

对象之间的四种关系

1. **一般-特殊关系** —— 又称**继承**关系，反映事物的分类。由这种关系可以形成**一般-特殊结构**。
2. **整体-部分关系** —— 即**聚合**关系。反映事物的构成。由这种关系可以形成**整体-部分结构**。
3. **关联关系** —— 对象实例集合（类）上的一个关系，其中的元素提供了被开发系统的应用领域中一组有意义的信息。
4. **消息关系** —— 对象之间的动态联系，即一个对象在执行其操作时，请求其他对象为它执行某个操作，或者向其他对象传送某些信息。反映了事物之间的行为依赖关系。

这些关系形成了类图的关系层

概念——同义词和近义词

继承 (inheritance) 是描述一般类和特殊类之间关系的最传统、最经典的术语。有时作为动词或形容词出现。

一般-特殊 (generalization-specialization) 含义最准确，而且不容易产生误解，恰切地反映了一般类（概念）和特殊类（概念）之间的相对（二元）关系；也用于描述结构，即一般-特殊结构。缺点是书写和阅读比较累赘。

泛化 (generalization) 取“一般-特殊”的一半，是UML的做法。比较简练，但是只反映了问题的一方面。作为关系的名称尚可，说结构是一个“泛化”则很勉强。

分类 (classification) 接近人类日常的语言习惯，体现了类的层次划分，也作为结构的名称。在许多的场合被作为一种原则。

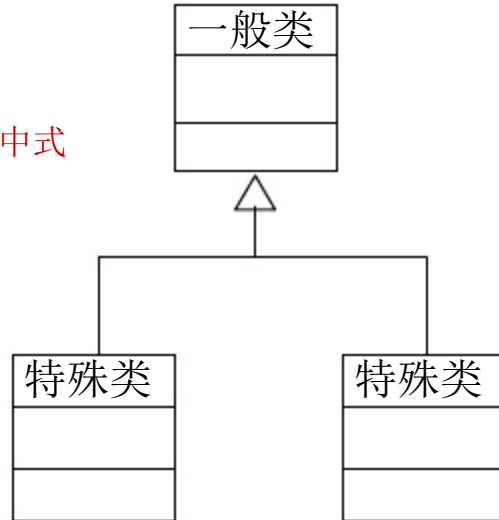
本书主要采用“一般-特殊”这个术语

相关概念：一般类、特殊类、继承、多继承、多态

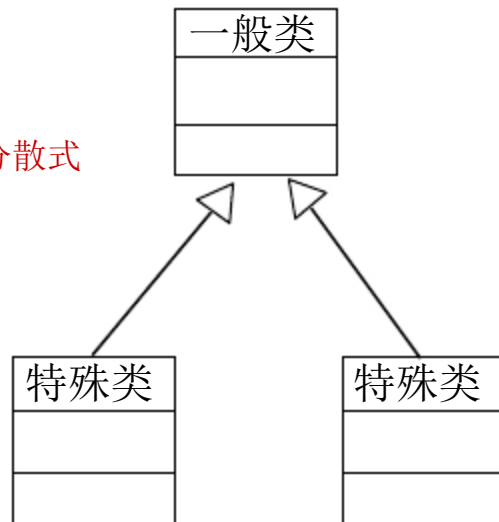
语义：“**is a kind of**”

表示法

集中式



分散式

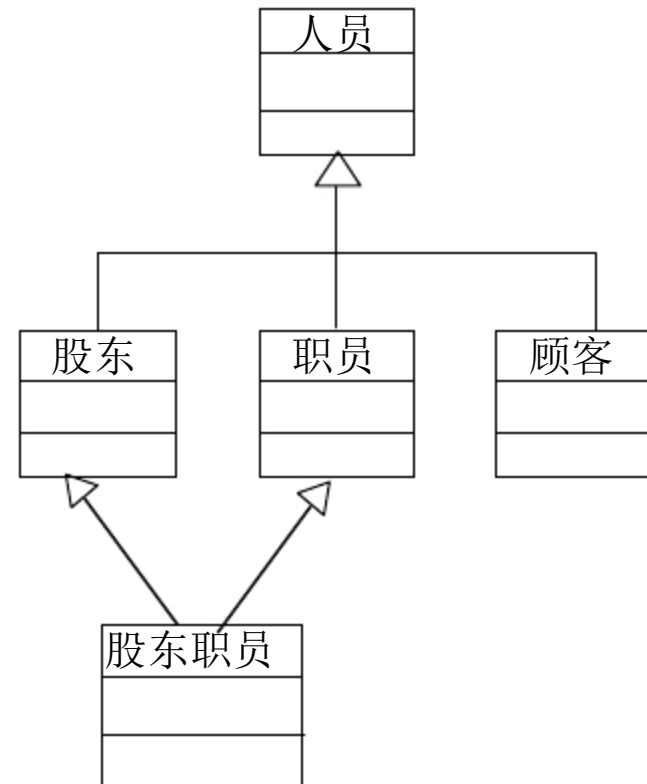


* 对继承的属性或操作重新定义

× 拒绝继承

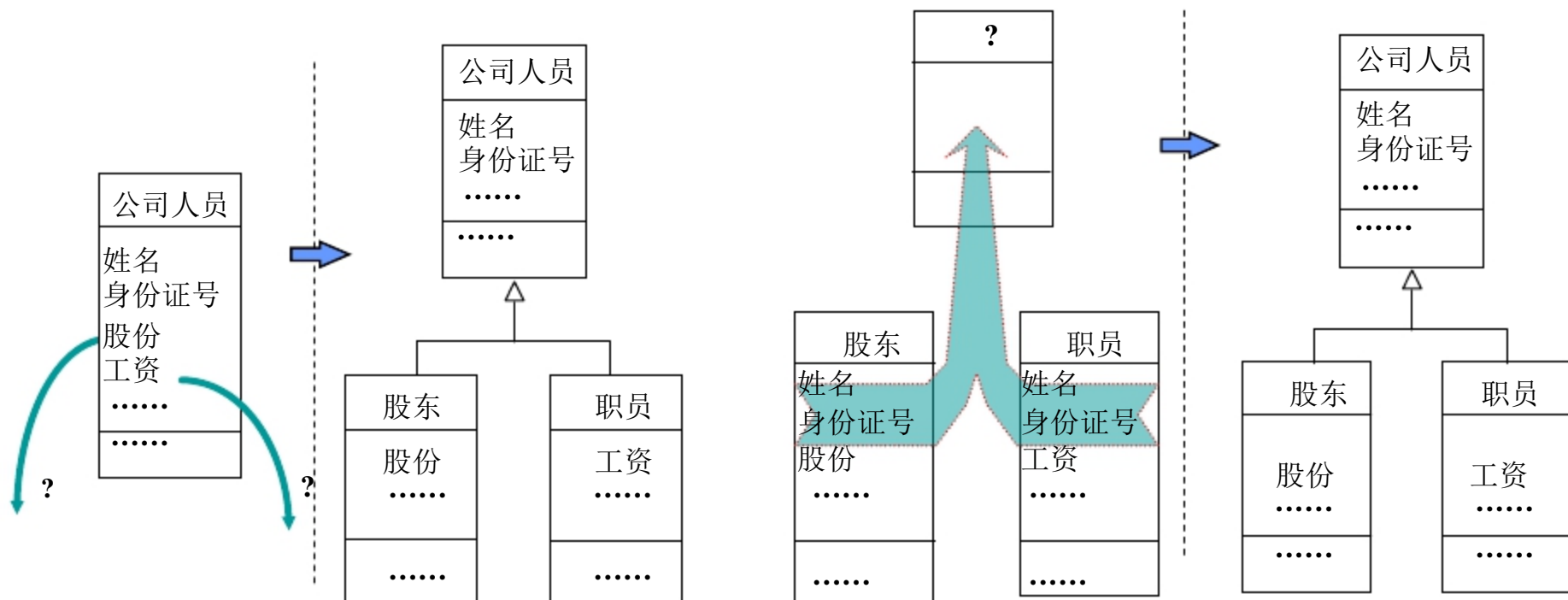
多态性的表示符号

例：

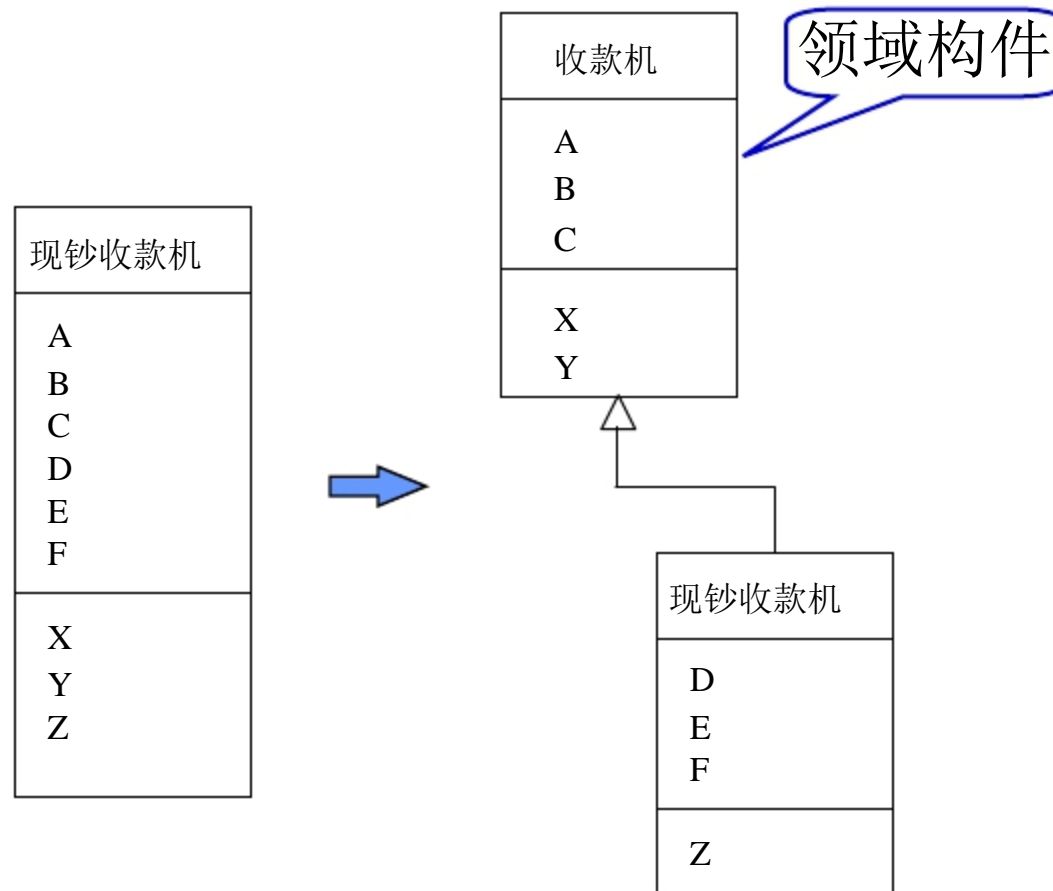


如何发现一般-特殊结构

- (1) 学习当前领域的分类学知识
- (2) 按常识考虑事物的分类
- (3) 根据一般类和特殊类的两种定义
- (4) 考察属性与操作的适应范围

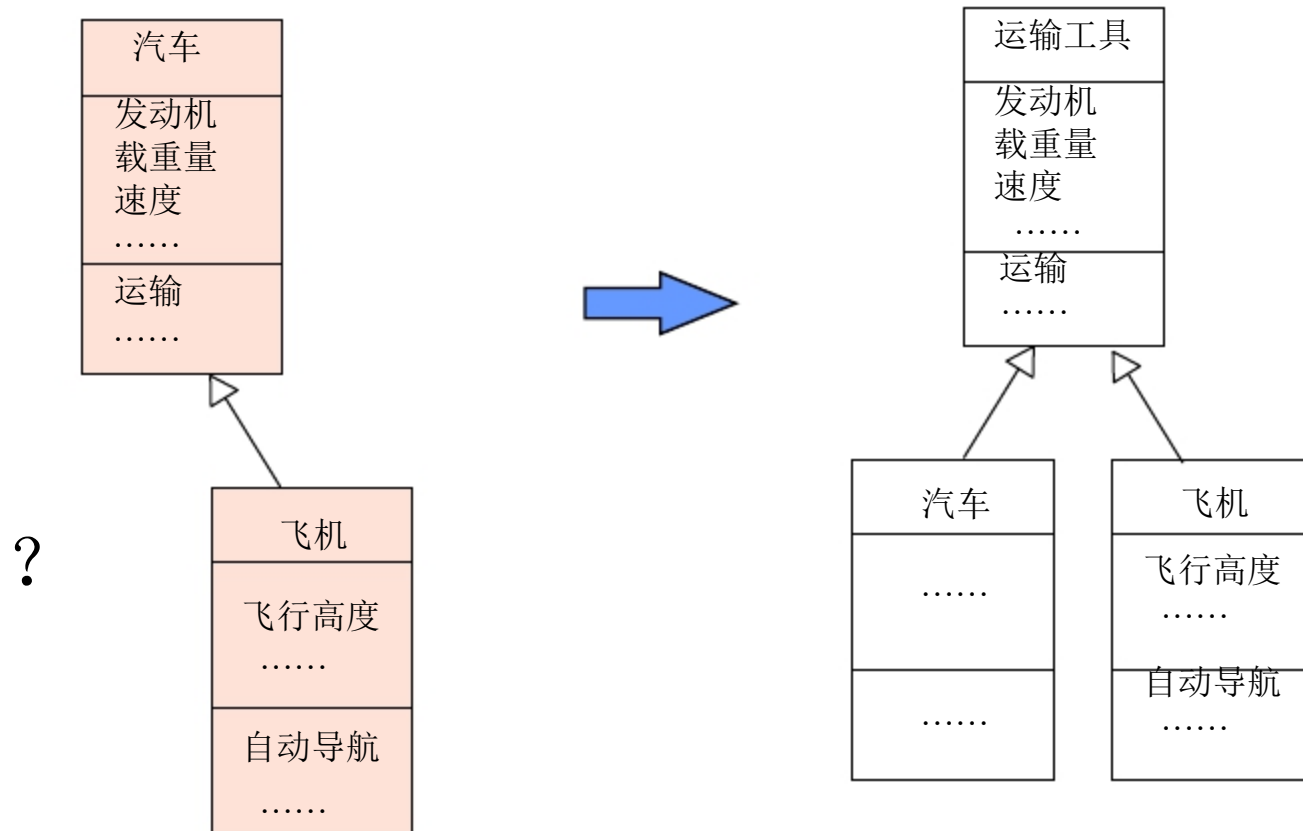


(5) 考虑领域范围内的复用



审查与调整

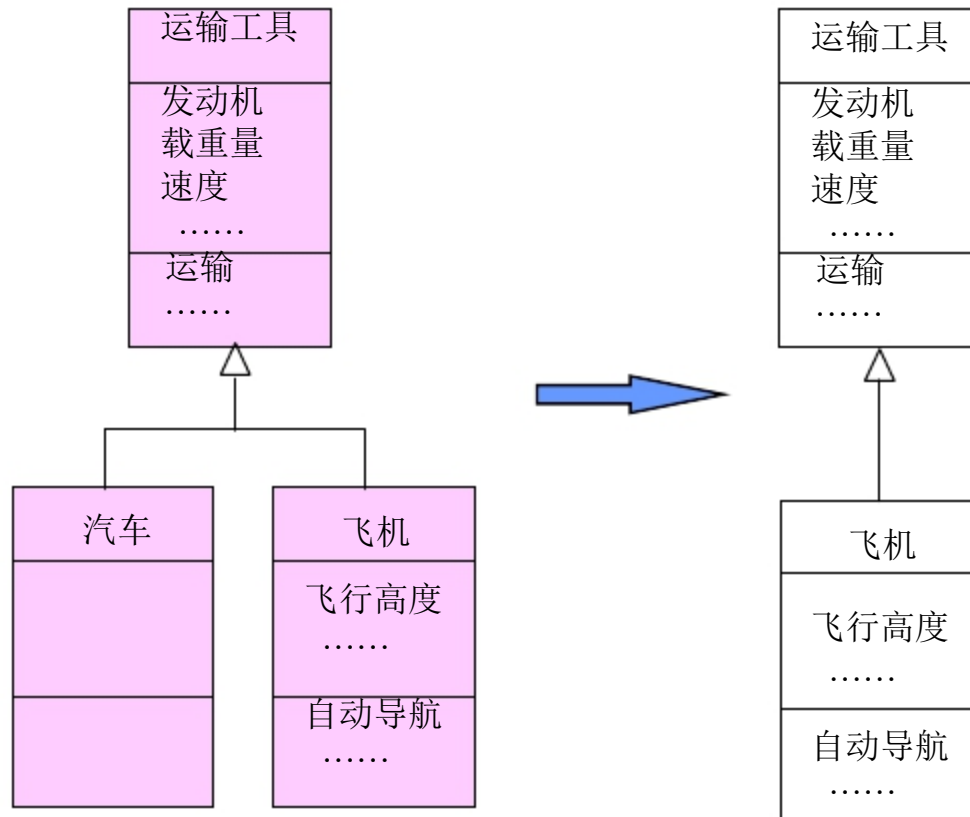
- (1) 问题域是否需要这样的分类？（例：书—线装书）
- (2) 系统责任是否需要这样的分类？（例：职员—本市职员）
- (3) 是否符合分类学的常识？（用“is a kind of ”来衡量）



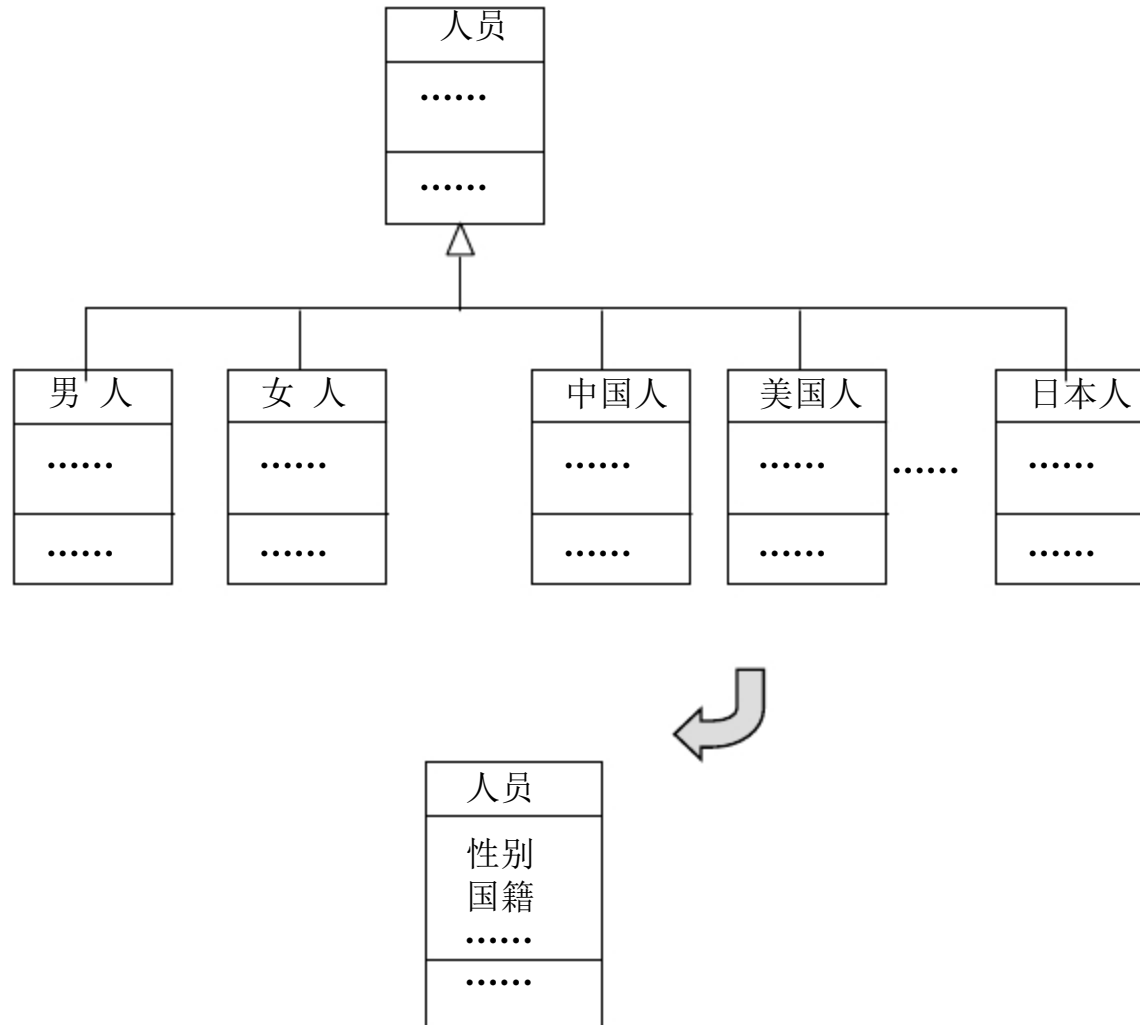
- (4) 是否真正的继承了一些属性或操作？

一般-特殊结构的简化

(1) 取消没有特殊性的特殊类

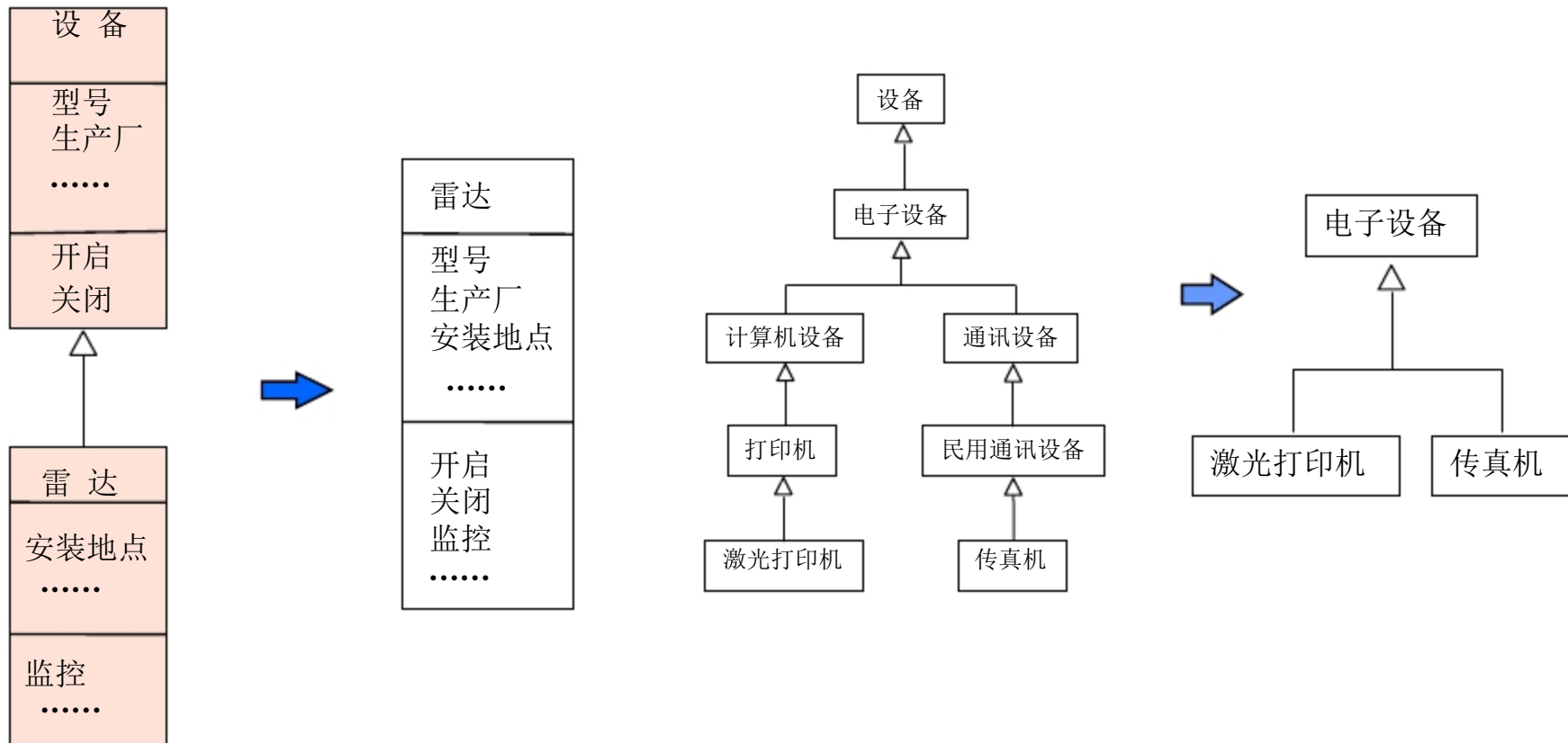


(2) 增加属性简化一般—特殊结构



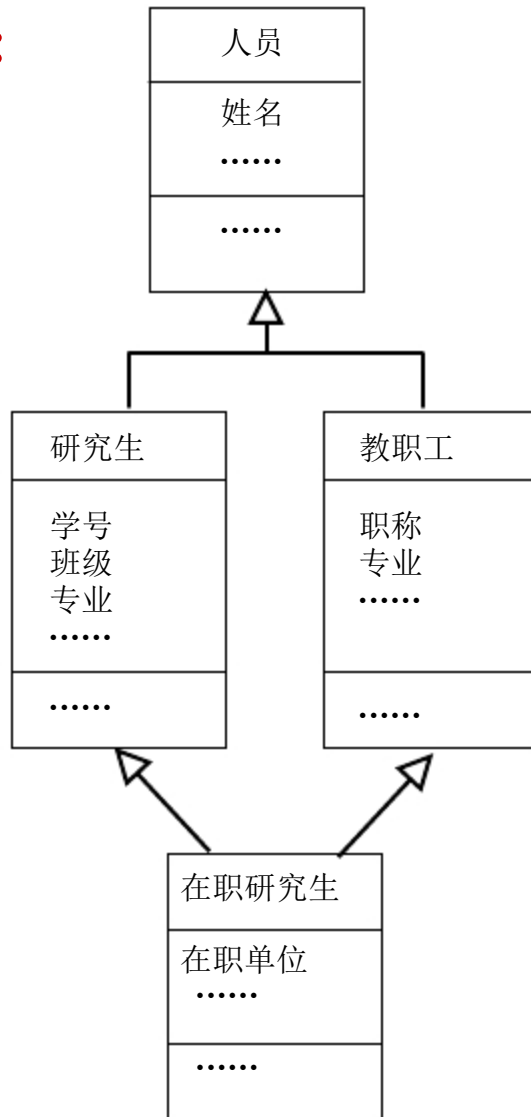
(3) 取消用途单一的一般类，减少继承层次 一般类存在的理由：

- * 有两个或两个以上的特殊类
- * 需要用它创建对象实例
- * 有助于软件复用

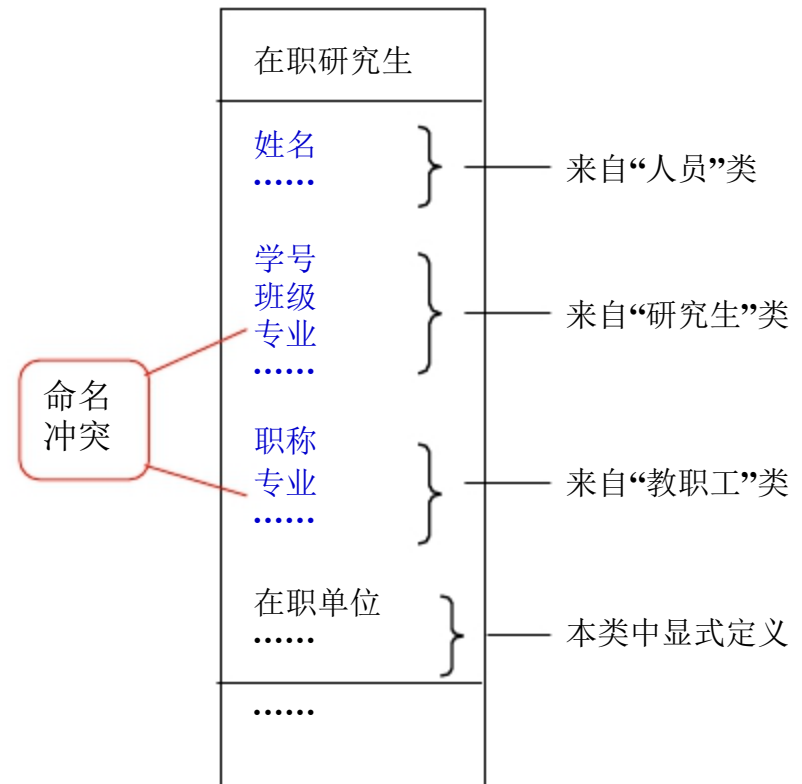


多继承：允许一个特殊类具有一个以上一般类的继承模式

例：

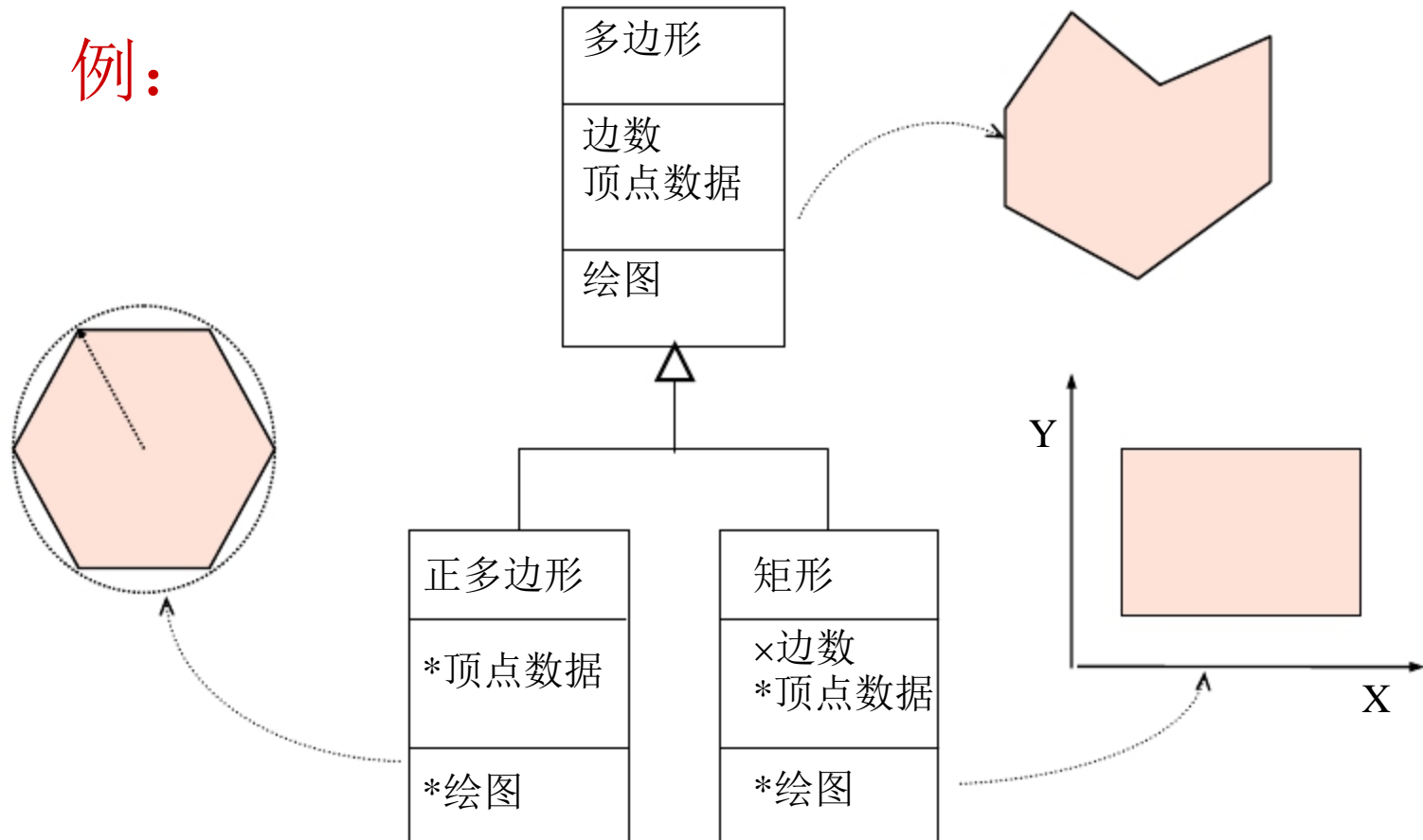


多继承特殊类的
内部情况



多态：多态是指同一个命名可具有不同的语义。**OO**方法中，常指在一般类中定义的属性或操作被特殊类继承之后，可以具有不同的数据类型或表现出不同的行为。

例：



概念:

聚合 (**aggregation**) , 组合 (**composition**)

整体-部分 (**whole-part**)

整体对象, 部分对象

语义: “**a part of**”或“**has a**”

聚合关系描述了对象实例之间的构成情况, 然而它的定义却是在类的抽象层次给出的。

——从集合论的观点看聚合关系

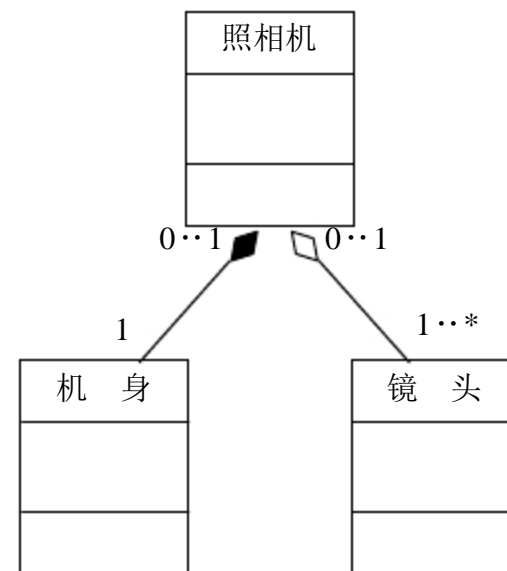
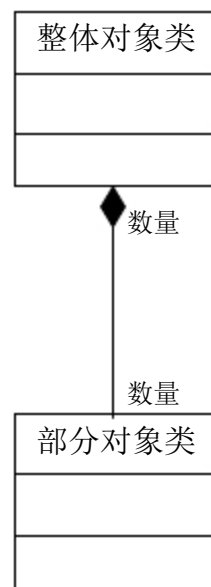
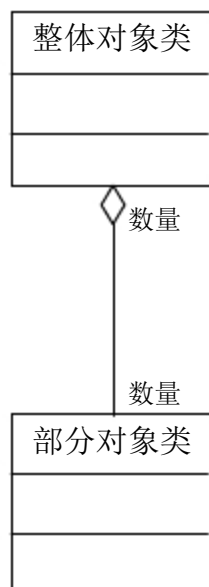
整体-部分关系 (聚合关系) 是两个类之间的二元关系, 其中一个类的某些对象是另一个类的某些对象的一部分。

整体-部分结构是把若干具有聚合关系的类组织在一起所形成的结构。它是一个以类为结点, 以聚合关系为边的连通有向图。

一种基本的
模型元素

由若干聚合关系形成的复合
模型成分

表示法



在连接符两端通过数字或者符号给出关系双方对象实例的数量约束，称为多重性（**multiplicity**）

确定的整数 —— 给出确定的数量

—— 例如：1, 2

下界..上界 —— 给出一个范围

—— 例如：0..1, 1..4

*

—— 表示多个，数量不确定

下界..*

—— 表示多个，下界确定 —— 例如 0..*, 1..*

多重性有以下**3**种情况：

一对一，一对多，多对多

如何发现整体-部分结构

基本策略——

考察问题域中各种具有构成关系的事物

(1) 物理上的整体事物和它的组成部分

例：机器、设备和它的零部件

(2) 组织机构和它的下级组织及部门

例：公司与子公司、部门

(3) 团体（组织）与成员

例：公司与职员

(4) 一种事物在空间上包容其它事物

例：生产车间与机器

(5) 抽象事物的整体与部分

例：学科与分支学科、法律与法律条款

(6) 具体事物和它的某个抽象方面

例：人员与身份、履历

审查与筛选

(1) 是否属于问题域？

例：公司职员与家庭

(2) 是不是系统责任的需要？

例：员工与工会

(3) 部分对象是否有一个以上的属性？

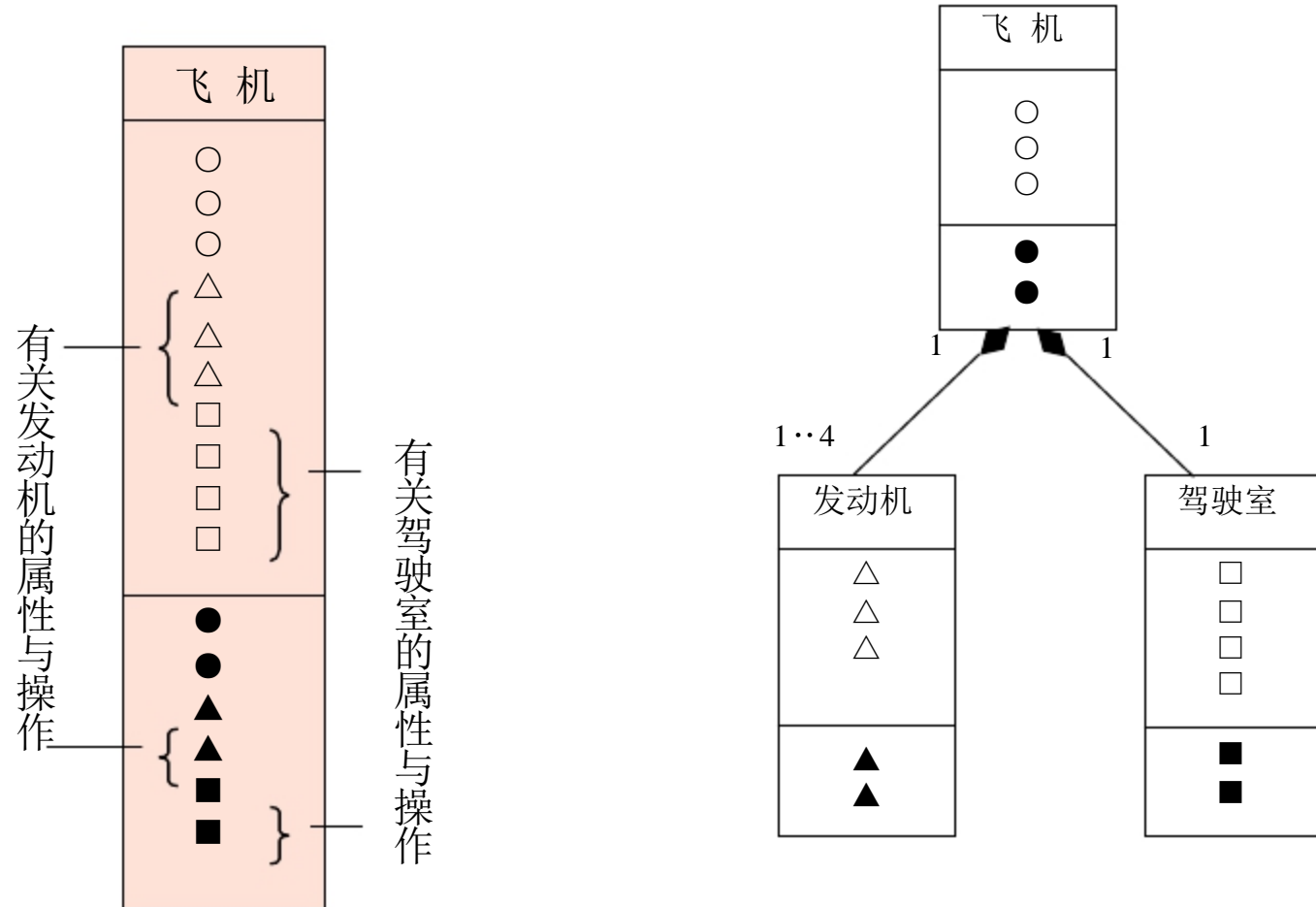
例：汽车与车轮（规格）

(4) 是否有明显的整体-部分关系？

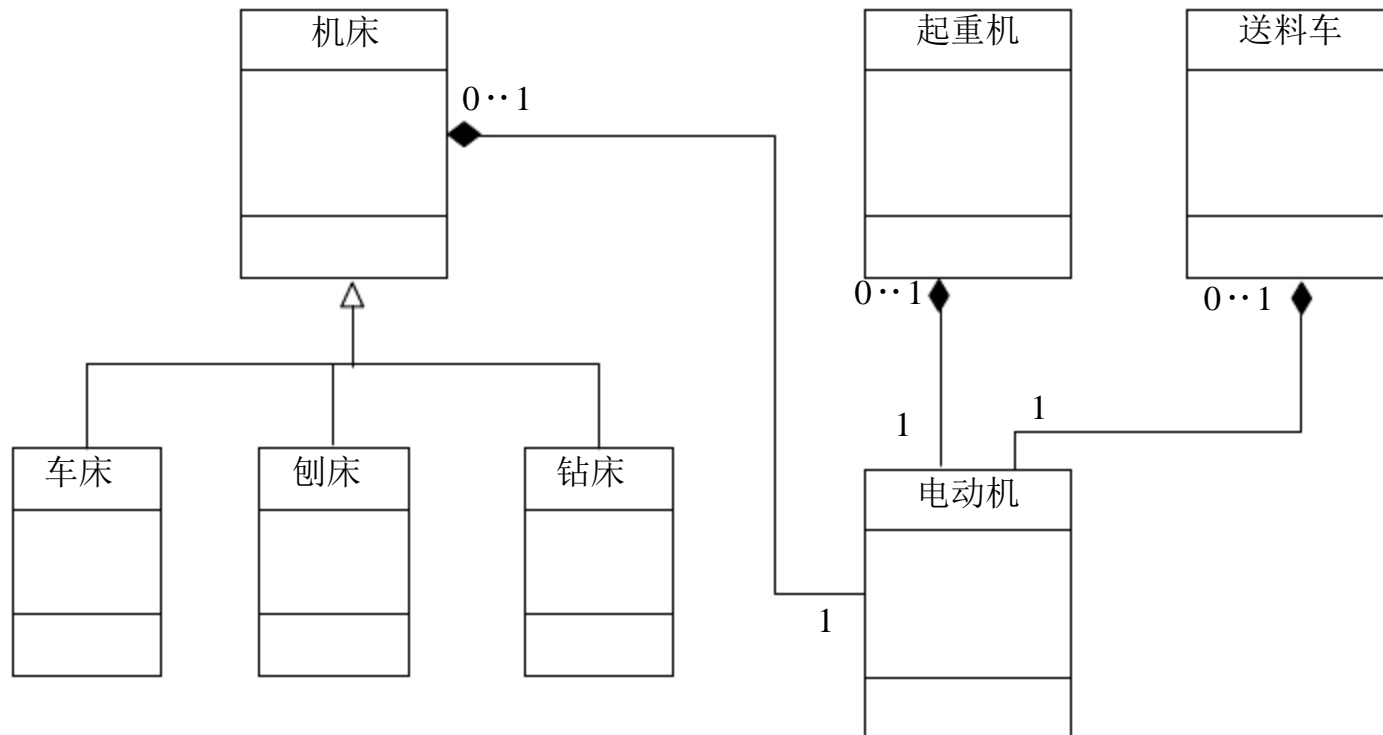
例：学生与课程

整体-部分结构的高级应用技巧

(1) 简化对象的定义



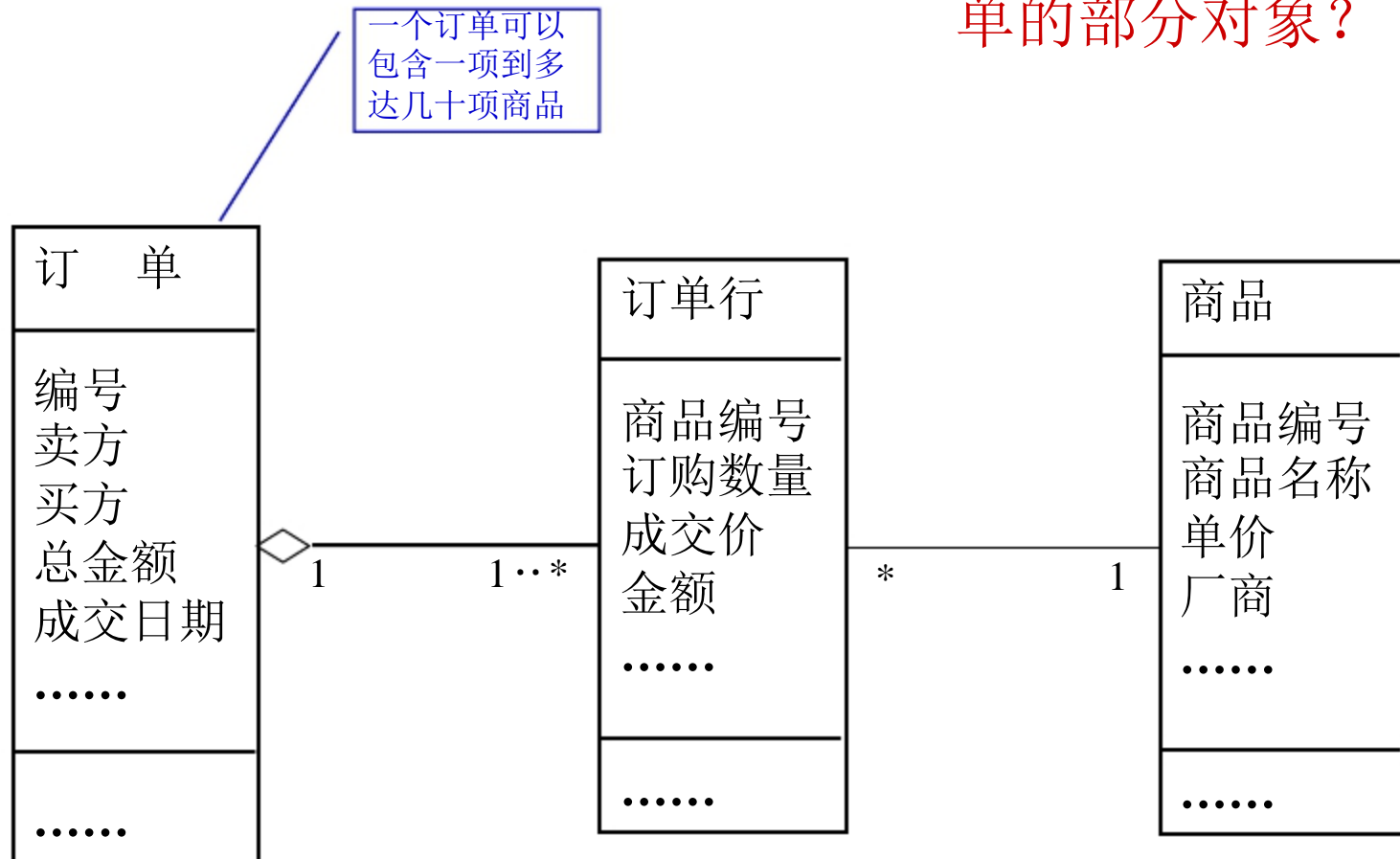
(2) 支持软件复用



(3) 表示数量不定的组成部分

提问：

能否不要订单行，
直接用商品作为订
单的部分对象？



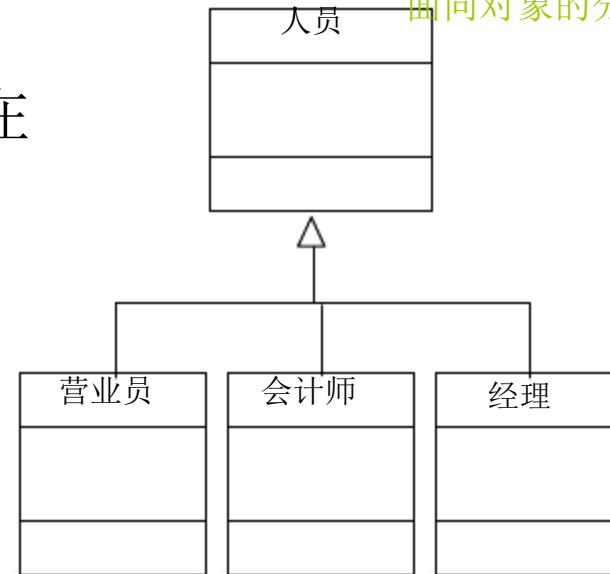
(4) 表示动态变化的对象特征

问题：对象的属性与操作定义在系统运行中动态变化，例如：

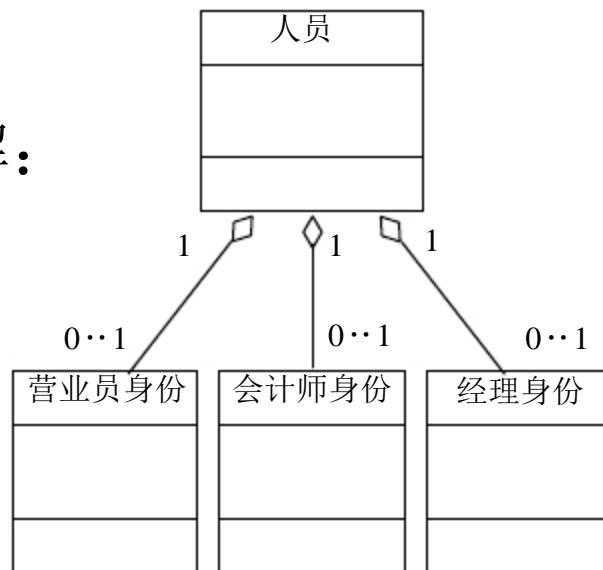
不理想的解决办法：

删除、重建

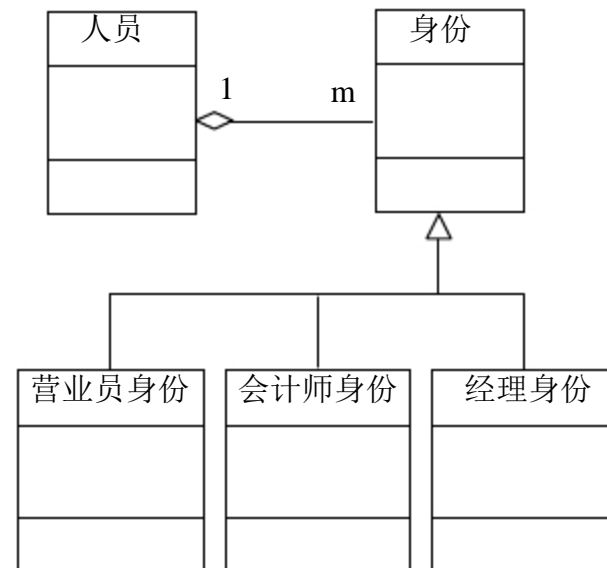
Shlaer/ Mellor的子类型迁移
“动态对象”



解：



或

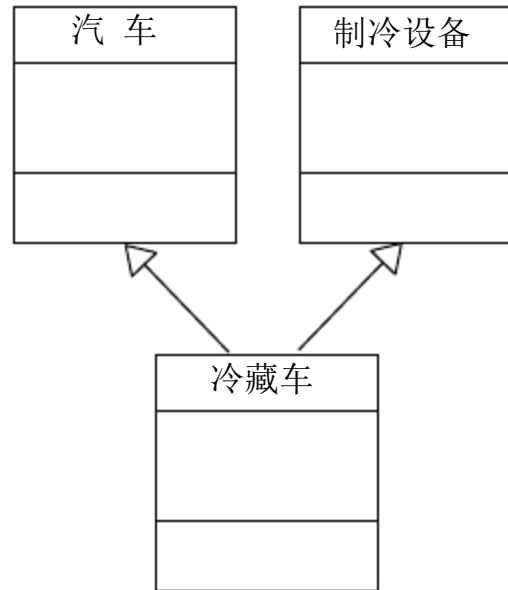


两种结构的变通

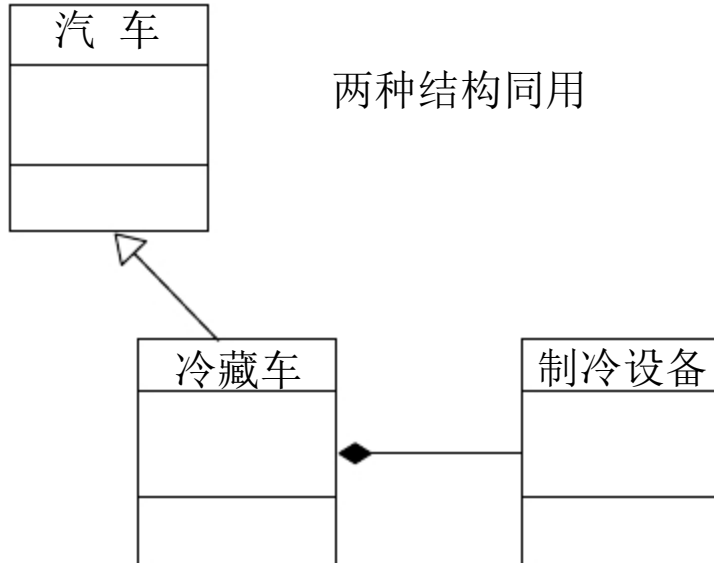
解释：

继承和聚合都是使一类对象获得另一类对象的特征，只是观察问题的角度不同。

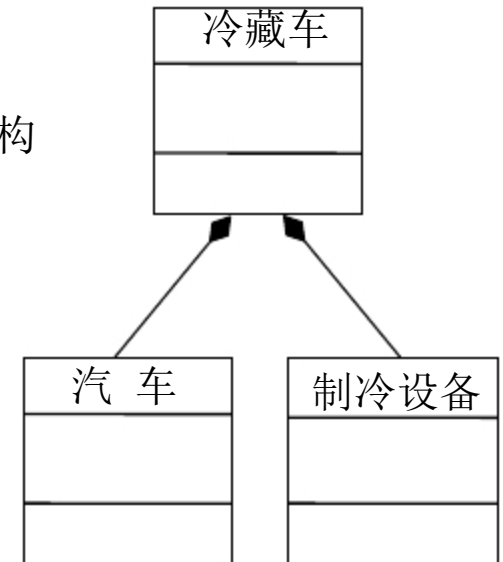
用一般—特殊结构



两种结构同用



用整体—部分结构



概念：

关联（**association**）是两个或者多个类上的一个关系（即这些类的对象实例集合的笛卡儿积的一个子集合），其中的元素提供了被开发系统的应用领域中一组有意义的信息。

二元关联（**binary association**）

n元关联（**n-ary association**）

关联的实例——**有序对** 或 **n元组**，又称**链**（**link**）

关联是这些有序对 或 **n元组**的集合

关联位于类的抽象层次，链位于对象的抽象层次

二元关联的表示法



数量约束

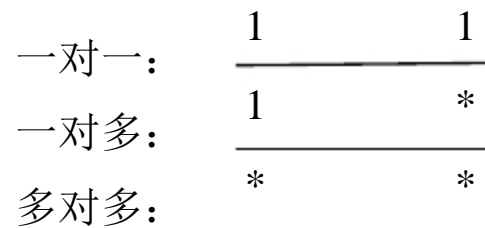
固定数值：例如 **1**

数值范围：例如 **0..1**

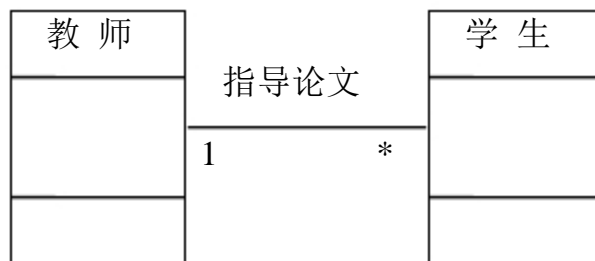
符号：***** 表示多个

0..* = ***** **1..*** 表示 **1**到多个

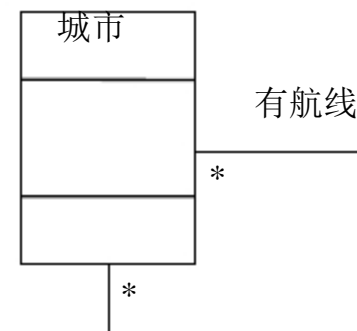
多重性的表示



例子



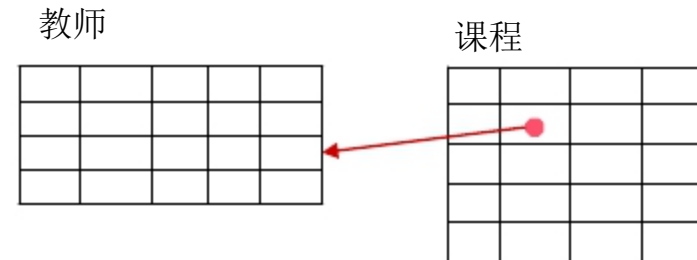
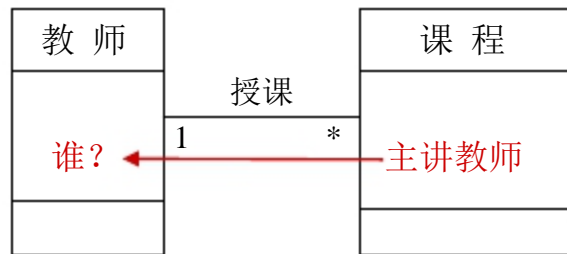
教师为学生指导论文



(d) 城市之间有航线

二元关联的实现（一对一和一对多）

编程语言：在程序中用两个类分别实现关联两端的类；以数量约束为“1”的类的对象实例为目标，在关联另一端的类中设置一个指向该目标的指针或者对象标识（源类的属性）。



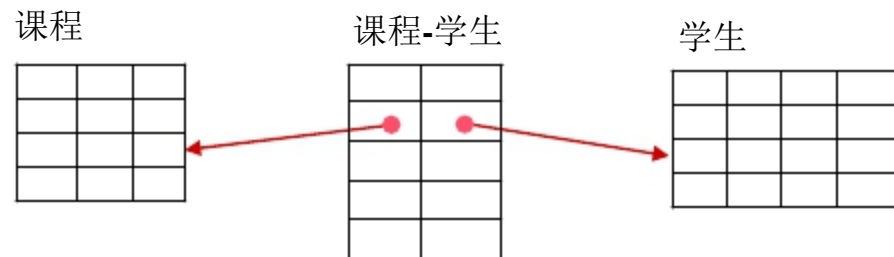
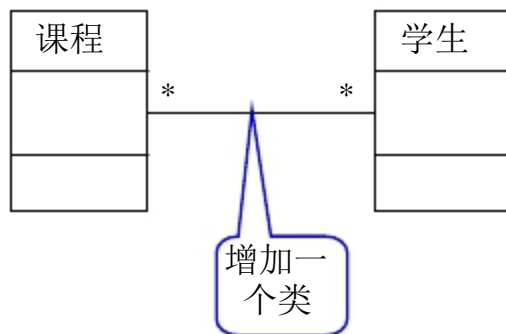
关系数据库：用两个数据库表分别实现关联两端的类；以数量约束为“1”的类对应的表的元组为目标，在关联另一端的类对应的表中设置一个指向该目标的外键（目标的主键）。

二元关联的实现（多对多）

问题：任何一端的一个对象实例的要和另一端多个对象实例发生关联，而且数量不确定。实现时不知道该设立多少个指针（或者对象标识、外键）才能够用。

编程语言：用两个类分别实现关联两端的类，同时用另外一个类来实现它们之间的关联。实现关联的类含有两个属性，分别是指向两端的类的对象实例的指针或者对象标识。

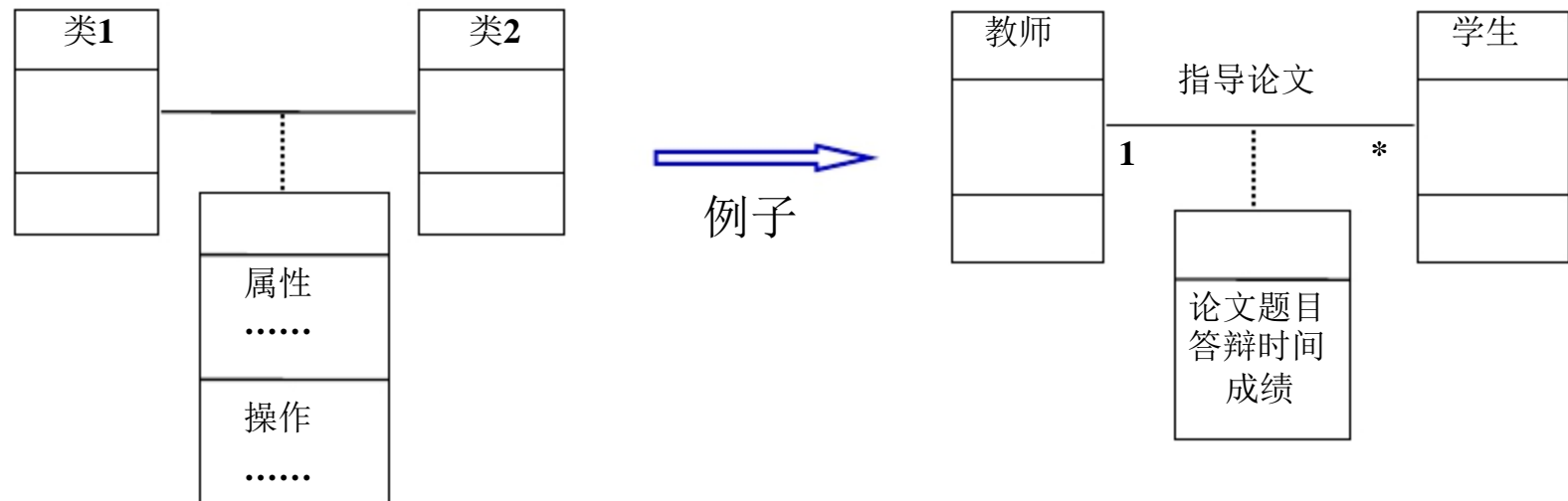
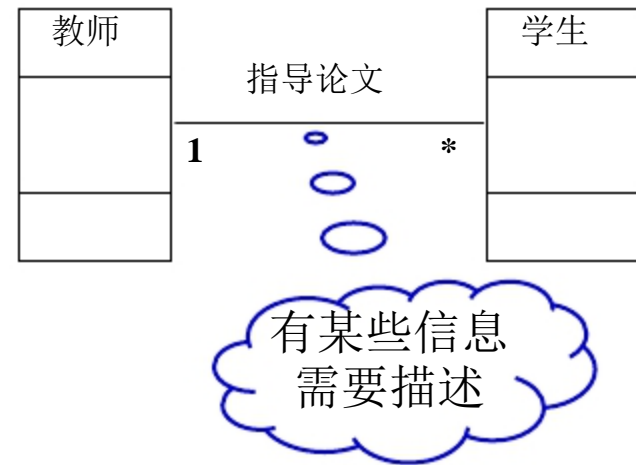
关系数据库：用两个数据库表分别实现关联两端的类，同时用另外一个数据库表来实现它们之间的关联。实现关联的数据库表含有两个属性，它们分别是指向两端的表的元组的外键。



运用简单的关联概念解决复杂的关联问题

(1) 带有属性和操作的关联

OMT（及UML）的概念扩充
关联类（**association class**）



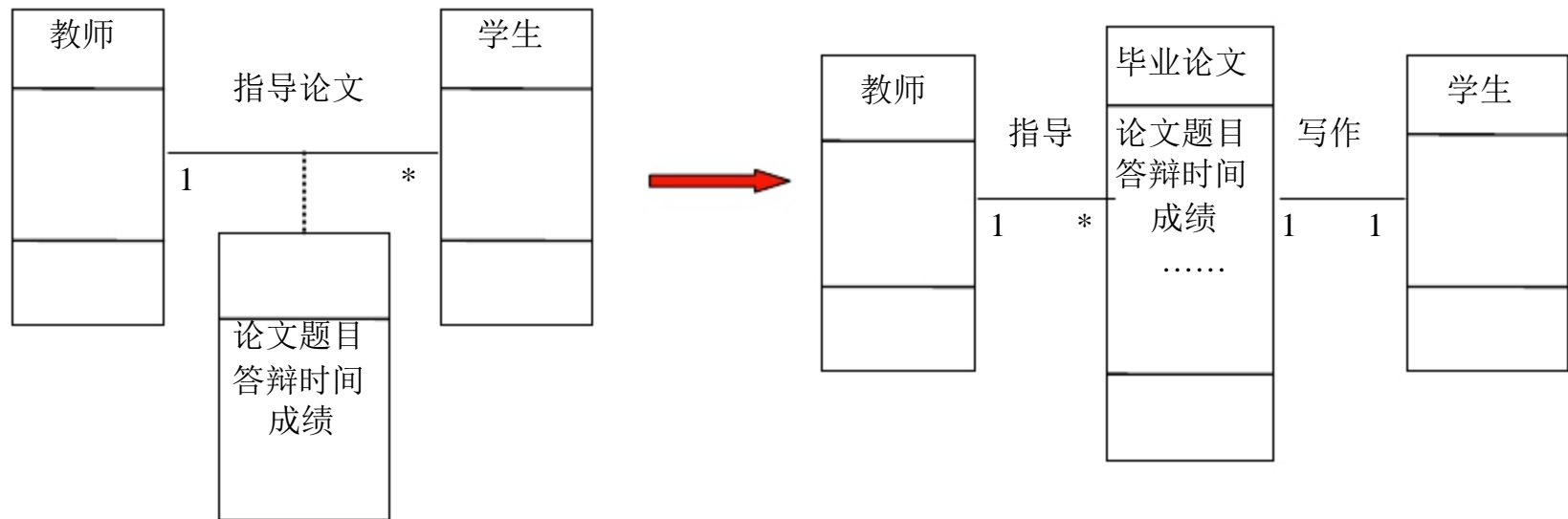
问题：增加了概念的复杂性，缺乏编程语言支持

换一种思路考虑问题：

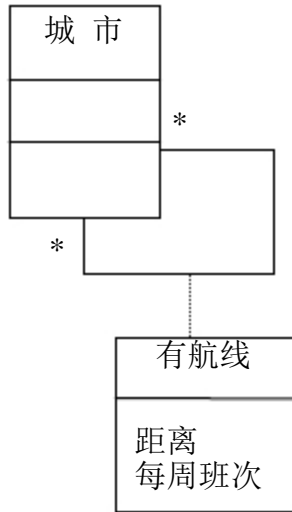
两类对象之间的关联带有某些复杂的信息，说明它们之间存在着某种事物（尽管可能是抽象事物）。

用普通的对象概念来表示这种事物，简化关联，减少概念，并加强与OOPL的对应。

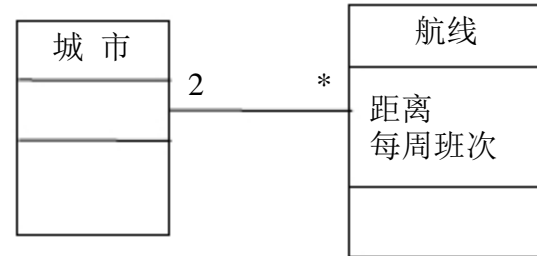
例1



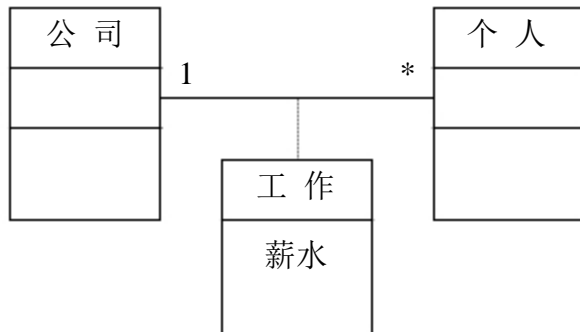
其他例子



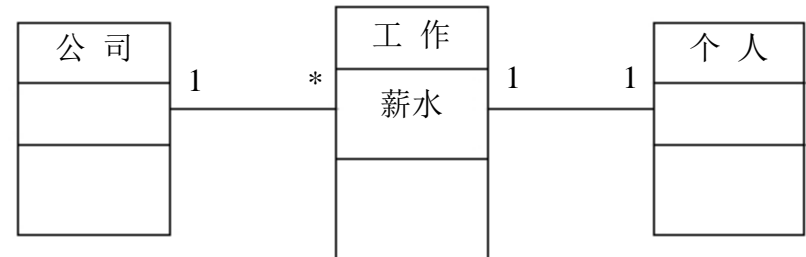
城市之间有航线



城市之间存在航线对象

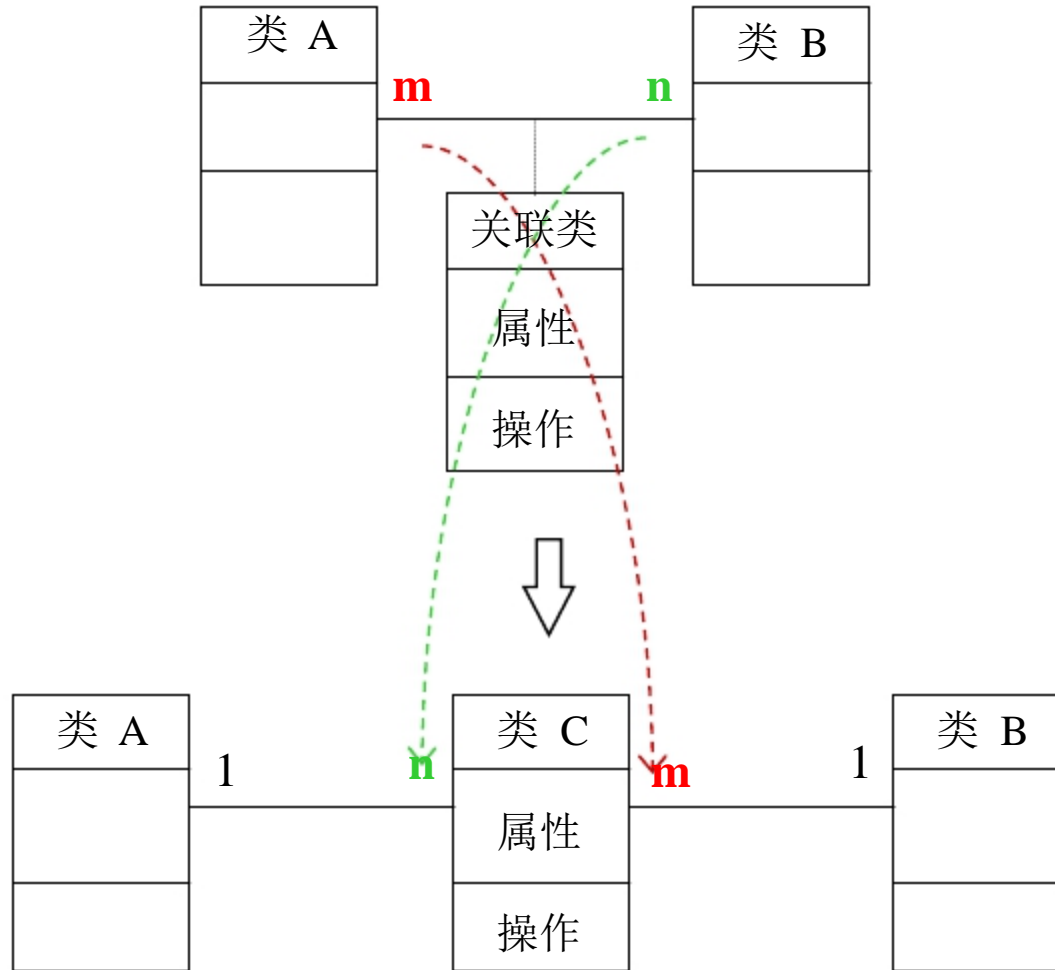


公司与个人



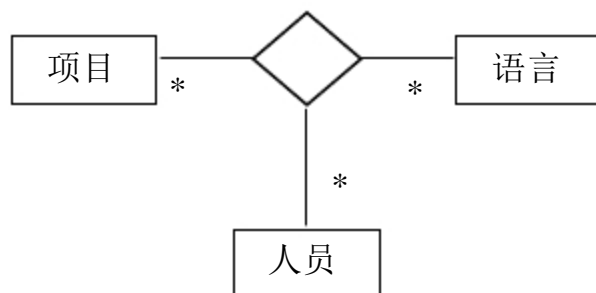
公司与个人之间存在工作对象

复杂关联表示法的转换

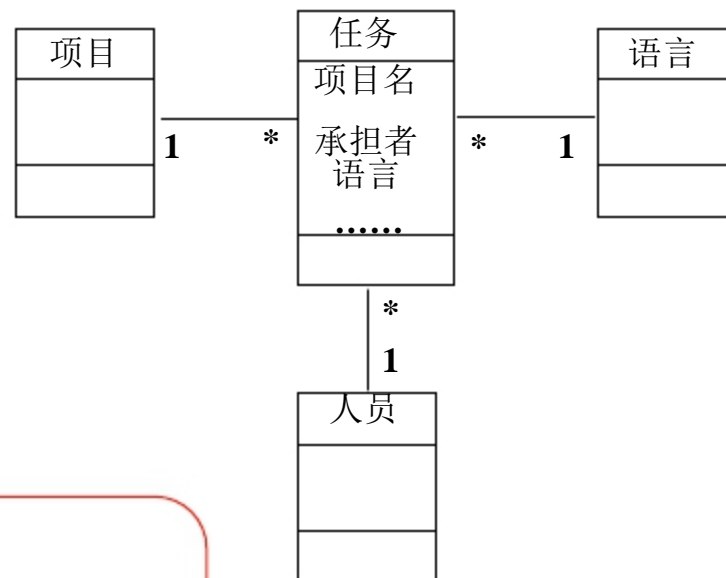


(2) n元关联

OMT的三元关联及其表示法



增设对象类表示多元关联



问题:

编程语言不能直接支持

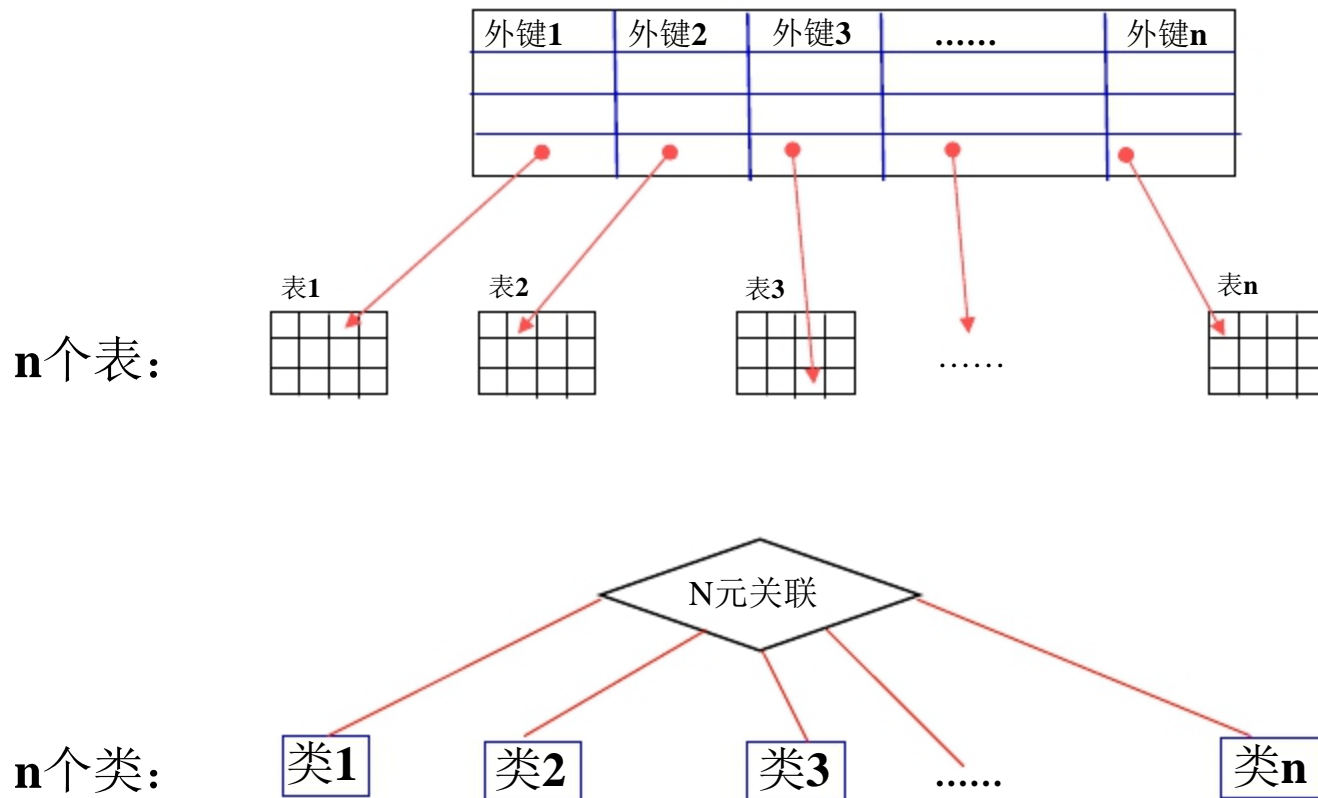
可推广到n元关联，是否要创造更多的符号？

多重性表示的困难（详后）

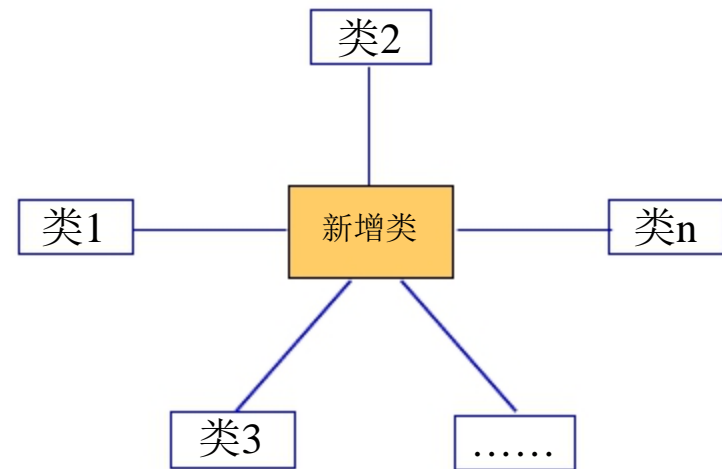
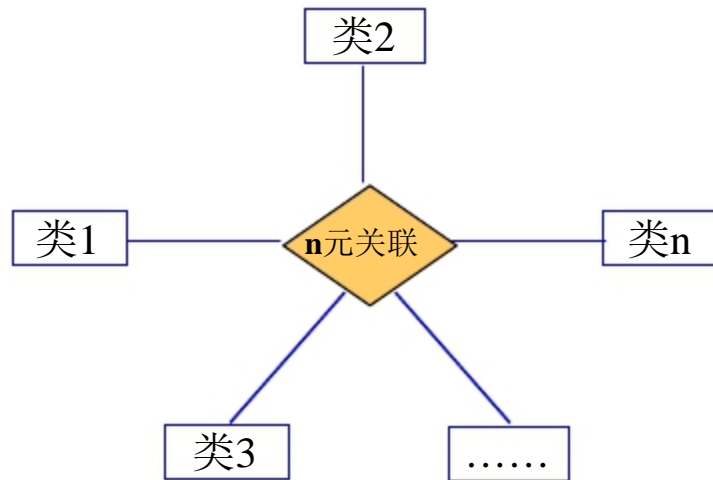
是一个类——是由每个**n**元组作为对象实例的类

从实现的角度看， 用类实现**n**元关联是最自然的选择

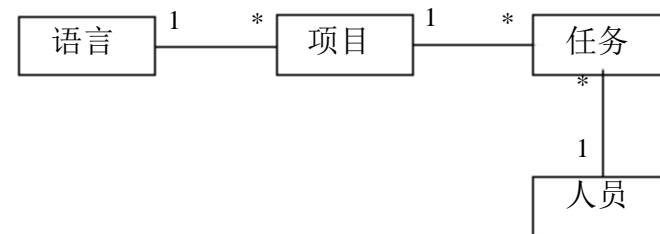
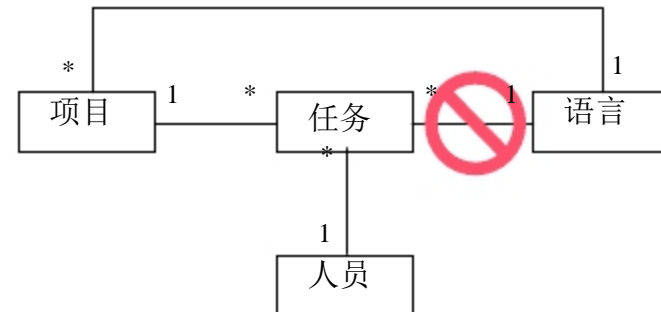
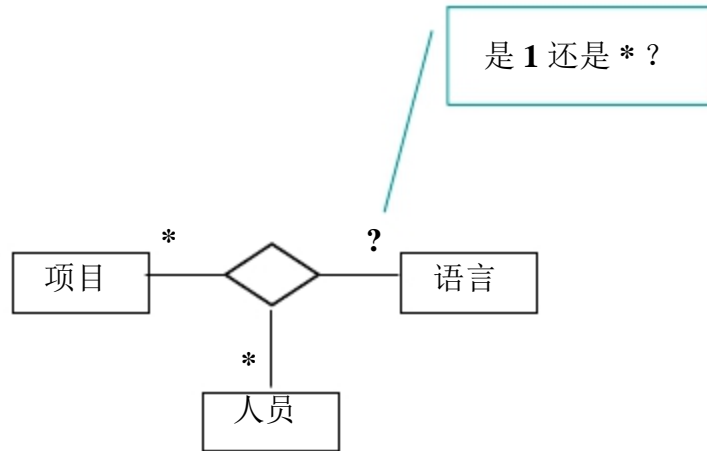
例如： 用一个数据库表存放**n**元关联的全部**n**元组



在模型中，把**n**元关联定义为一个类
并定义它与原有的各个类之间的关系——都是二元关联



n元关联多重性表示的困难和解决办法



(3) 一个类在一个关联中多次出现

例：课程实习中每两名学生在一台设备上合作完成一个题目

1) 若系统要求记录和查阅哪两名学生是合作者

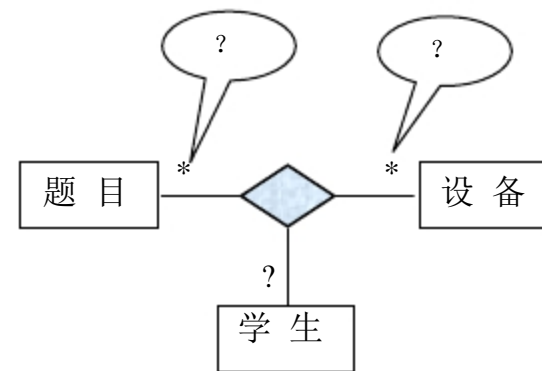
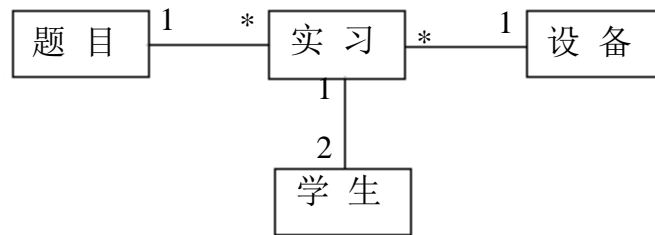
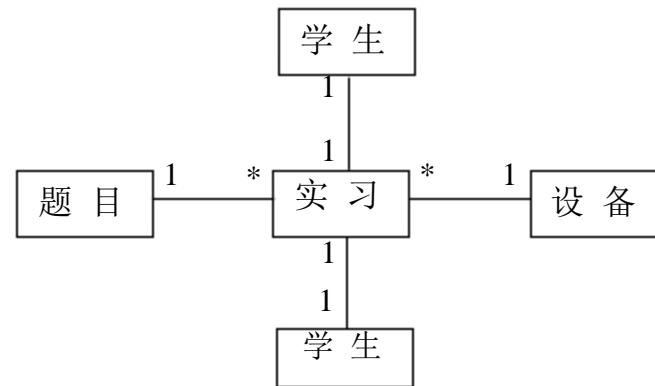
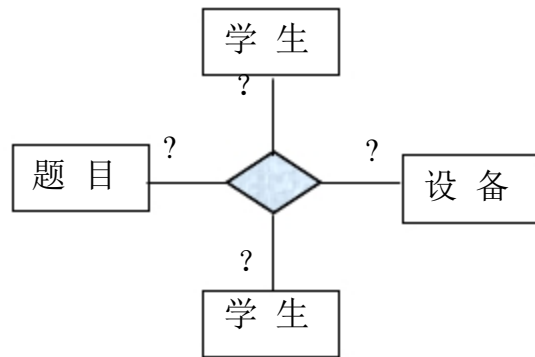
建立学生类到它自身的关联（如同城市之间有航线）
是一个二元关联，其中学生类在关联中出现了两次

2) 如果还要记录每组学生的实习题目和使用的设备

建立学生、题目、设备三个类之间的4元关联
学生类在这个关联中出现了两次

加入该系统的多重性要求是：

- 每两名学生在一台设备上合作完成一个题目；
- 一个题目可以供多组学生实习，可以在不同的设备上完成；
- 一台设备可以供多组学生使用，可以做不同的题目。



如何建立关联

1. 根据问题域和系统责任发现所需要的关联

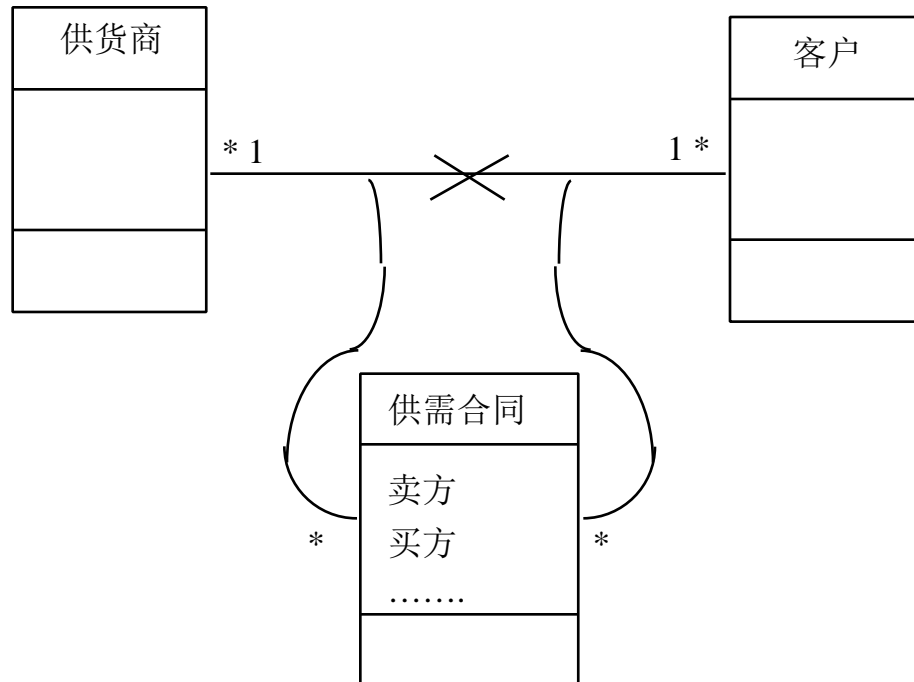
哪些类的对象实例之间存在着对用户业务有意义的关系？

- 问题域中实际事物之间有哪些值得注意的关系？
- 这种信息是否需要通过有序对（或者n元组）来体现？
- 这些信息是否需要在系统中进行保存、管理或维护？
- 系统是否需要查阅和使用由这种关系所体现的信息？

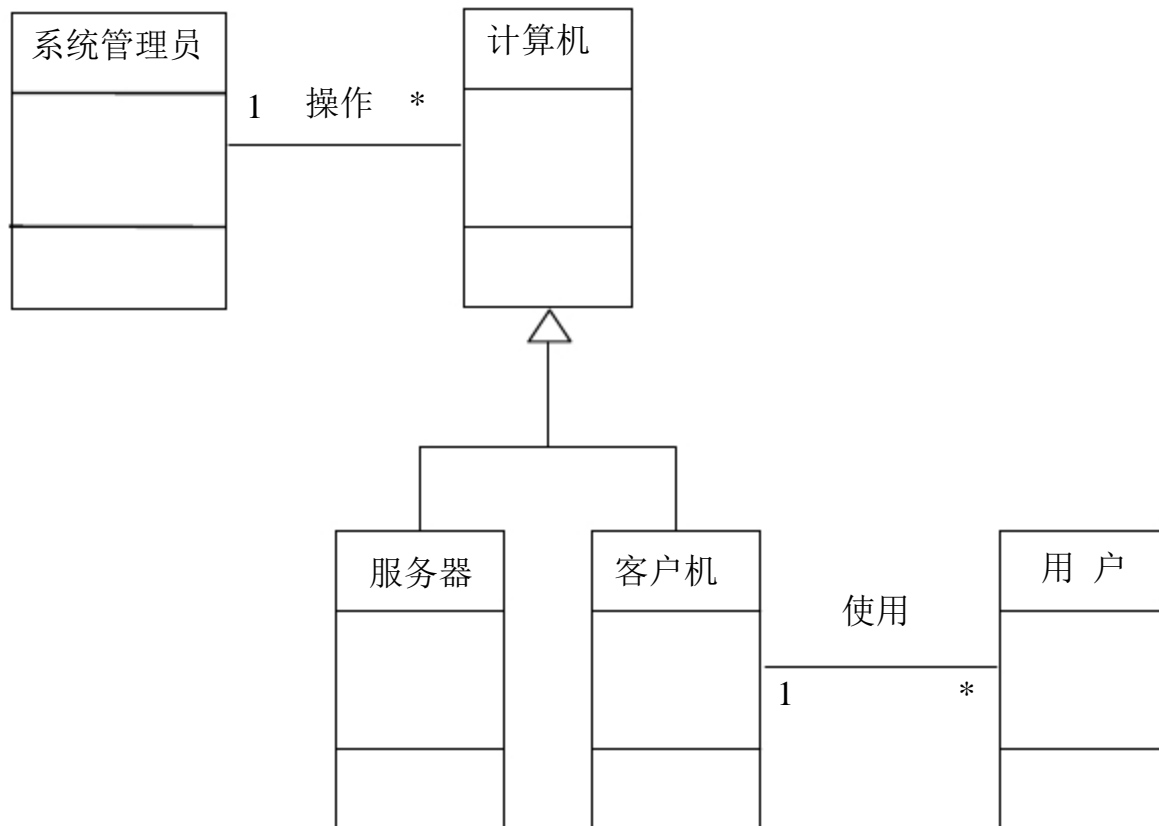
2. 关联的复杂情况处理

- 对关联属性和操作的处理
- 对**n**元关联的处理
- 避免一个类在关联中多次出现
- 多对多关联的处理

多对多关联
的处理



3. 关联定位



1、什么是消息（message）

现实生活中——人或其他事物之间传递的信息，例如：

人与人之间的对话、通信、发通知、留言

交通信号灯对车辆和行人发出的信号

人发给设备的遥控信号等.....

软件系统中——进程或软件成分之间传送的信息

控制信息 例如一次函数调用

数据信息 例如传送一个数据文件

面向对象的系统中——（按严格封装的要求）消息是对象之间在行为上的唯一联系方式

消息是向对象发出的服务请求（狭义）

消息是对象之间在一次交互中所传送的信息（广义）

消息有发送者和接收者，遵守共同约定的语法和语义

顺序系统中的消息

- 每个消息都是向对象发出的服务请求
 最常见的是函数调用
- 消息都是同步的。
- 接收者执行消息所请求的服务。
- 发送者等待消息处理完毕再继续执行。
- 每个消息只有唯一的接收者。

并发系统中的消息

控制流内部的消息——与顺序系统相同

控制流之间的消息——情况复杂得多

- 消息有多种用途

 - 服务请求，传送数据，发送通知，传递控制信号.....

- 消息有同步与异步之分

 - 同步消息（**synchronous message**）

 - 异步消息（**asynchronous message**）

- 接收者对消息有不同响应方式

 - 创建控制流，立即响应，延迟响应，不响应

- 发送者对消息处理结果有不同期待方式

 - 等待回应，事后查看结果，不等待不查看

- 消息的接收者可能不唯一

 - 定向消息（**directed message**）

 - 广播消息（**broadcast message**）

消息对面向对象建模的意义

消息体现了对象之间的行为依赖关系，是实现对象之间的动态联系，使系统成为一个能运行的整体，并使各个部分能够协调工作的关键因素。

在顺序系统中 消息体现了过程抽象的原则

一个对象的操作通过消息调用其他对象的操作

在OO模型中通过消息把对象操作贯穿在一起

系统实现后这些操作将在一个控制流中顺序地执行

在并发系统中

控制流内部的消息

使系统中的每个控制流呈现出清晰的脉络

控制流之间的消息




体现了控制流之间的通信关系

OO模型需要表示消息的哪些信息？（按重要性排序）

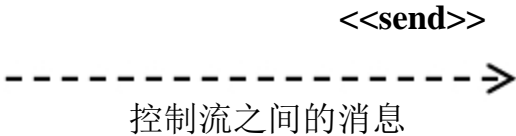
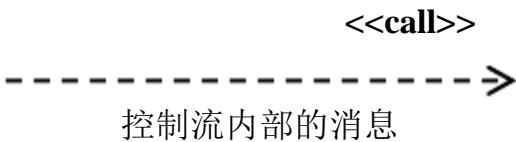
- （1）对象之间是否存在着某种消息？
- （2）这种消息是控制流内部的还是控制流之间的？
- （3）每一种消息是从发送者的哪个操作发出的？是由接收者的哪个操作响应和处理的？
- （4）消息是同步的还是异步的？
- （5）发送者是否等待消息的处理结果？

用什么符号表示消息

UML对各种箭头的用法

箭头种类	图形符号	用 途
实线开放箭头		关联的导航性（类图） 异步消息（顺序图）
虚线开放箭头		依赖（类图、包图、用况图、构件图） 从消息接收者的操作返回（顺序图）
实线封闭箭头		同步消息（顺序图、协作图）

借用依赖关系表示类图中的消息



什么是依赖（**dependency**）

在以往的OO方法中，只有**Firesmith**方法用到这个概念，其大意是：“客户/服务者（**client/server**）关系，表示客户对服务者的依赖。”列举的情况包括：

消息传送——其中客户发送消息给服务者；

聚合——其中聚合体（客户）的定义依赖它的构成部分（服务者）；

继承——其中派生类（客户）依赖它的基类（服务者）以继承其特征。

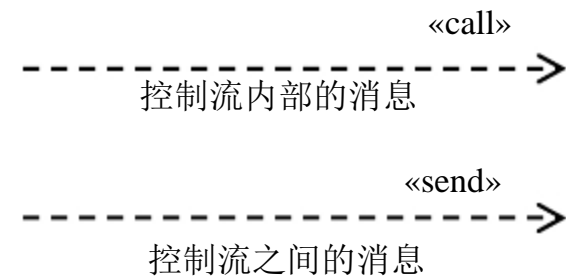
结论：在**Firesmith**方法中，依赖并不是对象之间的一种基本关系，而是为了指出在消息、聚合、继承等基本关系中哪个模型成分是客户（依赖者），哪个模型成分和服务者（被依赖者）所采用的一个概括性的术语。

依赖关系对面向对象建模的作用

继承、聚合、关联、消息这四种关系中都已经蕴涵了依赖的含义，不需要再用依赖关系再做重复的表示

除了上述关系之外，在**OO**建模中没有多少重要的信息必须用依赖关系表达

由于**UML**没有在类图中提供消息的表示法，可以借用依赖关系来表示类图中的消息



告戒：避免建立语义含糊不清的依赖关系，更要避免用这些含糊不清的依赖关系代替含义明确的**OO**关系。

建立一种依赖关系，就要具体地指出它是一种什么依赖。

习题

- 1、论述类与对象之间的关系以及关联与链之间的关系。这些概念之间还有什么关系吗？
- 2、针对下述问题，建立一个类图：有两种顾客，一种是常客，享受公司的一些优惠待遇；另一种是散户。
- 3、面包是由面包片组成的。讨论面包及其切片之间的关系。