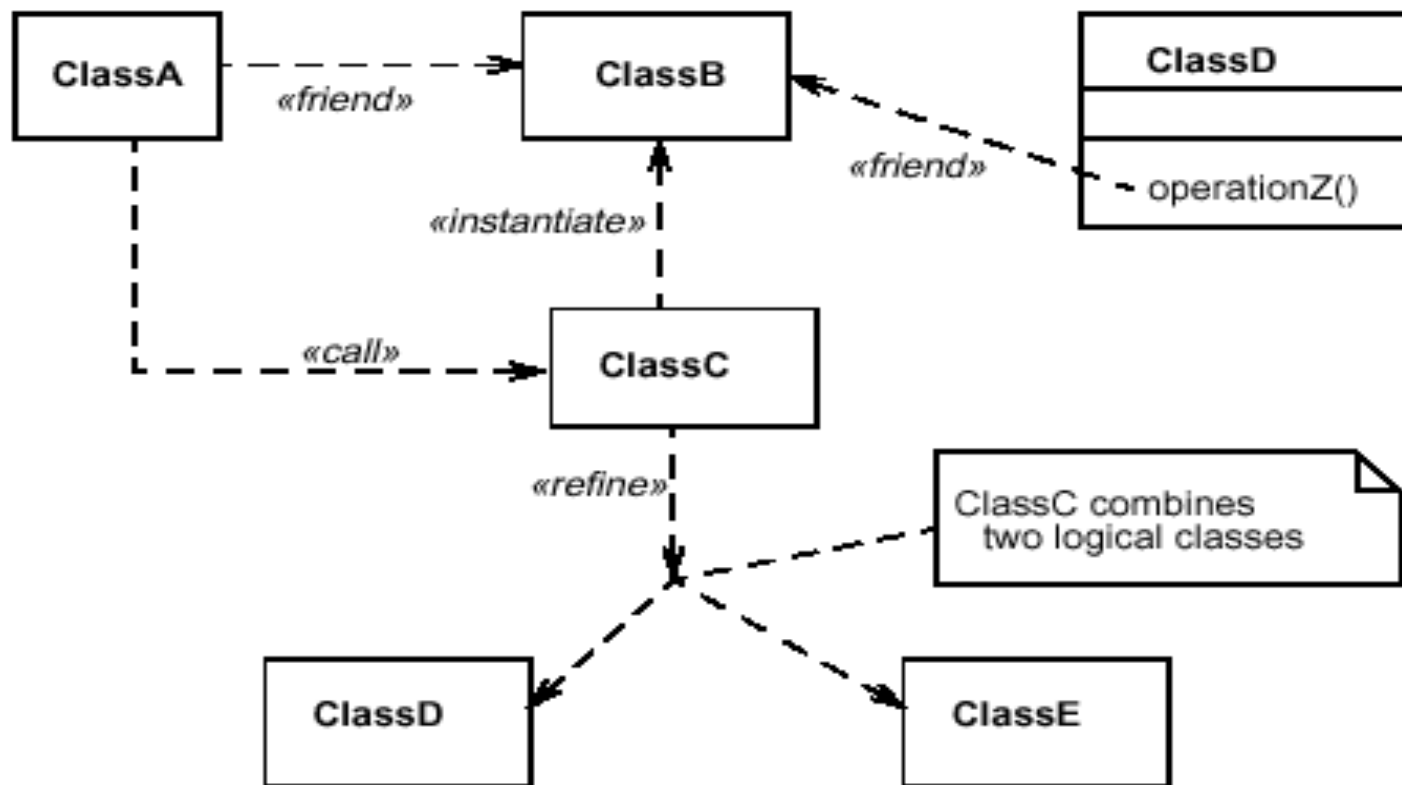


## 10) 依赖

### (1) 语义

一个依赖规约了两个模型元素（或两个模型元素集合）之间的一种语义关系，即：对目标元素的改变可能需要改变该依赖中的源元素。

### (2) 表示法。







❶ 把依赖表示为两个模型元素之间的虚线箭头。

箭头尾部的模型元素（客户）依赖箭头头部的模型元素（提供者）。

❷ 依赖的特定情况：一组元素作为客户或提供者

在这种情况下，可以自客户端引出一条带箭头的虚线，并把该虚线连接到多个提供者端。

如果需要的话，可以在连接处放置一个小的圆点，作为连接点。此时，依赖的注释应该依附在连接点上。

注意：在一些面向对象的建模语言中，预定义了一些特定的依赖。其关键字要用书名号括起，并表示在虚线箭头附近。





# 关于“关系”描述的小结

## 1、知识点

- (1) 给出了表达客观事物之间关系的基本概念：链。
- (2) 给出了关联的语义和表示，并且，
- (3) 还给出了一些特定的关联（分类）。主要包括：二元关联与N元关联，聚合与组合，泛化，依赖。

语义及其表示。

- (4) 给出了关联语义的描述：

①关联端点-多重性，排序，导航，角色名，接口说明符，可变性，可见性等；

②限定符-关联的属性；





③关联类；

④对泛化给出了子类划分的语义约束。

## 2、解决问题的基本思想和途径

紧紧围绕“关系”的表达（其中包括“形”与“义”），给出了一组概念。

①采用“二分法”：《关联，链》-控制信息组织的复杂性；

②通过对关系的分类，可以有效地支持关系的建模；

③为了弥补“形”的表达能力，给出了一些概念（多重性，可变性等），支持语义的表达；特别地，

④为了支持关系属性和操作的表示，引入了“关联类”和“限定符”。





## 11) 包

### (1) 语义

包是模型元素的一个分组。一个包本身可以被嵌套在其它包中，并且可以含有子包和其它种类模型元素。

包间只有依赖。

标准依赖：访问依赖和引入依赖。作用：使一个包可以访问和引入其它包。

其它依赖：通常隐含了元素间存在着的一个或多个依赖。

### (2) 表示法

包表示成一个大矩形，并且在这一矩形的左上角还有一个小矩形（作为一个“标签”）。





## 12) 包的访问或引入

### (1) 语义

一个元素可以引用其它包中的元素。

❶访问依赖：在包的层次上，依赖《access》表明：目标包的内容可以被客户包引用，或被递归嵌套在客户包中的其它包引用。

其中：●目标包必须具有适当的可见性，即应有可见性“public”；●对目标包的子孙，应有可见性“public”或“protected”；●对嵌套在目标包中的包，可有任意可见性

如果在提出访问的那个包中还存在包，那么嵌套在其中的包能得到与外层包同样的访问。

注意，一个访问依赖，●不修改客户的名字空间；

- 不以任何其它方式自动创建一些引用。

- 访问依赖仅准予建立一些引用。





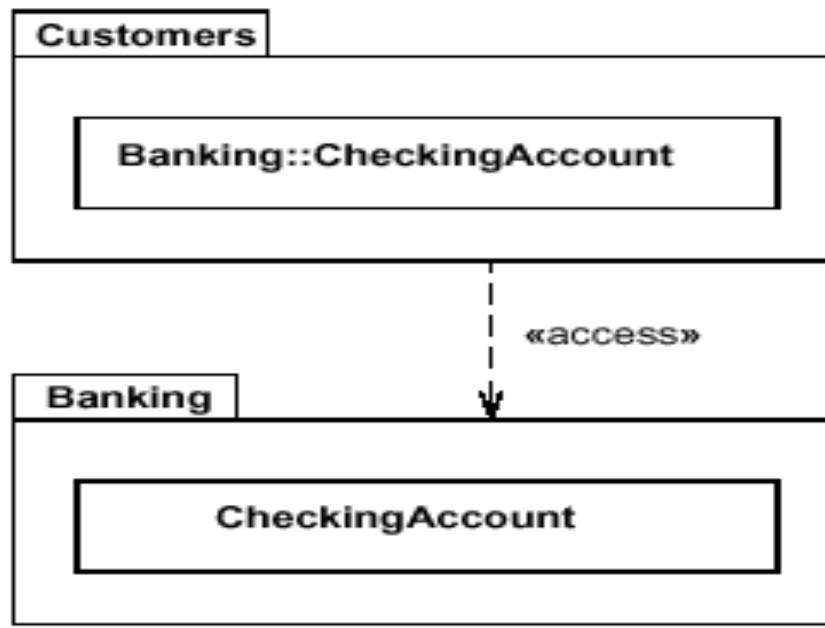
②引入依赖：一个引入依赖获得访问，并将目标名字空间中的那些具有合适可见性的名字引入到客户包（即对它们的引用可不需要一个路径名）。

## (2) 表示法

访问依赖表示为从客户包到目标包的依赖箭头。箭头上带有<<access>>：表明客户包中的元素可以合法地引用目标包中元素。

引用必须满足由提供者指定的可见性约束。

引入依赖除了箭头上的串为<<import>>外，具有和访问依赖一样的表示法。





## 关于“包”的小结

- (1) 包是控制文档组织复杂性的机制。
- (2) 包也可以作为“模块化”、“构件化”机制。
- (3) 包之间（在包的层次上）存在两种依赖：访问和引入

访问依赖表明：目标包的内容可以被客户包引用，或被递归嵌套在客户包中的其它包引用。

- 不修改客户的名字空间；
- 不以任何其它方式自动创建一些引用。

引入依赖表明：将目标包中那些具有合适可见性的名字引入到客户包（即对它们的引用可不需要一个路径名）。





## 13) 注释

### (1) 语义

注释是一个符号项，用以表示某一语义元素的一些文本信息。

一个注释是一个图形符号，该符号包含了一些文本信息（包含一些嵌入的图象），直接附属于一个模型元素，把任意的文本信息附属到一个模型元素。

注意：注释中的文本信息没有语义作用。

### (2) 表示法

把注释表示为带有折角（右上角）的矩形。通过虚线依附到多个模型化元素上，或单独存在。

这个模型是由安  
澜.怀特在与计划  
组商讨后构造的





## 2 顺序图

目的：为了表示实例之间按时间顺序组织的交互。支持实时系统和复杂场景的详细建模。

途径：❶引入实例生命线，表示参与交互的实例；

❷引入消息，并将消息按时间顺序排列，表示实例之间的交互。

局限性：只是对一组实例及之间的交互进行描述。

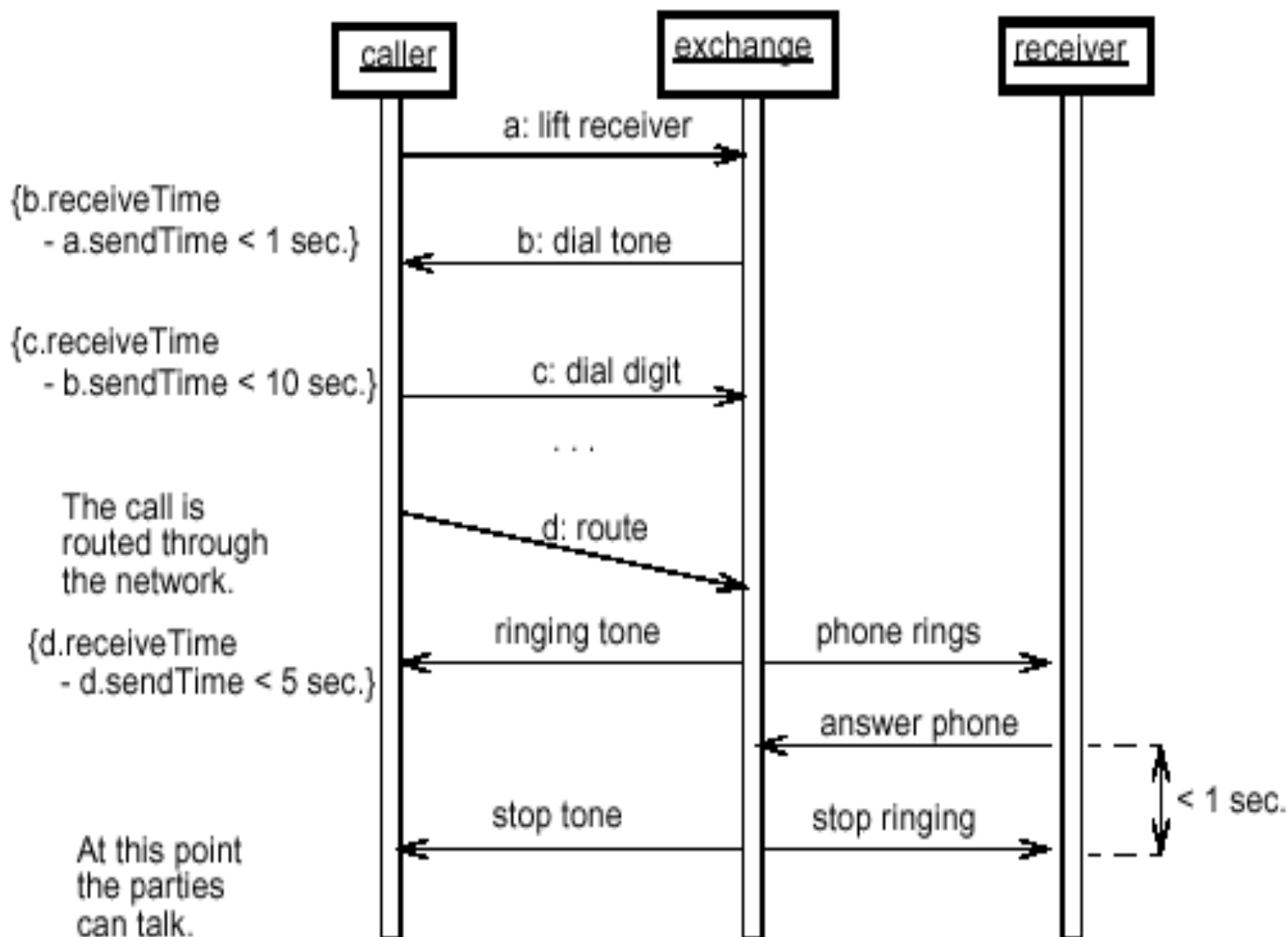
### 1) 顺序图

#### (1) 语义

顺序图是一种表达对象间交互的图，由一组对象以及其间顺序发送的消息组成。



## (2) 表示法







可见：

顺序图是二维的。其中：垂直方向表示时间，水平方向表示不同的对象。

### (3) 表示选项

- 生命线之间的顺序是任意的。
- 可以交换轴；
- 可以在消息名上使用时间表达式，表示计时约束。

其中：可以把函数**sendTime**（发送消息时间）和函数**receiveTime**（接收消息时间）应用于消息名，以计时间。

可以使用一些构造标记，用于指示时间间隔。

类似地，可以用消息名表示在计时表达式中的发送或接收消息的时间。





## 2) 对象生命线

### (1) 语义

在顺序图中，对象生命线表示扮演特定角色的对象。类的角色规定了对应的角色，它描述了扮演该角色的对象的性质。

与对象生命线相关的有：

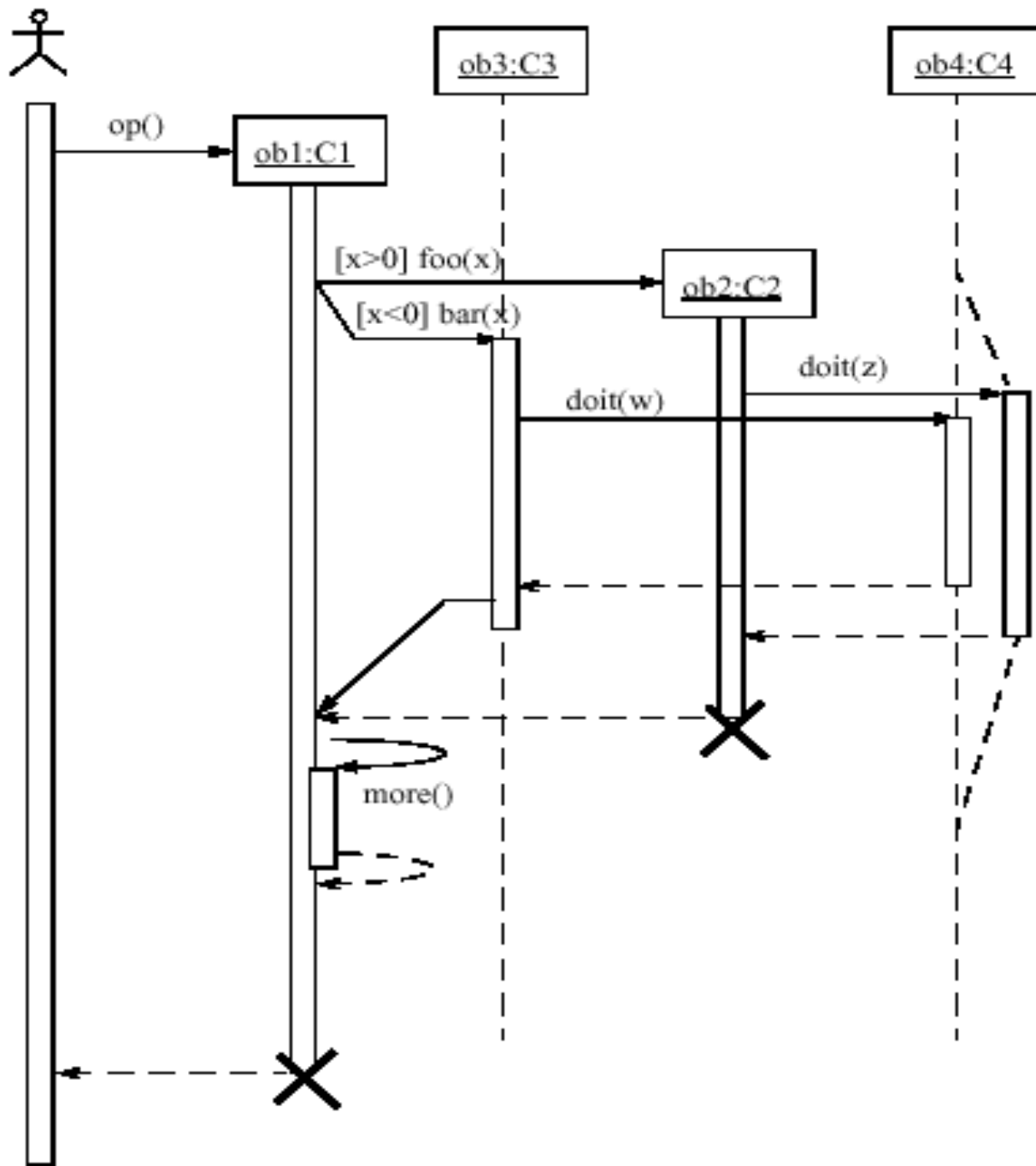
◆生命线之间的箭头（消息）：表示扮演这些角色的对象之间的通讯。

◆窄长的矩形（活化）：表示处于一个角色状态的对象的存在和持续。

### (2) 表示法

“对象生命线”表示为垂直虚线。生命线代表一个对象在特定时间内的存在。例如：





其中：

---：生命线

✱：激活（活化）

[X>0]：转换时间

——：消息

Foo(x):消息名

✱：撤消



其中：

①如果对象在图中所示的时间段内被创建或者销毁，那么它的生命线就在适当的点开始或结束。否则，生命线应当从图的顶部一直延续到底部。

②在生命线的顶部画对象符号。如果一个对象在图中被创建，那么就把创建对象的箭头的头部画在对象符号上。

③如果对象在图中被销毁，那么用一个大的“X”标记它的析构，该标记或者放在引起析构的箭头处，或者放在从被销毁的对象最终返回的箭头处（在自析构的情况下）。

④在图的顶部（第一个箭头之上）放置在交互开始时就存在的对象，而在整个交互完成时仍然存在的对象的生命线，要延伸超出最后一个箭头。

⑤生命线可以分裂成两条或更多条并发的生命线，以表示条件性。此时，每一个生命线对应通讯中的一个条件分支。生命线可以在某个后续点处合并。





### 3) 激活

#### (1) 语义

激活（控制焦点）表示一个对象直接或者通过从属例程执行一个行为的时期。它既表示了行为执行的持续时间，也表示了活动和它的调用者之间的控制关系。

#### (2) 表示法

用一个窄长的矩形表示激活（活化）。

- ①活化的始末：矩形顶端为活化的开始时刻，末端为结束时刻。
- ②激活的动作：可以用文本标注被执行的动作。







③并发：在每个拥有自己的控制线程的对象并发的情况下，一个激活说明了一个对象执行一个操作的持续时间，与其他对象的操作并不相关。

④嵌套：可以在一个特定的时间看到所有活动着的嵌套过程激活。在递归调用一个已激活的对象的情况下，第二个激活符号画在第一个符号稍微靠右的位置，并有重叠。（可以按任意的深度嵌套递归调用）

⑤如果直接计算和间接计算（由嵌套过程执行）的区别并不重要，那么可以用整条生命线表示一个激活。





## 4) 消息

### (1) 语义

消息是两个对象间的通讯，这样的通讯用于传输将产生的动作所需要的信息。一个消息会引起一个被调用的操作，产生一个信号，或者引起一个对象被创建或者被消除。

### (2) 表示法


在顺序图中，把消息表示为从一个对象生命线到另一个对象生命线的的一个水平实线箭头。对于到自身的消息，箭头就从同一个对象符号开始和结束。用消息（操作或信号）的名字及其参数值或者参数表达式标示箭头。

箭头也可以用一个序列数标示，以表示消息在整个交互中的顺序。序列数对于标示并发控制线程很有用处。另外，还可以用监护条件标示消息。



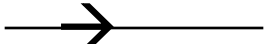
### (3) 表示选项

① 可以用如下种类的箭头表示不同种类的通讯：


● 实箭头： 

过程调用或其它的嵌套控制流。在外层控制恢复之前，要完成整个嵌套序列。

可以把它用于普通的过程调用。在某个主动对象发送信号并等待完成嵌套的行为序列时，也可以把它用于并发的主动对象。

● 枝状箭头： 

控制的平面流。每个箭头表示按顺序进入到下一步，不必考虑控制流的嵌套问题。通常所有的消息都是异步的。

● 半枝状箭头： 

用于显式地表示在过程顺序中的两个对象之间的异步通讯。





●虚的枝状箭头： ----->

从过程调用的返回。

在控制的过程流中，可以省略返回箭头（暗示激活结束），假设每个调用在任何消息后都有一个配对的返回，并可以把返回值标示在初始的箭头上。对于非过程控制流（包括并发处理和异步消息），都应当显式地标出返回。

②在并发系统中，一个实箭头表示产生一个控制线程；一个半箭头表示发送消息而不产生控制。

③通常，消息箭头都画成水平的。这表示发送消息所需要的持续时间是“原子的”（即，在传送消息的中间不能发生任何事情，它与交互的粒度相比是短暂的。）。如果消息需要一段时间到达，且这中间可能发生某些事情，则箭头向下倾斜。







④把分支画成从一个点出发的多个箭头，每个箭头由监护条件标示。依据监护条件是否互斥，这个结构可以表达条件或者并发。

⑤可以把一组相连的箭头闭合，表示为迭代。在一般的顺序图中，迭代表示一组消息的分发可以多次发生。

在一个过程中，可以在迭代的底部标出迭代持续的条件。

如果存在并发，那么图中的一些箭头可以是迭代的一部分，而另一些是独自执行的。此时，应对图进行布局，以使迭代箭头能容易地闭合在一起。





## 5) 转换时间

### (1) 语义

消息可以指定几个不同的时间（例如，发出时间和接受时间）。这些时间可以用在约束表达式中。

用户可以按需要为特定的目的给出时间表达，如**elapsedTime**（占用时间）和**startExecutionTime**（开始执行时间）。可以在约束中使用这些表达式，以给消息指派有效的具体时间约束。

### (2) 表示法

可以赋予消息一个名字。把时间约束写成为一个基于消息名字的表达式。例如，如果消息的名字是 **stim**，用 **stim.sendTime()** 表示发送时间，用 **stim.receiveTime()** 表达接收时间。可以把时间约束表示在与箭头对齐的图的左边上，也可以通过把布尔表达式（可能包括时间表达式）放在括号中表示约束。



## 顺序图要点

1、顺序图给出了显式的激发序列，表达了按时间顺序组织的交互。更适宜表达实时(实体)的规约和复杂的方案(scenarios)。

其中的概念：对象生命线，活化，消息与激发，转换时间。

### 1) 对象生命线

对象生命线表示一个特定角色的对象，指出该对象进行交互的生命期；

### 2) 活化(activation)

一个活化(集中于控制)，显示了一个周期。期间，一个对象正执行一个动作，或者是直接的，或者是通过低层的过程。

活化表示了三方面：第一，表示该动作执行的时间段，

第二，表示该活化和它的调用者之间的控制关系。

第三，嵌套和并发，并显示条件限制。

### 3) 消息与激发(message and stimulus)

一个激发是两个对象间的一个通讯。一个激发导致一个操作予以调用，引起了一个事件，或导致一个对象的创建或删除。

一个消息是激发的规格说明，即它说明了发送者和接受者应该与之一致的角色，也说明了一个动作，该动作在执行时，将分派一个与该消息一致的激发。

### 4) 转换时间

在顺序图上括号内的布尔表达式，可以用以说明这样的约束。





## 2、表示

- 对象：被显示为一个方框，在一个与之垂直的虚线顶部；
- “生命线”（lifeline）用虚线表示；
- 在两个对象生命线之间的箭头线（四种），表示一个消息；
- 每一消息有一个名字；还可以给出参数和一些控制信息；
- 消息出现的次序“由上到下”标记；
- 自调用的标记-发送消息的箭头返回自己的生命线；





- 对象的“活化”，由细框表示；
- 控制信息

例如：条件（condition）

### 重复标记(iteration marker)

- 条件：指出消息被发送的时候，例如：

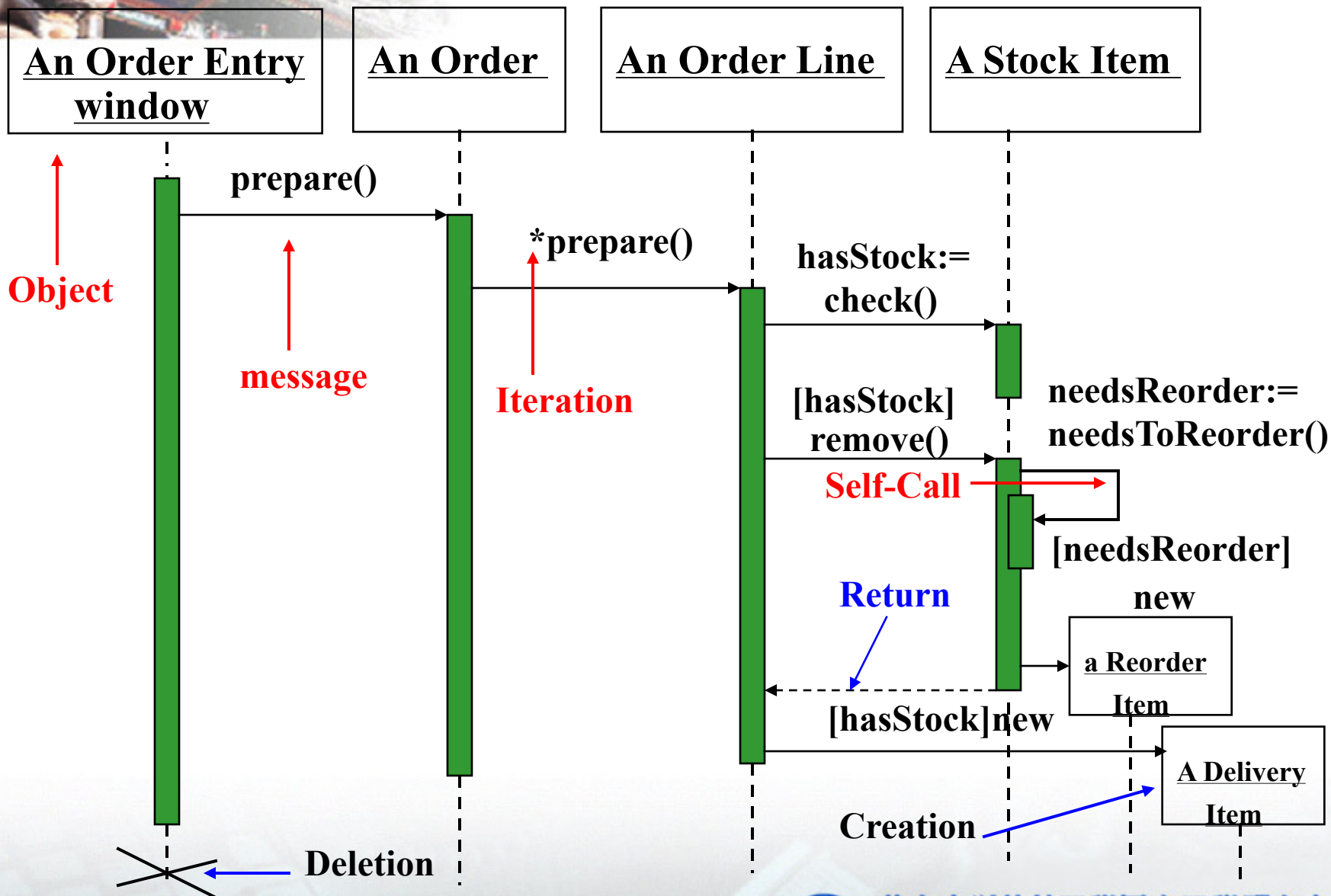
[needsReorder]，仅当该条件为真时，该消息才予发送。

- 迭代标记：指出一个消息对多个接受对象发送多次（当多次满足条件时），例如，

\*[for all order lines]











### 3 状态图

状态图用于描述模型元素（如对象）的行为。特别是，用它描述元素之状态的可能序列和动作的可能序列。因对特定事件（如信号和操作调用）的响应，元素在其生命期中要经历这样的状态，并执行相应的动作。

#### 1) 状态图

##### (1) 语义

通过描述对**事件实例**接收的响应，状态图描述了具有动态行为能力的实体之行为。

通常用状态图描述类的行为，也可以用它描述其它模型实体（如用况、参与者、子系统、操作或方法）的行为。

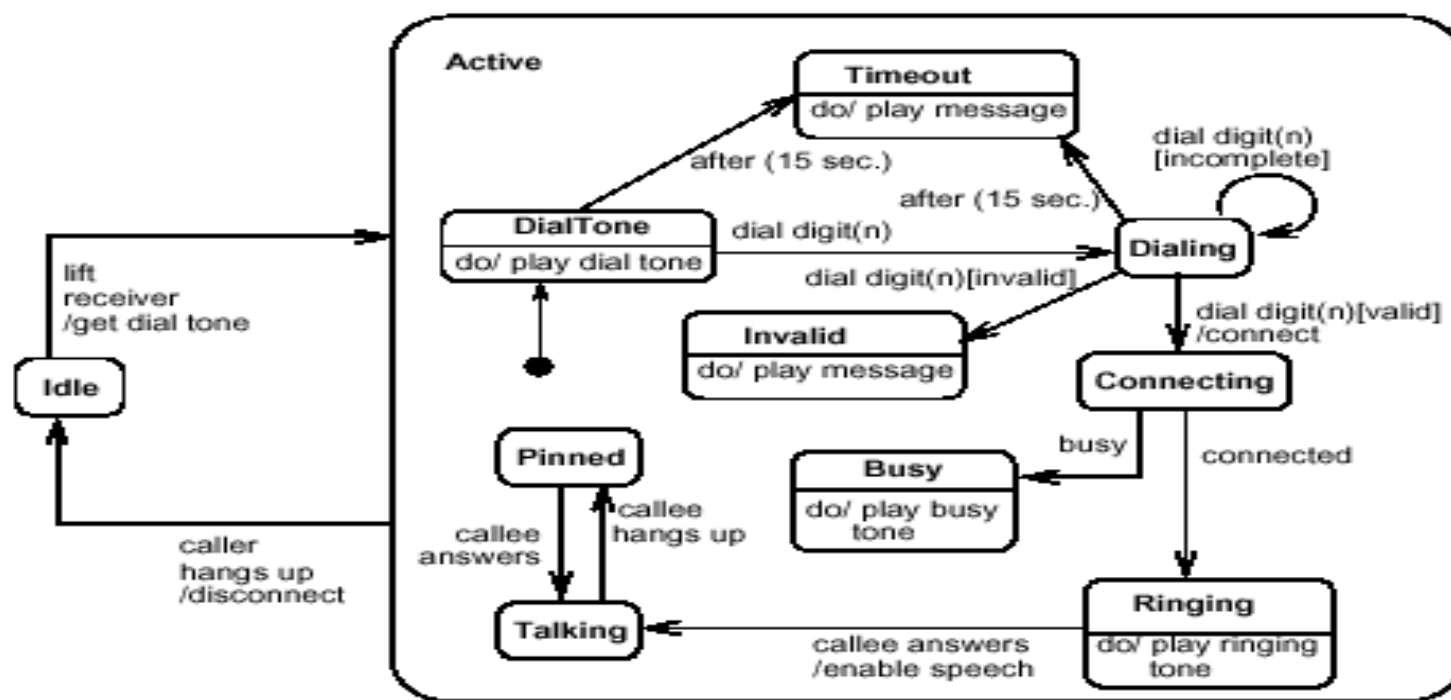




## (2) 表示法

状态图是表示状态机的图。用适当的状态表示状态机图中的状态；一般地，用连接状态的有向弧表示转换。

## (3) 实例 简单的电话状态的转换



其中：

□ 状态

— 转换



## 2) 状态

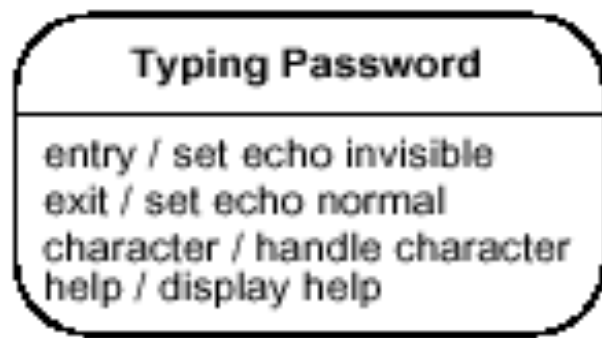
### (1) 语义

一个状态是对象在其生命期内的一个条件，或在对象满足某个条件、进行某个动作或等待某个事件的期间内的一个交互。在概念上，对象要在一个状态内维持一段时间。

在这一语义下，可以对瞬时状态建模，以及对非瞬时的交互建模。

### (2) 表示法

把一个状态表示成一个四角均为圆角的矩形。





根据需要，可以把状态划分成由水平线相互分隔的多个分栏：

### ①名称分栏

给出状态名。在同一张状态图里不应该出现具有相同名称的状态。

如果没有状态名称，那么该状态就是匿名的。同一张图中的匿名状态是各不相同的。

### ②内部转换分栏

给出在这个状态中对象所执行的内部动作或活动的列表

其一般格式为：

动作标号’/’ 动作表达式

其中：动作标号标识在该环境下要调用由动作表达式指定的动作。

动作表达式可以使用对象范围内的任何属性和链。若动作表达式为空，则可省略斜线分隔符。





下面给出专用的动作标号（注：它们不能用作事件名）：

- **entry**

该标号标识在进入状态时，执行由相应的动作表达式规定的动作（进入动作）。

- **exit**

该标号标识在退出状态时，执行由相应的动作表达式规定的动作（退出动作）。

- **do**

该标号标识正在进行的活动（“do 活动”）（只要被建模的元素是在状态中，没有完成由动作表达式指定的计算，就执行这个活动；当动作表达式指定的计算完成时，可能产生一个完成事件）。







### 3) 事件

#### (1) 语义

事件是值得注意的所发生的事情。按照状态图的具体用意，事件是指可以引发状态转换的所发生的事情。

事件可以分为：


a)条件（用布尔表达式描述）变为真。不论何时，只要条件变为真，事件都发生。（注意：这不同于监护条件。无论什么时候激发具有监护条件的事件，都对监护条件进行求值。如果求值的结果为假，转换就不发生，并且事件丢失。）

b)一个对象对另一个对象的显式信号的接收，导致一个信号事件。把这样的事件的特征标记放由它所触发的转换上。

c)对操作的调用的接收，导致一个调用事件。

d)在指定事件（经常是当前状态的入口）后，经过了一定的时间或到了指定日期/时间，导致一个时间事件。





## (2) 表示法

- 可以按如下的格式定义信号事件或调用事件：

事件名      ‘(‘用逗号分隔的参数列表‘)’

参数的格式如下：

参数名 ‘:’      类型表达式

在类图中，在类符号上用关键字<<signal>>声明信号。把该关键字放在信号名的上面，把参数说明为信号的属性。注意，信号是实例之间异步传送的消息的规格说明。

- 可以用关键词“after”和计算时间量的表达式表示时间事件，比如“after (5 秒)”或者“after (从状态A退出后经历了10秒)”。如果没指明时间起始点，那么从进入当前状态开始计时。可把其它的时间事件指定为条件，比如“when (date= 2000年1月1日)”。

- 用关键词“when”和布尔表达式表示变为真的事件。可以把它看作是连续测试条件，直到它为真。





## 4) 转换

### (1) 语义

转换是两个状态之间的关系，表示当一个特定事件出现时，如果满足一定的条件，对象就从第一个状态进入第二个状态，并执行一定的动作。


--对于这样的状态的改变，称为“触发”转换。

转换的触发器就是标注在转换上的事件。事件可能有参数，这样的参数可由转换指定的动作访问，也可由与源和目标相联系的退出和进入动作分别访问。

在状态图中，每次处理一个事件。如果事件没有触发任何转换，就丢弃它。

如果在同一个简单状态图中触发了多个转换，就只对优先级最高的那个转换点火。如果这些相冲突的转换具有相同的优先级，就随机地选择一个转换，进行触发。





## (2) 表示法

把转换表示成从源状态出发并在目标状态上终止的带箭头的实线。它可以由转换串标记。转换串的格式为：

事件特征标记 ‘[‘监护条件’]’ ‘/’ 动作表达式

其中：事件特征标记描述带参数的事件：

事件名            ‘(‘由逗号分隔的参数表‘)’

监护条件：是布尔表达式，根据触发事件的参数和拥有这个状态机的对象的属性和链来书写这样的布尔表达式。也可用监护条件显式地指定某个可达对象的状态（例如，”in State1”或”not in State2”）。

如果触发了转换，就执行动作表达式。可以根据对象的属性、操作和链以及触发事件的参数，或在其范围内的其它特征书写动作表达式。

动作表达式可以由一些有区别的动作组成的动作序列，其中包括显式地产生事件的动作，如发送信号或调用操作。表达式的细节与为模型选择的动作语言有关。

