关于其它可表达结构化事物的术语／符号

6.2.1.2 接口 -- 体现功能抽象

（1）定义：

接口（interface）是一组操作的集合，其中每个操作描述了类或构件的一个服务。

（2）接口的基本作用：模型化系统中的“接缝”

即，

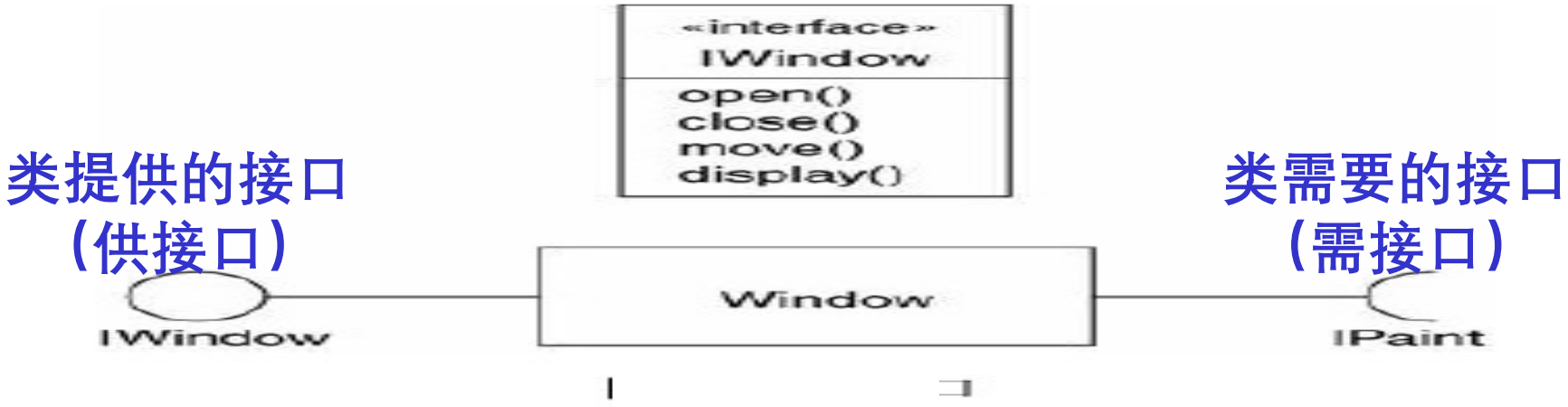
❶ 通过声明一个接口，表明一个类、构件、子系统提供了所需要的、且与实现无关的行为；

❷ 表明一个类、构件、子系统所要得到的、且与实现无关的行为。

（3）接口的表示

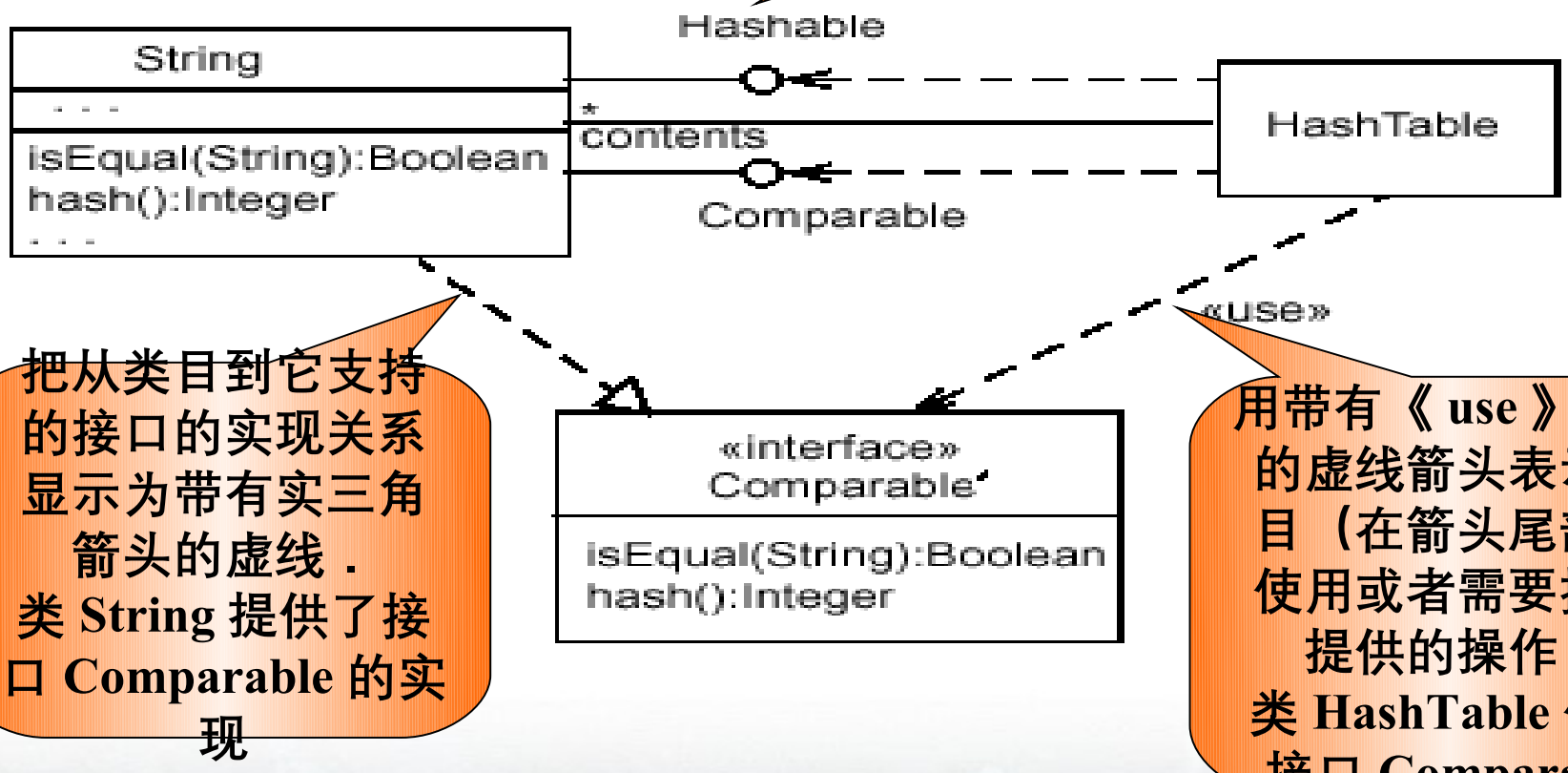
❶ 可以用带有分栏和关键字 <<interface>> 的矩形符号来表示接口。其中：●在操作分栏中给出接口支持的操作列表

●接口的属性分栏总是空的



❷ 可以用小圆圈来表示接口：

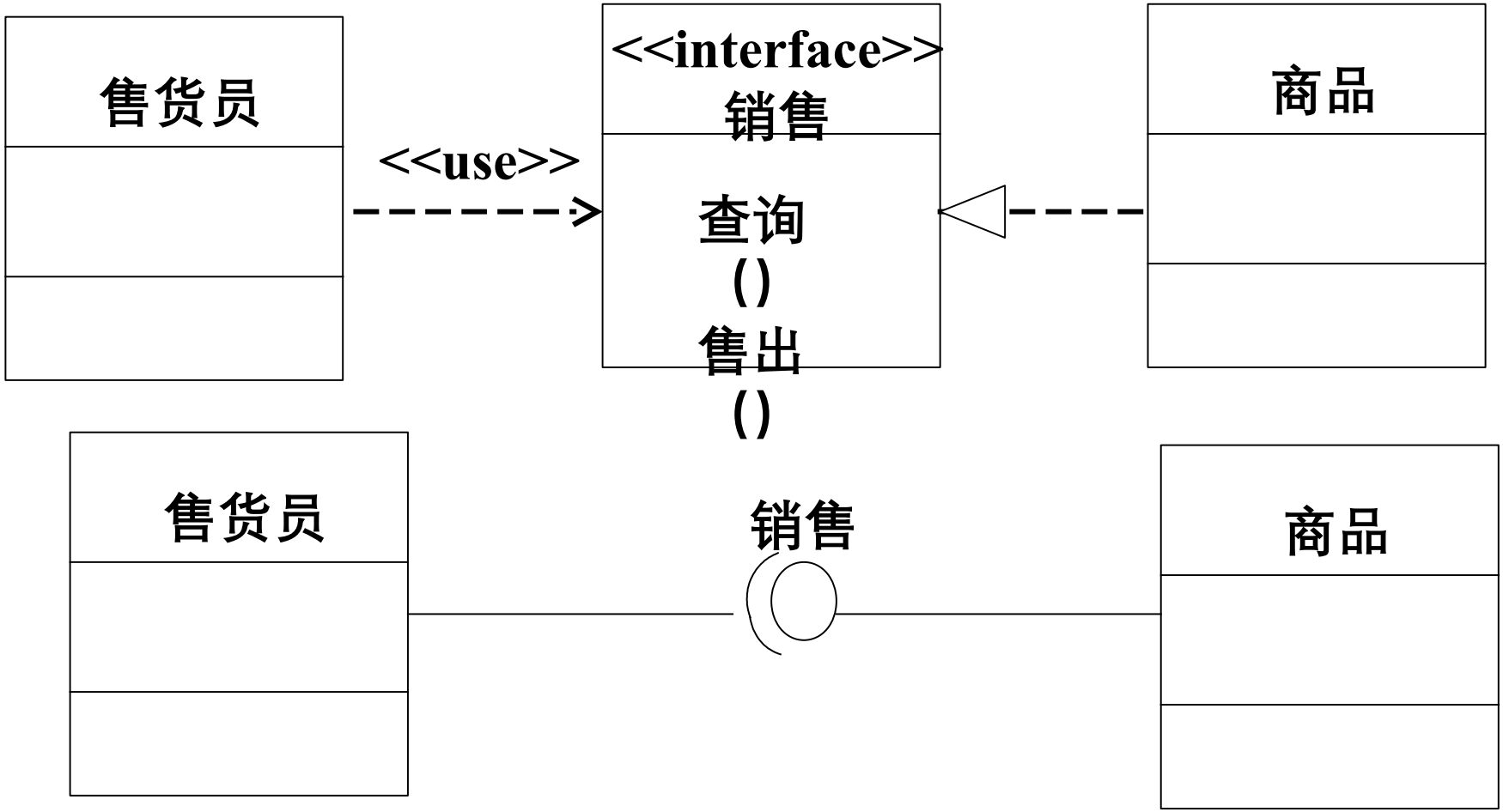
接口名放在圆圈的下面，并用实线把圆圈连接到支持它 的类目上。
类 String 支持接口 Hashable 、 Comparable ，而类 HashTable 使用接口 Hashable 、 Comparable



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．
类 String 提供了接口 Comparable 的实现

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．
类 HashTable 使用接口 Comparable

接口举例



- “销售”接口是“商品”类的供接口，是“售货员”类的需接口。
- “商品”类实现了“销售”接口，“售货员”类使用“销售”接口。

其中：

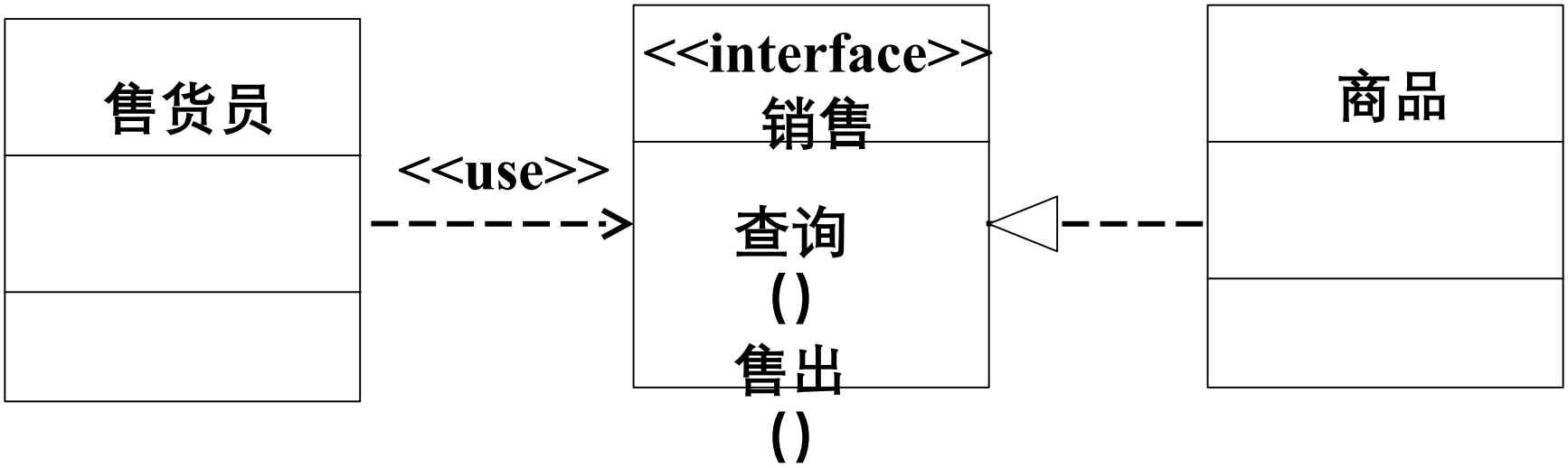
- 若用圆圈表示接口，接口名放在圆圈的下面，并用实线把圆圈连接到支持它的类目上。这意味着这个类目要提供在接口中的所有操作，其实类目提供的操作可能要更多。
- 把从类目到它支持的接口的实现关系显示为带有实三角箭头的虚线。
- 用带有《use》标记的虚线箭头表示类目（在箭头尾部）使用或者需要接口提供的操作。

显然，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。

（4）几点说明(仅以类为例)

- 接口只描述类(构件或子系统)的外部可见操作，并不描述内部结构。
- 通常，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。
- 接口只可以被其它类目使用，而其本身不能访问其它类目。
- 接口之间没有关联、泛化、实现和依赖，但可以参与泛化、实现和依赖关系。



例如：上图中接口既参与了依赖关系，又参与了实现关系。

面向对象分析和设计为什么要用接口？

6.2.1.3 协作（collaboration） -- 体现行为结构抽象

协作是一组类、接口和其他元素的群体，它们共同工作以提供比各组成部分的总和更强的合作行为。

协作是一个交互，涉及交互三要素：交互各方、交互方式以及交互内容。交互各方的共同工作提供了某种协作行为。

表示：

2点说明：

❶ 协作有两个方面：一个是结构部分，详细说明共同工作以完成该协作的类、接口和其他元素，经常用组合结构图或类图来表示；二是行为部分，详细说明这些元素如何交互，经常用交互图来表示。

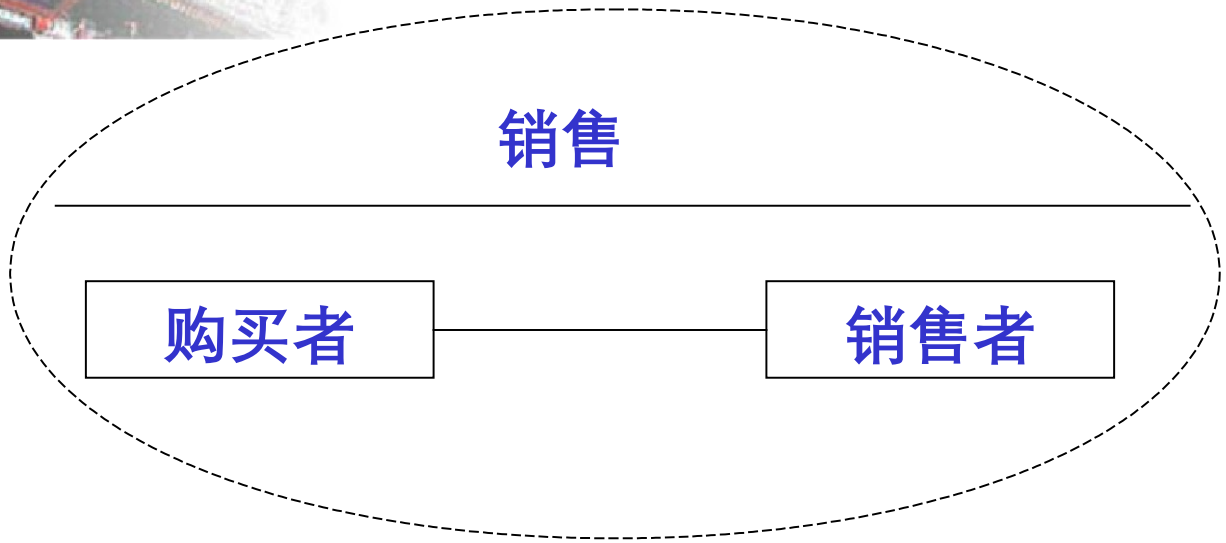


图 一个协作的组合结构图

❷ 由于一个给定的类或对象可以参与多个协作，因此协作表现了系统细化的构成模式。

注意：协作是系统体系结构的概念组块，不能拥有自己的结构元素，而仅引用或使用在其他地方声明的类、接口、构件、结点和其他结构元素。

6.2.1.4 用况（use case）

-- 体现功能抽象

是对一组动作序列的描述，系统执行这些动作产生对特定的参与者一个有值的、可观察的结果。

表示：

2点说明：

❶ 用况用于模型化系统中的行为，是建立系统功能模型的重要术语。一个用况描述了系统的一个完整的功能需求。

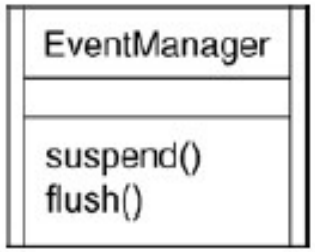
❷ 用况是通过协作予以细化的。

6.2.1.5 主动类（active class）

-- 体现并发行为抽象

是一种至少具有一个进程或线程的类，因此它能够启动控制活动。

表示：



主要特性：

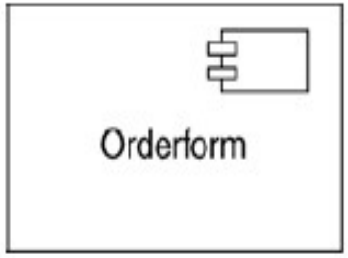
主动类对象的行为通常与其他元素的行为是并发的。

构件描述比特世界的软件制品的系统单位

6.2.1.6 构件（component）

构件是系统中逻辑的并且可替换的成分，它遵循并提供了一组接口的实现。

表示：



说明：

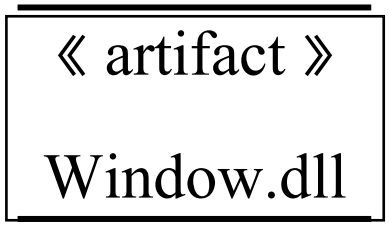
❶ 在一个系统中，共享相同接口的构件可以相互替代，但其中要保持相同的逻辑行为。

❷ 构件可以包含更小的构件。

6.2.1.7 制品（artifact）

是系统中物理的、可替代的部件，其中包含物理信息 (比特).

表示：



2 点说明

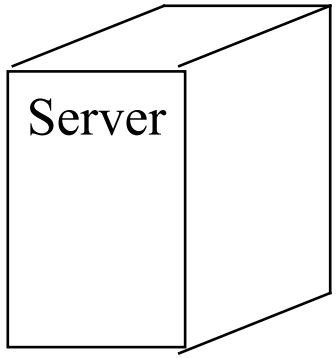
❶ 在一个系统中，可能会存在不同类型的部署制品，例如源代码文件、可执行程序和脚本等。

❷ 制品通常代表对源代码信息或运行时信息的一个物理打包

6.2.1.8 节点（node）

是在运行时存在的物理元素，通常它表示一种具有记忆能力和处理能力的计算机资源。

表示：



1 点说明：

❶ 一个构件可以驻留在一个节点中，也可以从一个节点移到另一个节点。

结构化地表达客观事物的术语小结

- 抽象客观世界中任何实体的基本术语

  UML 给出了以上八个术语 (模型化概念)

  -- 类、接口、协作、用况、主动类、构件、制品、节点，

  它们是可包含在一个 UML 模型中的基本模型化元素 .

  它们存在一些变体，例如：

  类的变体 - 参与者、信号、实用程序；

  主动类的变体 - 进程和线程；

  制品的变体 - 应用、文档、库、页和表等。

- 在 UML 中，把以上结构化概念统称为类目 （ classifier ）

6.2.2 包

为了组织类目，控制信息组织和文档组织的复杂性，UML 引入了术语 - 包。

6.2.2.1 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的模型元素。

一个包元素对外的可见性，可以通过在该元素名字前加上可见性符号（+ ：公共的， - ：私有的， # ：受保护的）来指示：

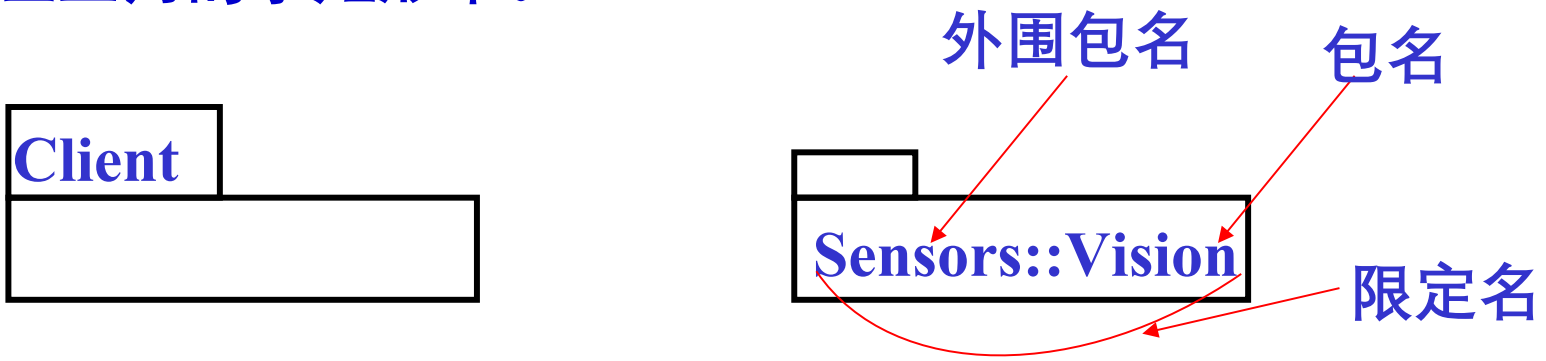
+ ：对其他包而言都是可见的；

- ：对其他包而言都是不可见的；

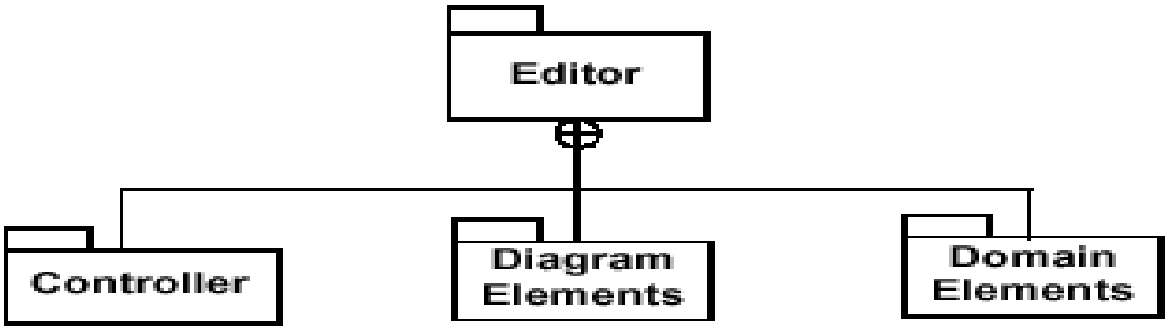
：对子孙包而言是可见的；

6.2.2.2 表示

❶ 通常，在大矩形中描述包的内容，而把该包的名字放在左上角的小矩形中。



❷ 可以把所包含的元素画在包的外面，通过符号⊕，将这些元素与该包相连。这时可把该包的名字放在大矩形中。



包拥有在其内所声明的模型元素，它们可以是类、接口、构件、协作、用况、节点，甚至可以是其他包。

6.2.2.3 包之间的关系

两种依赖：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。

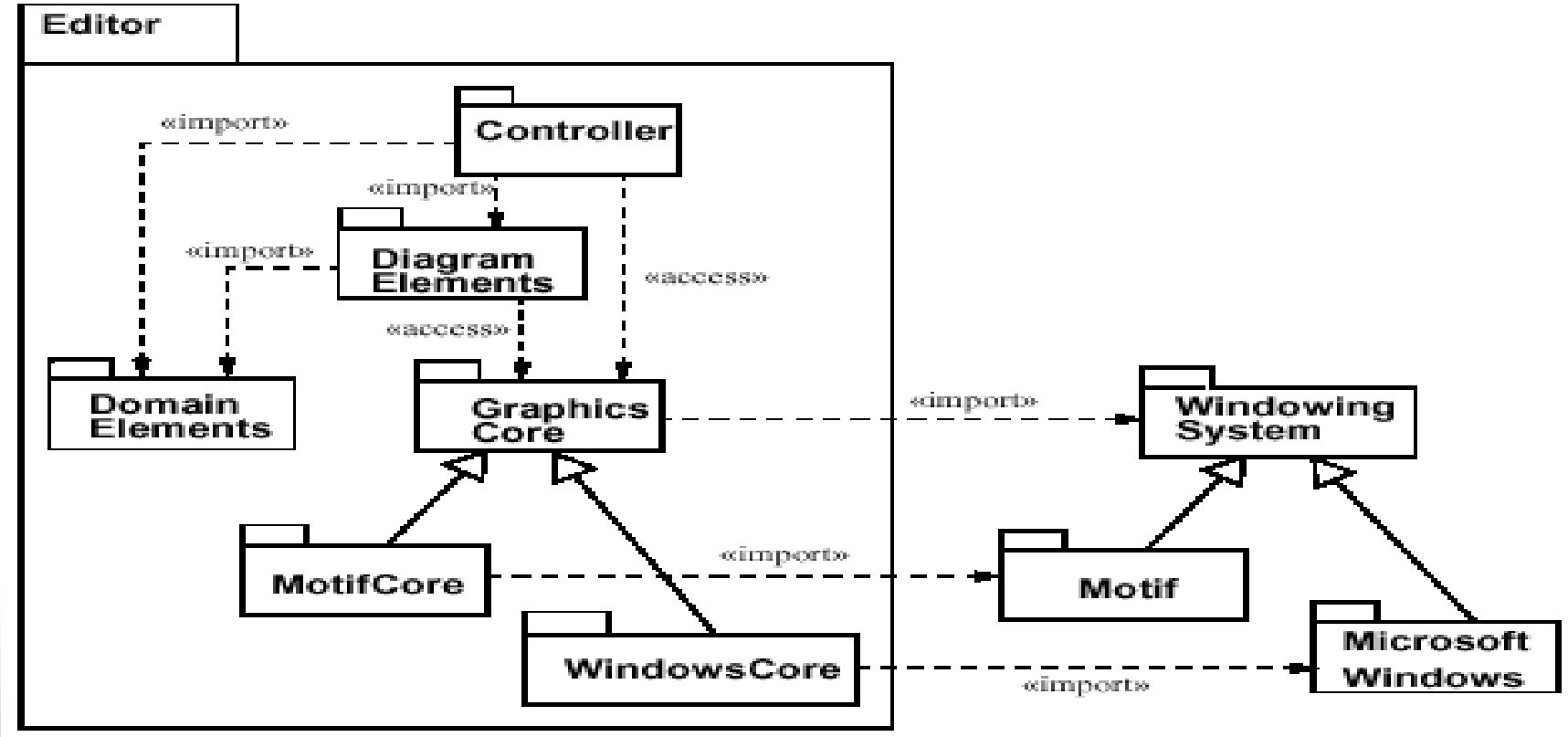
注：包间的依赖通常隐含了各包中元素之间存在着的一个或多个依赖。

（1）引入依赖：《import》

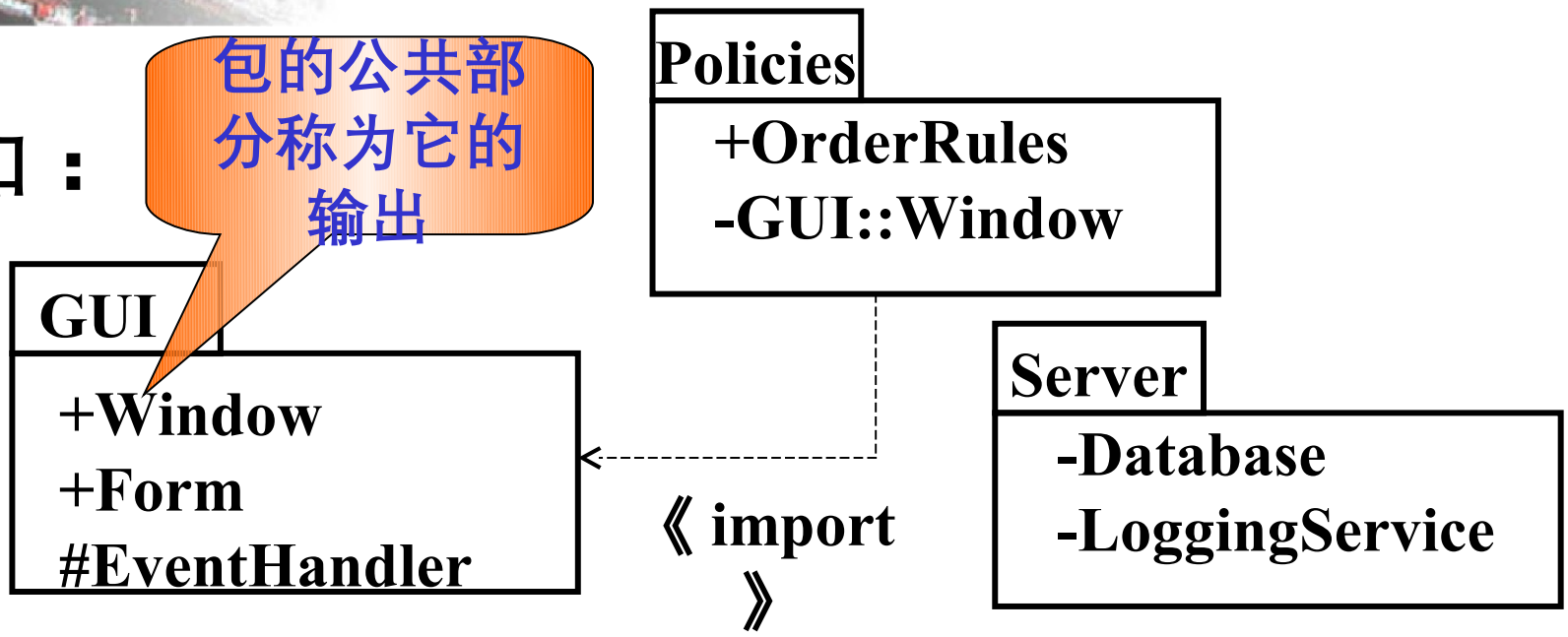
从源包到目标包的引入依赖表明：目标包中有适当可见性的内容被加入到源包的公共命名空间中，这相当于源包对它们做了声明（即对它们的引用可不需要一个路径名）

引入：《import》

表示为从源包到目标包的一条带箭头的线段，并标记为《import》，如下图所示：



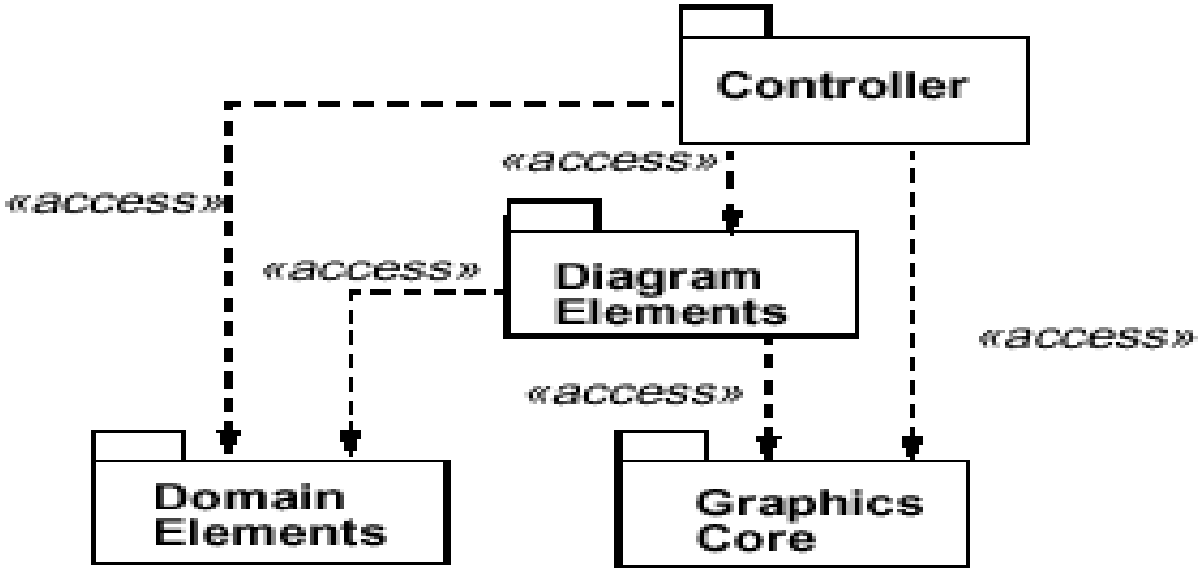
例如：



注：包 Policies 引入包 GUI，因此，对于类 GUI::Window 和类 GUI::Form，包 Policies 的内容使用简单名 Window 和 Form 就能访问它们，然而，由于 GUI::EventHandler 是受保护的，因此它是不可见的．由于包 Server 没有引入包 GUI， Server 中的内容必须用限定名才能访问 GUI 的公共内容，如 GUI::Window 。由于 Server 的内容是私有的， GUI 的内容无权访问 Server 中的任何内容，即使用限定名也不能访问它们．

（2）访问依赖：《access》

从源包到目标包的访问依赖表示：目标包中具有可见性的内容增加到源包的私有命名空间里（即源包可以不带限定名来引用目标包中的内容，但不可以输出之，即如果第三个包引入源包，就不能再输出已经被引入的目标包元素）。



注：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。

6.2.2.4 对成组的元素建模策略：

- 浏览特定体系结构视图中（如类图）的建模元素，找出概念或语义上相互接近的元素所定义的组块．
- 把每一个这样的组块围在一个包内．
- 对每一个包判别哪些元素要在包外访问，把这些元素标记为公共的，把所有其他元素标记为受保护的或私有的．
- 用引入依赖显示地连接建立在其他包之上的包．
- 在包的家族中，用泛化关系把特殊包连接到它们的较一般的包．

6.2.3 表达关系的术语

在 UML 中，提供了以下 4 种关系，作为 UML 模型中的基本关系构造块，表达类目之间的关系，以构造一个结构良好的 UML 模型 .

① 关联 (association)

② 泛化 （ generalization ）

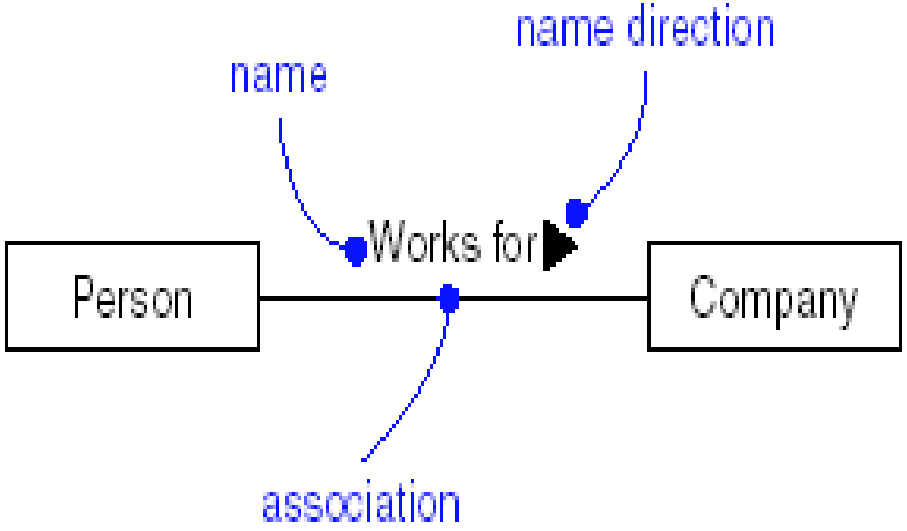
③ 实现 （ realization ）

④ 依赖 (dependency)

① 关联 (association)

定义：关联是类目之间的结构关系，描述了一组具有相同结构、相同语义的链（links）.

链是对象之间的连接（connection）。例如：



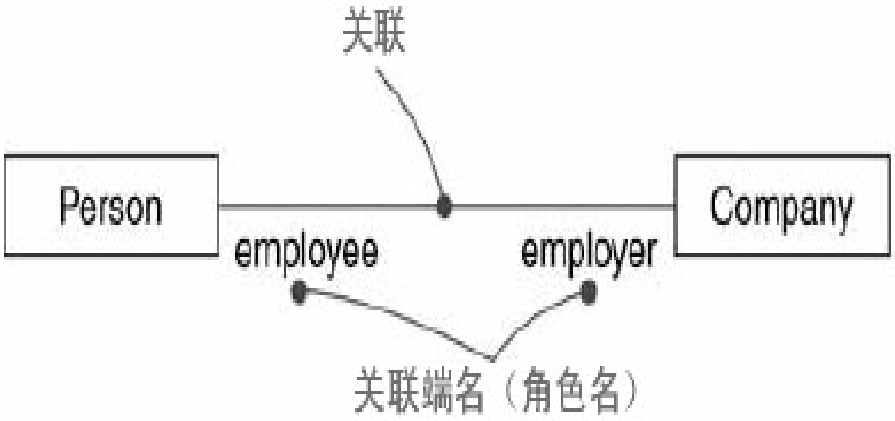
注：如一个关联只连接两个类目，称为二元关联；

如一个关联连接 n 个类目，称为 n 元关联。

关联的语义表达（6点）：

❶关联名 (name): 关联的标识，用于描述该关联的“涵义”。为了避免该关联涵义上的歧义性，可给出其关联方向。

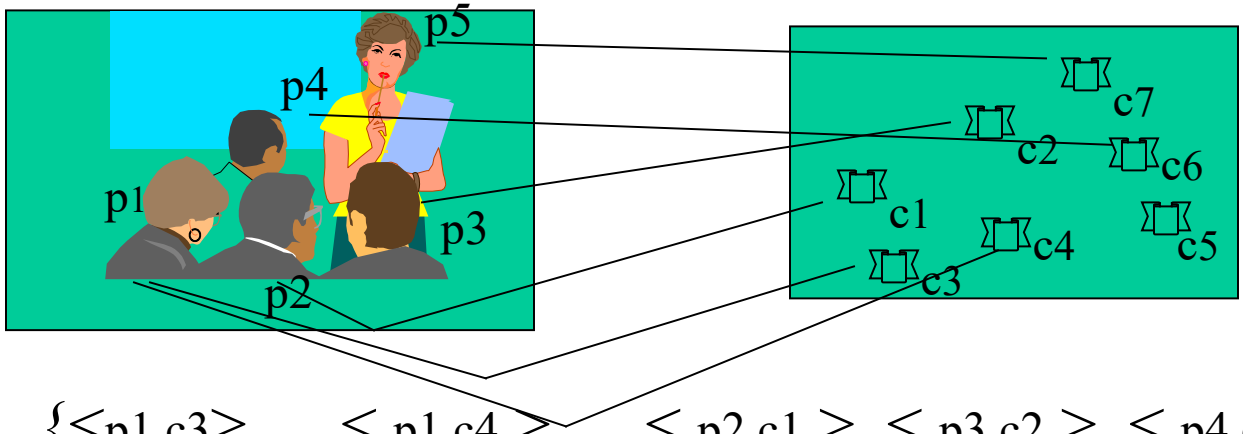
❷ 角色名（role）：一个类参与一个关联的角色标识。在类的一个关联中，可以显式地命名该角色，如下所示：



注：

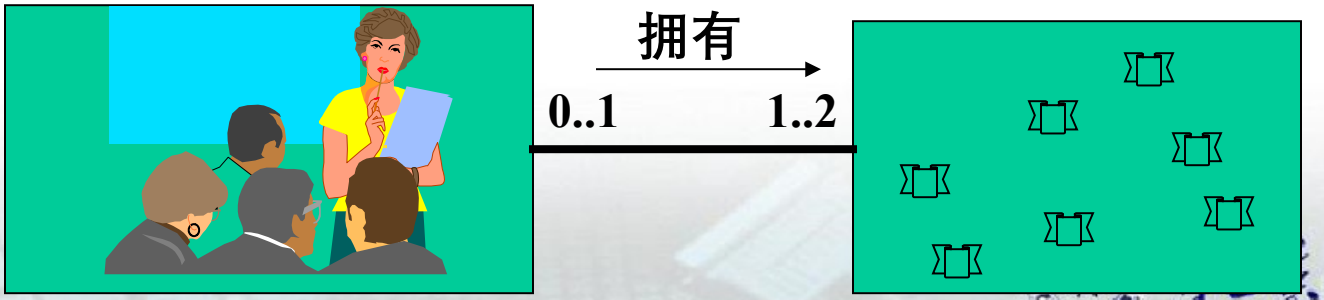
- 在明确给出关联端名的情况下，通常可以不给出该关联名。但若一个类有多个关联，可使用关联名或端点名来区分它们。若一个类有多个端点，可使用端点名来区分它们 .
- 同一个类可以在其它关联中扮演相同或不同的角色 .

❸ 多重性（multiplicity）：类中对象参与一个关联的数目，称为该关联角色的多重性。例如：



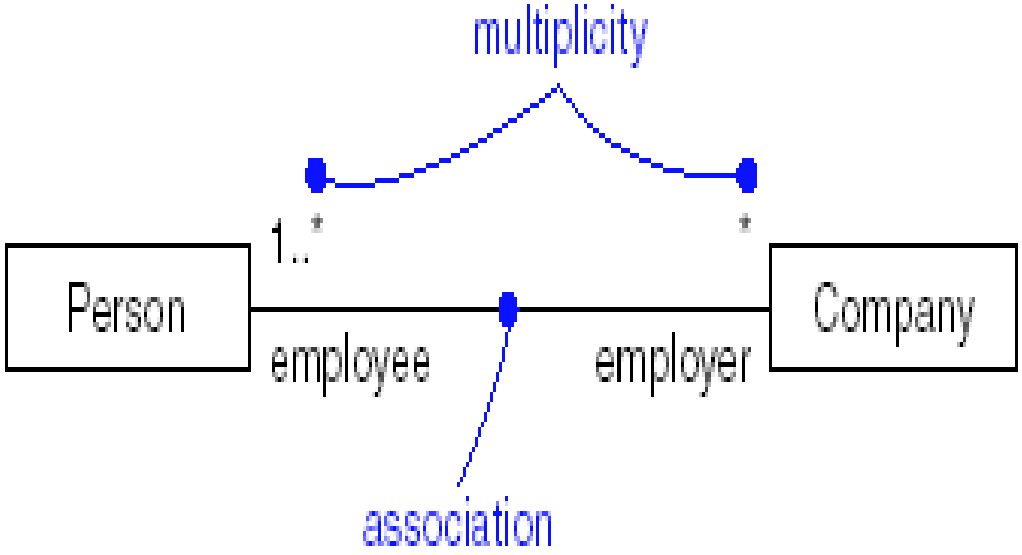
拥有关系：{<p1,c3>，<p1,c4>，<p2,c1>,<p3,c2>,<p4,c6>,<p5,c7>}

模型化为



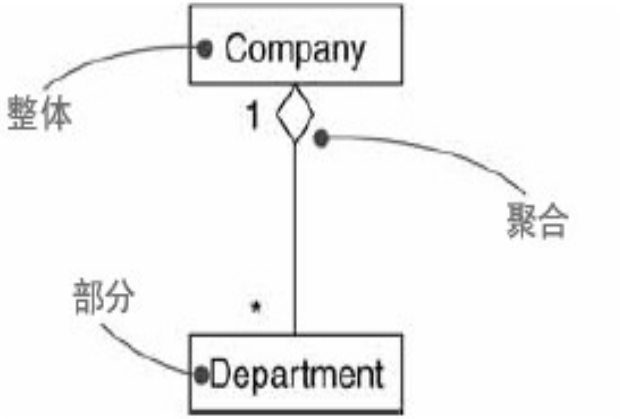
多重性的表达：

关联的一端的多重性，说明：对于关联另一端的类的每个对象，本端的类可能有多少个对象出现。



上图多重性解释：每个公司对象可以雇佣一个或多个人员对象（多重性为 1..* ）；每个人员对象受雇于 0 个或多个公司对象（多重性为 *, 它等价于 0..*）

❹ 聚合（aggregation）：一种特殊形式的关联，表达一种“整体/部分”关系。即一个类表示了一个大的事物，它是由一些小的事物（部分）组成的。



注意：不论是整体类还是部分类，它们在概念上是处于同一个层次的。

- 在建模实践中，这是区分是否把一类事物标识为一个部分类还是把它标识为一个类的属性的基本准则。

组合（composition）

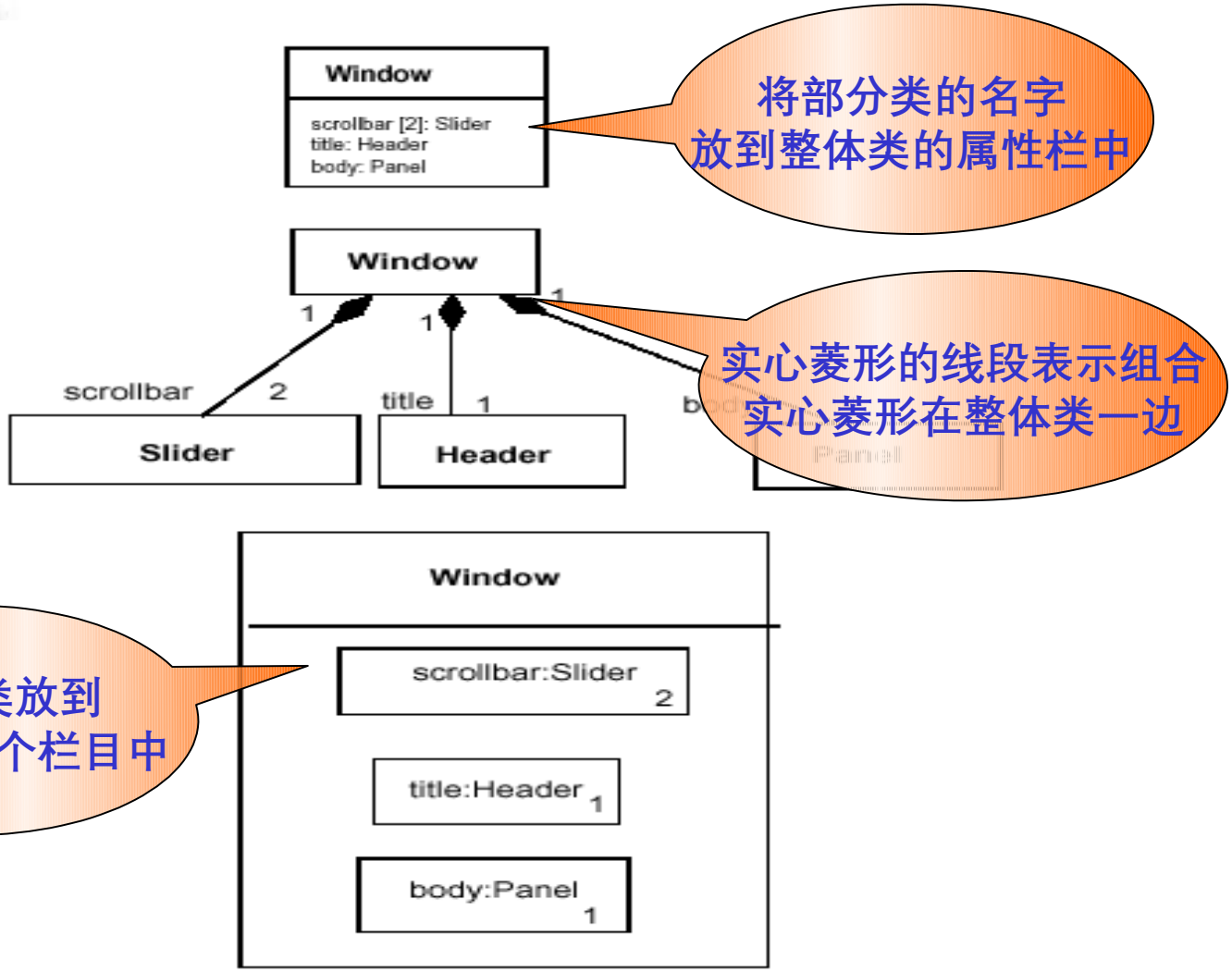
定义：如果整体类的实例和部分类的实例具有相同的生命周期，这样的聚合称为组合。

4点说明：

- 组合是聚合的一种形式。部分和整体之间具有很强的“属于”关系，即具有一致的生存期；
- 组合的末端，其多重性显然不能超过1；
- 在一个组合中，由一个链所连接的对象而构成的任何元组，必须都属于同一个整体类的对象；
- 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。

表示：

表示组合的
不同方法



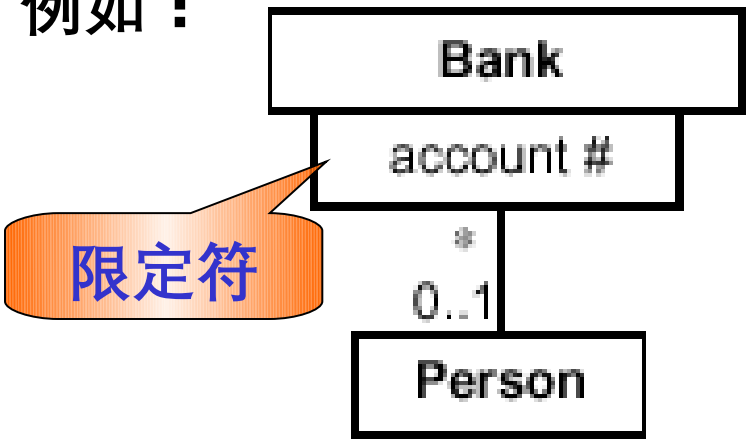
该例给出了三种表示组合的方法。

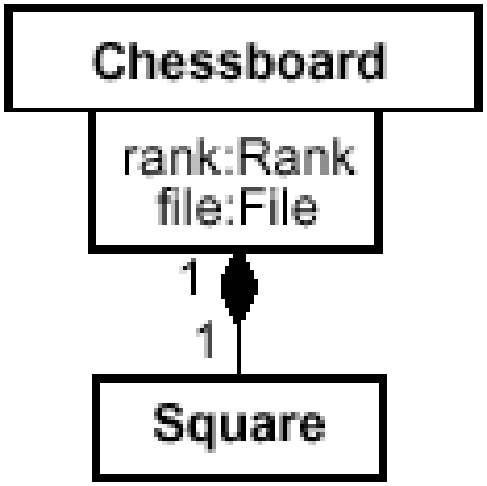
其中，类 Window 由类 Silder （角色为 Scrollbar ）、 Header （角色为 title ） 和 Panel （角色为 body ）组成。

❺ 限定符：

一个限定符是一个关联的属性或属性表，这些属性的值将对该关联相关的对象集做了一个划分。

例如：



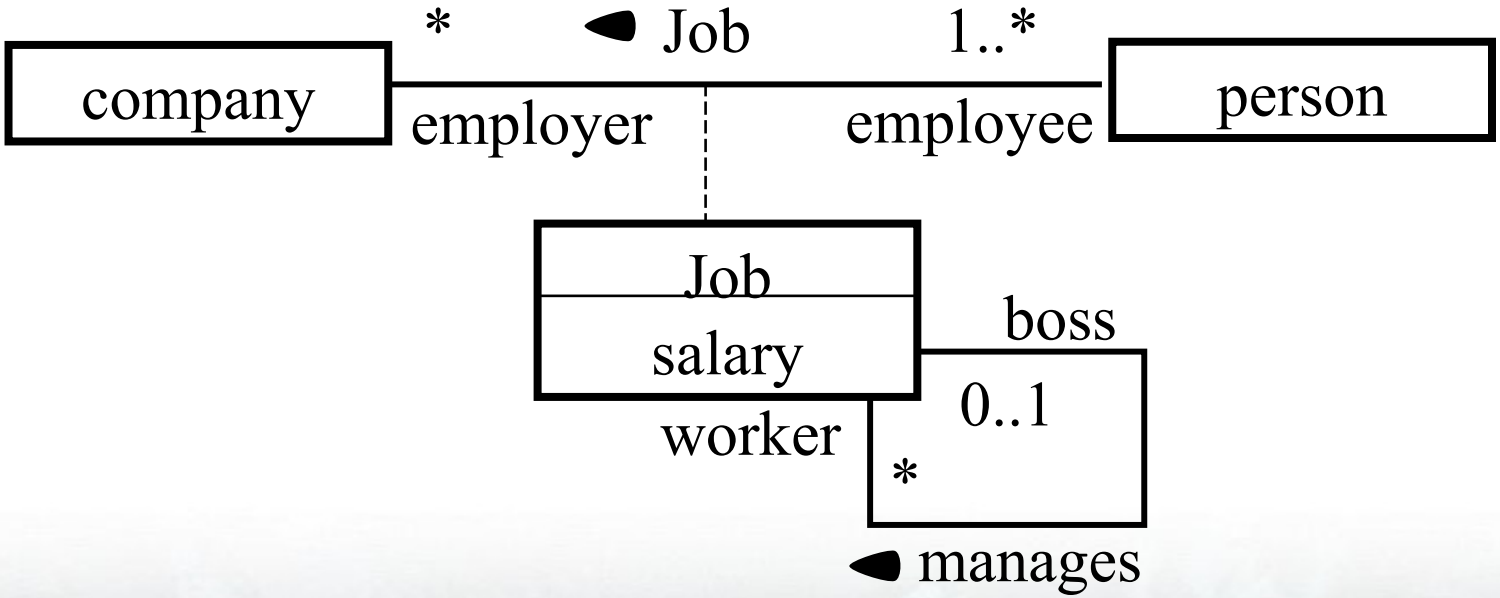


左图的限定符有一个属性 account#，表明：在一个银行中，一个帐户对应
一个用户，或没有对应人员。

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。

❻ 关联类

一种模型元素，它有关联和类的特性。一个关联类，可以被看作是一个关联，但还有类的特性；或被看作是一个类，但有关联的特性。例如：



注意：

- 如果关联类只有属性而没有操作或其他关联，名字可以显示在关联路径上，从关联类符号中省去，以强调其“关联性质”。
- 如果它有操作和其他的关联，那么可以省略路径中的名字，并将他们放在类的矩形中，以强调其“类性质”。
- 在关联路径的两端可能都具有通常的附属信息，类符号也可以具有通常的内容，但在虚线上没有附属信息。
- 尽管把一个关联类画成一个关联和一个类，但它仍然是一个单一的模型元素。

② 泛化（ generalization ）

定义：

泛化是一般性事物（称为超类或父类）和它的较为特殊种类

（称为子类）之间的一种关系，有时称为“ is-a-kind-of” 关系。

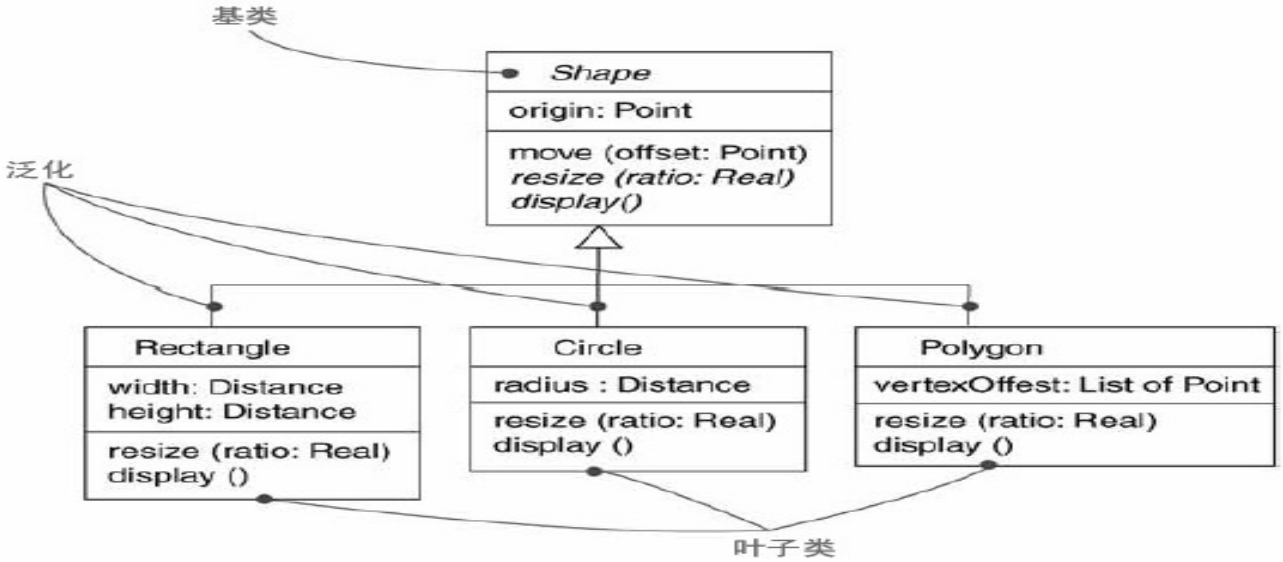
4 点说明：

❶ 子类可继承父类的属性和操作，并可有更多的属性和操作；

❷ 子类可以替换父类的声明；

❸ 若子类的一个操作的实现覆盖了父类同一个操作的实现，这种情况被成为多态性，但两个操作必须具有相同的名字

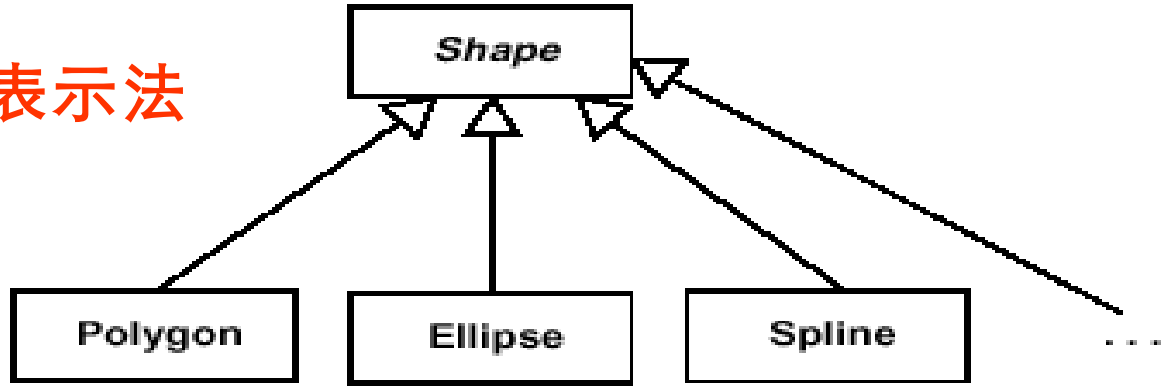
❹ 一个类可以有 0 个、 1 个或多个父类。没有父类且最少有一个子类的类被称为根类或基类；没有子类的类称为叶子类。如果一个类只有一个父类，则说它使用了单继承；如果一个类有多个父类，则说它使用了多继承。



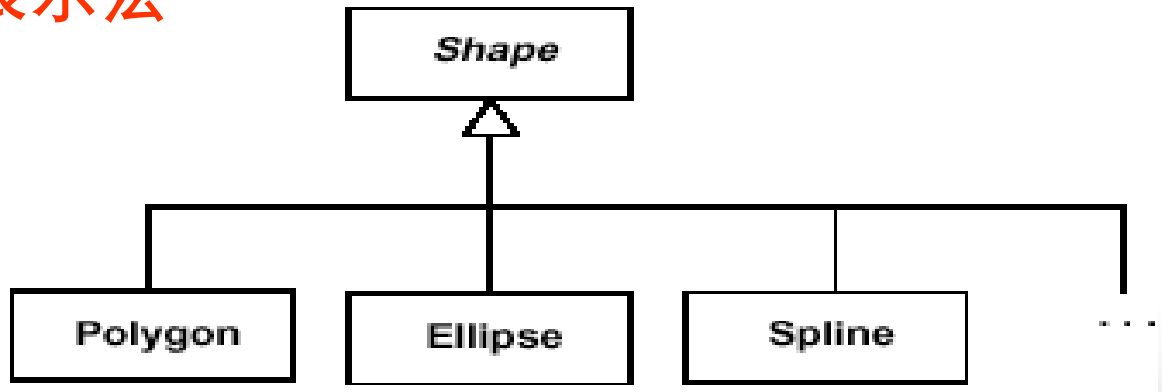
注：在大多数情况中，用类和接口之间的泛化来表明继承关系。在 UML 中，也可在其他类目之间创建泛化，例如在结点之间。

表示：

分离表示法



共享表示法



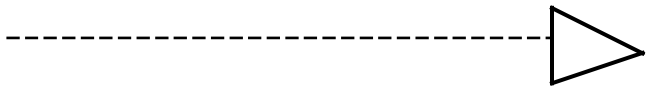
③ 细化（也称为实现）（ realization ）

定义：细化是类目之间的一种语义关系，其中一个类目规约了保证另一个类目执行的契约。

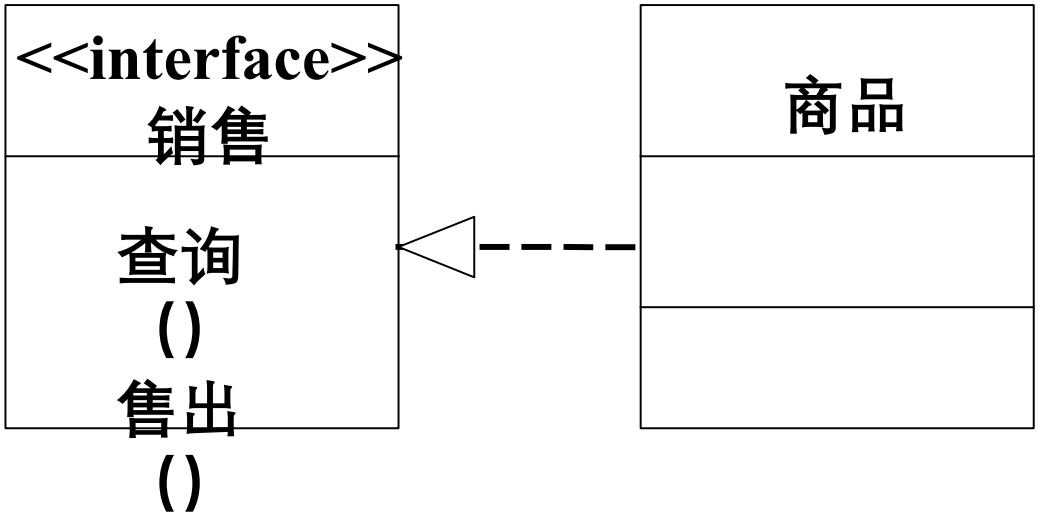
说明：在以下 2 个地方会使用细化关系：

- 接口与实现它们的类和构件之间；
- 用况与实现它们的协作之间。

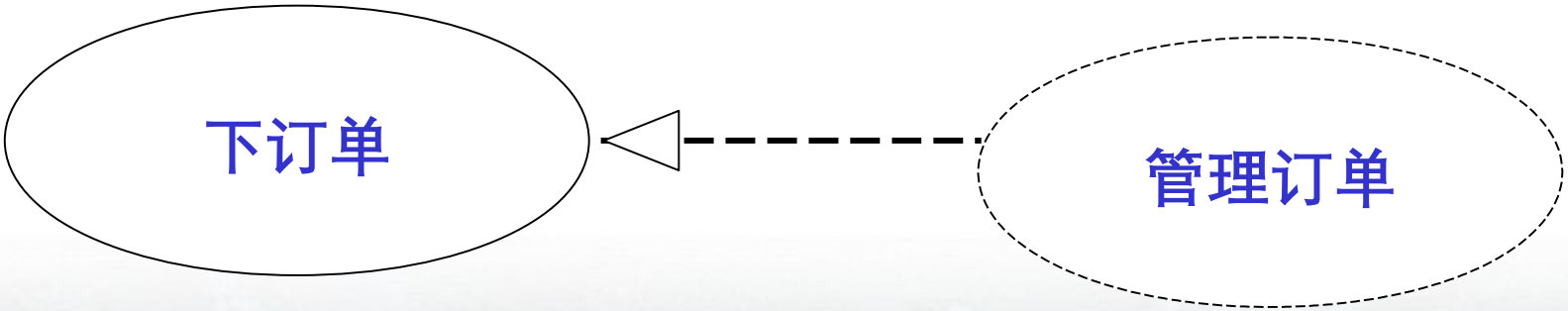
表示：



例 1 ：接口和实现它们的类之间的关系



例 2 ：用况和实现它们的协作之间的关系



④ 依赖

定义：依赖是一种使用关系，用于描述一个事物（如类 Window ）使用另一事物（如类 Event ）的信息和服务。

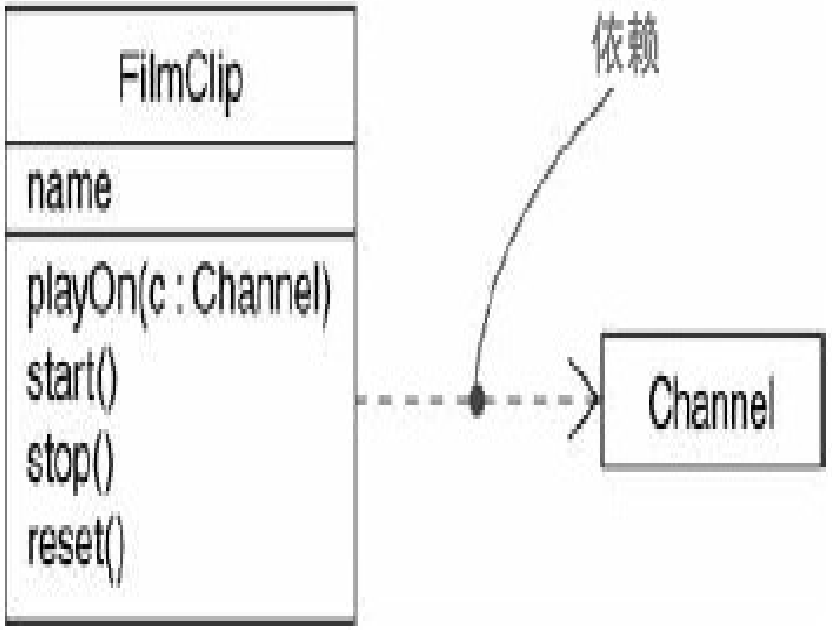
3 点说明：

❶ 在大多数情况里，使用依赖来描述一个类使用另一个的操作；

❷ 如果被使用的类发生变化，那么另一个类的操作也会受到影响；

❸ 依赖可用于其它事物之间，例如注解之间和包之间。

表示：一条有向虚线。例如：



为了进一步表达依赖的语义，UML 对依赖进行了分类，并给出了相应的标记。

❶ 绑定（bind）：表明源的实例化是使用目标给定的实际参数来达到的。例如，可以把模板容器类（目标）和这个类实例（源）之间的关系模型化为绑定。其中绑定涉及到一个映射，即实参到形参的映射。

❷ 导出（derive）：表明可以从目标推导出源。例如类 Person 有属性“生日”和“年龄”，假定属性“生日”是具体的，而“年龄”是抽象的，由于“年龄”可以从“生日”导出，因此可以把这两个属性之间的这一关系模型化为导出。

❸ 允许（permit）：表明目标对源而言是可见的。一般情况下，当许可一个类访问另一个类的私有特征时，往往把这种使用关系模型化为允许。

❹ 实例（instanceOf）：表明源的对象是目标的一个实例。

❺ 实例化（instantiate）：表明源的实例是由目标创建的。

❻ 幂类型（powertype）：表明源是目标的幂类型。幂类型是一个类目，其对象都是一个给定父类的子类。

❼ 精化（refine）：表明源比目标更精细。例如在分析时存在一个类 A，而在设计时的 A 所包含的信息要比分析时更多。

❽ 使用（use）：表明源的公共部分的语义依赖于目标的语义。

以上谈到的 4 个术语，是 UML 模型中可以包含的基本关系。

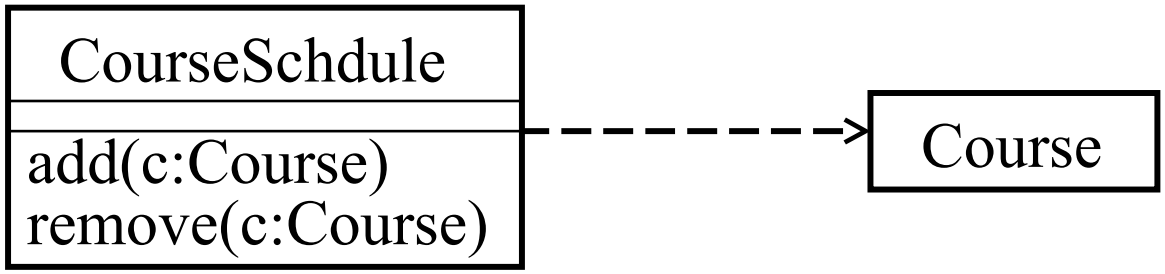
它们也有一些变体，例如精化、跟踪、包含和扩展等。

四种关系的一般用法：

① 模型化简单依赖

例如，一种常见的依赖关系是：一个类只是使用另一个类作为它的操作参数。

对此，可从含有操作的类到被该操作用做参数的类创建一个依赖。即：



注：如果操作 add 和 remove 给出了明显的操作标记（c:Course，如上所示），则一般就不需要给出这个依赖；但当省略操作标记时或一个模型还描述了被使用类的其它关系时，就应显示这一依赖。

② 模型化单继承

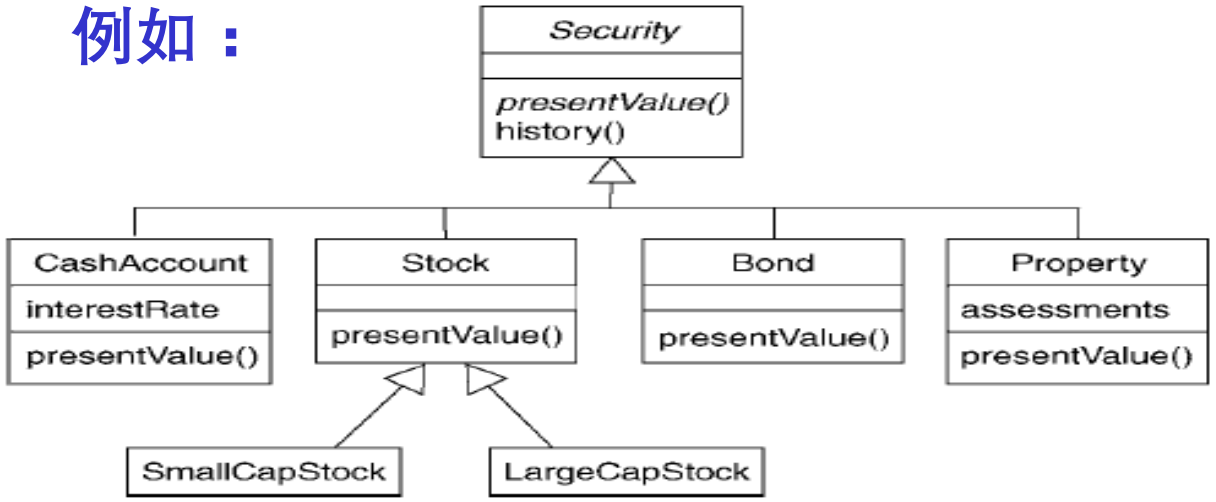
第一步：对于给定的一组类，发现 2 个或 2 个以上类的共同责任、属性和操作。

第二步：把发现的共同责任、属性和操作放到一个一般类中

其中要注意，不要引入过多的层次。

第三步：画出从每个特殊类到一般类（父类）的泛化关系。

例如：



注：

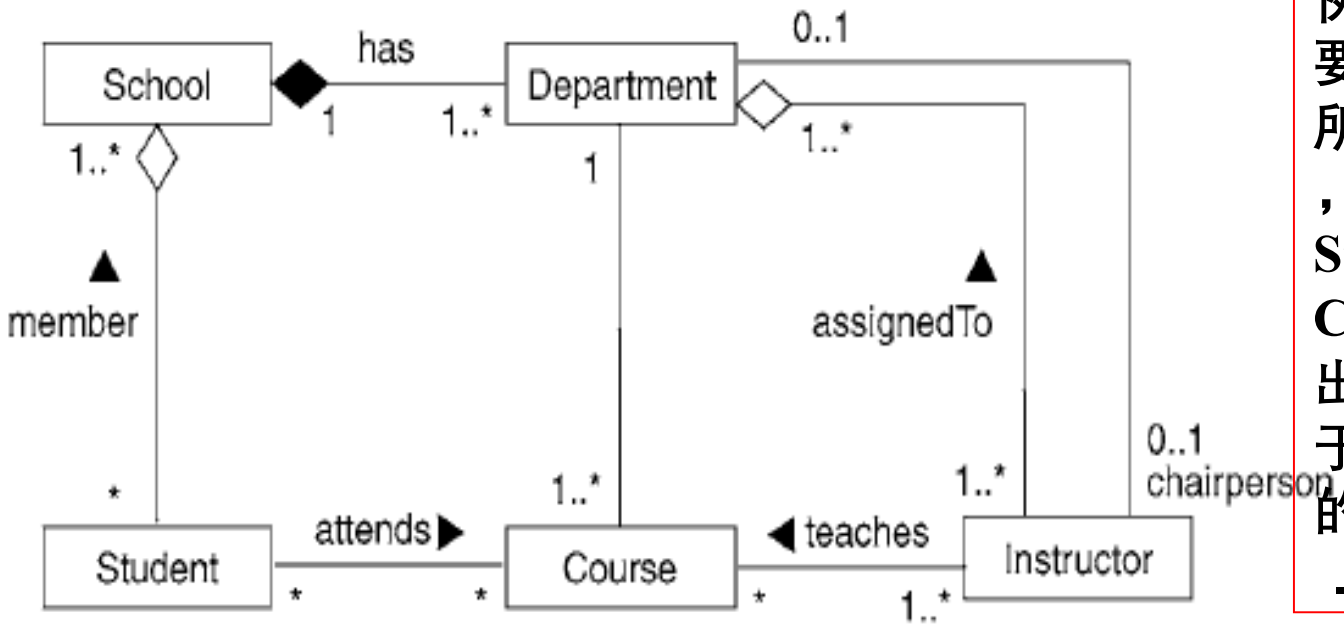
- 斜体字表明是一个抽象类或抽象操作；
- 子类中给出的操作为非斜体字，表明给出了操作的实现。

③ 模型化结构关系

第一步：标识关联

若对于每一个类，需要导航到另一个类的对象，那么就要在这 2 个类之间给出一个关联。

——这是关联的数据驱动观点。



例如：
要了解 Student 所要参与的课程，因此就应在 Student 和 Course 之间给出一个关联，用于描述学生参与的课程；

. . . . . .

若对于每一个类的对象需要与另一个类的对象进行交互，并且后一个对象不作为前一个对象的局部变量或操作参数，那么就要在这 2 个类之间给出一个关联。

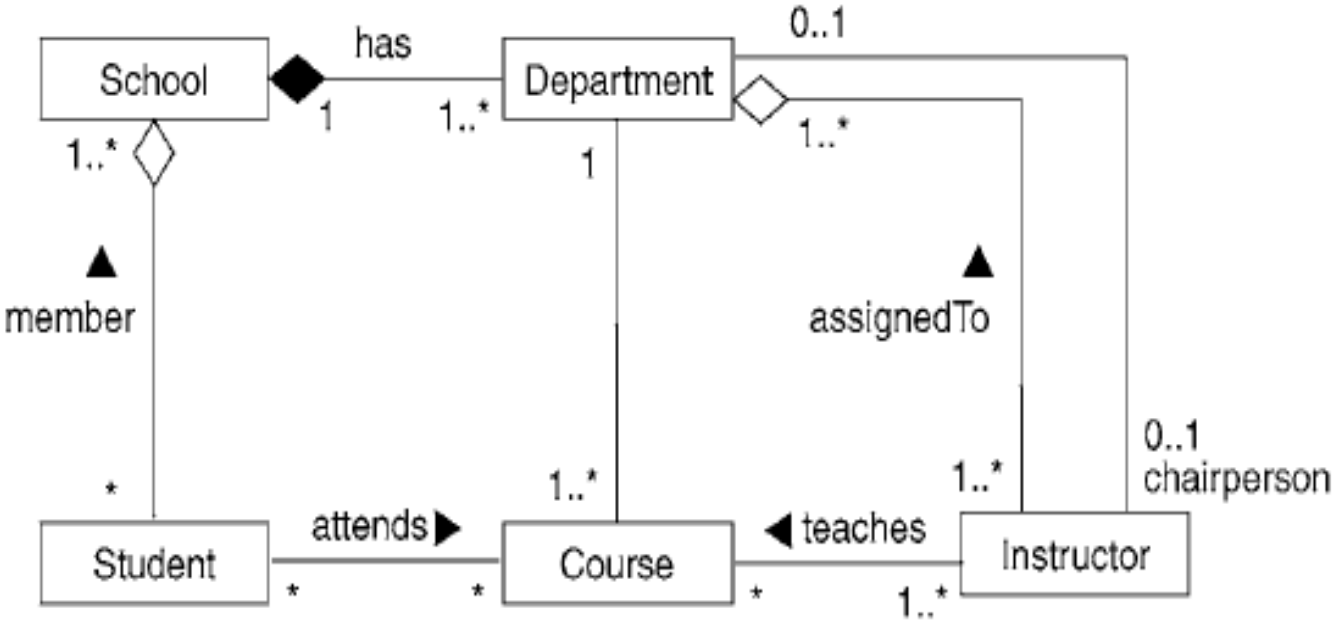
－－这是关联的行为驱动观点。

第二步：对于标识的每一个关联，添加语义描述

例如，就下图而言，给出关联的多重性：

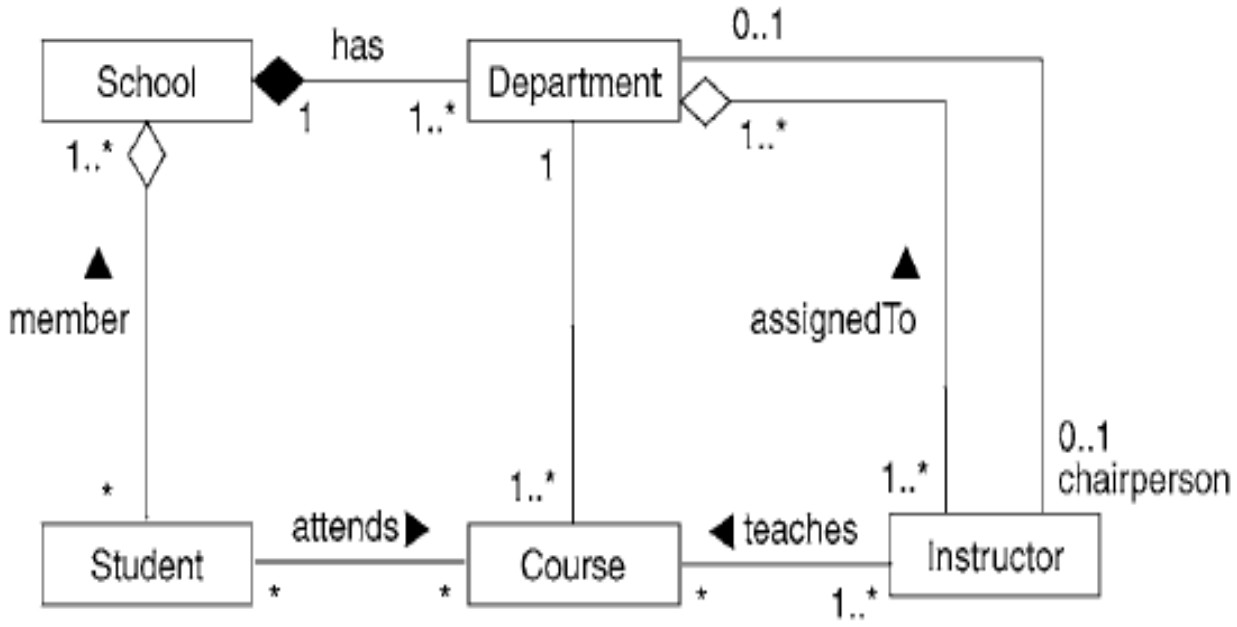
－－每门课程至少有一名教师，而一名教师可以教多门课程。

－－每门课程是精确地属于一个系的。



第三步：标识“整体／部分”

如果关联中的一个类与另一端的类相比，前者在结构上或组织上是一个整体，而后者似乎是它们的一部分，那么就要把它们标识为聚合，例如，见下图：



聚合：一所学校可以有 0 到多名学生，一个学生可以注册在一所或多所学校学习；

聚合：一所学校可以有一个或多个系，而每个系只能属于一所学校；

注意：在该例中， Department 和 Instructor 之间有两个关联，其中：一个关联（聚合）说明可以指派一名教师到一个或多个系中工作，而一个系可以有一名或多名教师；另一关联表明一个系只能有一名教师作系主任，而某些教师不是系主任。

基本策略

在用 UML 对关系建模时，要遵循以下策略：

- 仅当要建模的关系不是结构关系时，才使用依赖。

  这条策略意味着什么？

- 仅当关系是“ is-a-kind-of” 关系时，才使用泛化。

  聚合可否替代多继承？

- 一般不要引入循环的泛化关系。
- 应保持泛化关系的平衡：继承的层次不要多深，不要过宽（如果出现这种情况，就要寻找可能的中间抽象类）。

小结

UML 的术语表 - 元信息，包括：

- 可用于抽象客观世界中任何实体的基本术语

  类、接口、协作、用况、主动类、构件、制品、节点，

  以及相关的变体。

  在 UML 中，把以上结构化概念统称为类目（classifier）
- 可用于组织信息的术语—包
- 可用于解释信息的术语 —注解
- 可用于抽象客观世界中任何实体关系的基本术语

  关联，泛化，细化，依赖，以及相关的特殊形式.

  其中为了增强关系语义的表达，还给出了一些基本概念，例如