



## 关于其它可表达结构化事物的术语 / 符号

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的塞服务用：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。



北京大学

### (3) 接口的表示

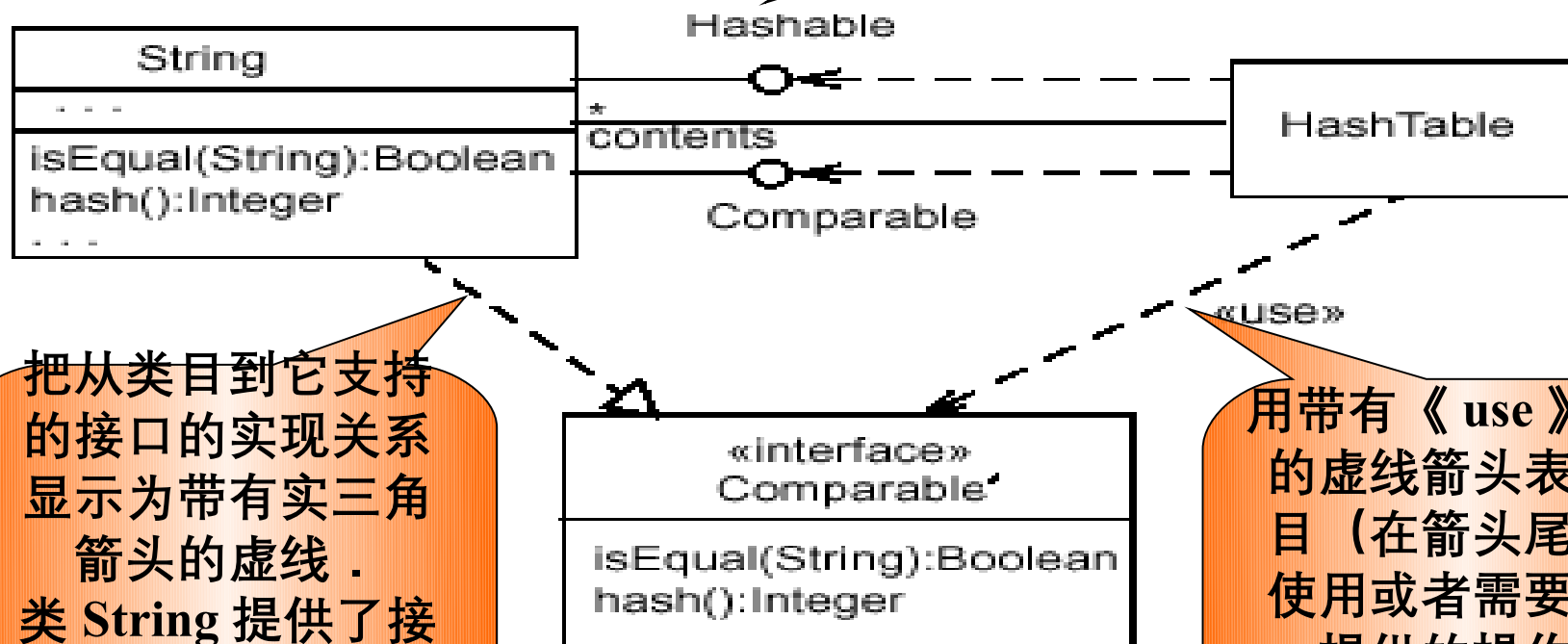
- ① 可以用带有分栏和关键字 <<interface>> 的矩形符号来表示接口。其中：
  - 在操作分栏中给出接口支持的操作列表
  - 接口的属性分栏总是空的





接口名放在圆圈的下边，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

## ② 可以用小圆圈来表示接口：



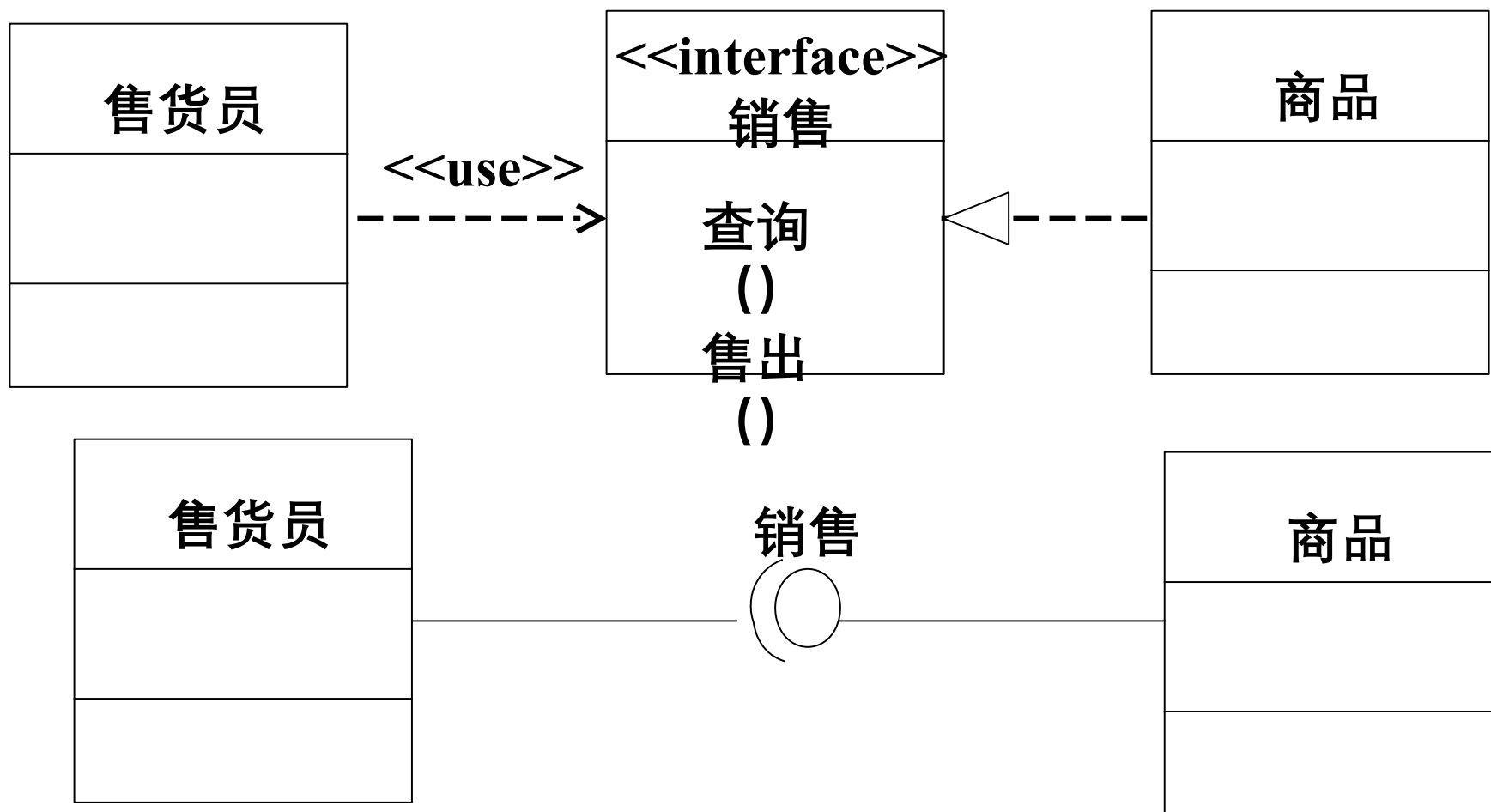
把从类目到它支持的接口的实现关系显示为带有实三角箭头的虚线。  
类 String 提供了接口 Comparable 的实现

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



北京大学

# 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。



北京大学

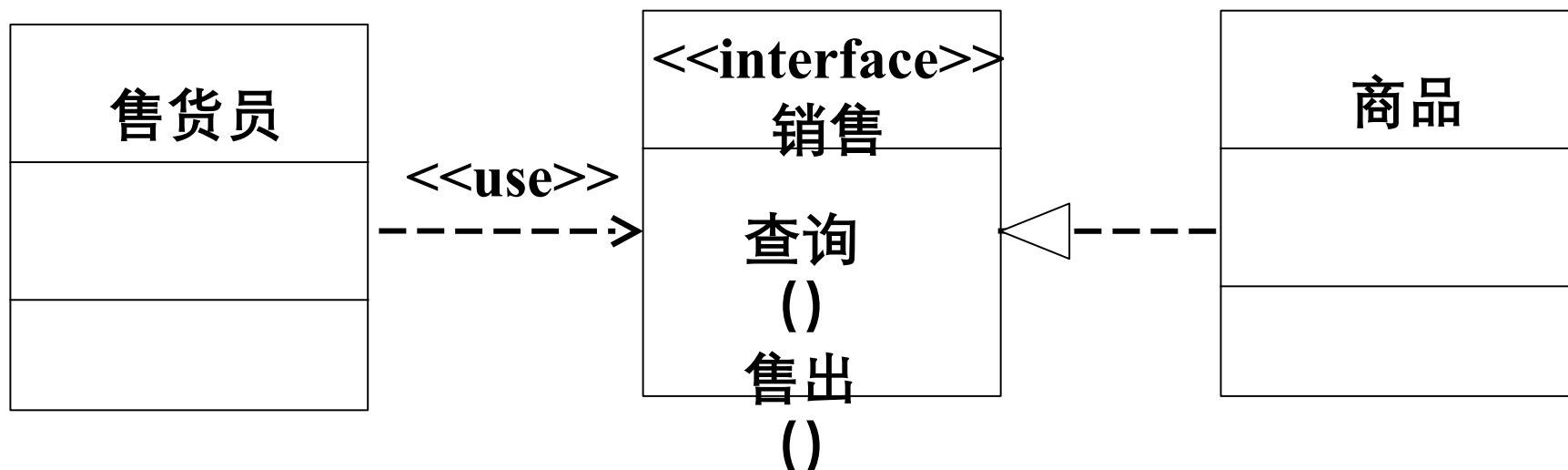


#### (4) 几点说明 ( 仅以类为例 )

- 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？



北京大学

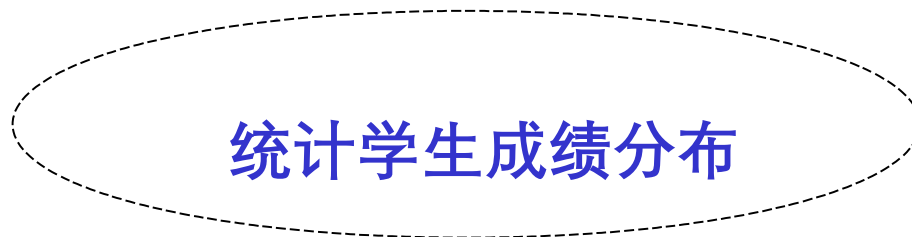


### 6.2.1.3 协作（collaboration） -- 体现行为结构抽象

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。

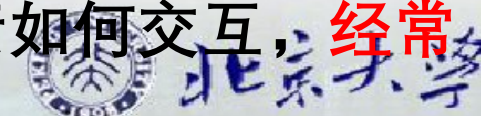
协作是一个交互，涉及交互三要素：交互各方、交互方式以及交互内容。交互各方的共同工作提供了某种协作行为。

表示：



2点说明：

① 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。





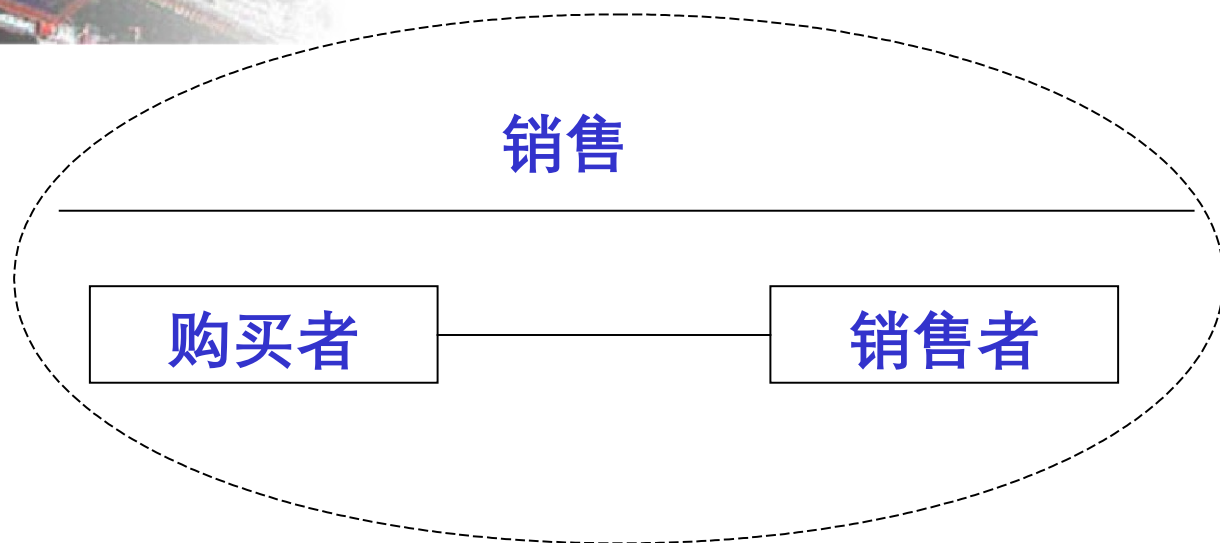


图 一个协作的组合结构图

② 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。



北京大学

## 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：

① 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。

② 用况是通过协作予以细化的。



北京大学

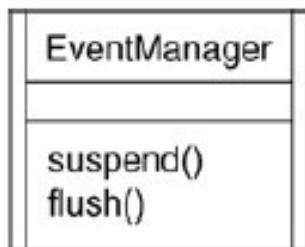


## 6.2.1.5 主动类（active class）

-- 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。

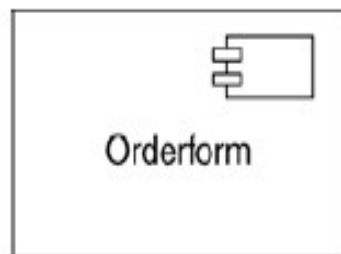


## 构件描述比特世界的软件制品 的系统单位

### 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：

- ① 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。
- ② 构件可以包含更小的构件。



北京大学



### 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：

《 artifact 》
Window.dll

#### 2 点说明

- ① 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 and 脚本等。
- ② 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

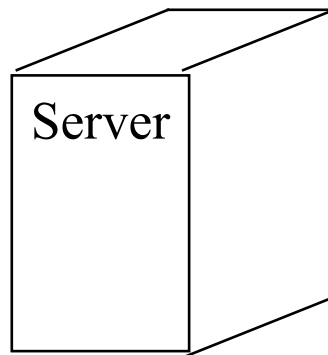




### 6.2.1.8 节点（node）

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：

- ① 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）



北京大学



## 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：

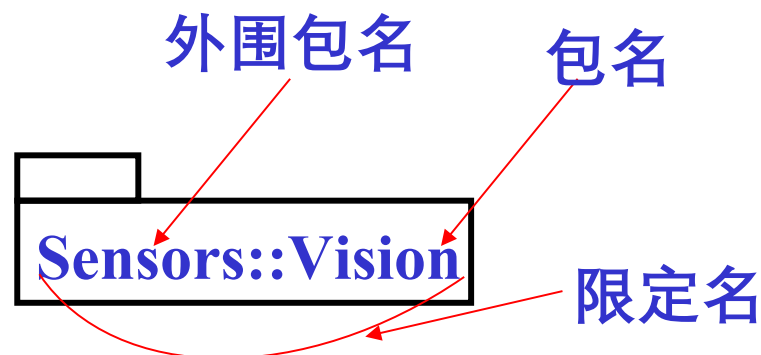
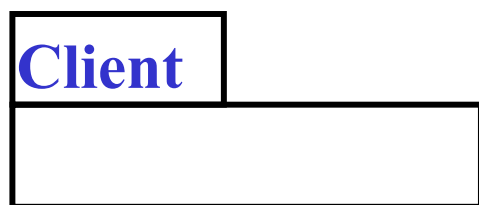
- +：对其他包而言都是可见的；
- ：对其他包而言都是不可见的；
- #：对子孙包而言是可见的；



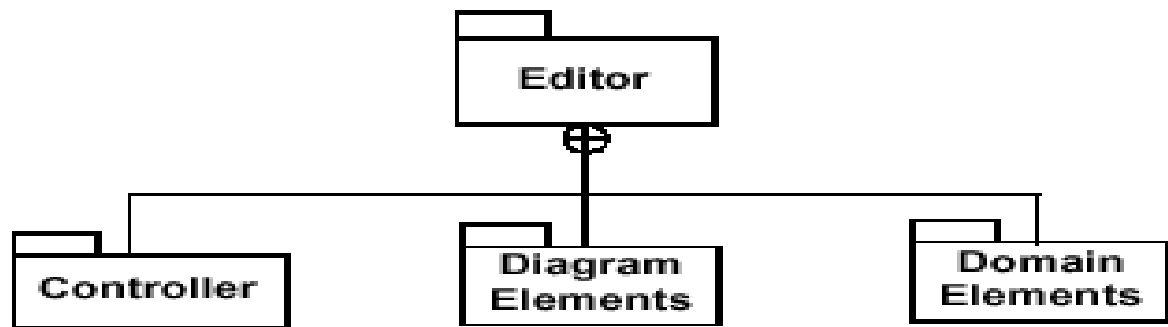


### 6.2.2.2 表示

- ① 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。



- ② 可以把所包含的元素画在包的外面，通过符号 $\oplus$ ，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。



北京大学



### 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》

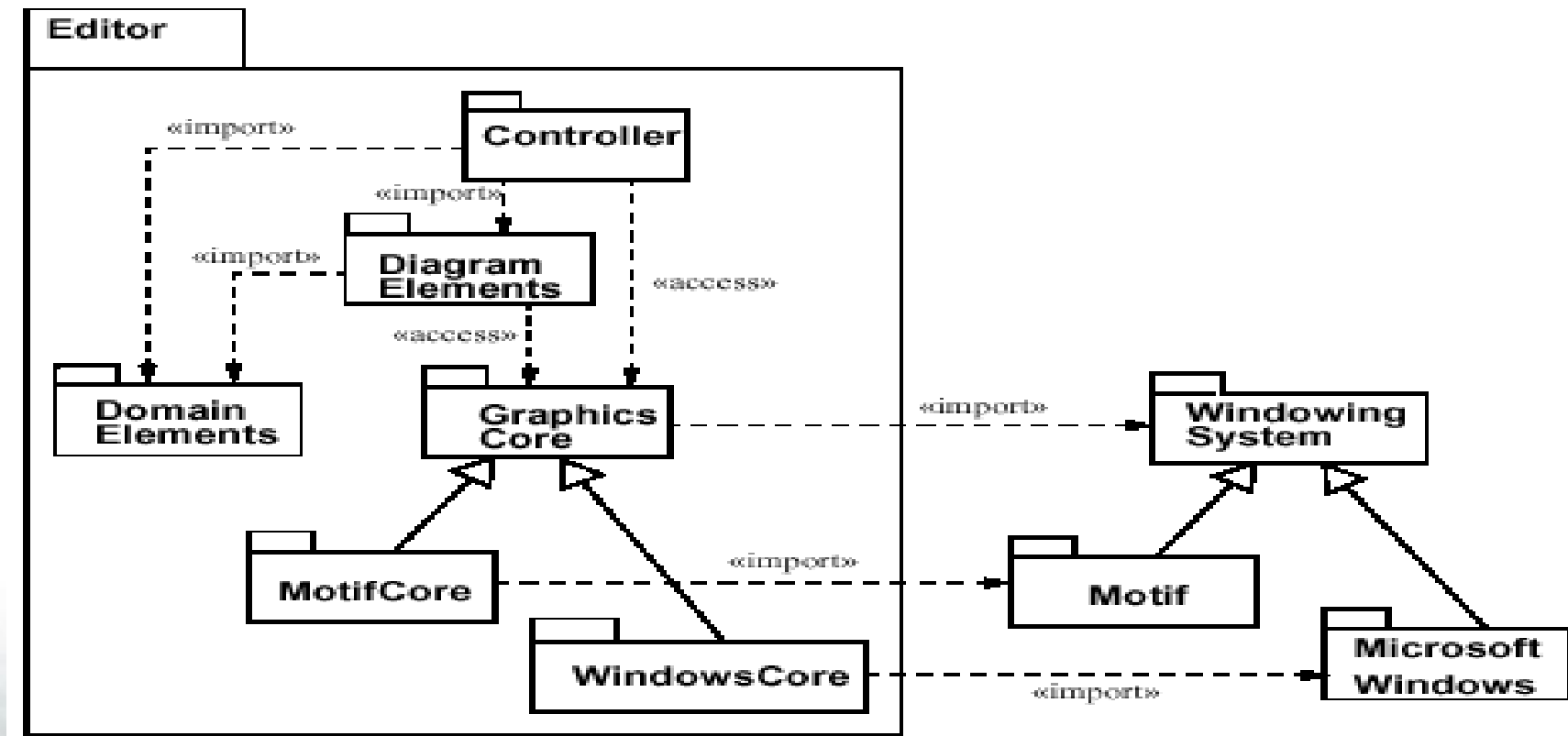
从源包到目标包的引入依赖表明：目标包中有适当可见性的内容被加入到**源包的公共命名空间**中，这相当于源包对它们做了声明（即对它们的引用可不需要一个路径名）





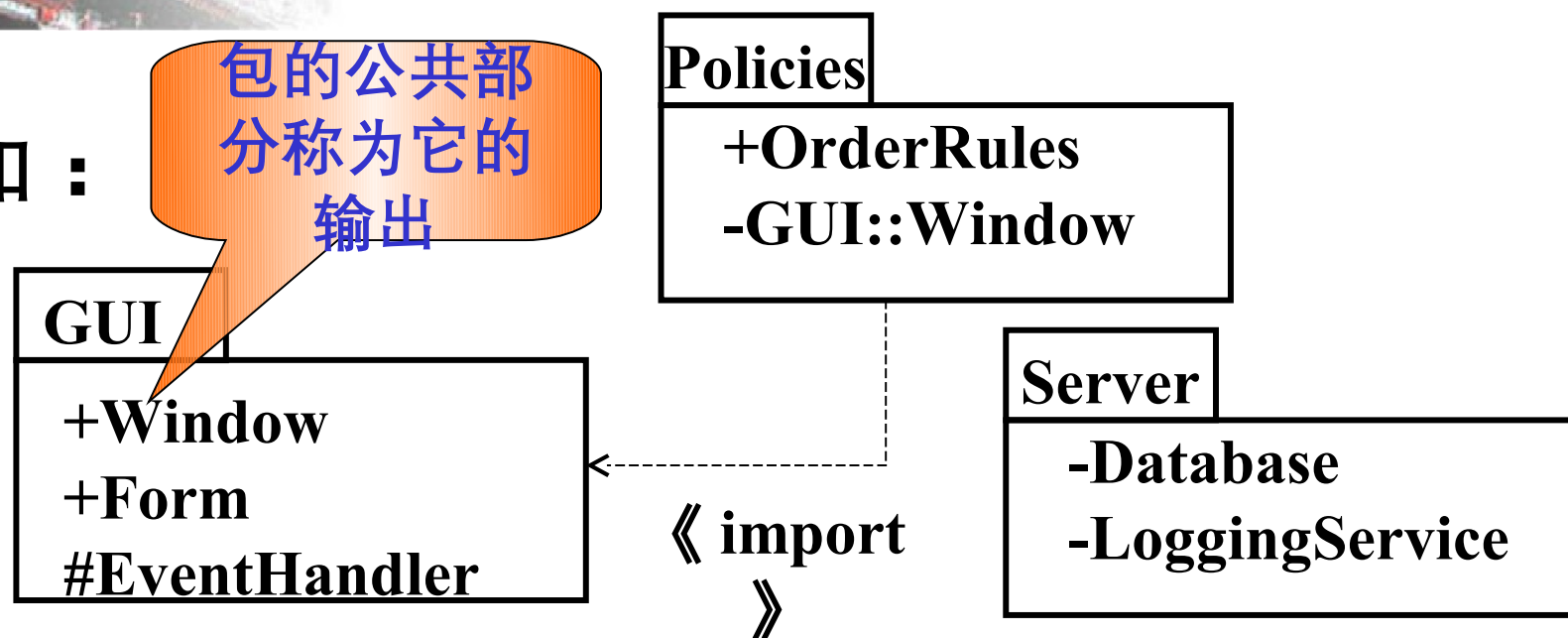
## 引入：《 import 》

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。

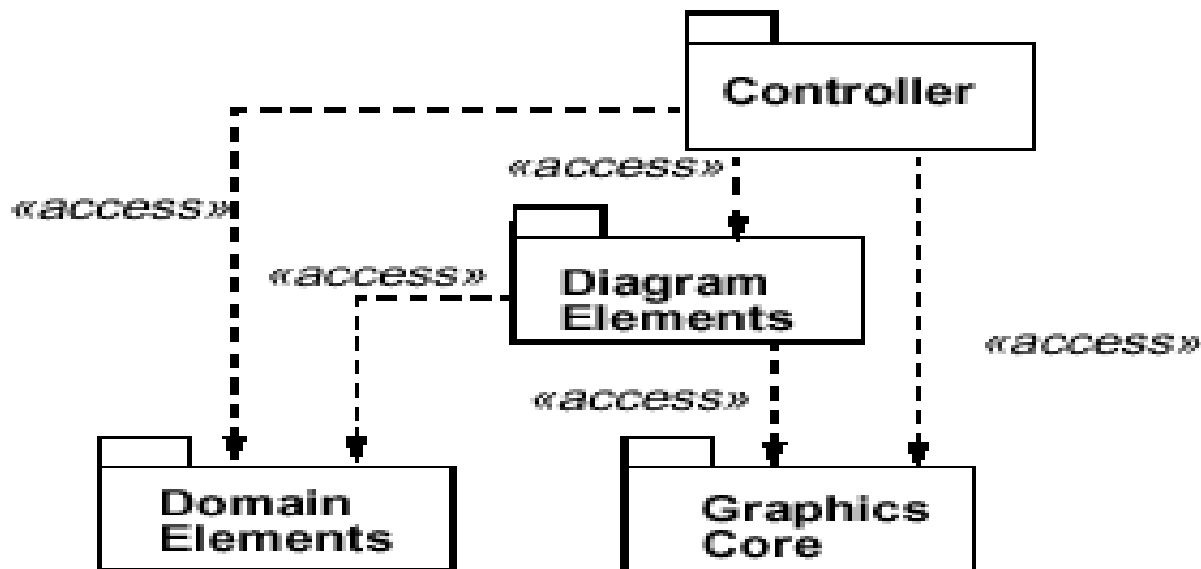


北京大学



## （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。

