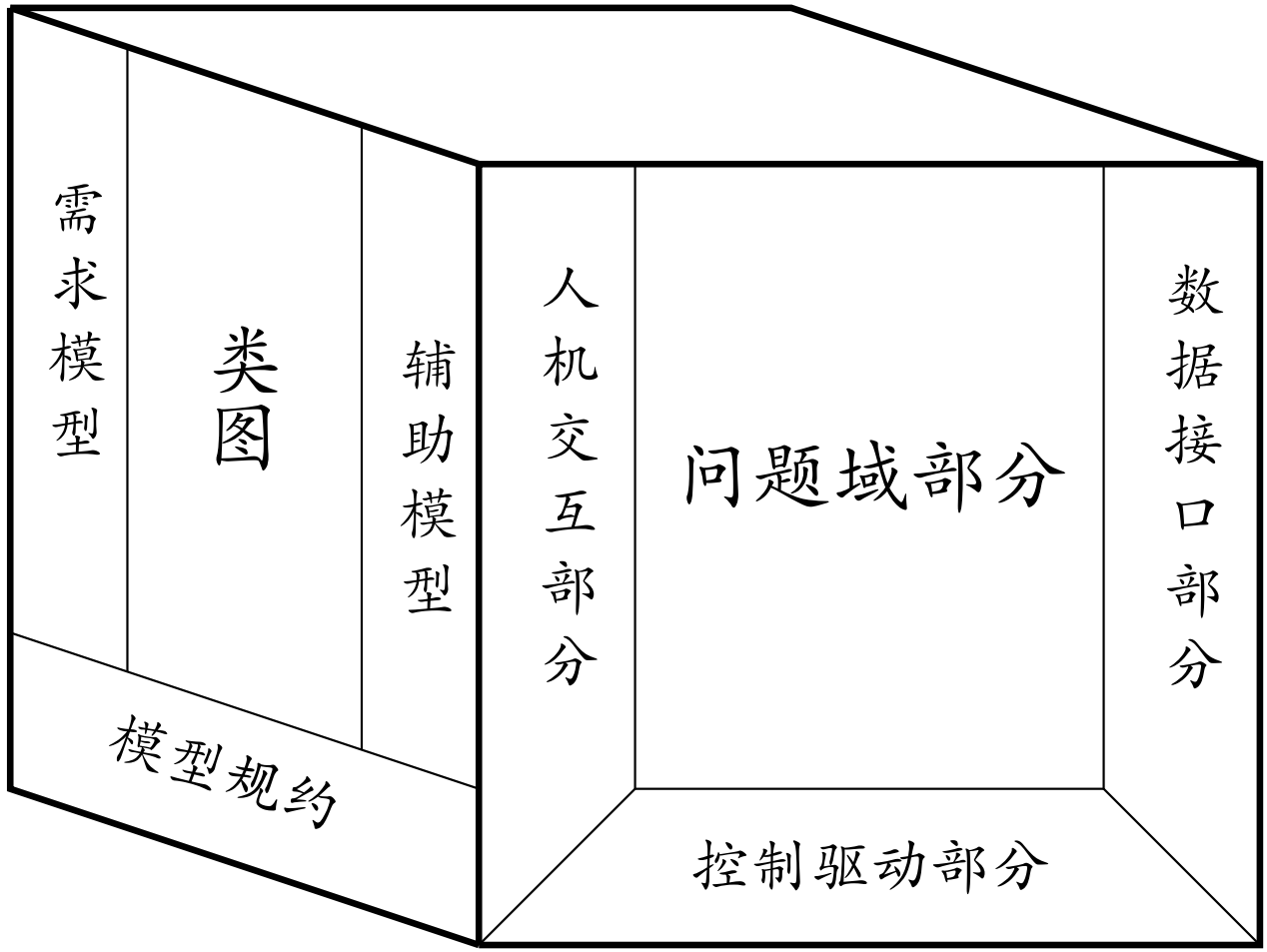
第七章　面向对象的系统设计 I

曹东刚
caodg@pku.edu.cn

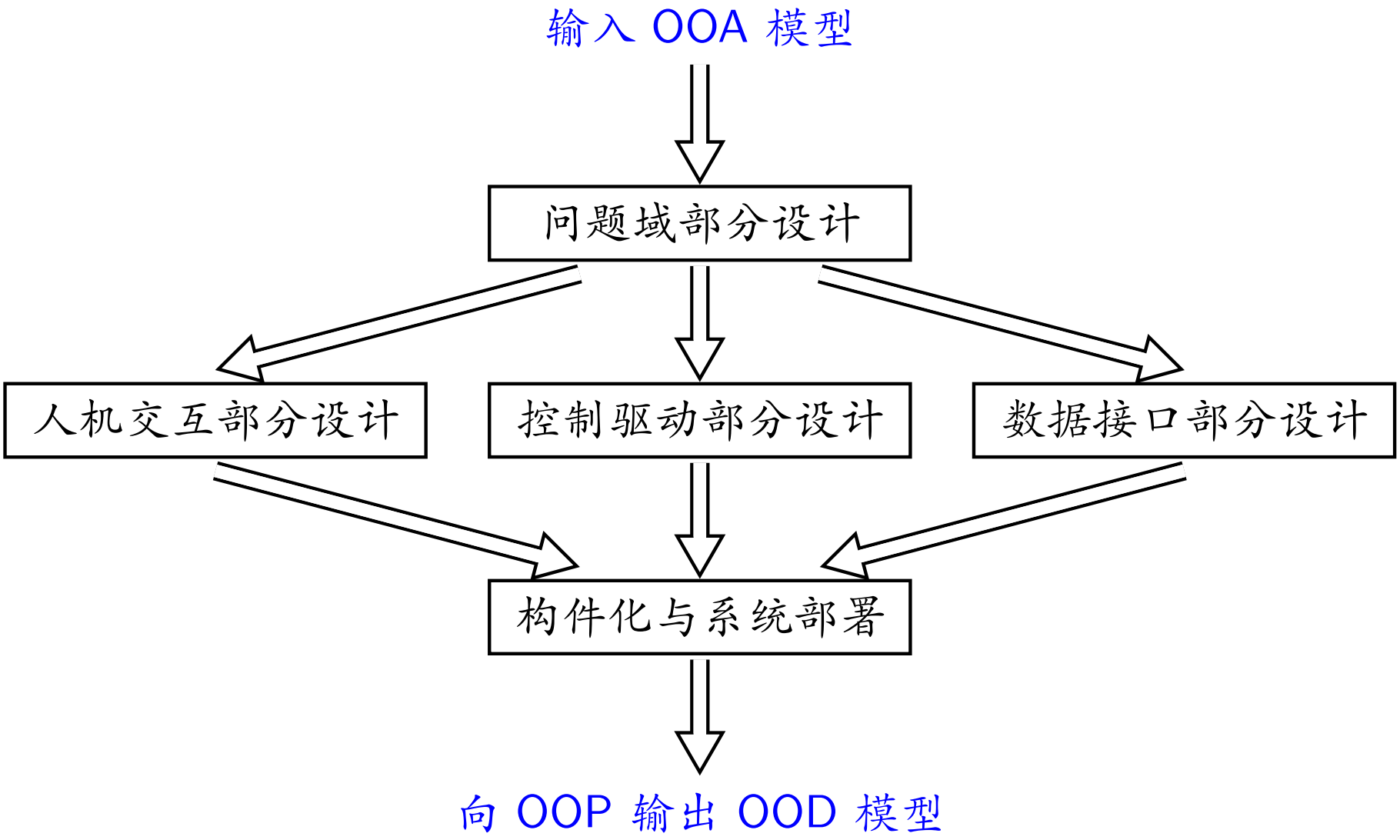
北京大学信息学院研究生课程 - 面向对象的分析与设计
http://sei.pku.edu.cn/~caodg/course/oo

OOD 框架—两个侧面

- OOD 模型包括几个主要部分?
- OOD 模型每个部分如何用 OO 概念表达?



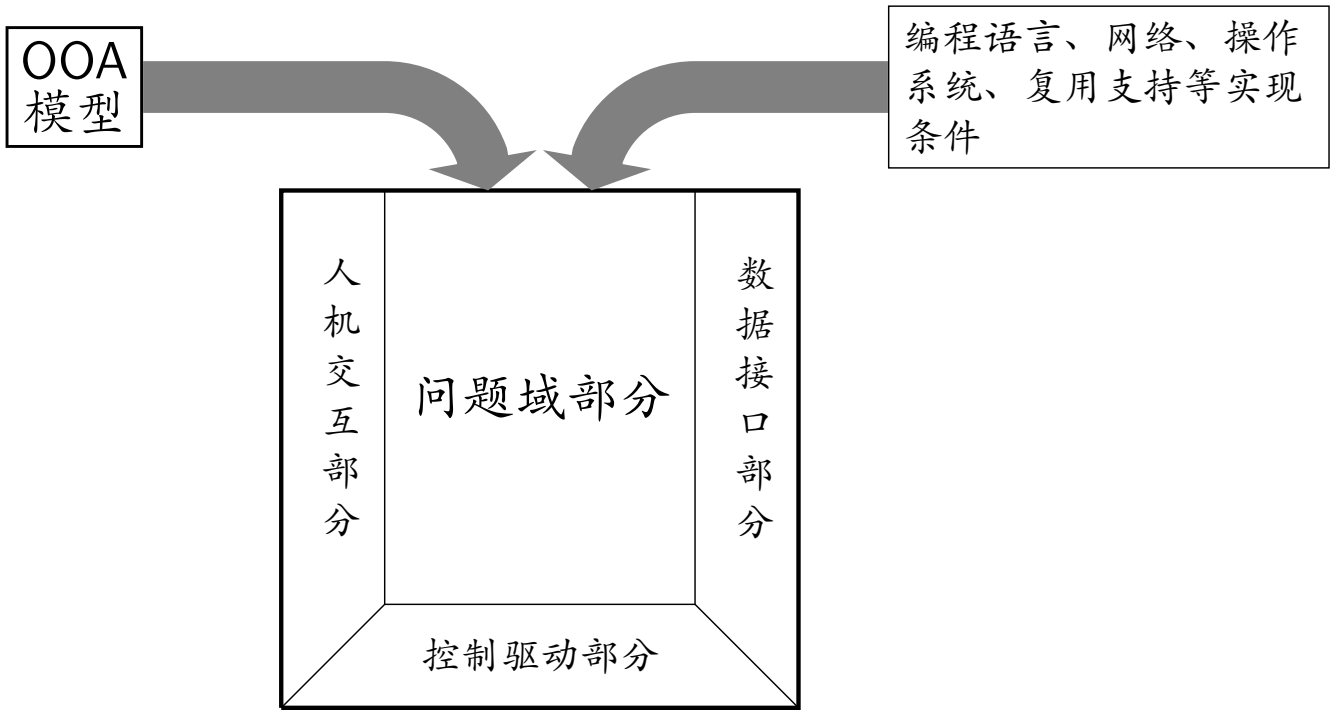
OOD 过程


输入 OOA 模型
问题域部分设计
人机交互部分设计
控制驱动部分设计
数据接口部分设计
构件化与系统部署
向 OOP 输出 OOD 模型

内容提要

什么是问题域部分

问题域部分是 OOD 模型的四个组成部分之一，由来自问题域的对象构成，是在 OOA 模型基础上，按照具体的实现条件进行必要的修改、调整和细节补充而得到的



实现条件对问题域部分的影响

- 编程语言
  - 语言的实现能力
- 硬件、操作系统及网络设施
  - 对象分布、并发、通信、性能
- 复用支持
  - 根据复用支持对模型做适当调整，以实现复用
- 数据管理系统
  - 为实现对象的持久存储，对问题域部分做某些修改
- 界面支持系统
  - 问题域部分与人机界面之间的消息传输

设计过程

- 设计准备
  - 保留 OOA 文档
  - 复制 OOA 文档，作为 OOD 的输入
    - 根据需求的变化和发现的错误进行修改

设计过程

- 设计内容与策略
  - 针对编程语言支持能力的调整
  - 增加一般类以建立共同协议
  - 实现复用
  - 提高性能
  - 为实现对象持久存储所做的修改
  - 完善对象的细节
  - 定义对象实例
  - 对辅助模型、模型规约的修改和补充
- 建立 OOD 文档与 OOA 文档的映射

内容提要

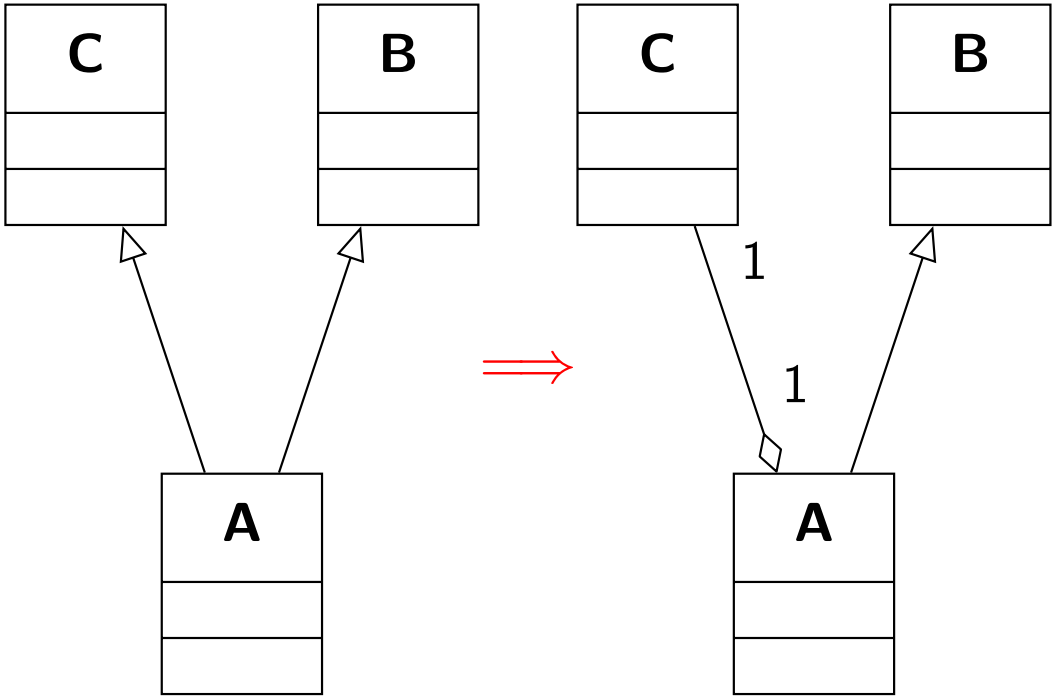
1. 按编程语言调整继承与多态

OOA 强调如实地反映问题域，OOD 考虑实现问题

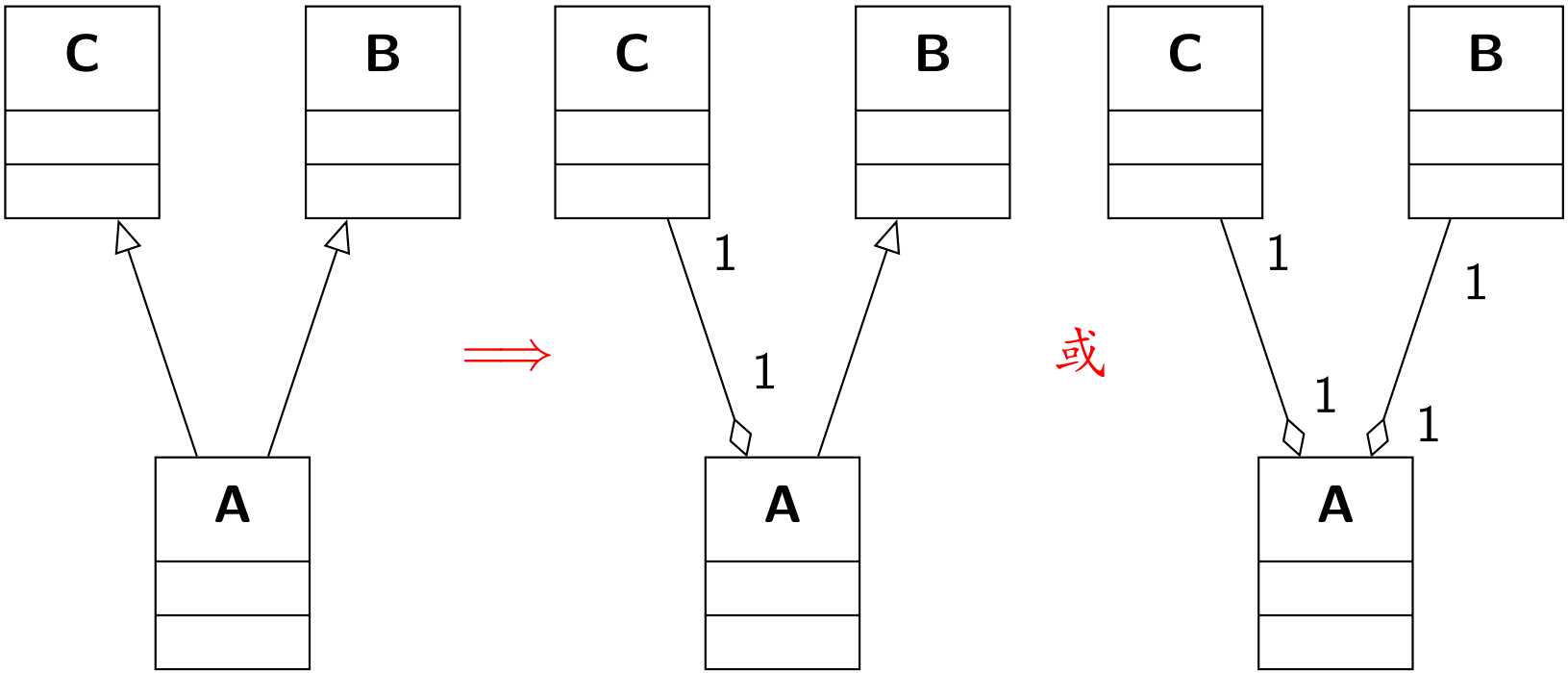
如果语言不支持多继承或多态，就要进行对模型调整

多继承化为单继承

多继承简单转换为单继承

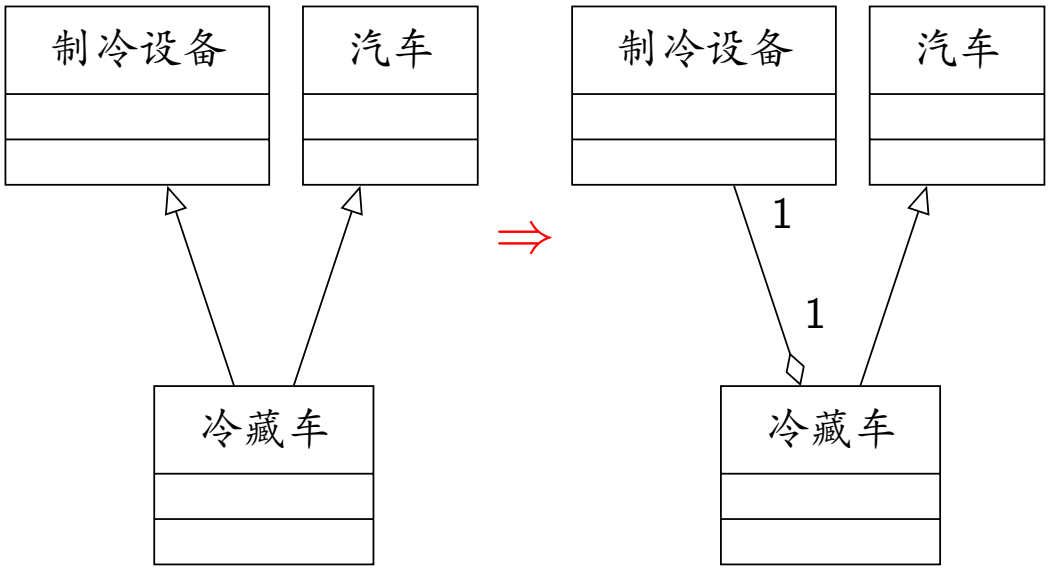


多继承简单转换为单继承



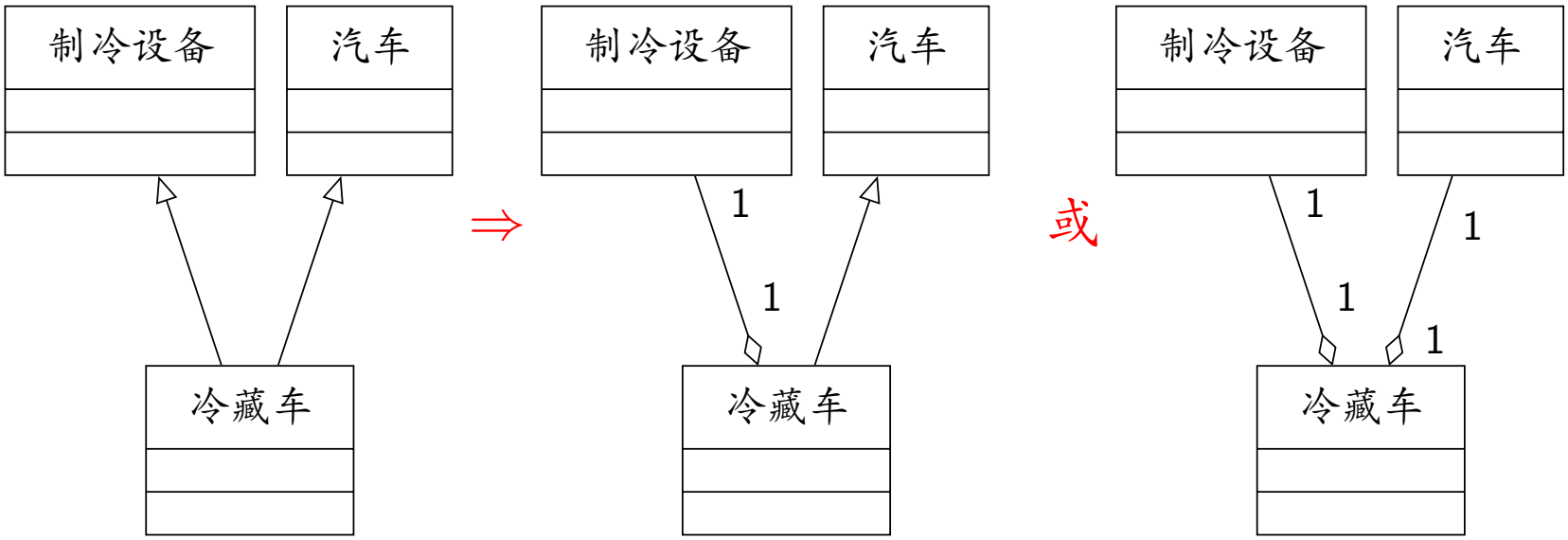
多继承简单转换为单继承

例子:



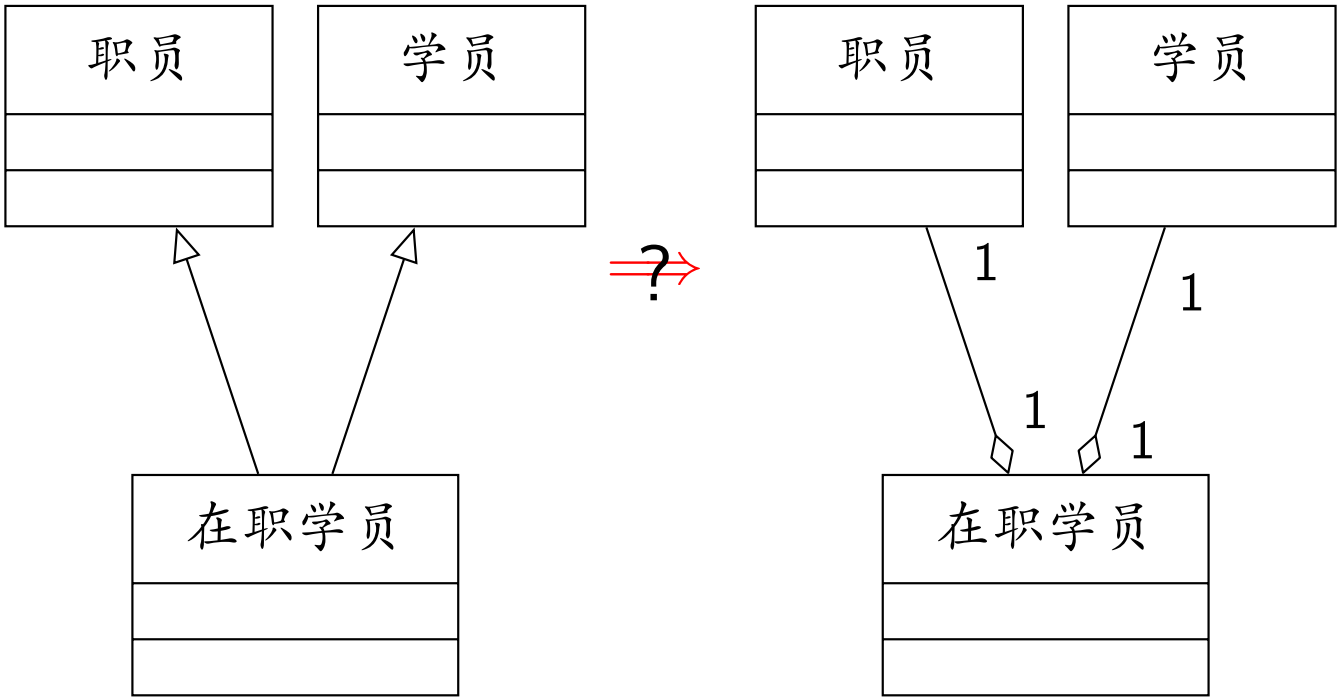
多继承简单转换为单继承

例子:



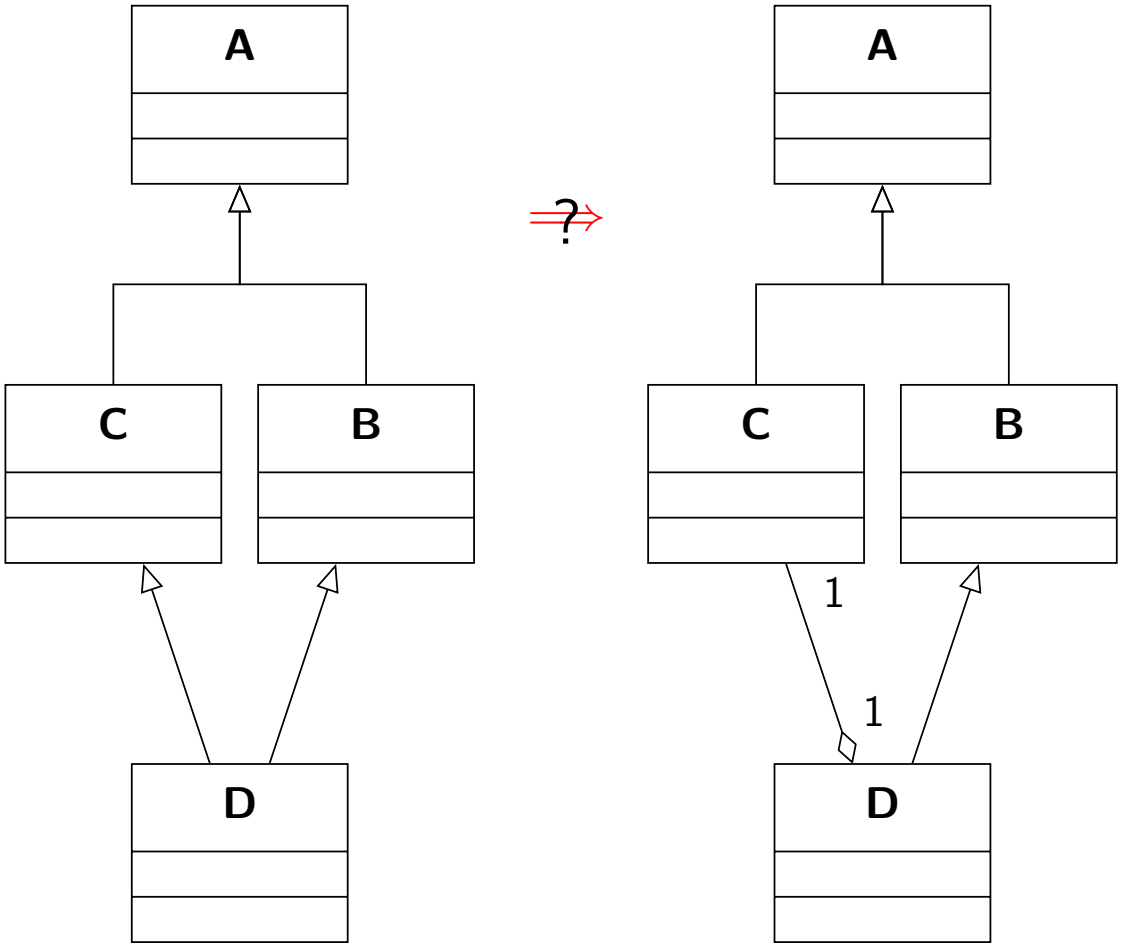
多继承简单转换为单继承

例子:

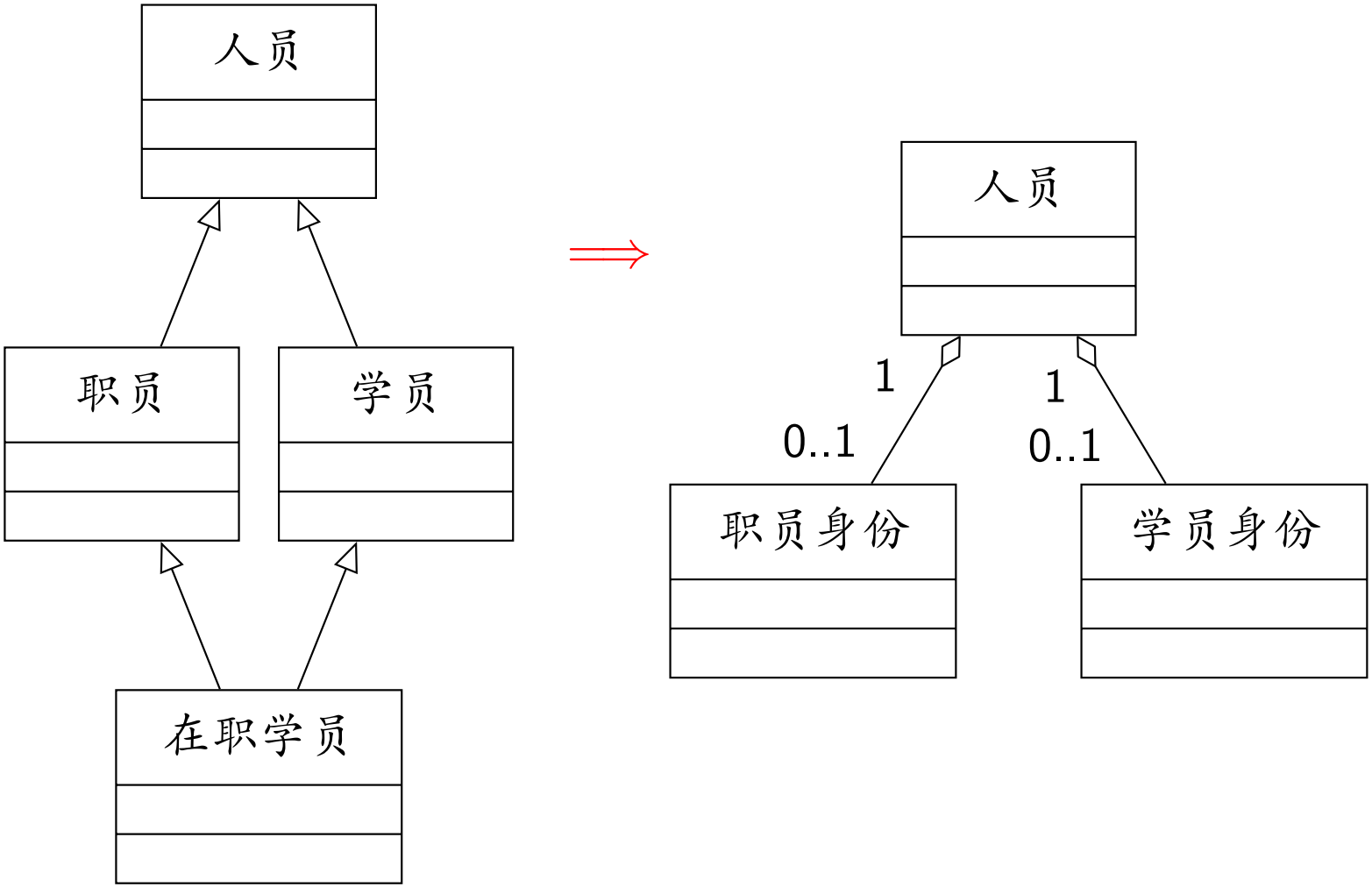


多继承简单转换为单继承

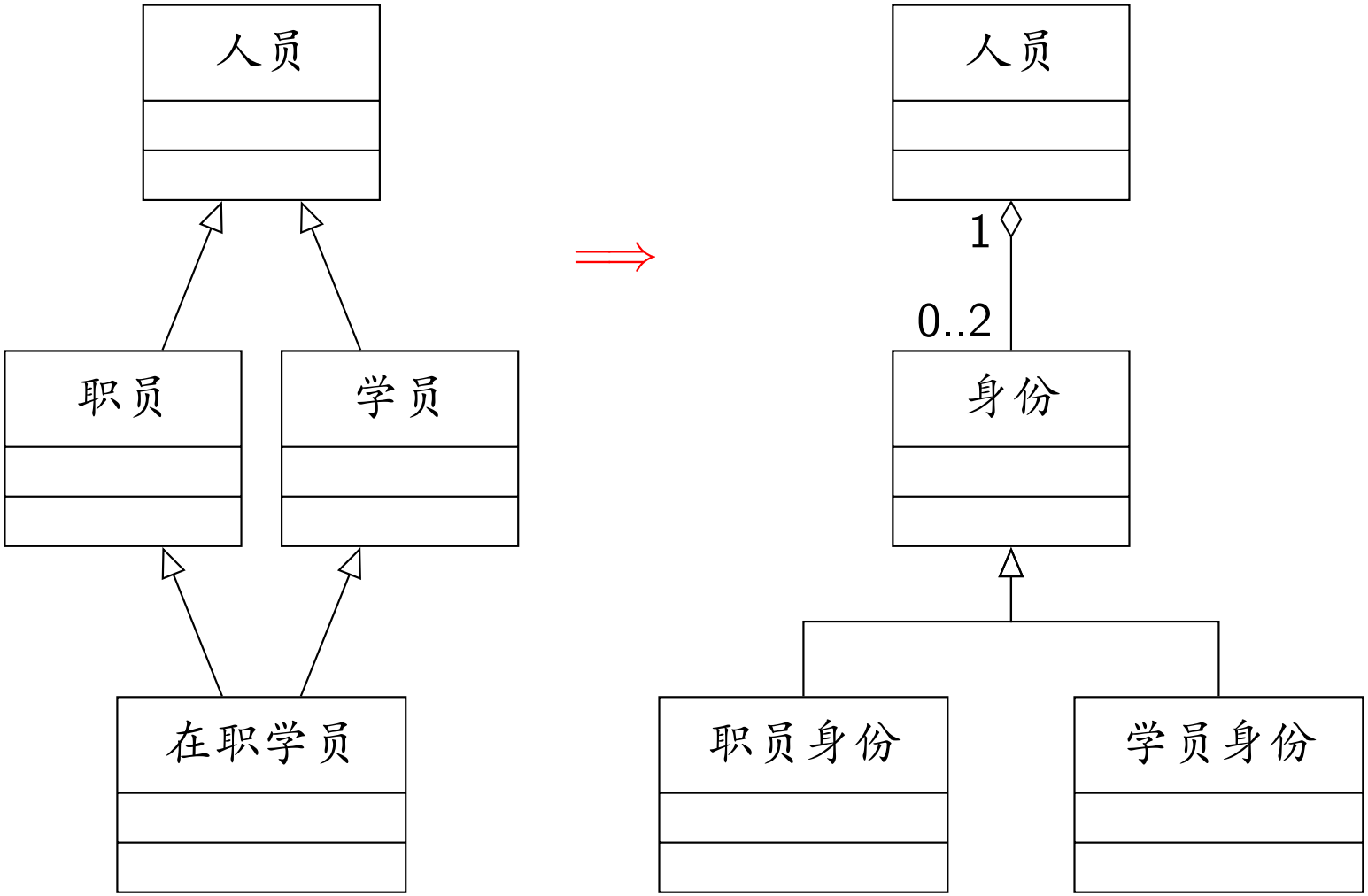
转换产生信息重复:



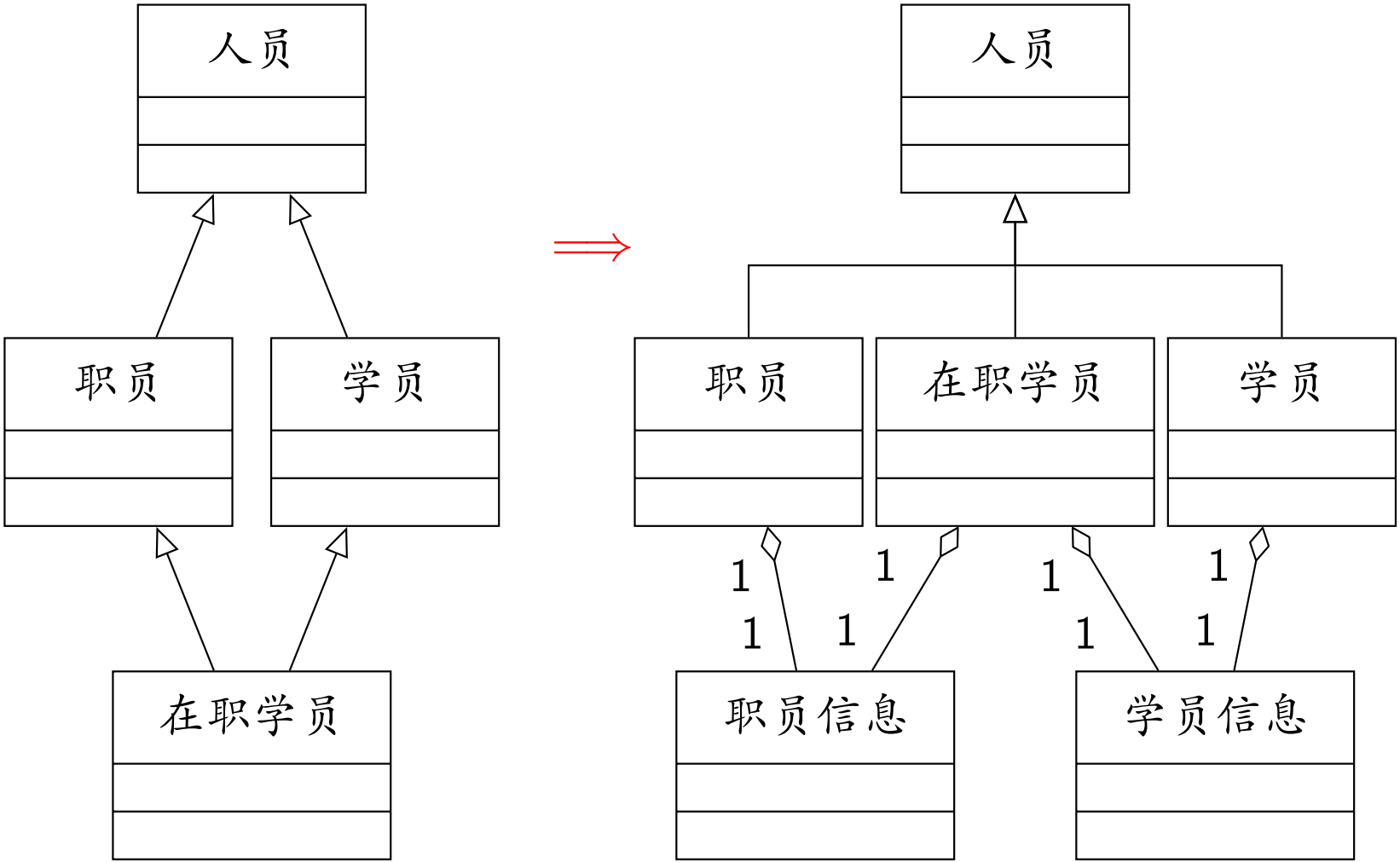
重新定义类化解多继承



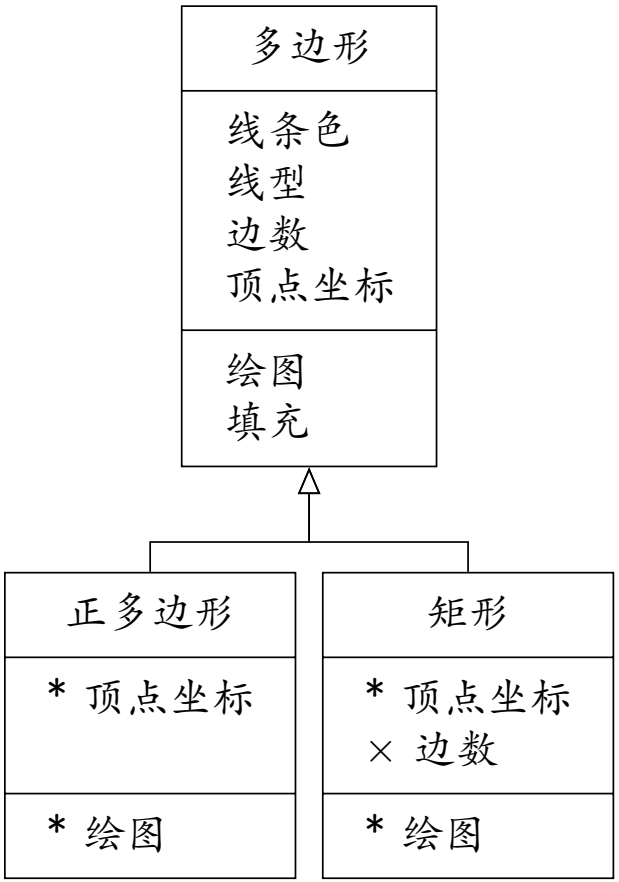
重新定义类化解多继承



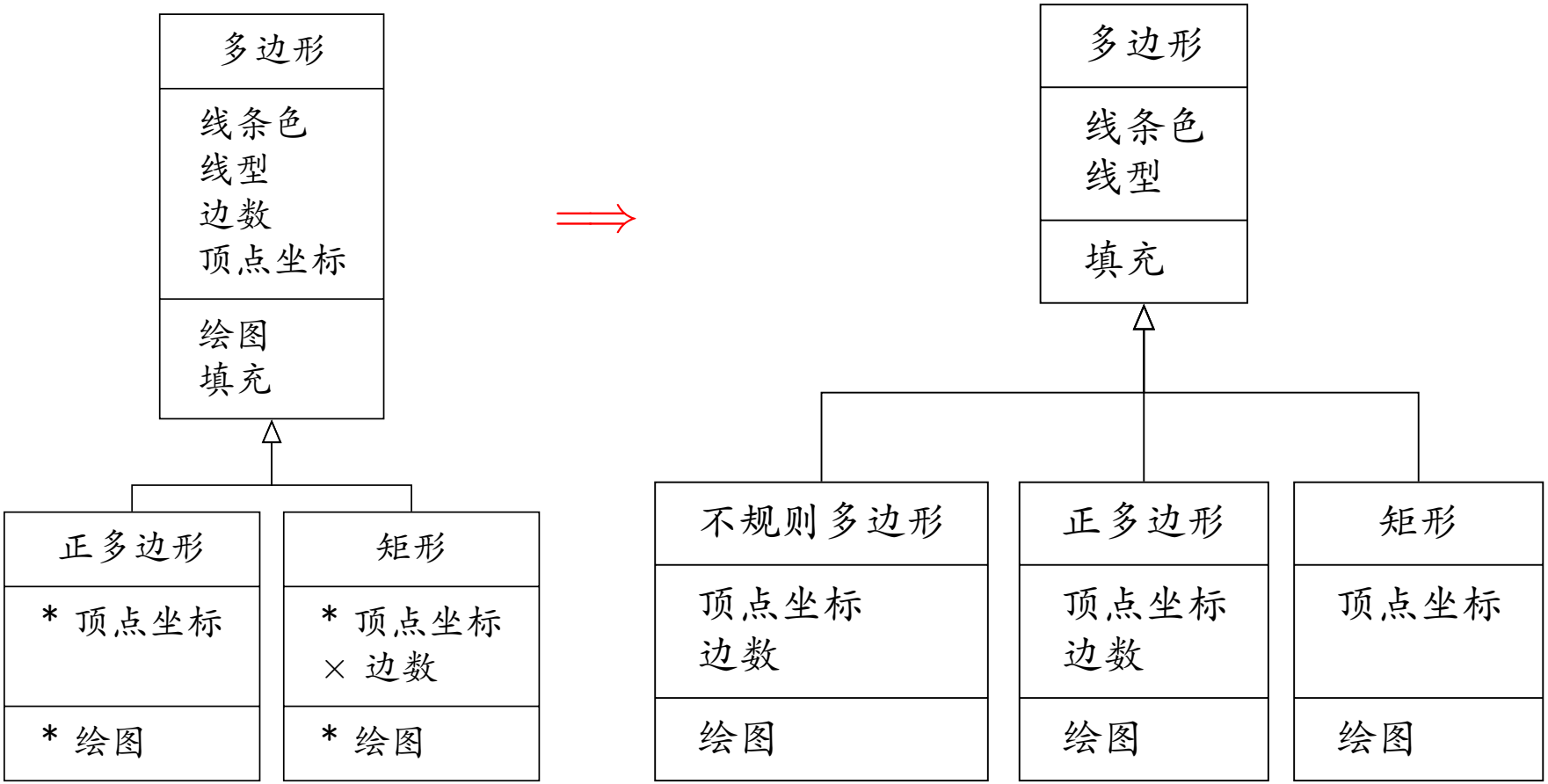
保持分类，剥离多继承信息



取消多态性



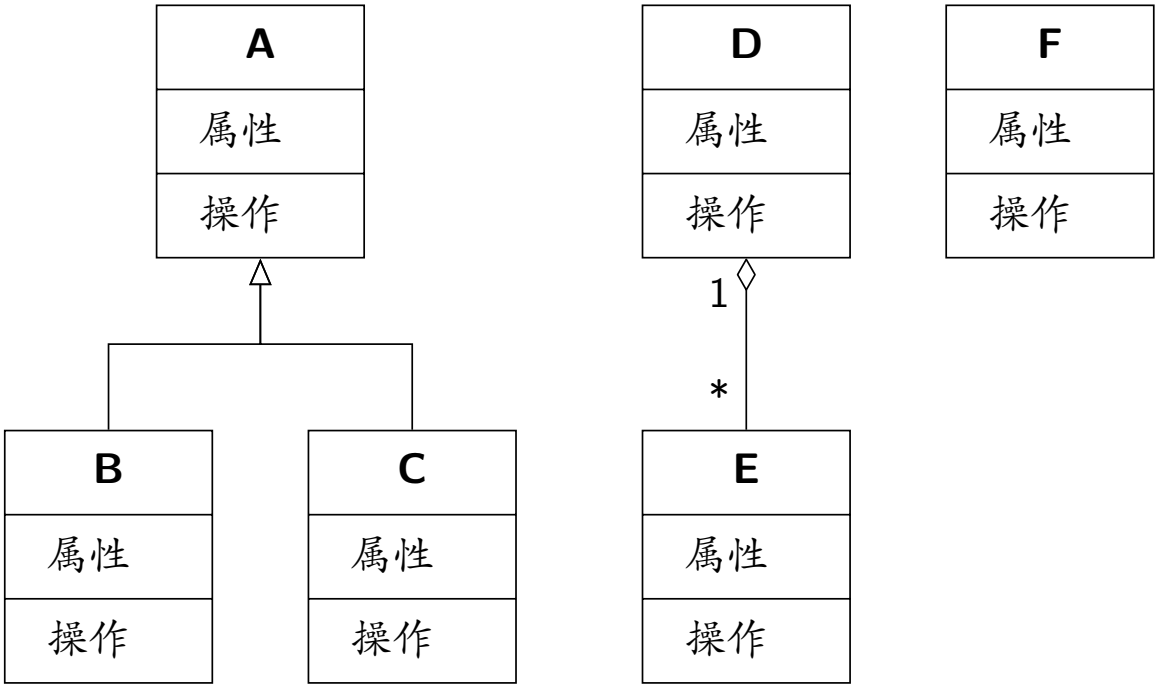
取消多态性


多边形
线条色
线型
边数
顶点坐标
绘图
填充
正多边形
* 顶点坐标
* 绘图
矩形
* 顶点坐标
× 边数
* 绘图
多边形
线条色
线型
填充
不规则多边形
顶点坐标
边数
绘图
正多边形
顶点坐标
边数
绘图
矩形
顶点坐标
绘图

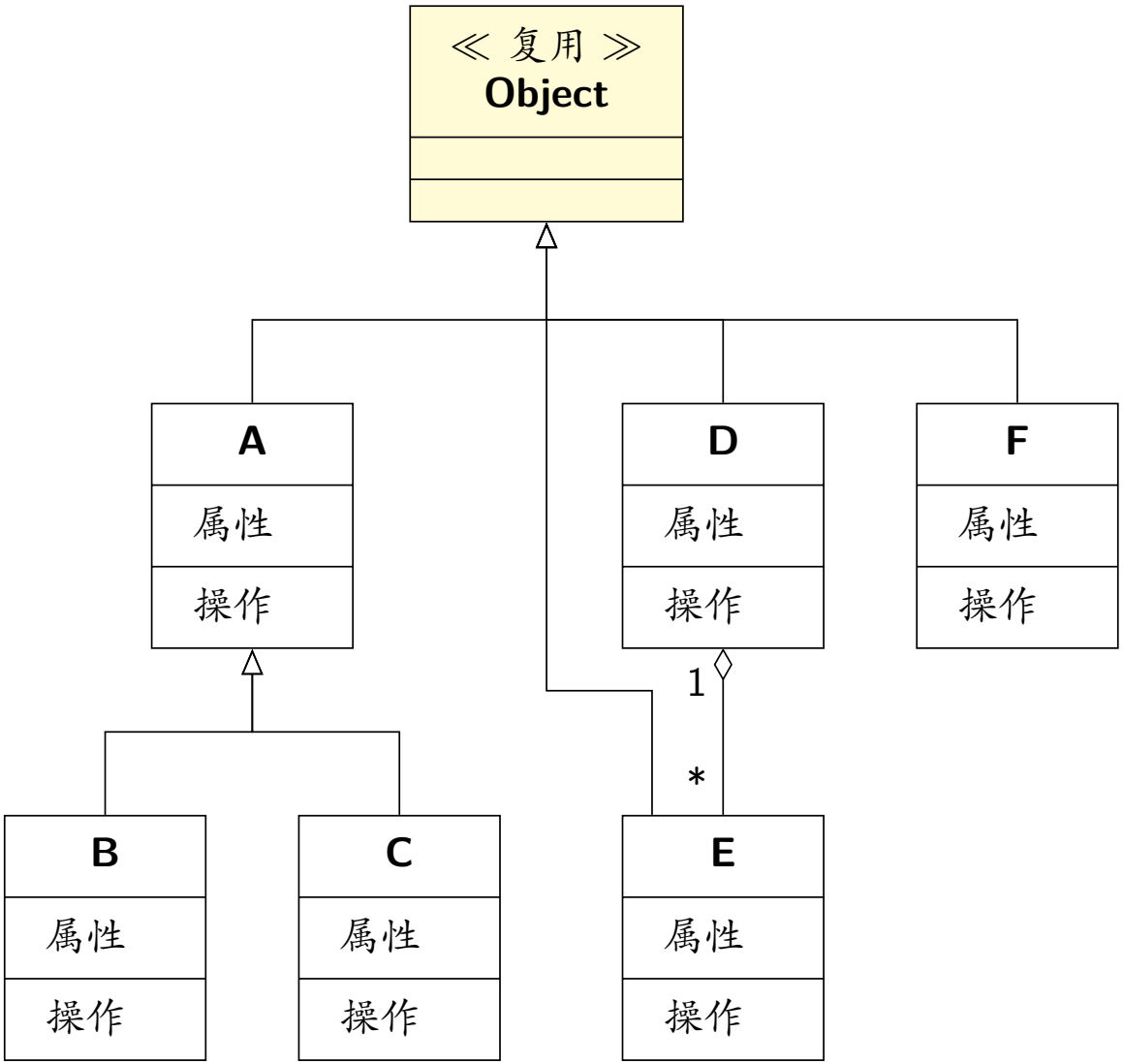
2. 增加一般类以建立共同协议

- 增加根类：将所有的类组织在一起，提供全系统通用的协议
  例：提供创建、删除、复制等操作
- 增加其他一般类：提供局部通用的协议
  例：提供持久存储及恢复功能

2. 增加一般类以建立共同协议



2. 增加一般类以建立共同协议



3. 实现复用的设计策略

如果已存在一些可复用的类，而且这些类既有分析、设计时的定义，又有源程序，那么，复用这些类可提高开发效率与质量

目标：尽可能使复用成分增多，新开发的成分减少

可复用类定义的信息 VS 当前所需的类的信息：

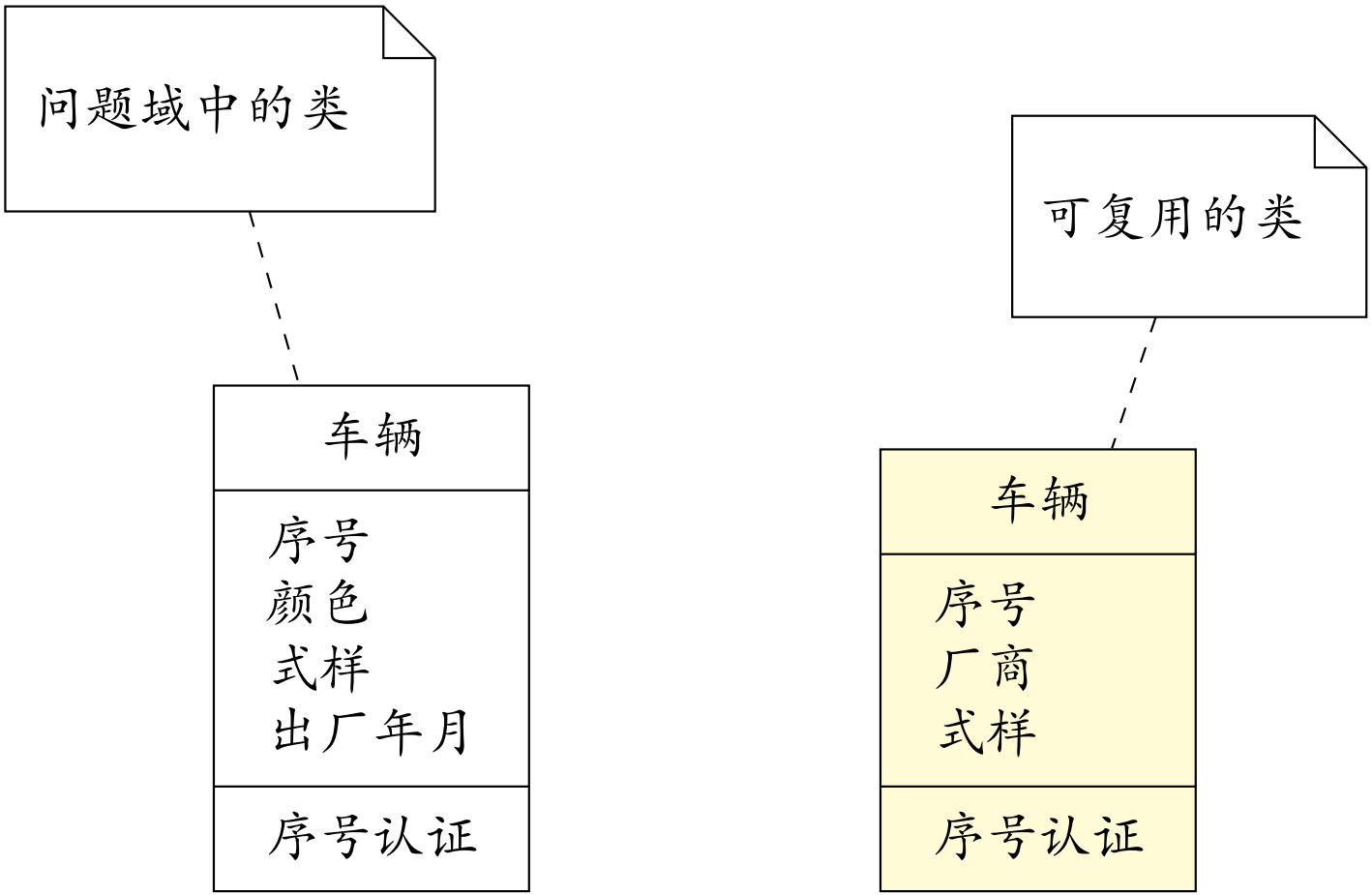
$=$ 直接复用

$<$ 通过继承复用

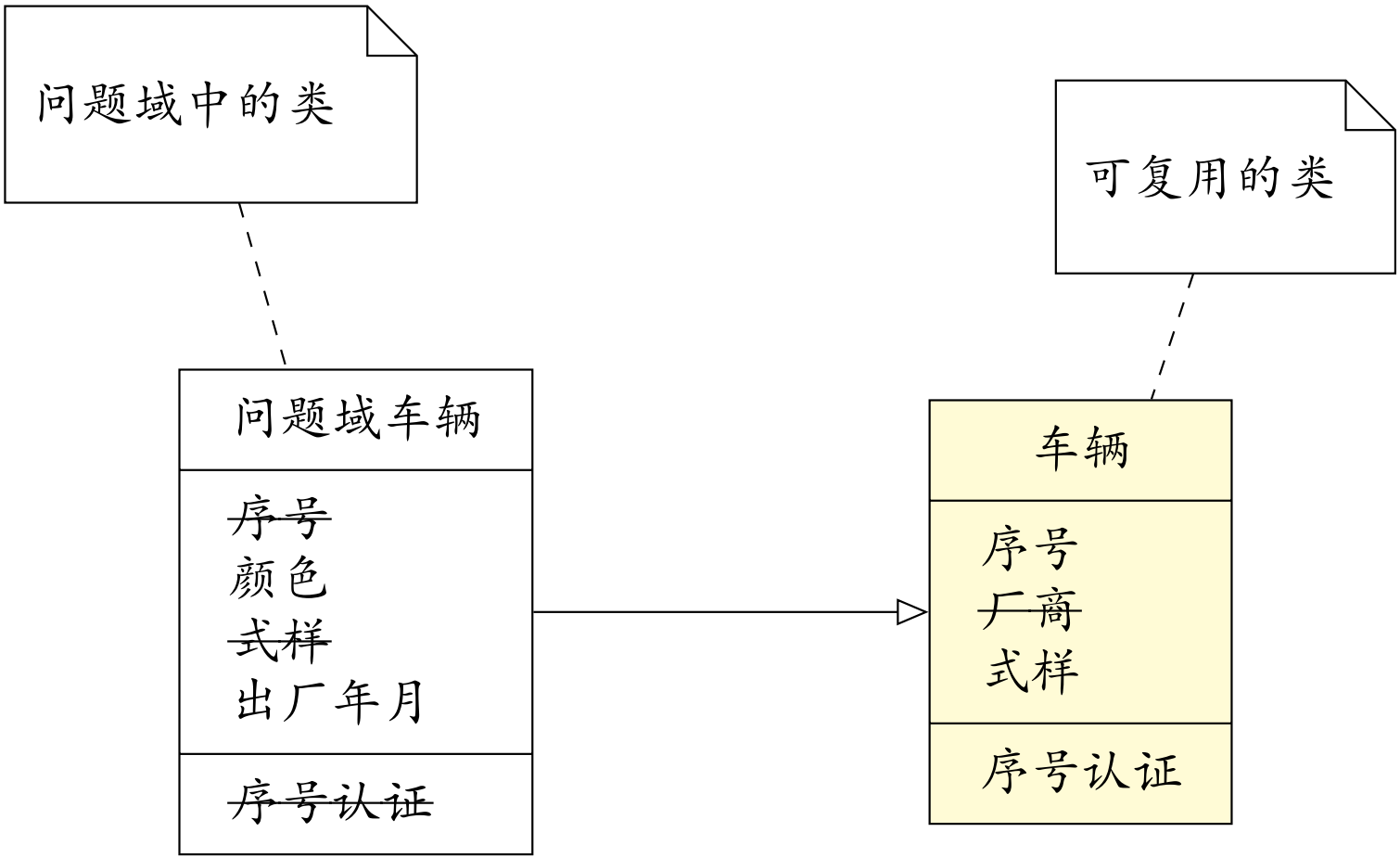
$>$ 删除可复用类的多余信息

$\approx$ 删除多余信息，通过继承而复用

示例



示例



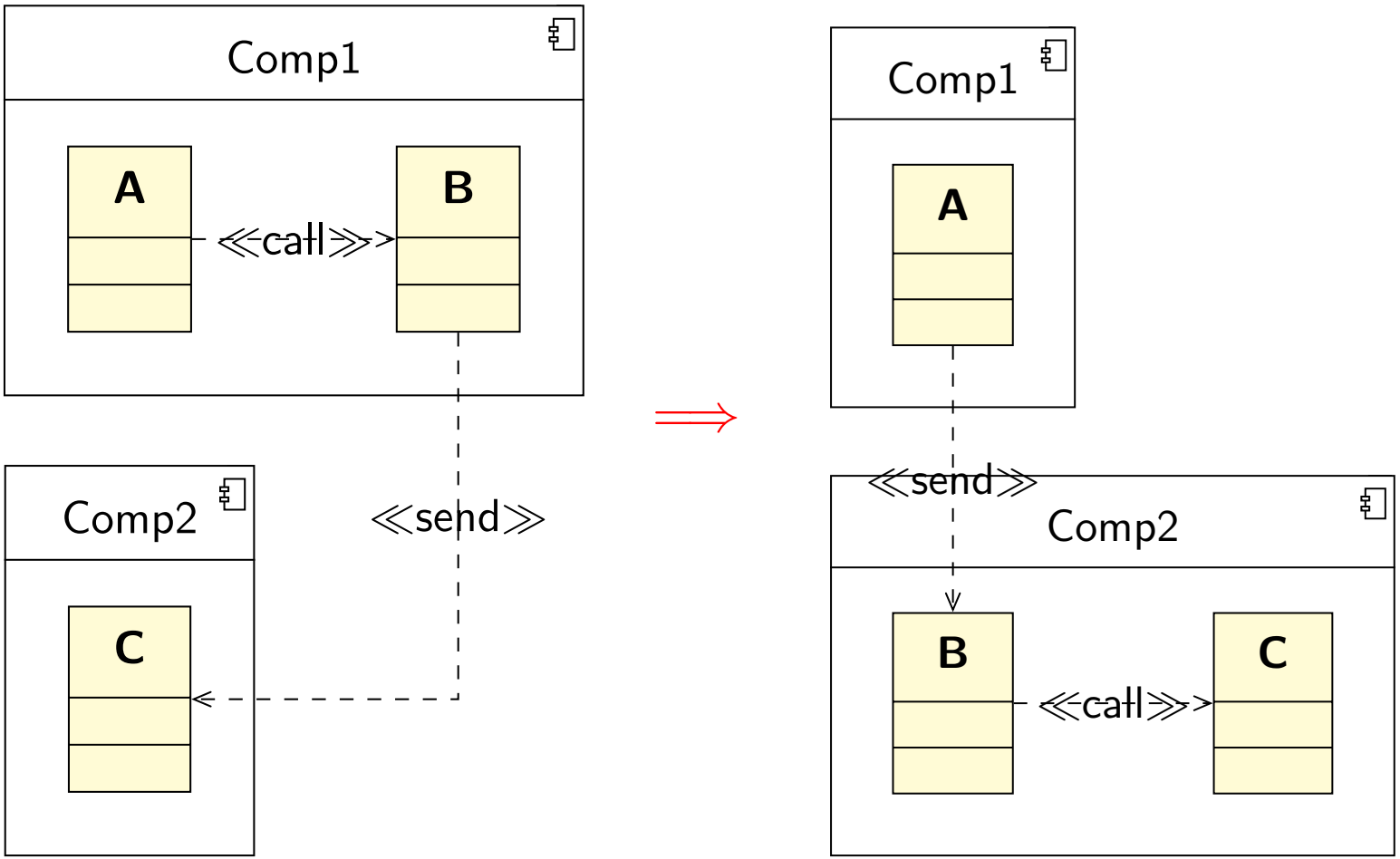
4. 提高性能

影响性能的因素

- 数据传输时间
- 数据存取时间
- 数据处理时间

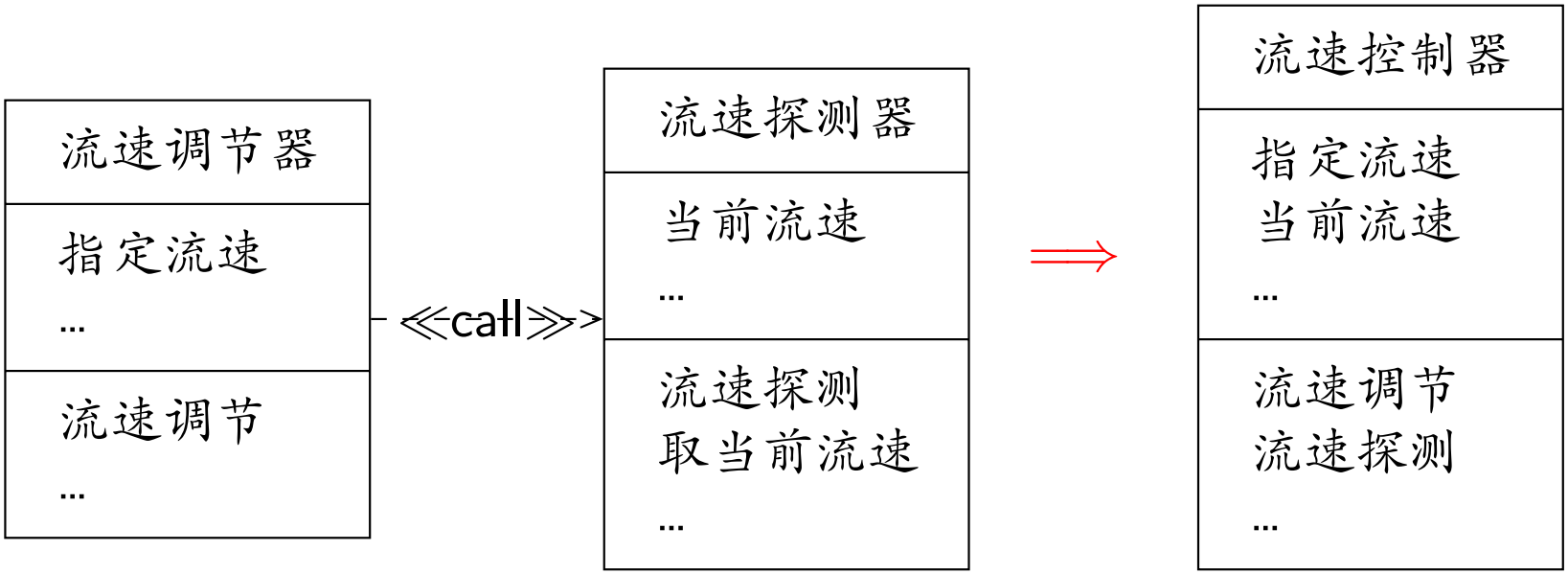
4. 改善性能的策略

1 调整对象分布



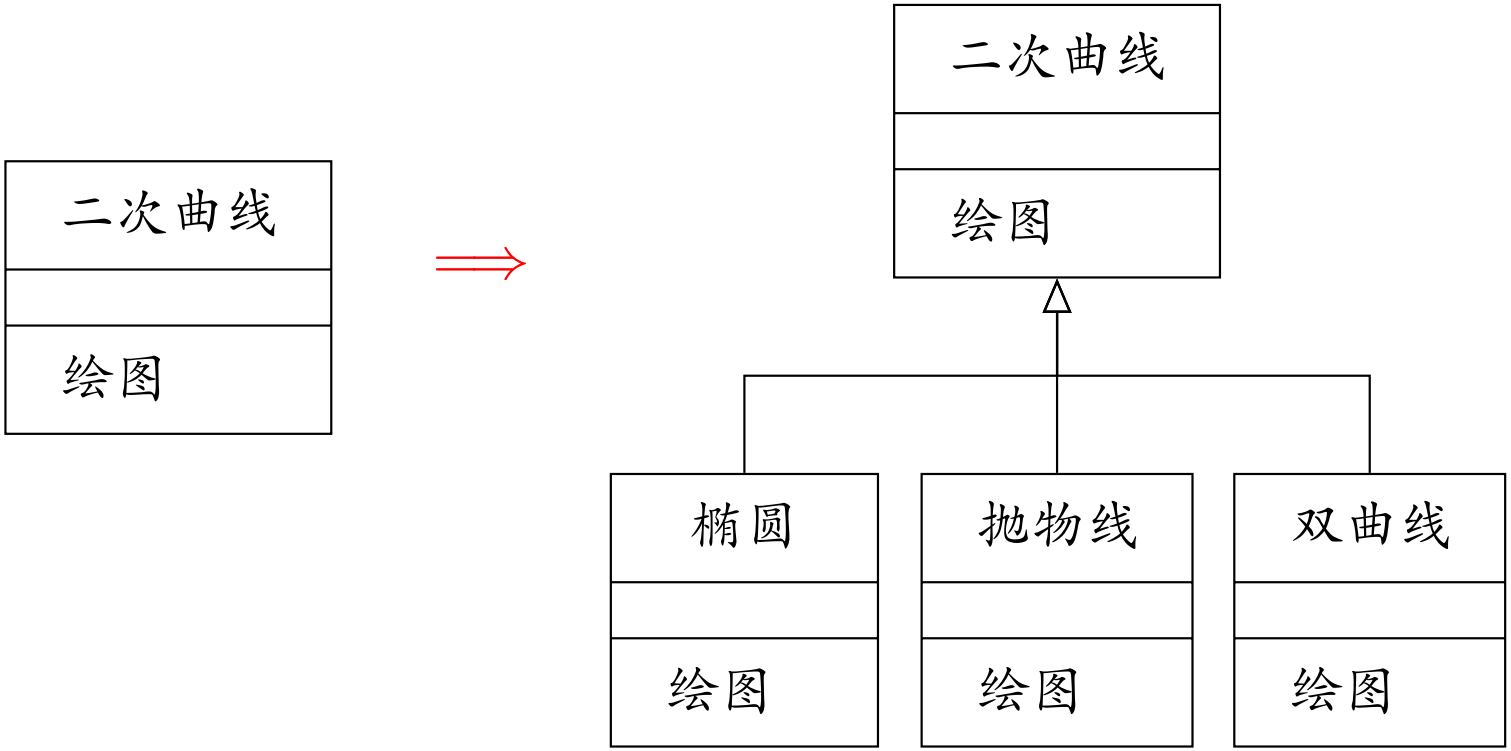
4. 改善性能的策略

2 合并通讯频繁的类



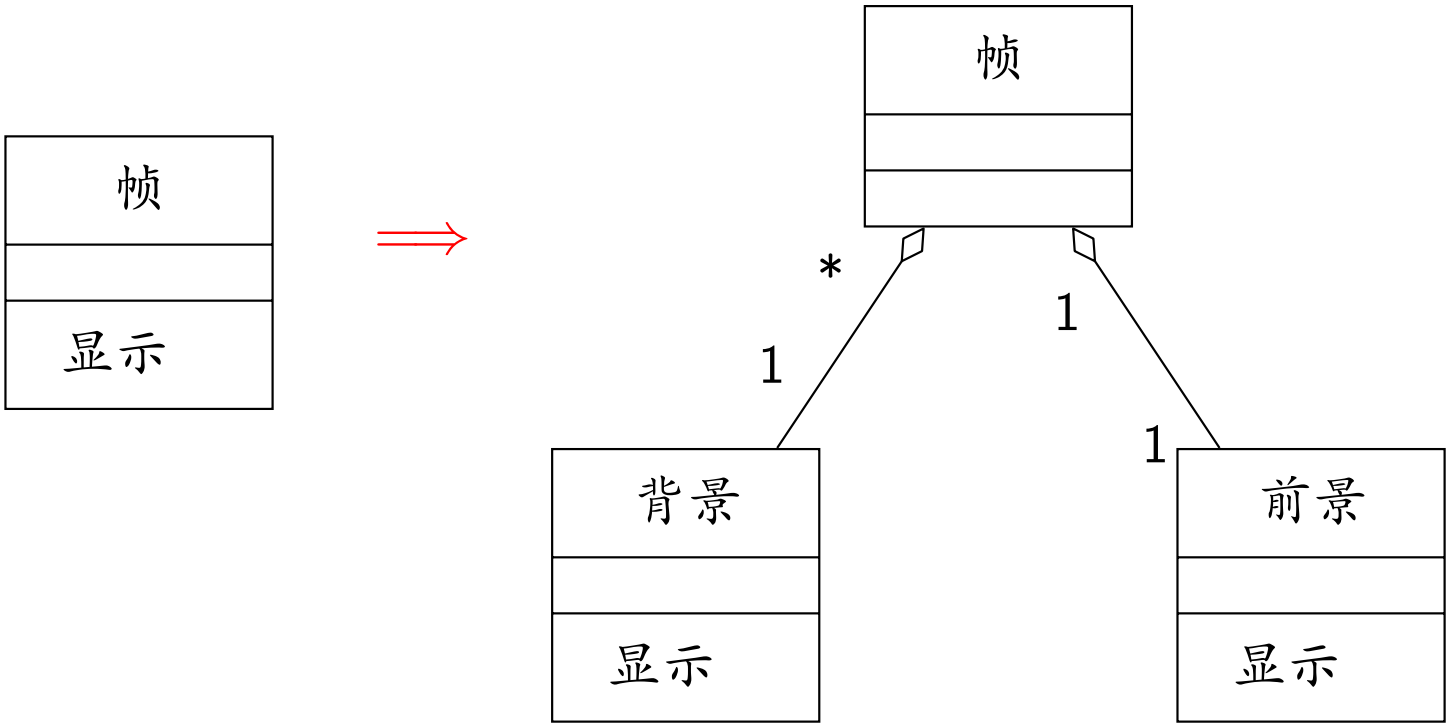
4. 改善性能的策略

3 细化对象的分类



4. 改善性能的策略

4 将复杂对象化为整体-部分结构



4. 改善性能的策略

5. 缩短对象存取时间，如设立缓冲区
6. 增加属性以减少重复计算
7. 降低算法的计算复杂性

5. 为数据存储管理增补属性与操作

在数据接口部分设计中介绍

6. 完善对象的细节

OOD 在 OOA 模型基础上所做的主要工作，不能用“细化”二字概括，但细化是不可缺少的

1. 完善与问题域有关的属性和操作
   - 在 OOA 阶段允许不详尽，OOD 必须加以完善
2. 解决 OOA 阶段推迟考虑的问题，包括：
   - 因封装原则而设立的对象操作
   - 与 OOD 模型其他部分有关的属性和操作
3. 设计类的每个操作
   - 必要时用流程图或者活动图表示

6. 完善对象的细节

OOD 在 OOA 模型基础上所做的主要工作，不能用“细化”二字概括，但细化是不可缺少的

4. 设计表示关联的属性
   - 区分多重性的 3 种情况，决定属性设置在哪一端
5. 设计表示聚合的属性
   - 区分组合与松散的聚合
     - 对于组合，用嵌套对象实现
     - 对于松散的聚合，采用与关联相同的策略

7. 定义对象实例

- 在逻辑上，一个类的对象实例是：
  - 问题域中所有可用这个类描述的实际事物
- 在物理上，一个类的对象实例可以是：
  - 内存中的对象变量
  - 文件的一个记录，或数据库表的一个元组
- 一个类的对象实例可以分布到不同的处理机上，对每一台处理机
  - 说明在它之上创建的每一个（或组）内存对象
  - 说明在它之上保存的外存对象

7. 定义对象实例

```
1  类的对象实例说明:
2  {
3        处理机: <结点名>{,<结点名>}
4        内存对象: {<名称>[（n元数组）][<文字描述>]}
5        外存对象: {<名称>[<文字描述>]}
6  }
```

8. 修改或补充辅助模型及模型规约

- 包图
  - 类的增减、拆分、合并以及各个类之间关系的变化
- 顺序图
  - 操作与消息
- 活动图
  - 操作流程
- 其他模型图
  - 状态机图、定时图、交互概览图、组合结构图
- 模型规约
  - 类的属性、操作及其对外关系的修改或细化

建立与 OOA 文档的映射

映射方式	**OOA** 类	**OOD** 类
$1 = 1$		
1 to 1		
1 to m		
m to 1		
m to m		
0 to 1		

内容提要

什么是人机交互部分

人机交互部分

是 OOD 模型的外围组成部分之一，是系统中负责人机交互的部分。其中所包含的对象（称作界面对象）构成了系统的人机界面

人机交互部分既取决于需求，又与界面支持系统密切相关

- 界面支持系统：支持人机界面开发的软件系统
- 人机界面：图形方式 vs 终端方式
- 图形界面开发工具

图形用户接口 GUI

一种用户接口，允许用户通过图形图标及各种可视化元素和电子设备交互

和 GUI 对应的其他接口有 CLI 和 TUI

界面支持系统

窗口系统（windowing system）

一种图形用户接口，为用户界面定义了 WIMP（窗口，图标，菜单，光标）风格，提供 API 支持应用系统界面开发。通过使用控件工具箱可简化应用程序界面开发工作
例：X11，Wayland，Quartz，ExtJS

控件工具箱（widget toolkit, widget library）

支持图形界面开发的控件（界面元素）集，包括标签、菜单、按钮、滚动条等基本控件，以及窗口、面板等容器控件
例：Motif，Mac OS X Cocoa，Windows API，QT，GTK+

界面支持系统

桌面环境（Desktop Environment）

桌面环境为系统提供了一套完整的图形用户接口，并通常集成了若干应用和工具，这些应用和工具有一致的 WIMP 风格和视感，基于共同的控件工具箱开发而成

例：CDE，GNOME, KDE, XFCE，MS Aero，Mac OS Aqua

界面支持系统

可视化编程环境

将窗口系统、可视化开发工具、编程语言以及控件工具箱等结合为一体的可视化开发平台，支持用户以“所见即所得”的方式构造用户界面

例：MS Visual Studio, NetBeans, Apple Xcode, QT Creator, App Inventor for Android

人机界面的开发

人机界面的开发不仅是设计和实现问题，也包括分析问题——对人机交互需求的分析。

人机界面的开发也不纯粹是软件问题，它还需要心理学、美学等许多其它学科的知识。

把人机交互部分作为系统中一个独立的组成部分进行分析和设计，有利于 隔离界面支持系统的变化对问题域部分的影响

内容提要

人机交互部分的需求分析

- 对使用系统的人进行分析
  以便设计出适合其特点的交互方式和界面表现形式
- 对人和机器的交互过程进行分析
  核心问题是人如何命令系统，以及系统如何向人提交信息

分析与系统交互的人（参与者）

人对界面的需求，不仅在于人机交互的内容，而且在于他们对界面表现形式、风格等方面的爱好。前者是客观需求，对谁都一样；后者是主观需求，因人而异

- 列举所有的人员参与者
- 区分人员类型
- 统计（或估算）各类人员的比例
- 调查研究使用者的情况
- 了解使用者的主观需求

从用况分析人机交互

用况的构成：参与者的行为和系统行为按时间顺序交替出现，左右分明，形成交叉排列的段落：

- 每个段落至少含有一个输入语句或输出语句；
- 有若干纯属参与者自身或系统自身的行为陈述；
- 可能包含一些控制语句或括号。

抽取方法：

- 删除所有与输入、输出无关的语句
- 删除不再包含任何内容的控制语句与括号
- 剩下的就是对一项功能的人机交互描述

示例：用况描述 1

收款用况 p1

```
1   收款
2   输入开始本次收款的命令;
3           作好收款准备，应收款总数
4           置为 0，输出提示信息;
5   for  顾客选购的每种商品 do
6     输入商品编号;
7     if  此种商品多于一件 then
8       输入商品数量
9     end if;
10          检索商品名称及单价;
11          货架商品数减去售出数;
```

示例：用况描述 2

收款用况 p2

```
12          if 货架商品数低于下限 then
13            通知供货员请求上货
14          end if;
15          计算本种商品总价并打印编号、
16          名称、数量、单价、总价;
17          总价累加到应收款总数;
18  end for;
19          打印应收款总数;
20  输入顾客交来的款数;
21          计算应找回的款数,
22          打印以上两个数目,
23          收款数计入账册。
```

示例：去除非交互内容 1

收款用况 p1

```
1   收款
2   输入开始本次收款的命令;
3           作好收款准备，应收款总数
4           置为 0，输出提示信息;
5   for   顾客选购的每种商品 do
6     输入商品编号;
7     if   此种商品多于一件 then
8       输入商品数量
9     end if;
10          检索商品名称及单价;
11          货架商品数减去售出数;
```

示例：去除非交互内容 2

收款用况 p2

```
12          if 货架商品数低于下限 then
13            通知供货员请求上货
14          end if;
15          计算本种商品总价并打印编号、
16          名称、数量、单价、总价;
17          总价累加到应收款总数;
18  end for;
19          打印应收款总数;
20  输入顾客交来的款数;
21          计算应找回的款数,
22          打印以上两个数目,
23          收款数计入账册。
```

示例：得到的人机交互描述

收款．人机交互

```
1  收款员．收款（人机交互）
2  输入开始本次收款的命令;
3                  输出提示信息;
4  for 顾客选购的每种商品 do
5    输入商品编号;
6      if 此种商品多于一件 then
7        输入商品数量
8      end if;
9                  打印商品编号、名称、数量、单价、总价;
10 end for;
11                 打印应收款总数
12 输入顾客交来的款数
13                 打印交款数及找回款数;
```

人机交互的细化

输入相比输出在人机交互中起到主导作用
一条输入相当于一条对系统的命令

输入的细化

- 输入步骤的细化
- 输入设备的选择
- 输入信息表现形式的选择

输出的细化

- 输出步骤的细化
- 输出设备的选择
- 输出信息表现形式的选择

命令的组织

不怎么受欢迎的命令组织方式：

- 一条命令含有大量的参数和任选项
- 系统有大量命令，不加任何组织和引导

命令的组织措施——分解与组合

分解：将含有许多参数和选项的命令分解为若干 命令步

组合：将基本命令组织成 高层命令，从高层命令引向基本命令

命令的组织

基本命令

使用一项独立的系统功能的命令

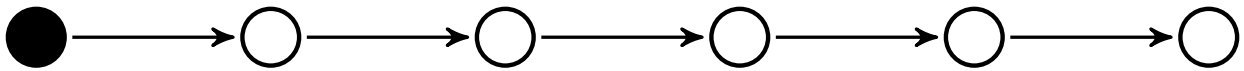
命令步

基本命令交互过程中所包含的具体输入步骤

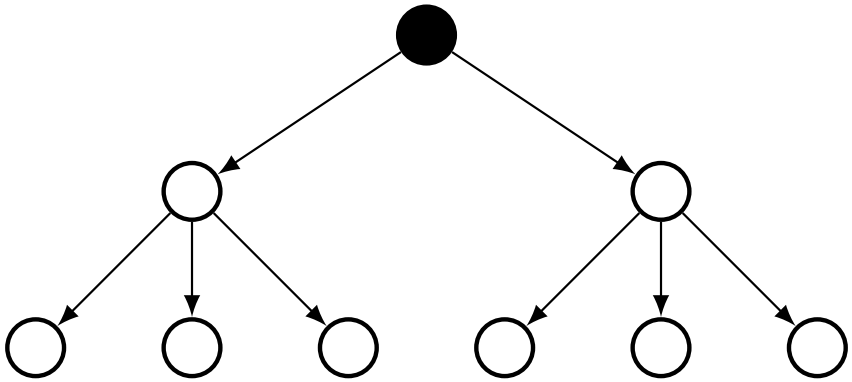
高层命令

由其他若干命令组合而成，起组织和引导作用

基本命令及其命令步的结构

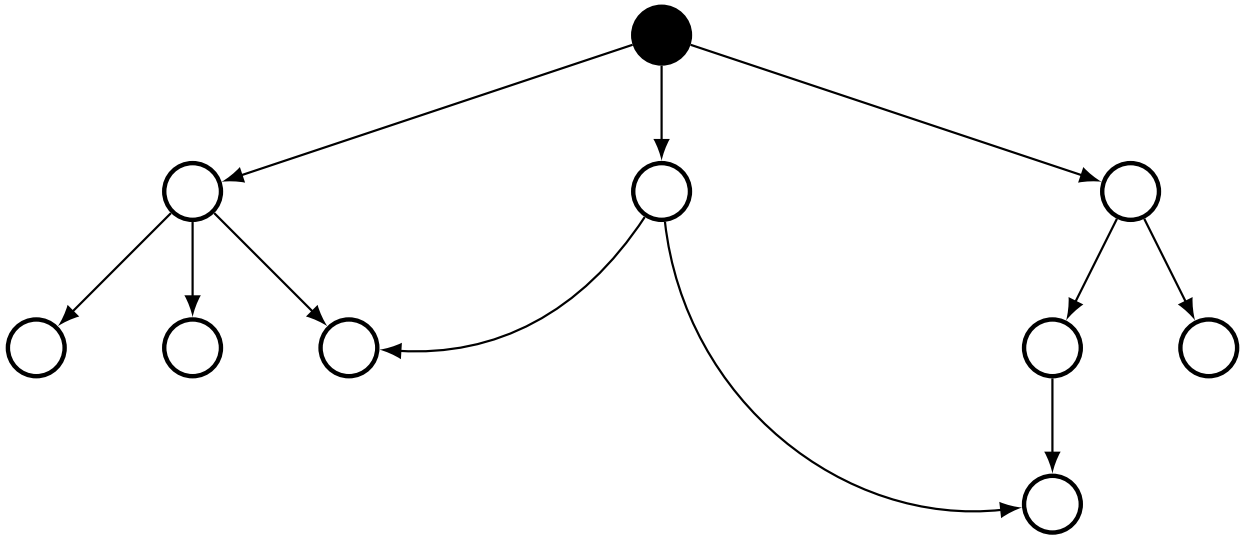


线性结构



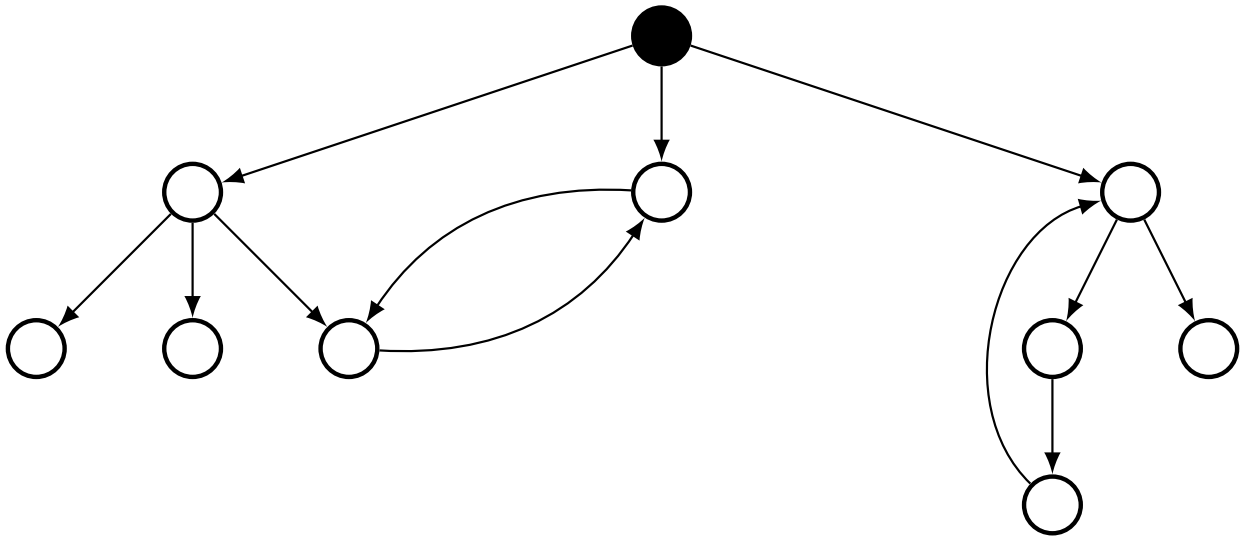
树形结构

基本命令及其命令步的结构



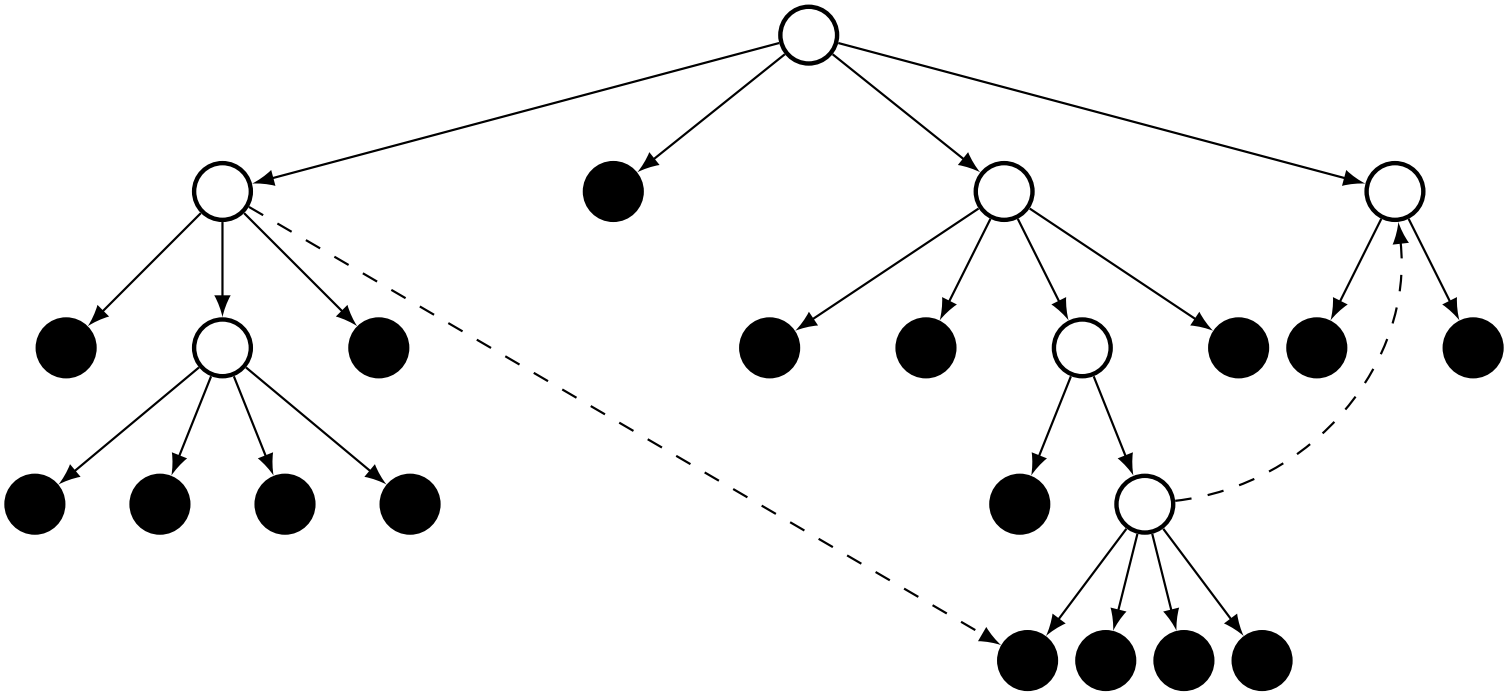
半序网状结构

基本命令及其命令步的结构

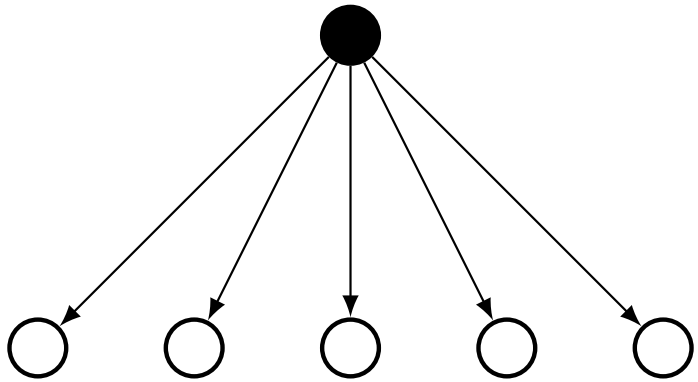


一般网状结构

高层命令的组织结构

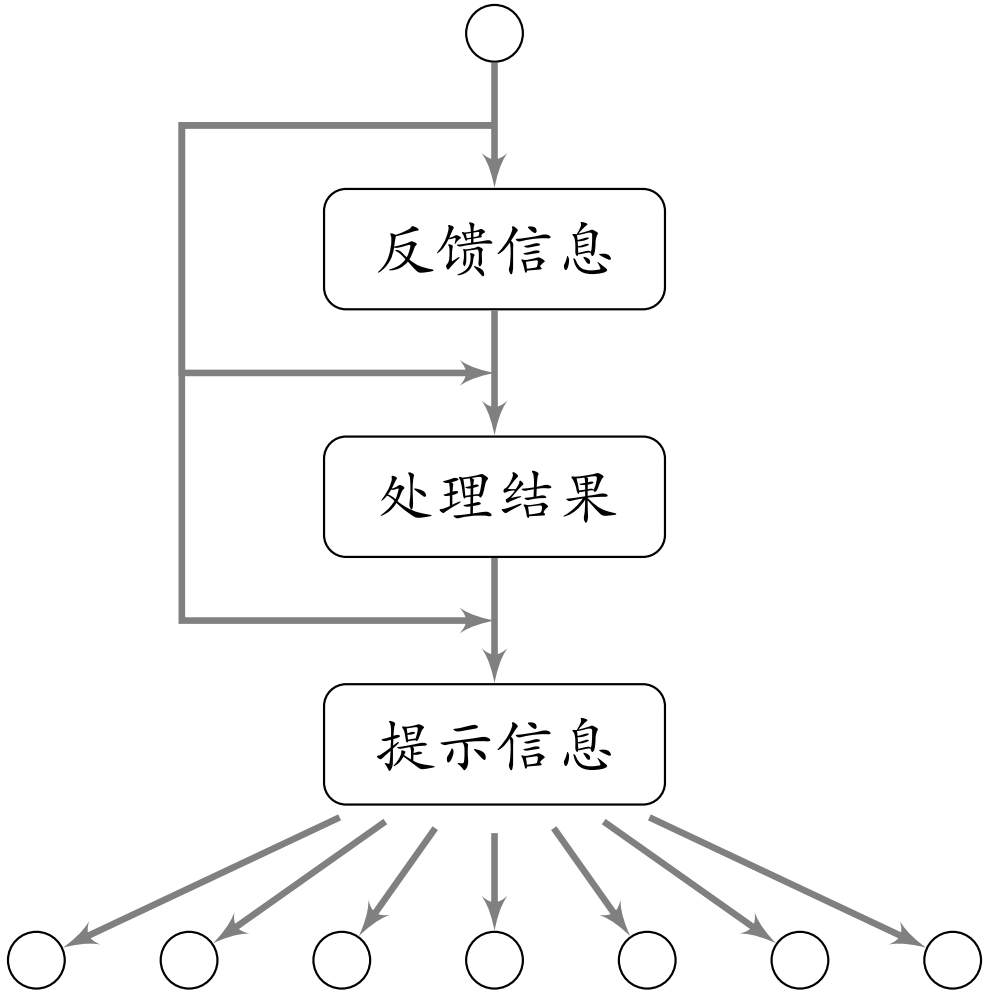


输出信息的组织结构



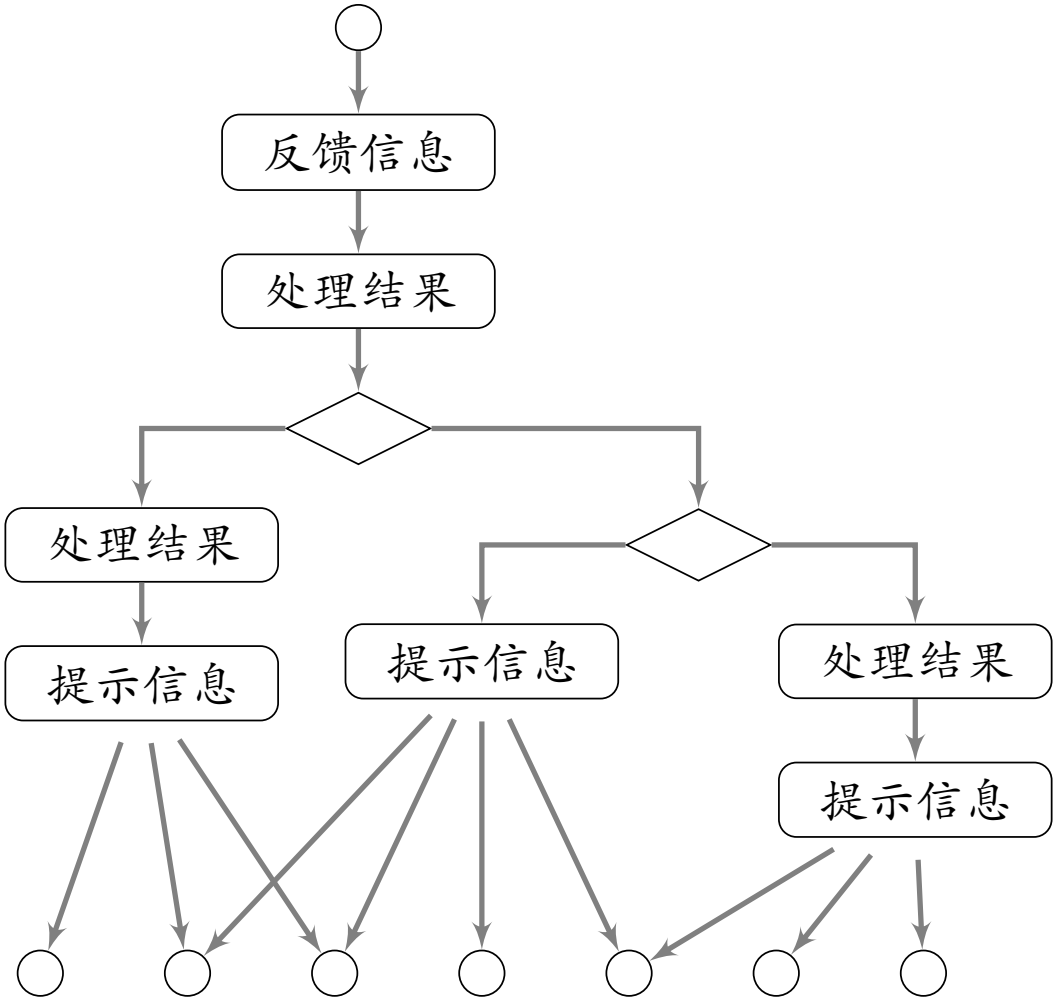
两层命令

输出信息的组织结构



典型的输出信息结构

输出信息的组织结构



复杂的输出信息结构

内容提要

人机界面的设计准则

- 使用简便
- 一致性
- 启发性
- 减少人脑记忆的负担
- 减少重复的输入
- 容错性
- 及时反馈
- 其它：艺术性、趣味性、风格、视感 ...

人机界面的 OO 设计

- 选择界面支持系统
- 根据人机交互需求选用界面元素
- 用 OO 概念表示界面元素

选择界面支持系统

选择界面支持系统考虑的因素：

- 硬件
- 操作系统及编程语言
- 开发工具
- 支持级别
- 风格与视感
- ...

根据人机交互需求选用界面元素

不同的界面支持系统提供不同的界面元素，常用的界面元素有：窗口、菜单、对话盒、图符、滚动条等，大多数界面元素使用事件驱动的交互方式

- 系统的启动
  - 选用实现主界面的界面元素，如框架窗口、对话框窗口
- 高层命令组织结构的实现
  - 通过界面元素的构造层次体现高层命令的组织结构
    例如：窗口– 菜单– 下级菜单 …

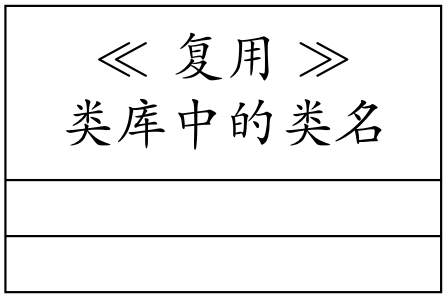
根据人机交互需求选用界面元素

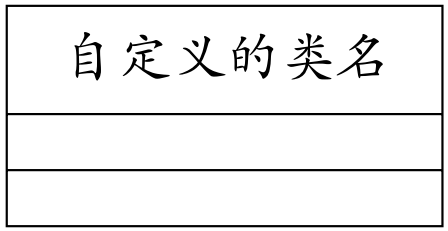
不同的界面支持系统提供不同的界面元素，常用的界面元素有：窗口、菜单、对话盒、图符、滚动条等，大多数界面元素使用事件驱动的交互方式

- 基本命令的执行
  - 通过高层命令引向基本命令
    例如：窗口—菜单—菜单选项
- 详细交互过程的输入与输出
  - 选择适当的界面元素完成每个命令步的输入与输出
- 异常命令的输入
  - 使用支持异常命令输入的界面功能，如鼠标右键菜单

用 OO 概念表示界面元素

对象和类：尽可能使用界面类库中提供的可复用类





用 OO 概念表示界面元素

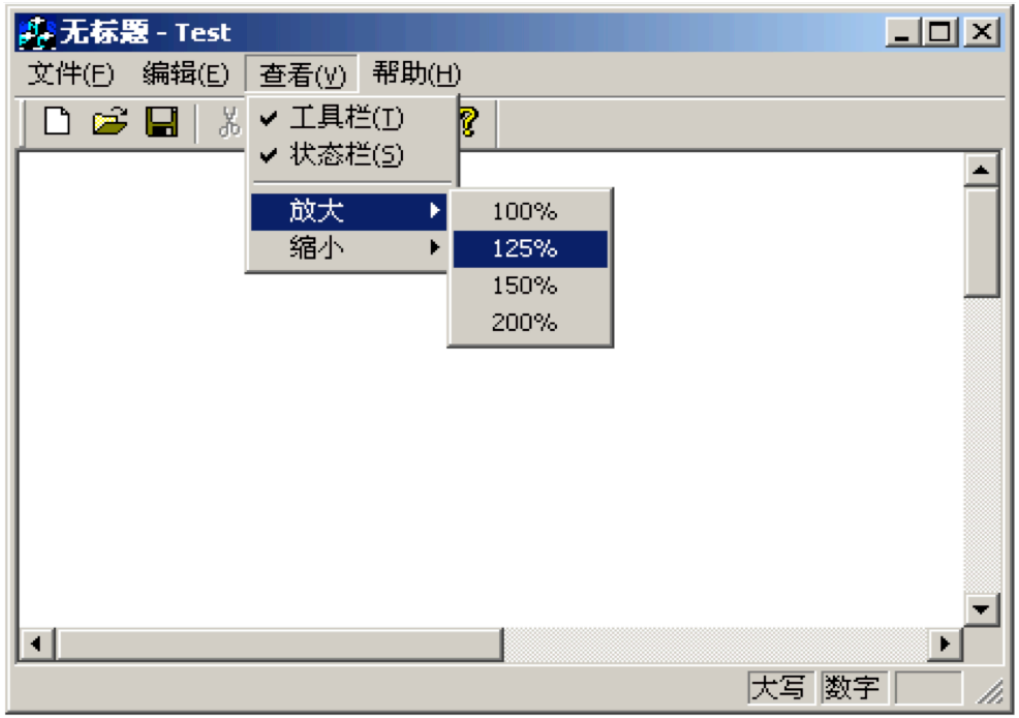
类的属性与操作：

- 用属性表示界面对象的静态特征
  - 物理特征——如：位置、尺寸、颜色、立体效果
  - 逻辑特征——聚合、关联
- 用操作表示界面对象的行为
  - 例如：创建、激活、最大化、最小化、移动、选中、单击、双击……

用 OO 概念表示界面元素

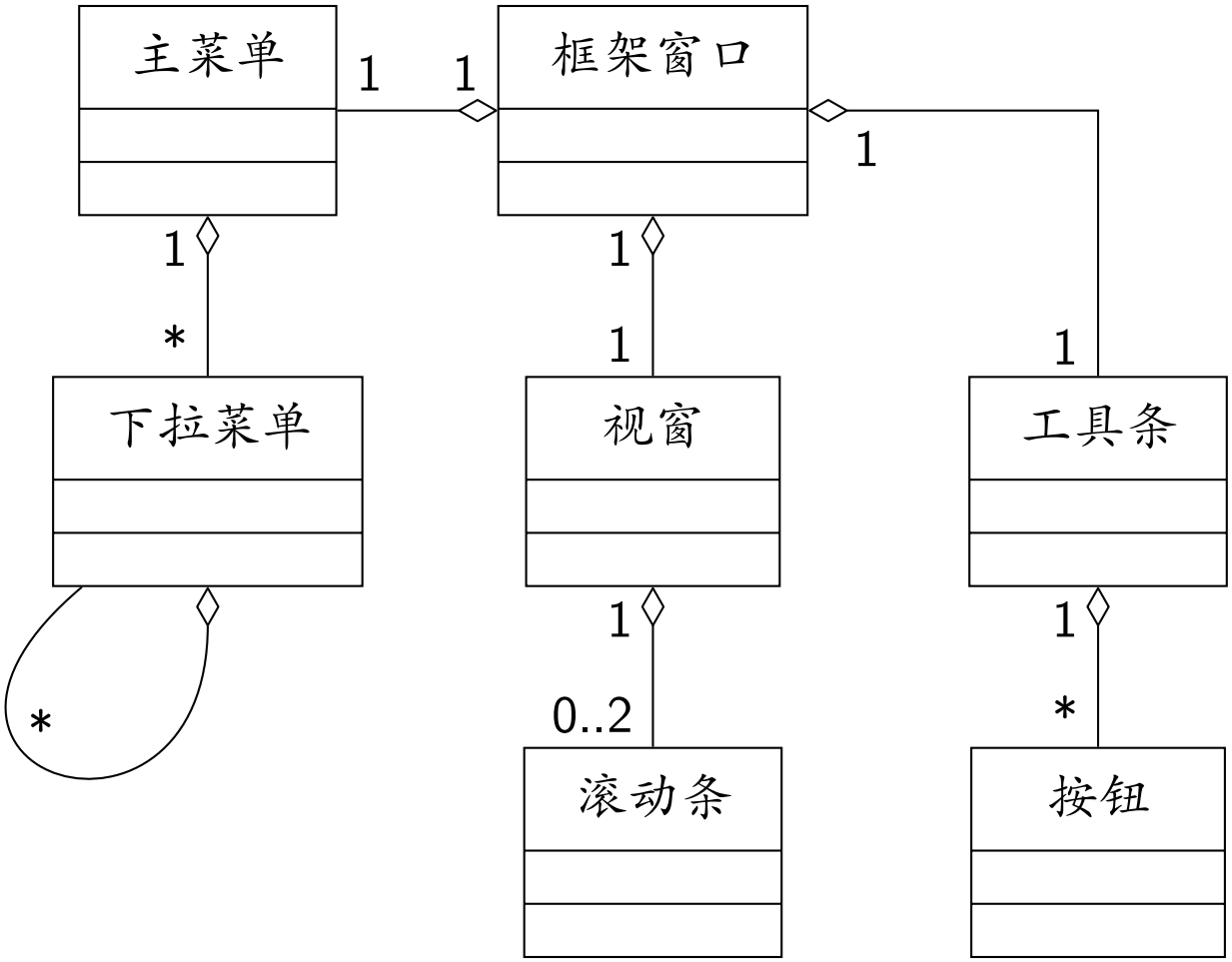
整体-部分结构：

- 表示界面元素之间的构成关系，例如：
  - 窗口与其中的菜单、按钮、图符、对话框、滚动条
- 表示界面对象在操作中的逻辑层次
  - 反映上、下两层命令之间的关系



用 OO 概念表示界面元素

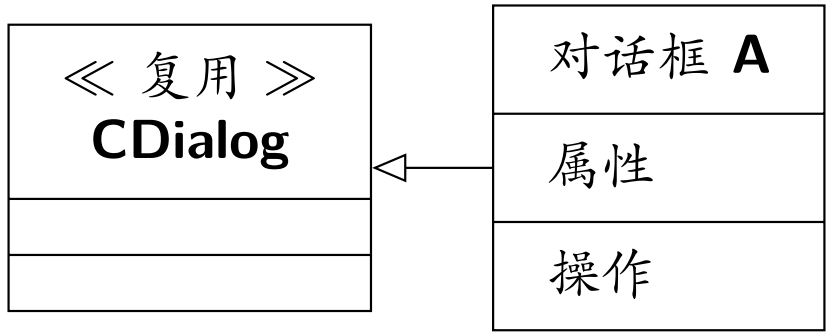
整体部分结构：



用 OO 概念表示界面元素

一般-特殊结构：表示较一般的界面类和较特殊的界面类之间的关系

- 自定义的类之间的一般-特殊关系
- 用一般-特殊结构特化可复用类



用 OO 概念表示界面元素

关联：表示界面类之间一个有特定意义的关系
例如：

用 OO 概念表示界面元素

消息:

- 高层命令到低层命令
  - 界面对象之间的消息
- 基本命令的执行
  - 从界面对象向功能对象发消息
- 信息输出
  - 从功能对象向界面对象发消息

内容提要

可视化编程环境下的人机界面设计

- 问题的提出
- 所见即所得的界面开发
- 设计的必要性
- 基于可视化编程环境的设计策略

设计的必要性

1. 为实现提供依据
   - 为了满足人机交互的需求，人机界面中要使用哪些界面对象?
   - 交互过程中的各项输入和输出应由哪些界面对象完成?
   - 如何通过界面对象类之间的各种关系体现人机交互命令的组织结构与层次?
   - 如何通过界面对象和功能对象之间的消息实现它们之间的动态联系?

设计的必要性

2. 降低失败的风险
3. 设计和实现分离，符合软件工程良好实践经验
4. 可视化编程环境下的设计策略有所不同
   - 类库的存在促进了复用
   - 以所见即所得的定义界面对象的各种物理属性方式更为直接

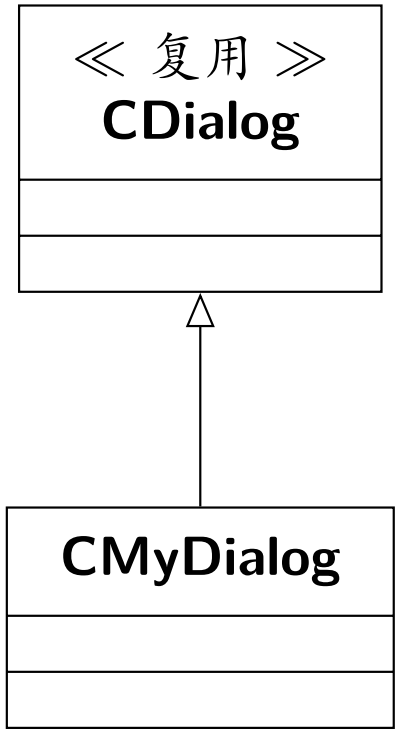
基于可视化编程环境的设计策略

1. 学习可视化编程环境及其类库
2. 根据人机交互需求选择界面元素
3. 根据可视化编程环境的特点，建立类图

建立类图—做什么，不做什么

类的设立—首先想到复用

直接复用



通过继承复用

建立类图—做什么，不做什么

属性—忽略物理特征，着重表示逻辑特征

设计阶段不必关心描述界面物理特征的属性，诸如：大小、形状、位置、颜色、边框、底纹、图案式样、三维效果等，由实现人员去自主处理效果更好，效率更高

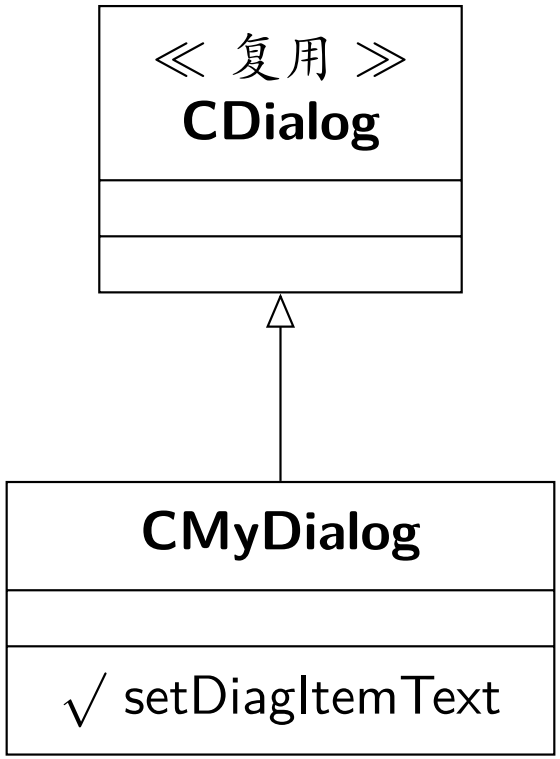
以主要精力定义描述界面逻辑特征的属性

- 表现命令的组织结构的属性
  - 例如：菜单类的每个选项表示什么命令
- 表现界面元素之间组成关系和关联的属性
  - 例如：对话框中包含哪些控件

建立类图—做什么，不做什么

操作—显式地表示从高层类继承的操作

例：



建立类图—做什么，不做什么

整体-部分结构—表现界面的组织结构和命令层次

与采用其它界面支持系统的策略相同，需区分界面对象的普通属性和它的部分对象

有些组成部分被作为对象的一个普通属性

例如：下拉菜单的选项，窗口的边框

有些组成部分则被作为一个部分对象

例如：对话框的一个下拉菜单或按钮

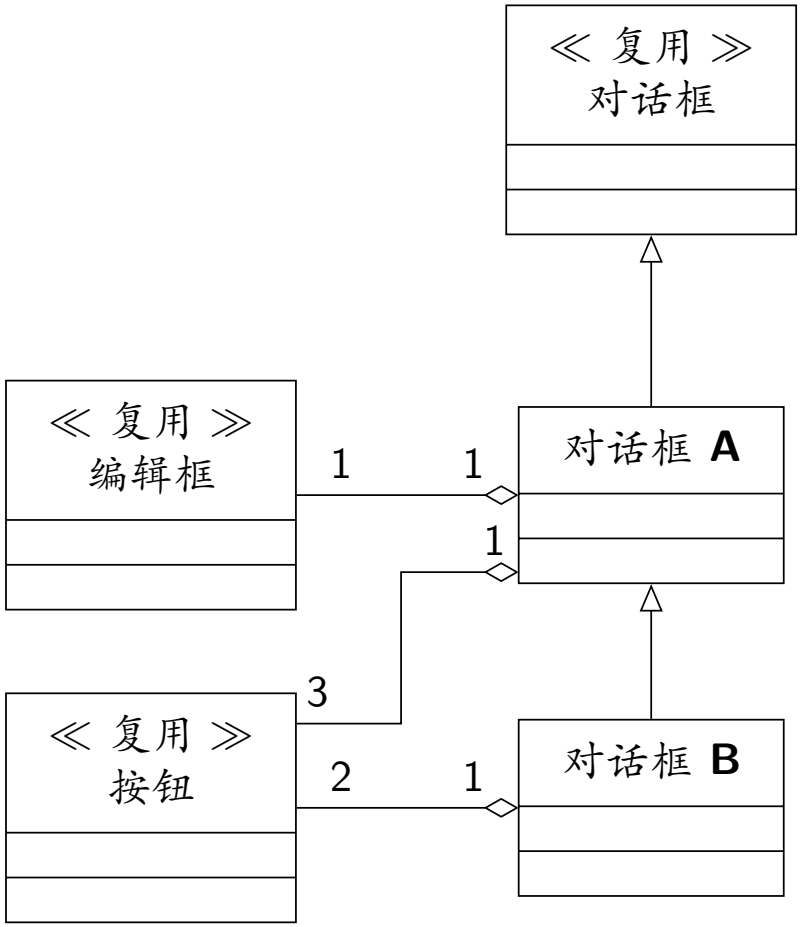
区分两种情况的依据

——环境类库有没有给出这种组成部分的类定义

建立类图—做什么，不做什么

一般-特殊结构—多从可复用类直接继承

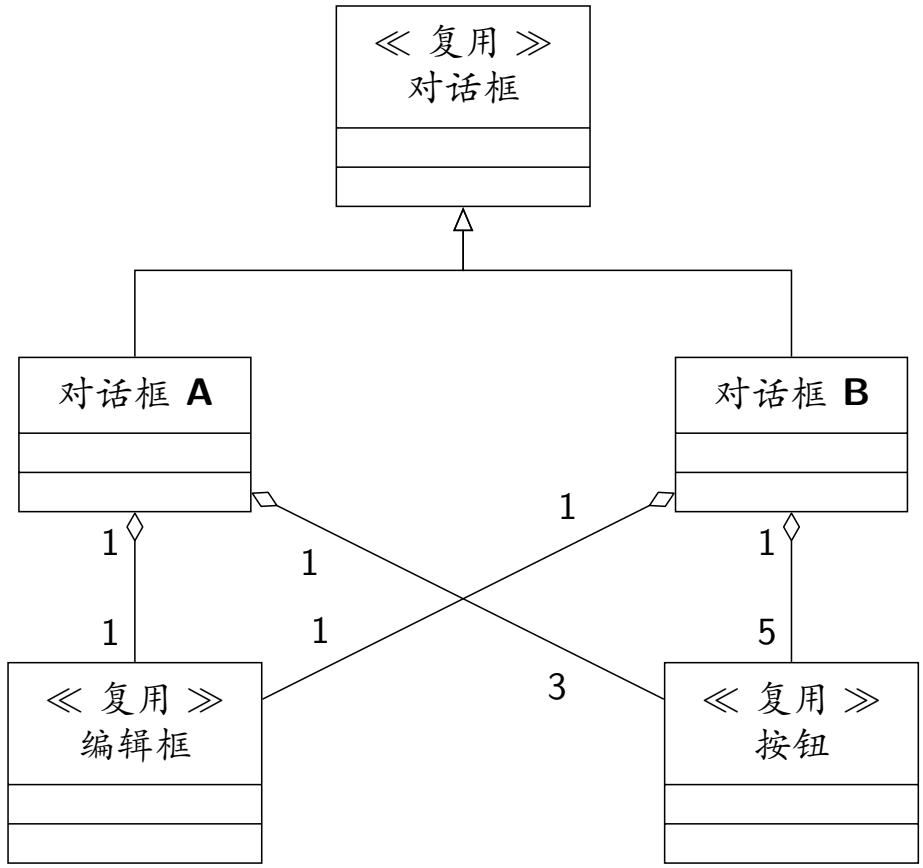
普通策略



建立类图—做什么，不做什么

一般-特殊结构—多从可复用类直接继承

直接继承可复用类的策略



建立类图—做什么，不做什么

消息—忽略自动实现的消息，注意需要编程实现的消息

- 界面对象接收到一个操作事件，通过它的一个操作向处理该事件的功能对象所发送的消息
- 从功能对象向完成其输入/输出的界面对象发送的消息
- 其它：凡是需要通过手工编程来实现的消息，都要在设计中加以表示