



# 面向对象的设计



北京大学



# 第一部分 什么是面向对象的设计（OOD）

概而言之，但不同时期有不同内容及特点。

## 一、早期的 OOD（八十年代至九十年代初）：

**历史：从 OOP 发展到 OOD**

**G. Booch** 1982 年发表 “Object-Oriented Design”，首次称“面向对象的设计”。

**G. Booch** 1986 年发表 “Object-Oriented Development” 较完整地阐述了 OOD 思想。

两个术语都用 OOD 作为缩写，内容上也没有根本区别

**R. J. Abbott** 1983 年提出正文分析方法，用规范的英语对问题进行陈述，然后从描述中提取对象及其特征。例：名词——对象，动词——操作。被后来的许多 OOD 方法所采用。

1986 年后，相继出现了一批（早期的）OOD 方法



北京大学



## 早期的 OOD 方法：

**Booch86——Object-Oriented Development**  
面向对象的开发

**GOOD——General Object-Oriented Development**  
通用面向对象的开发

**HOOD——Hierarchical Object-Oriented Design**  
层次式面向对象的设计

**OOSD——Object-Oriented Structured Design**  
面向对象的结构设计

.....



北京大学



## 早期 OOD 的特点：

- 1、不是基于 OOA 的  
大多基于结构化分析结果（数据流图）
- 2、是 OO 编程方法的延伸  
多数方法与编程语言有关，特别受 Ada 影响很大
- 3、不是纯 OO 的  
对某些 OO 概念（如继承）缺少支持，  
掺杂一些非 OO 概念（如数据流、包、模块等）
- 4、不是只针对软件生命周期的设计阶段  
OOD 中的“D”——指的是 Design 或 Development  
多少涉及分析问题（如识别问题域的对象），但很不彻底

早期的  
OOD 可  
看作现今  
OOA&D  
方法的雏  
形





## 二、现今（90年代）的 OOD

### 背景：

从结构化分析文档识别 OOD 的对象并非良策，识别对象的关键问题在于用 OO 方法进行系统分析。

OO 方法从设计发展到分析，出现 OOA 方法。

OOA 和 OOD 构成完整的 OOA&D 方法体系。

OOD 基于 OOA，  
识别对象由 OOA 完成，  
OOD 主要定义对象如何实现。

### 有多种 OOA&D 方法：

Booch 方法  
Coad-Yourdon 方法  
Firesmith 方法  
Jacobson 方法(OOSE)  
Martin-Odell 方法  
Rumbaugh 方法(OMT)  
Wirfs-Brock 方法

.....



北京大学



## 特点：

1. 以面向对象的分析为基础，一般不依赖结构化分析。
2. 与相应的 OOA 方法共同构成一种 OOA&D 方法体系。OOA 和 OOD 采用一致的概念与原则，但属于软件生命周期的不同阶段，有不同的目标及策略。
3. 较全面地体现面向对象方法的概念与原则。
4. 大多数方法独立于编程语言，通过面向对象的分析与设计所得到的系统模型可以由不同的编程语言实现。

## 定义：

面向对象的设计（OOD）就是在 OOA 模型基础上运用面向对象方法进行系统设计，目标是产生一个符合具体实现条件的 OOD 模型。



北京大学





- 在面向对象软件开发中运用的较为普遍的开发过程有：统一软件开发过程 (Unified Software Development Process, USDP)、基于特定活动组织的开发过程和面向特征的开发过程等。
- 这些开发过程都分为 OOA、OOD 和 OOP 等阶段。本课程所讲述的开发过程，是一种基于特定活动组织的开发过程。





### 三、OOD 的根本目标：(coad/yourdon)

#### 1、提高软件生产率

设计的投入在编程、测试时得到回报

OO 方法使系统更易于理解

分析文档、设计文档、源程序对应良好

功能变化引起的全局性修改较少

OOD 结果的复用

#### 2、提高质量

现今的质量观点：

- 不仅是事后通过测试排除错误，而是着眼于软件开发过程的每个环节，从分析、设计阶段开始质量保证。

- 高质量不只是没有错误，而是好用、易用、可移植、易维护，用户由衷地满意。



北京大学





### 3、加强可维护性

需求是不断变化的（尽管可阶段性地“冻结”）

因素：客户业务、竞争形式、技术发展、规章制度……  
——要求设计结果对变化有弹性

设计如何适应不可预见的变化？

——把易变部分和较稳定的部分隔离，  
将变化的影响限制在局部

易变性：服务 > 接口 > 属性 > 类



北京大学



## 四、OOA 与 OOD 的关系 第二种观点的理由：

### 1、OOA 与 OOD 的分工 —— 两种不同的观点

第二种观点

		第二种观点	
		分析 问题域与 系统责任	设计 与实现有 观的因素
第一种观点	分析 做什么		
	设计 怎么做		

(1) 在各种分析 / 设计方法中“做什么”和“怎么做”实际上没有严格的划分”。

(2) 过分强调“分析不考虑怎么做”将使某些必须在 OOA 考虑的问题得不到完整的认识。

(3) 由于 OO 方法表示形式的一致，不存在把细化工作留给设计人员的必然理由。

(4) 避免重复地认识同一事物，并有利于分析结果的复用。

例：Rumbaugh 方法 (OMT)  
和 Coad/ Yourdon 方法

关键问题：对象的特征细节  
(如属性的数据类型和操作流程  
图)，是在分析时定义还是在  
设计时定义？



北京大学



## 第二部分

## OOD 方法概貌

基本按 Coad/Yourdon 方法讲授，做适当的改进和补充。

**概念：**运用与OOA部分相同的概念  
——没有增加新概念

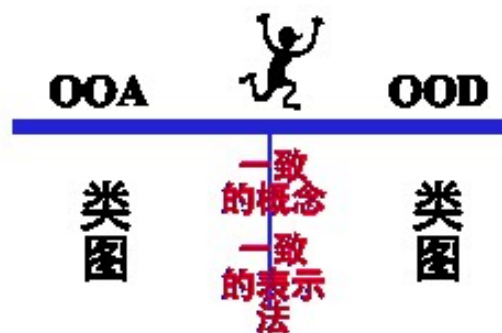
对象、类、属性、服务(操作)  
、封装、继承、消息、关联、  
聚合、多态、主动对象 等

**表示法：**采用与OOA一致的表示法

使分  
析与  
设计  
之间  
不存  
在鸿沟



传统方法分析与  
设计之间的鸿沟



面向对象的分析与设计  
之间不存在鸿沟



北京大学



**OOD——按实现条件对 OOA 模型进行调整，并补充几个新的组成部分（也是由对象构成）**

**与实现有关的因素：**

图形用户界面系统

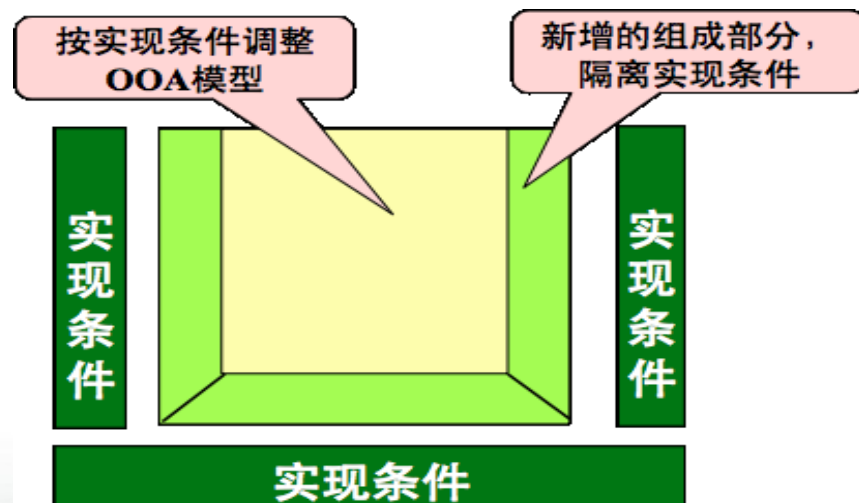
硬件、操作系统及网络

数据管理系统

其他——编程语言、可复用构件库……

**基本思想：**

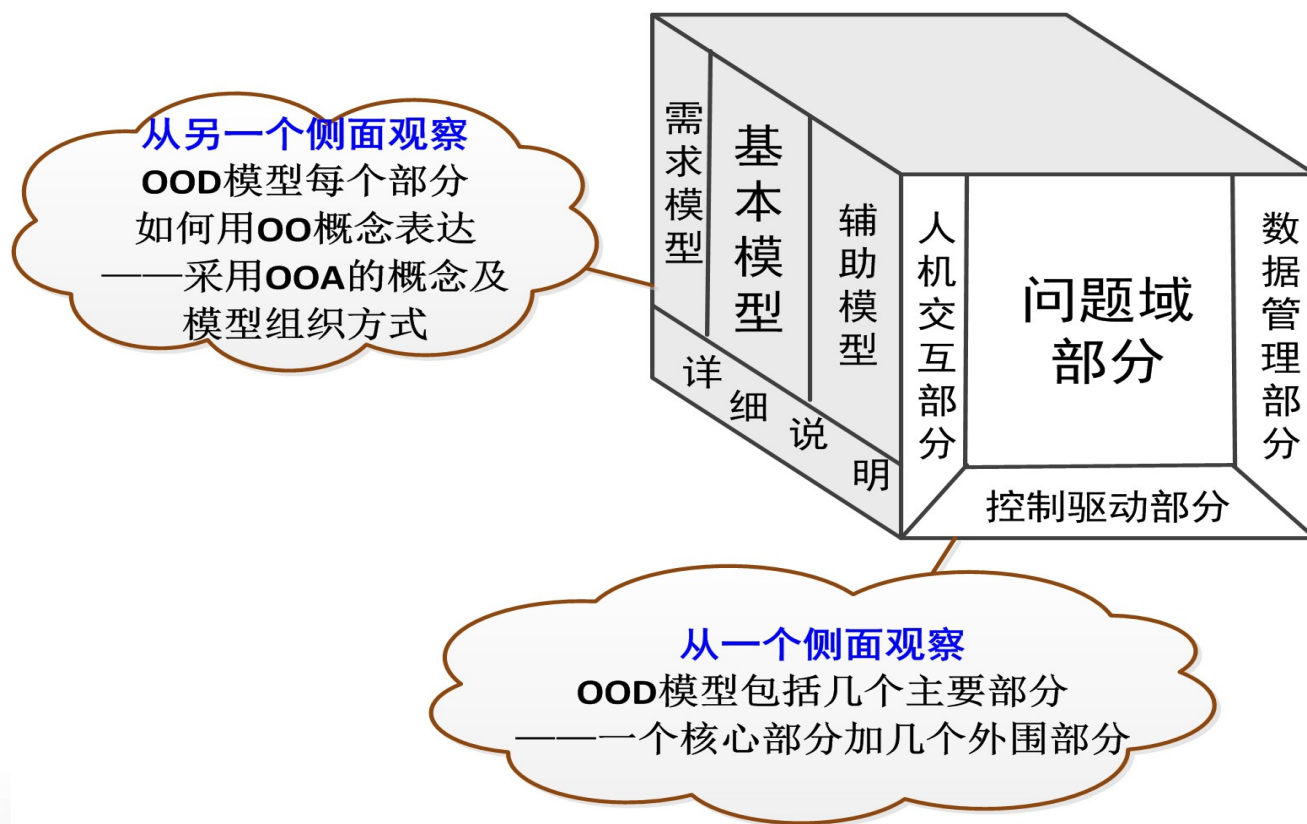
尽可能隔离实现条件对系统的影响——提供独立的接口  
对不可隔离的因素，按实现条件调整 OOA 模型



北京大学

# OOD 模型

## ——从两个侧面来描述



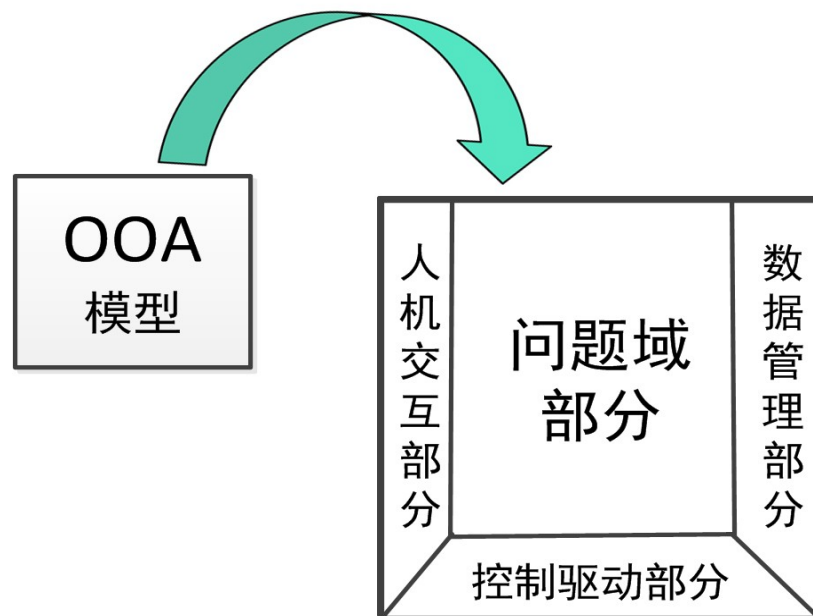




## OOA 与 OOD 的关系：

1、从 OOA 到 OOD 不是转换，不是细化；  
——是调整和增补

将 OOA 模型搬到  
OOD；  
进行必要的调整，  
作为 OOD 模型的问题  
域部分，  
增补其它三个部分  
，  
成为完整的 OOD 模型







## 2、采用一致的概念和表示法 ——不存在分析与设计之间的鸿沟

## 3、有不同的侧重点和不同的策略

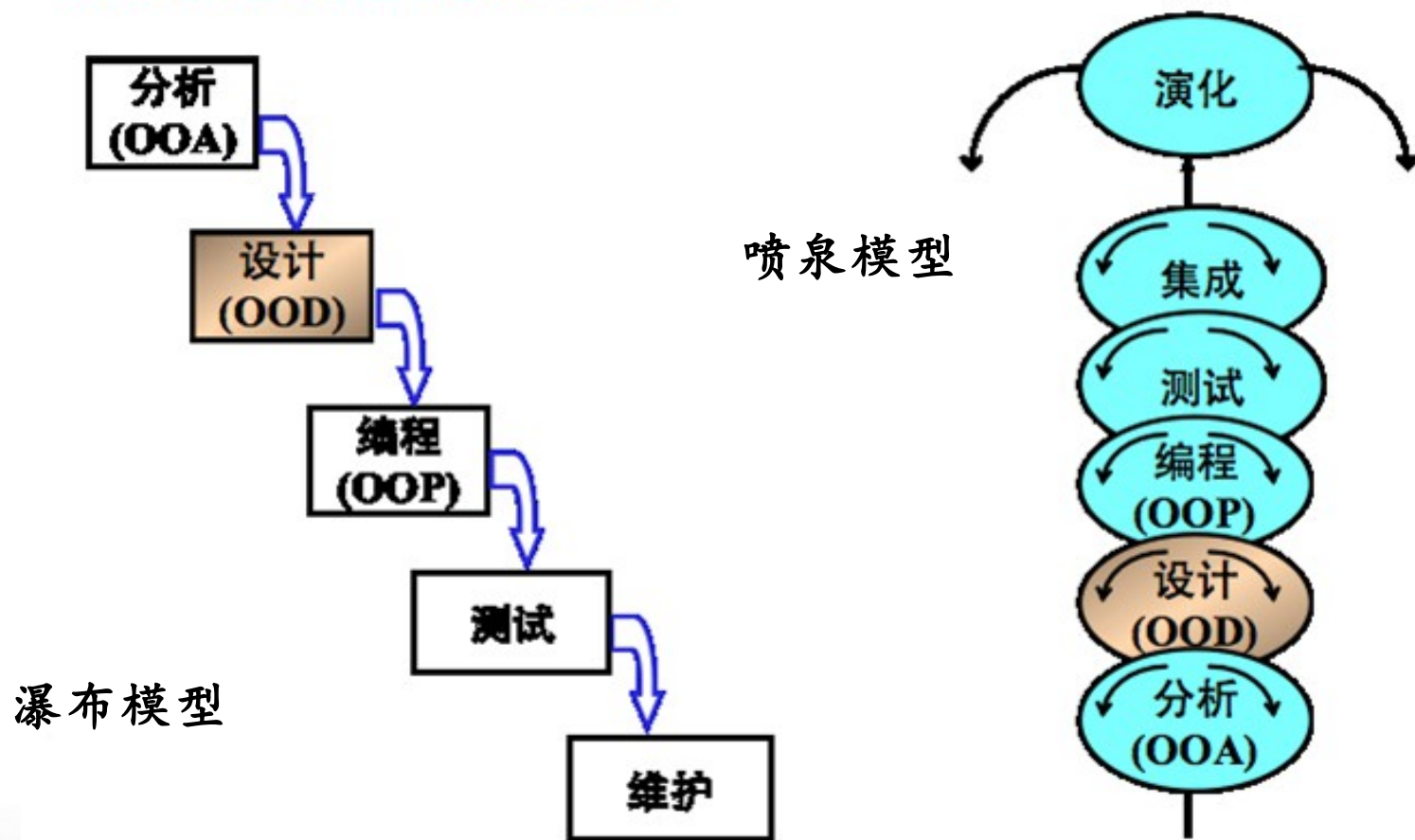
OOA 主要针对问题域，识别有关的对象以及它们之间的关系，产生一个映射问题域，满足用户需求，独立于实现的 OOA 模型。

OOD 主要解决与实现有关的问题，基于 OOA 模型，针对具体的软、硬件条件（如机器、网络、OS、GUI、DBMS 等）产生一个可实现的 OOD 模型。

## 4、OOA 与 OOD 可适合不同的生命周期模型 ——瀑布模型、螺旋模型、增量模型、喷泉模型 OOA 与 OOD 之间可以顺序进行，也可交叉进行



## 不同过程模型中的OOD





**OOD 过程：**

**逐个设计 OOD 模型的四个部分**

问题域部分的设计

人机交互部分的设计

控制驱动部分的设计

数据接口部分的设计

**不强调次序**

每个部分均采用与 OOA 一致的概念、表示法及活动，  
但具有自己独特的策略。



北京大学

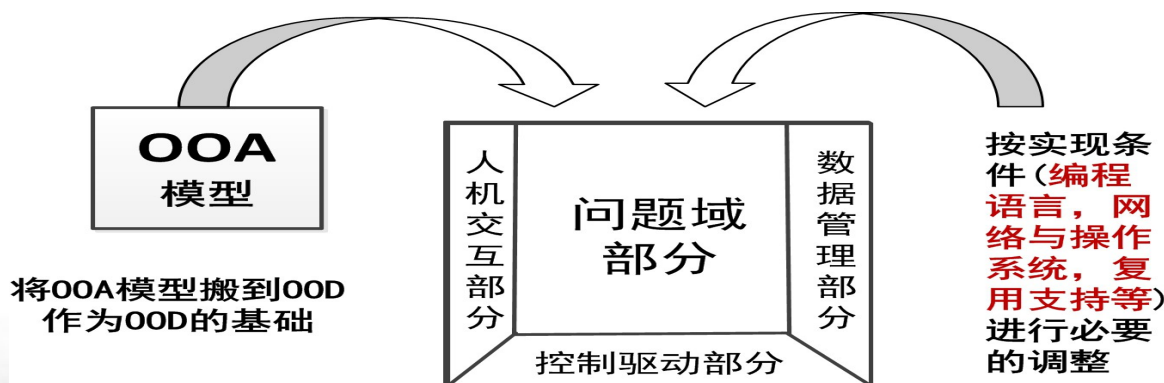
# 第三部分问题域部分（PDC）的设计

## 一、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，是由问题域有关的对象构成，并且在特定的实现条件下提供用户所需功能的组成部分。它是在 OOA 模型基础上按实现条件进行必要的修改、调整和细节补充而得到的。

不是传统方法的“转换”，不存在鸿沟。主要不是细化，但 OOA 未完成的细节定义要在 OOD 完成。

**是对 OOA 模型的补充与调整。**





并非所有的实现因素都能通过一些在 OOD 中新定义的独立组成部分而实行有效的隔离。有些实现因素将不可避免地影响到 OOA 阶段识别的对象，影响到它们的内部特征和相互关系，因而要求在 OOD 阶段按照这些条件对它们做必要的修改、调整和细节上的补充。这正是问题域部分的设计所要解决的问题。

- 1、编程语言
- 2、硬件、操作系统及网络设施
- 3、复用支持
- 4、数据管理系统
- 5、界面支持系统







## 二、为什么需要问题域部分的设计？

- ◆在 OOA 阶段只考虑问题域和系统责任，OOD 则要考虑与具体实现有关的问题，需要对 OOA 结果的补充与调整；
- ◆使反映问题域本质的总体框架和组织结构长期稳定，而细节可变；
- ◆稳定部分（PDC）与可变部分（其它部分）分开，使系统从容地适应变化；
- ◆有利于同一个分析用于不同的设计与实现；
- ◆支持系统族和相似系统的分析设计及编程结果复用；
- ◆使一个成功的系统具有超出其生存期的可扩展性







### 三、实现条件对问题域部分的影响

现在分析和讨论各种实现条件将对 OOD 模型产生什么影响：

#### 1、编程语言：

用于实现的编程语言对问题域部分的影响最大，其中包括两方面问题：

(1) 选定的编程语言可能不支持某些面向对象的概念与原则。此时要根据编程语言的实际表达能力对模型进行调整，以保证设计模型与源程序一致。

(2) OOA 阶段可能将某些与编程语言有关的对象细节推迟到 OOD 阶段来定义。如对象的创建、删除、复制、转存、初始化等系统行为、属性的数据类型等。

编程语言确定之后，这些问题都要给出完整的解决。





## 2、硬件、操作系统及网络设施

选用的计算机、操作系统及网络设施对 OOD 的影响包括：对象在不同的站点上的分布、主动对象的设计、通信控制以及性能改进措施等。这些问题对问题域部分和控制驱动部分都将有影响。

## 3、复用支持

如果存在已经进行设计和编码的可复用类构件，用以代替 OOA 模型中新定义的类无疑将提高设计与编码效率。但这需要对模型做适当的调整与修改。

## 4、数据管理系统

选用的数据管理系统主要影响 OOD 模型的数据管理部分的设计，但也需要对问题域部分的某些类补充访问该数据管理部分所要求的属性与操作。





## 5、界面支持系统

指支持用户界面开发的软件系统。主要影响人机交互部分的设计，对问题域部分影响很少，只是两部分之间需要互传消息而已。





## 四、如何进行问题域部分的设计？

### 1、继续运用 OOA 的方法

—— 概念、表示法及策略

### 2、使用 OOA 结果，并加以修改

—— 需求的变化，新发现的错误

### 3、使用 OOA 结果，并进行补充与调整（本节的重点）

（1）为复用设计与编程的类而增加结构

（2）增加一般类以建立共同协议

（3）按编程语言调整继承

（4）提高性能

（5）为实现对象永久存储所做的修改

（6）为编程方便增加底层细节



北京大学



## (1) 为复用设计与编程的类而增加结构

----OOA 识别和定义的类是本次开发中新定义的，需要进行编程。

---- 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，复用这些类即可提高开发效率与质量。

---- 可复用的类可能只是与 OOA 模型中的类相似，而不是完全相同，因此需对二者进行修改。

目标：尽可能使复用成分增多，新开发的成分减少



北京大学



## 不同程度的复用

定义的 信息	可复用类	比	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
				≈	删除多余信息，通过继承而复用

### 对第四种情况的做法：

- (1) 把要复用的类加到问题域。
- (2) 在类名后加{复用}，划掉不用的属性与操作。
- (3) 建立从复用类到问题域原有的类之间的泛化关系
- (4) 由于问题域的类继承了复用类的特征，所以有些属性和操作不需要了——划掉。
- (5) 修改问题域原有类的结构和关联，必要时移到复用类。





例：

可复用的类

车辆{ 复用 }

序号

~~厂商~~

式样

序号认证

问题域中的类

车辆

~~序号~~

颜色

~~式样~~

出厂年月

~~序号认证~~

可复用的类

车辆

序号

厂商

式样

序号认证



北京大学



## (2) 增加一般类以建立共同协议

- 增加一般类：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
- 增加一般类：提供局部通用的协议  
例：提供永久存储及恢复功能

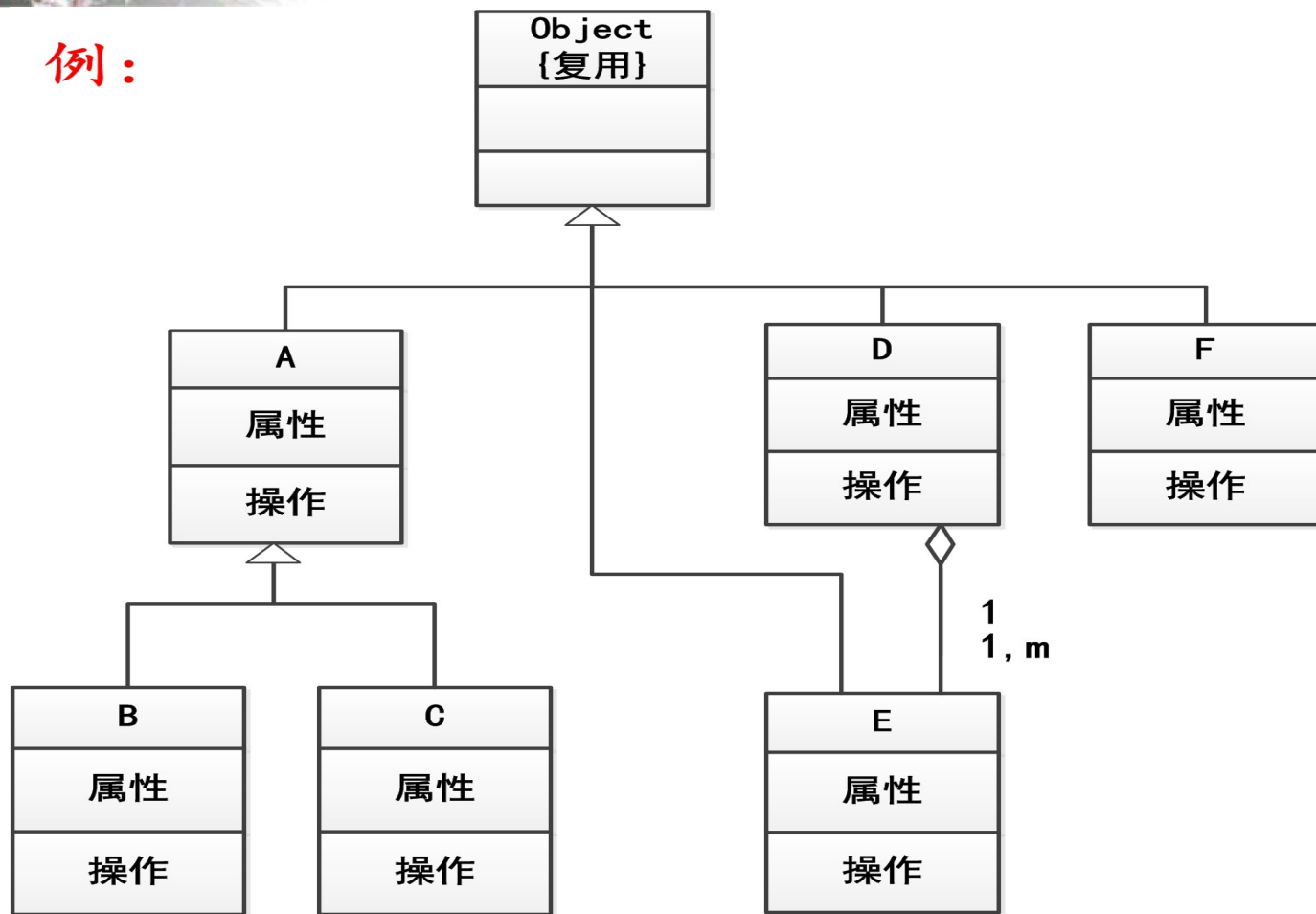
### 注意：

- 1、如果新增加的类是自己定义的，则表示法和其它类一样；
- 2、如果新增加的类是编程语言提供的预定义类，则只在类符号的名字栏填写一个和语言提供的类完全相同的类名，并标上{复用}的字样，属性和操作栏不必填写任何属性和操作；
- 3、在类描述模版的“类整体说明”部分的“其它”项中用文字加以说明，比如“利用编程语言提供的同名类”。





例：



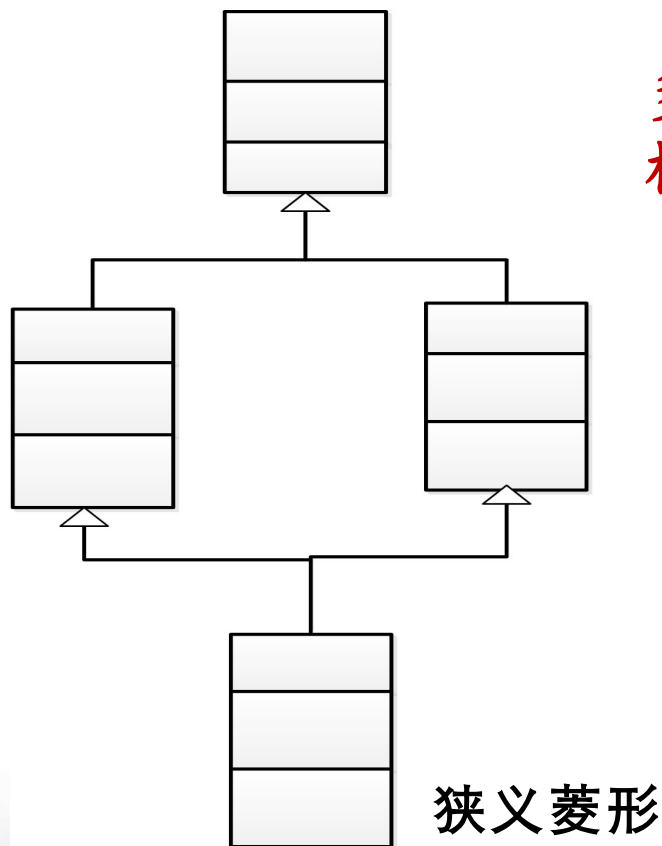
上图表示“Object”是模型中新增的一般类，其它类都是模型中原有的类。这样的表示法表明，“object”是由语言提供的可复用类，实现时不需对它进行任何编程就可让A、D、E、F等类直接引用它作为一般类（B和C则通过A间接地继承。）



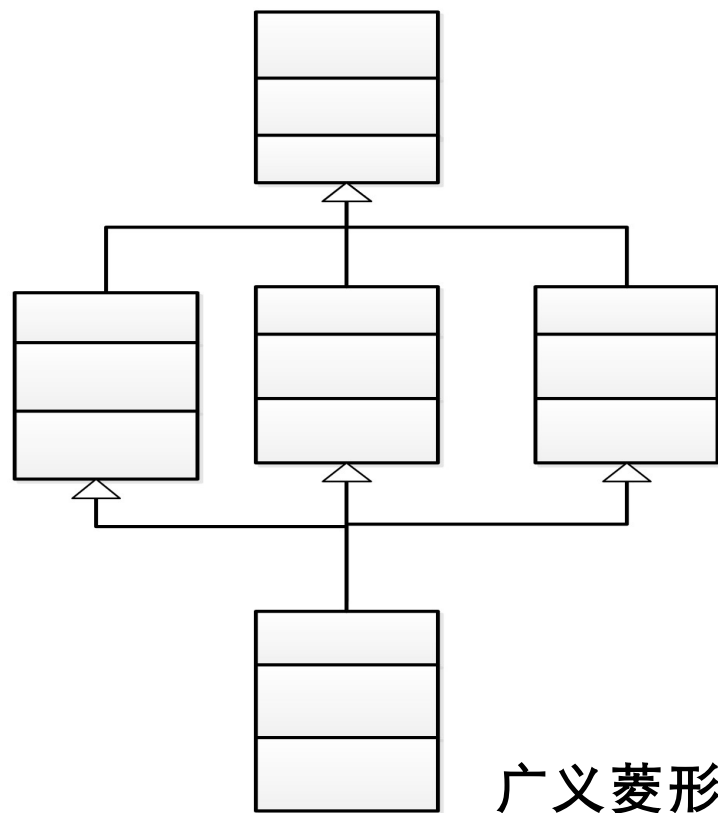
北京大学

### (3) 按编程语言调整继承和多态

起因：OOA 强调如实地反映问题域，OOD 考虑实现问题，所用语言不支持多继承和多态



多继承  
模式

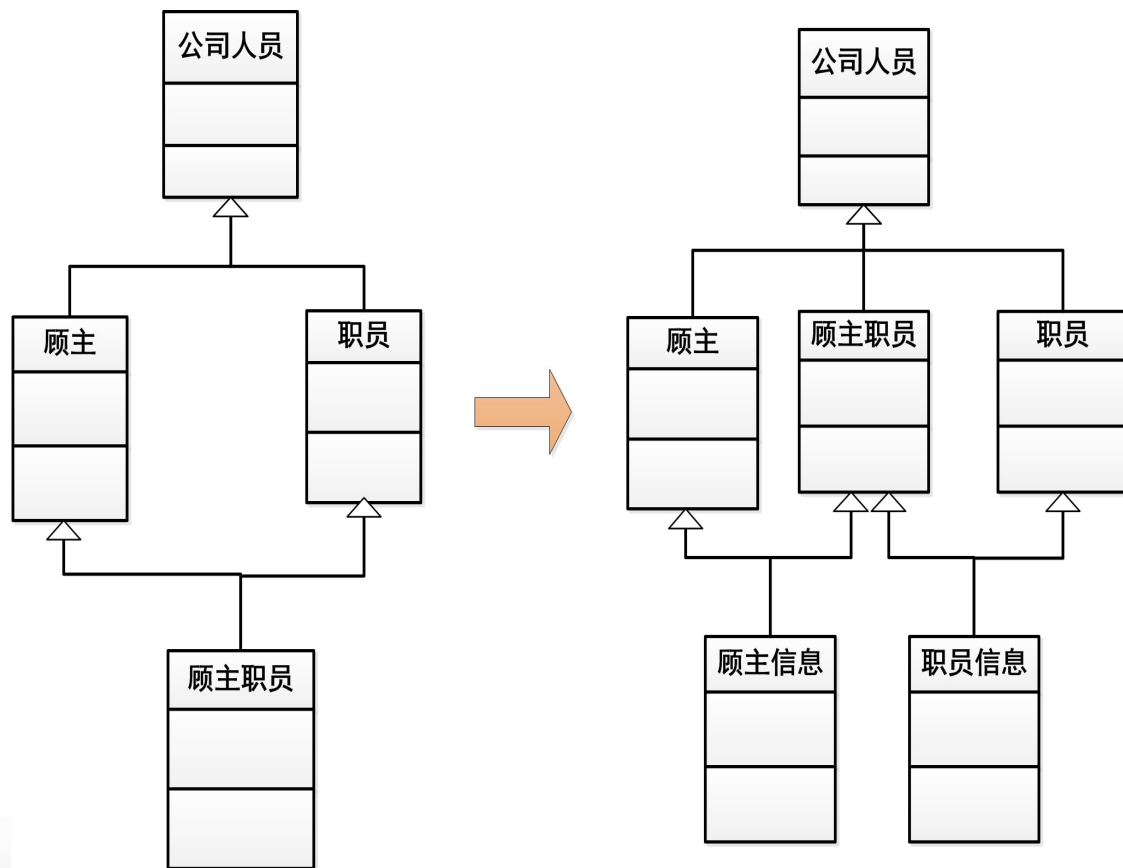


北京大学



## ① 对继承的调整

方法 1：采用聚合，将多继承转换为单继承



- 保持原来多继承结构中的每个类
- 把形成多继承的每一组特殊信息从有关的类中剥离出来，定义为部分对象类
- 再通过聚合使各个特殊类能拥有这些信息

问题：道理何在？



北京大学



一般－特殊结构和整体－部分结构在某些情况下可以互相变通的道理如下：

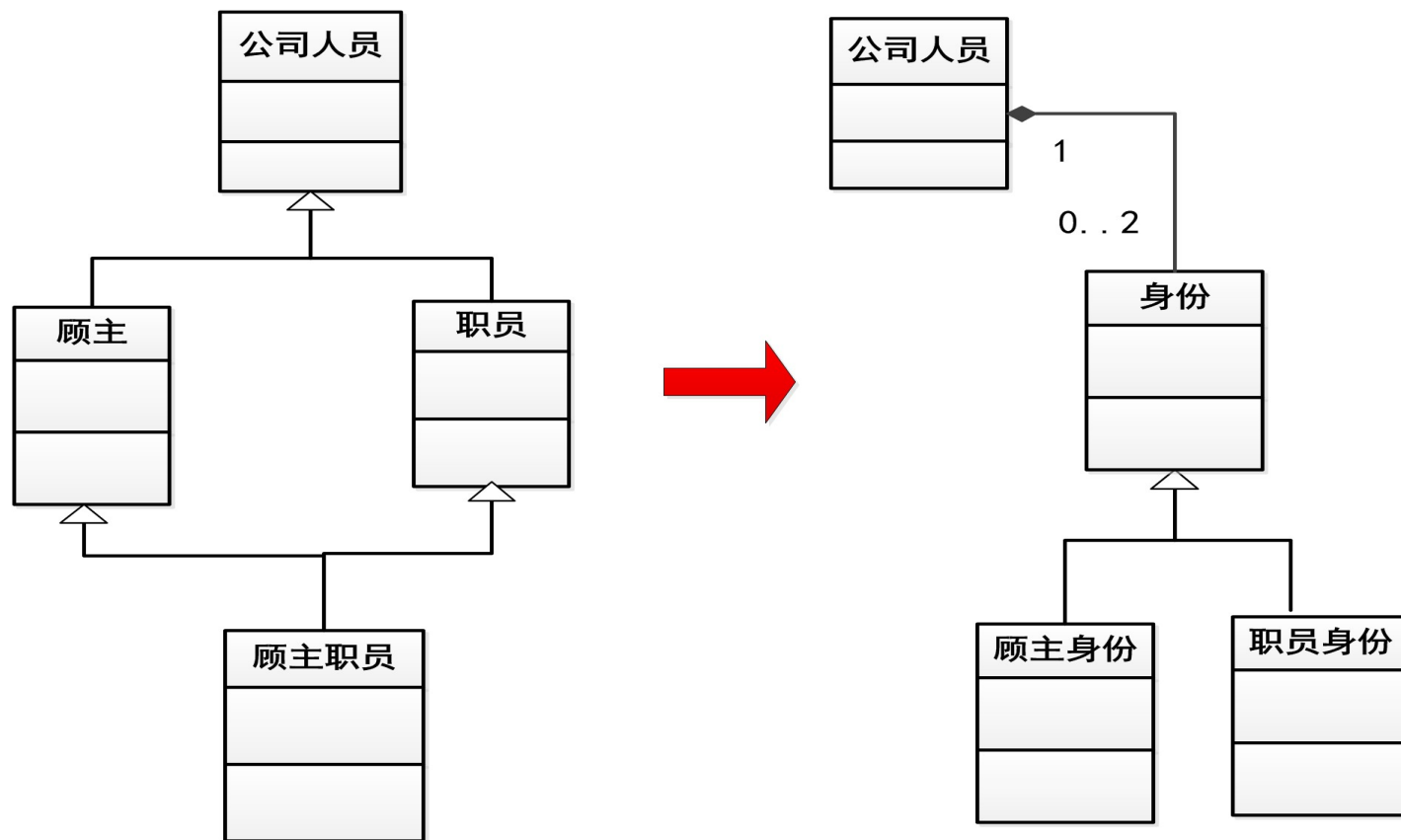
尽管继承和聚合反映了现实世界中两种不同的关系，**但是从最终效果来看却存在共性－都是使一个类的对象能够拥有另一个（一些）类的属性和操作。**



北京大学



## 方法 2：重新定义对象类



问题：与方法 1 有何区别？



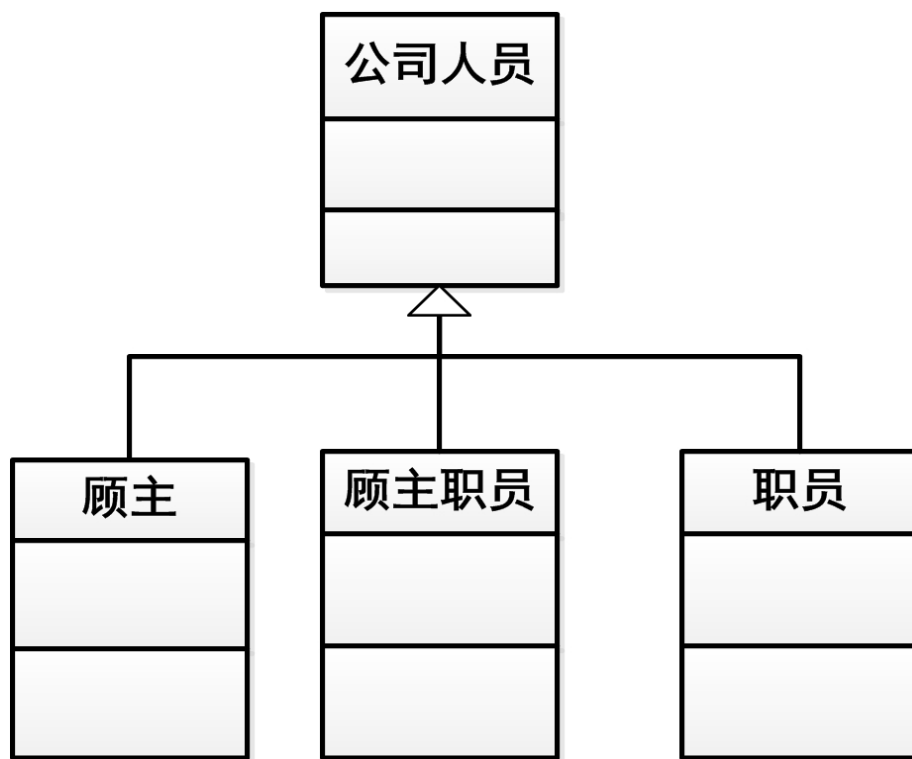
北京大学



❖这种方法不是简单地将多继承结构中的某两个类之间的继承关系替换为聚合关系，而是用一种新的思路，一种不运用多继承概念的思路，**重新审视原来用多继承结构表达的实际事物及它们之间的关系。例如，上述例子换一个角度看问题：形成这种分类的原因是什么？从而增加“身份”类，构成单继承。**



### 方法 3 : 压平



问题：有什么缺点？





## ② 取消继承

若编程语言不支持继承，有两种方法应对。

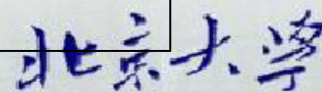
方法一：把继承结构展平

顾主

顾主职员

职员

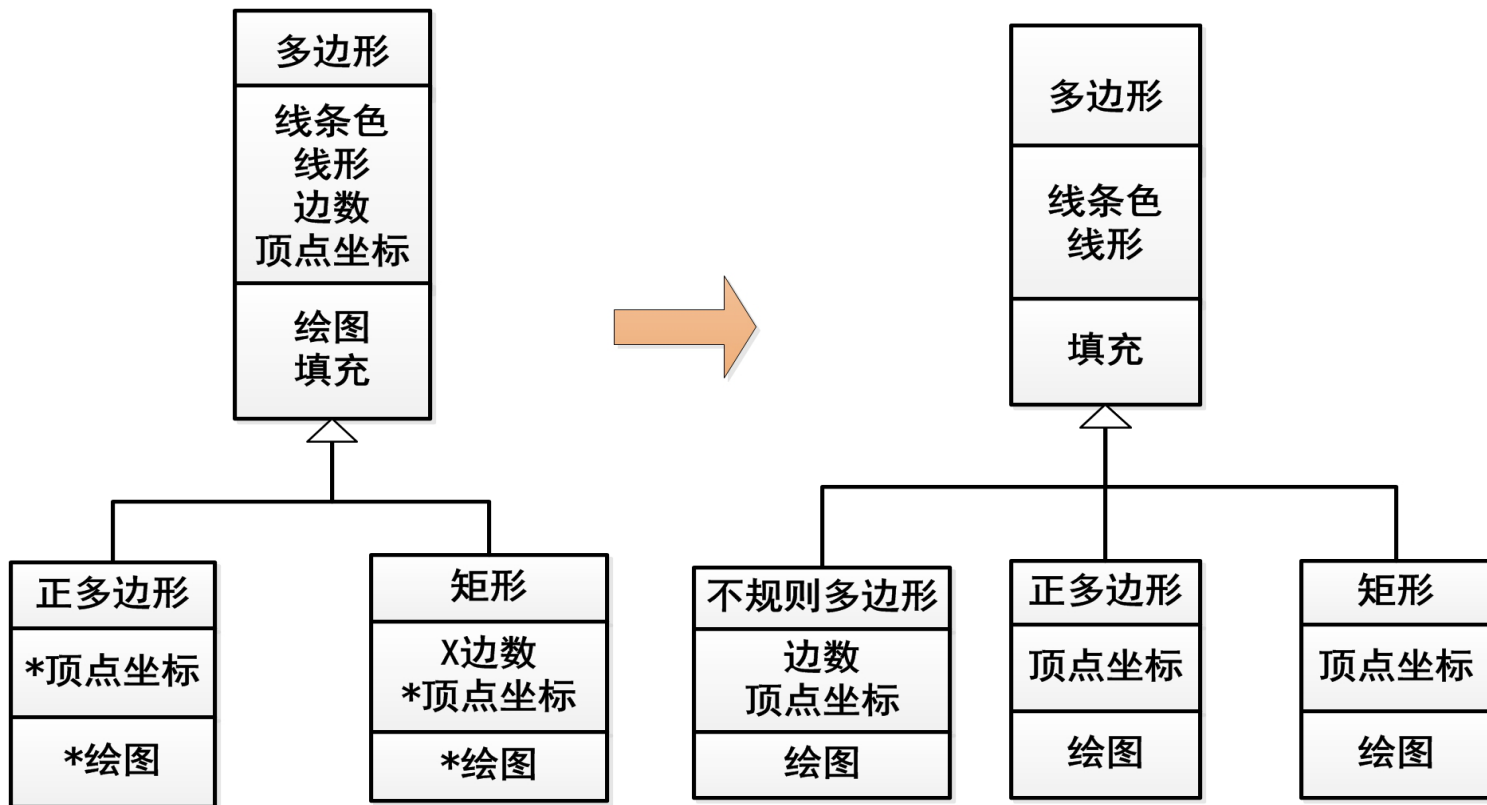








### ③ 对多态性的调整



多态：在继承结构中，具有相同的属性和操作，在不同的类中可以具有不同的类型和行为。



北京大学



## 注意：多态和重载的区别：

重载是指相同的操作名在同一个类中可以被定义多次，按参数的个数、种类或次序等的不同对它们进行区分。

多态是指在继承结构中，具有相同的属性和操作，在不同的类中可以具有不同的类型和行为。





## (4) 提高性能

### a、影响性能的因素：

**数据传输时间**：主要是在网络环境下不同处理机之间为完成某项系统功能进行必要的数据传输所专用的时间。影响数据传输时间的主要因素有：系统分布方案、网络拓扑结构、从发送者到接收点经由的每一段线路的传送速率和数据拥挤程度。

**数据存取时间**：是指完成一项系统功能时，在外部存储设备上读取或保存数据所用的时间。影响因素有：存储设备的物理性能（各种设备都有明确的性能指标说明）、访问外存的次数和每次访问的数据量。

**数据处理时间**：指计算机主机为完成一项功能，进行处理、计算所花费的时间。



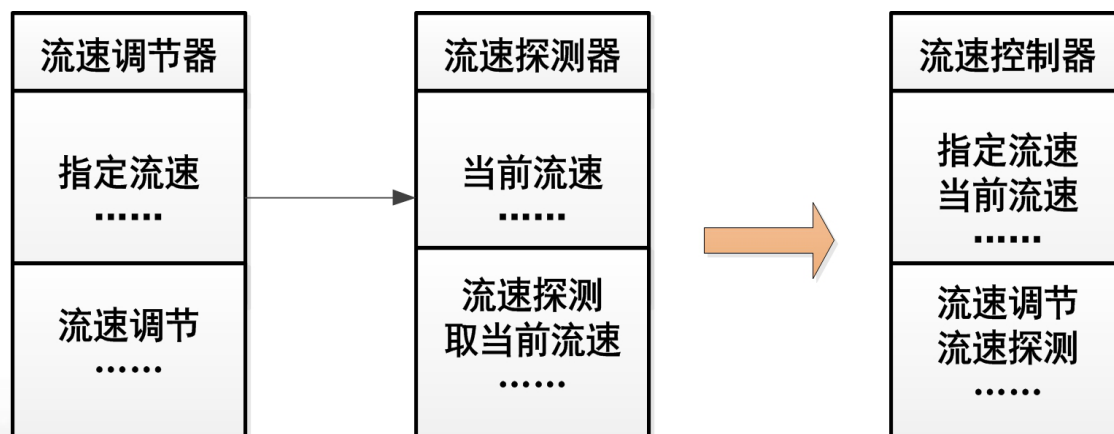


## b、改进性能的设计策略

设计人员为改进性能所做的设计决策，就是对 OOD 模型采取某些提高性能的策略，只有这些策略仍不能满足性能要求时，才考虑改变计算机、网络、DBMS 等基本配置。

\* 调整对象分布：不同处理机间的数据传输成为性能瓶颈

\* 合并通讯频繁的种类



合并前

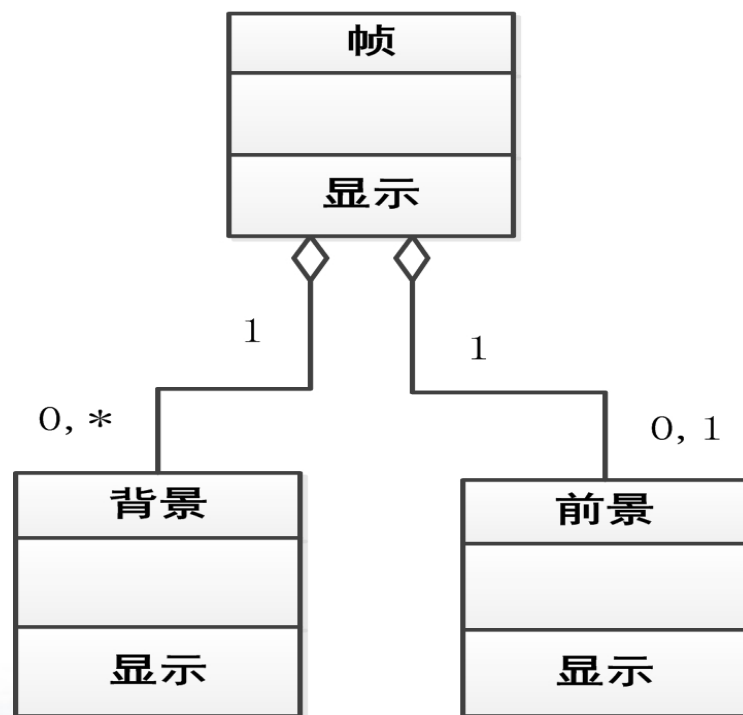
合并后



北京大学



- \* 增加属性以减少重复计算
- \* 降低算法的计算复杂性
- \* 用聚合关系描述复杂类







## (5) 为实现对象永久存储所做的修改

有些类的对象实例需要被永久存储。如果选用文件系统或关系数据库管理系统实现对象的永久存储，则需要对这些类做某些修改，如为实现对象的存储和恢复操作而增加属性和操作。这些修改在数据管理部分设计中介绍。

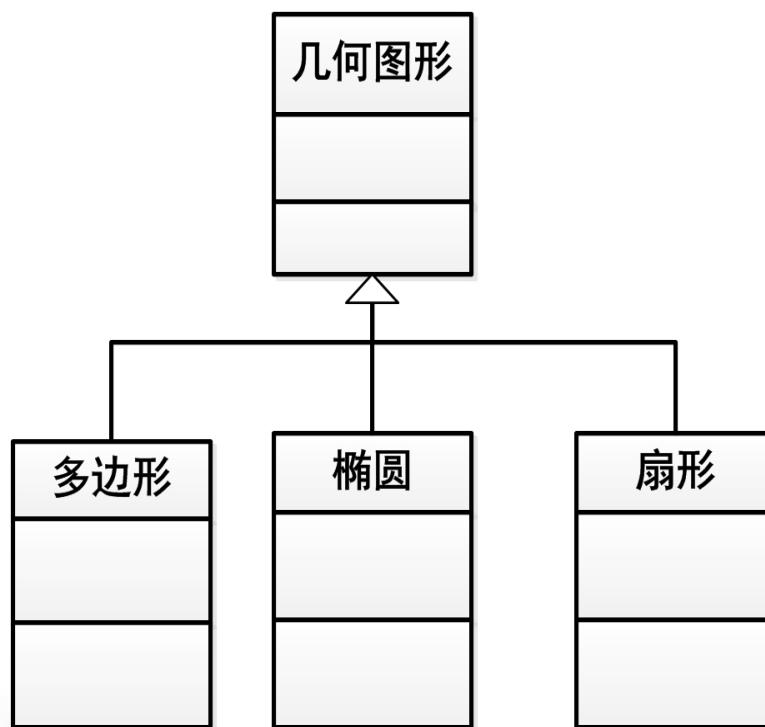





## (6) 为编程方便增加底层成分

### —— 细化对象的分类

例：将几何图形分成多边形、椭圆、扇形等特殊类





# 第四部分 人机交互部分的设计

## 一、什么是人机交互部分

系统中负责人机交互的部分（由一些对象类构成），突出人如何命令系统以及系统如何向用户提交信息。

OOA 和 OOD 都要考虑人机交互，但目的不同

OOA：通过人机界面反映需求（原型开发）

OOD：设计人机交互的细节

人机交互部分既取决于需求，又与 GUI 密切相关

人机交互部分的设计可以与 OOA 同时进行，但要互相分离

—— 为了隔离实现条件的影响





## 二、为什么需要人机交互部分

为了隔离 GUI 的变化对问题域部分的影响

人机交互部分是系统中一个比较独立的部分

集中表现：

人如何向系统下达命令  
系统如何向人提交信息



北京大学



### 三、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。

#### 1、分析与系统交互的人——人员活动者

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。因此要针对界面使用者的具体情况做具体的分析。

(1) 列举所有的人员活动者

(2) 调查研究：

人机界面的开发者要向使用者进行调查，了解他们的基本情况、具体要求、习惯及爱好。



北京大学



### (3) 区分人员类型

不同类型的人对人机界面有不同的要求、期望和爱好。可以从以下不同的角度对使用者进行分类：

**熟练程度：**初级、中级和高级。

**职业：**像技术人员、管理人员、行政人员、办事员、教师和学生等不同类型的人，使用计算机的方式、习惯、频繁程度、依赖程度等均有所不同。

**与系统的关系：**可分为系统管理员、维护人员、操作员、超级用户、一般用户以及用户业务的客户等。

**年龄：**老、中、青、少、幼不同年龄段的人对界面风格的爱好不太相同。

### (4) 统计（或估算）各类人员的比例

无论按以上哪种观点对人进行分类，使用界面的人都可能不止一类。这样就需要通过统计或估算给出各类人员的比例，以便在设计时重点考虑比例最大的人员情况，并适当地兼顾其他人。



北京大学





## (5) 了解使用者的主观需求

对使用界面的人进行分类之后，查阅有关人机工程、人机交互、人机界面等方面的研究资料，以了解本系统所面对的人员类型对交互方式、界面风格等方面的一些共同的要求和爱好。

## 2、从 use case 分析人机交互

人机交互包括两个方面：一是人对系统的输入，包括向系统下达的命令、提供的命令参数和系统所需的其他输入信息；另一方面是系统向人提供信息，即输出。

关于交互内容的需求是客观的，主要是由系统的功能需求决定的，与人的主观意识没有太大关系，但交互过程和交互方式则可以根据人的主观因素做不同的决策。





## 1) 从 use case 抽取人机交互内容及过程

### 抽取方法：

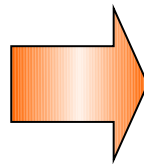
删除所有与输入、输出无关的语句和不再包含任何内容的控制语句与括号，  
剩下的就是对一个活动者（人）使用一项系统功能时的人机交互描述。





收款员收款 ( use case )  
输入开始本次收款的命令;  
作好收款准备, 应收款总数  
置为 0 , 输出提示信息;  
for 顾客选购的每种商品 do  
    输入商品编号;  
    if 此种商品多于一件 then  
        输入商品数量  
    end if ;  
    检索商品名称及单价;  
    货架商品数减去售出数;  
    if 货架商品数低于下限 then  
        通知供货员请求上货  
    end if ;  
计算本种商品总价并打印编号、  
名称、数量、单价、总价;  
总价累加到应收款总数;  
end for ;  
打印应收款总数;  
输入顾客交来的款数;  
计算应找回的款数,  
打印以上两个数目,  
收款数计入账册。

( a ) 一个 use case 的例子



收款员收款 (人机交互)  
输入开始本次收款的命令;  
    输出提示信息;  
for 顾客选购的每种商品  
    输入商品编号;  
if 此种商品多于一件 then  
    输入商品数量  
end if ;  
打印商品编号、名称、  
数量、单价、总价;  
end for ;  
打印应收款总数  
输入顾客交来的款数  
打印交款数及找回款数;

( b ) 人机交互描述

从 use case 提取人机交互描述



北京大学



## 2) 人机交互的细化

从 use case 抽取的人机交互只是定义了使用一项系统功能时的基本交互内容与步骤。它只是反映了人机交互的客观需求，而没有反映人的主观需求，还要针对系统使用者的特点进行细化。

人机交互的细化包括对交互过程中每一次输入和每一次输出的细化。细化所考虑的主要问题是如何将基本交互过程（即从 use case 抽取的、未经细化的交互过程）中的每一项输入和输出组织得更加符合人的习惯与爱好。

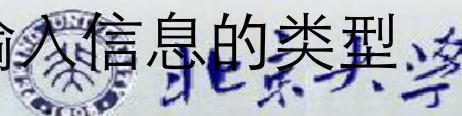
### （1）输入的细化

#### ① 输入步骤的细化

向系统输入同样一条信息，既可以一次输入完毕，也可以分若干细小的步骤完成，对此要进行权衡。

#### ② 输入设备的选择

输入设备的选择在很大程度上决定于输入信息的类型，也在一定程度上受人的因素影响。





### ③ 输入信息表现形式的选择

输入可分为两类：一类是人对系统的命令；另一类是向系统提供的数据。后一类输入要原本地输送给系统，这里只考虑前一类输入——人对系统的命令。

命令表示形式和输入方式的选择主要考虑以下因素：

- 适合使用者的特点。
- 以文字方式表达的命令要求所使用的词汇能够较准确地反映命令的语义。以图形、符号等形式表达的命令要求形象、直观，使人容易联想到它们的寓意。
- 与流行的、大家已经习惯的命令表示形式和操作方式相符。

### (2) 输出的细化

人机交互过程中的每一项输出都是机器向人提供的必要信息。输出可分为三类：

**第一类：是提示信息**，是根据输入的要求设置的，旨在告诉用户应进行何种输入以及如何输入；



北京大学





**第二类是系统向人报告的计算或处理结果；**

**第三类是系统对输入操作的反馈信息，**表示系统已接收到用户的输入，仅用于对该项输入的预计处理时间较长的情况。一、三两类输出一般都比较简单，这里主要讨论第二类输出的细化。

### **① 输出步骤的细化**

输出信息量如果不是很大，一般可以一次输出完毕。大量的输出可以分若干步骤给出。

**② 输出设备的选择：**显示器、打印机和绘图仪。

**③ 输出信息表现形式的选择：**有文本、表格、图形、图象、声音、视频片段等多种形式的输出形式。采用何种形式的输出主要决定于系统的功能需求，也在一定程度上决定于使用者的特点。



北京大学





### 3、命令的组织

#### 命令的组织措施——分解与组合

- (1) 分解：将一条复杂的命令分解成一系列较为简单的命令。
- (2) 组合：当命令很多时，将它们按功能或所属的子系统组合成若干命令组。

**基本命令：**使用一项独立的系统功能的命令。一个从 use case 提取的人机交互过程是针对一项系统功能的，基本命令正是开始其交互过程并使用该项系统功能的命令。

**命令步：**在执行一条基本命令的交互过程中所包含的具体输入步骤。从 use case 提取的交互过程中的各项输入都是这样的命令步。这里不再区分一项输入是对系统的指示，还是命令参数或其他输入信息，总之都是基本命令的一个步骤。

**高层命令：**如果一条命令是在另一条命令的引导下被选用的，则后者称作前者的高层命令。高层命令相当于一个索引目录，其作用是将若干低层命令组织在一起，并在人机交互中引导用户选用在它之下的低层命令。





### 三、人机界面的设计准则

#### ➤使用简便

#### ➤一致性：

界面的各个部分及各个层次，在术语、风格、交互方式、操作步骤等方面尽可能保持一致。此外，要使自己设计的界面与当前的潮流一致。

#### ➤启发性：

能够启发和引导用户正确、有效地进行界面操作。

#### ➤减少人脑记忆的负担

使人在与系统交互时不必记忆大量的操作规则和对话信息。

#### ➤减少重复的输入

记录用户曾经输入过的信息，特别是那些较长的字符串；当另一时间和场合需要用户提供同样的信息时，能自动地或通过简单的操作复用以往的输入信息，而不必人工重新输入。



北京大学



## ➤容错性

对用户的误操作有容忍能力或补救措施。包括：对可能引起不良后果的操作给出警告信息或请求再次确认；提出撤消和恢复功能，使系统方便地回到以往的某个状态，或重新进入新的状态；对无意义的操作最好不予理睬。

## ➤及时反馈

对那些需要较长的系统执行时间才能完成的用户命令，不要等系统执行完毕时才给出反馈信息。系统应及时地给出反馈信息，说明工作正在进展。

## ➤防止灾难性的错误

系统要采取保证措施，防止由于用户误操作或因为其他原因造成数据丢失这样的错误。例如，对可能引起不良后果的操作（如删除、不存盘退出）给出警告，对重要的操作要能撤消（undo），对数据要定时进行备份。

## ➤其它：如艺术性、趣味性、风格、视感等。



北京大学



## 四、人机界面的 OO 设计

人机界面的设计，一般是以一种选定的界面支持系统为基础，利用它所支持的界面构成成分，设计一个可满足人机交互需求、适合使用者特点的人机界面设计模型。在 OOD 中要以面向对象的概念和表示法表示界面的构成成分以及它们之间的关系。

### 1、选择界面支持系统

**(1) 窗口系统：**是控制位映像显示器与输入设备的系统软件，它所管理的资源有屏幕、窗口、色彩表、字体、光标、图形资源及输入设备。

●**窗口系统的特点：**屏幕上可显示重叠的多个窗口，采用鼠标器确定光标位置和各种操作，屏幕上用弹出式或下拉式菜单、对话框、滚动框、图标等交互机制供用户直接操作。





- **窗口系统既是一种开发平台，也是一种运行平台**
  - 对开发者而言，它提供了支持应用系统开发的支撑机制、库函数、应用程序接口、工具箱和供开发者使用的人机交互界面；
  - 对应用系统的用户而言，它提供了系统运行的环境，包括对应用系统用户界面的显示和操作的支持。
- **在窗口系统的发展史上知名的窗口系统**
  - PARC 开发的 Smalltalk，它将语言、类库、窗口系统结合为一体，既是第一个较完善的 OOPL，也是第一个窗口系统；
  - Apple 公司的 Macintosh，是第一个流行较广，影响较大的商品化窗口系统；
  - 麻省理工学院和 DEC 公司开发的 X Windows，是迄今在工作站上使用最广泛的窗口系统；
  - 微软公司的 MS-Windows，是目前在微机上使用最广泛的窗口系统。



北京大学





(2) **图形用户界面 (GUI)**：把一种在窗口系统之上提供层次更高的界面支持功能，具有特定的视感和风格，支持应用系统用户界面开发的系统。

- 典型的窗口系统（如 X Window）一般不为用户界面规定某种特定的视感及风格，而在它之上开发的 GUI（例如在 X Window 之上开发的 OSF/Motif 和 Openlook）则通常要规定各自的界面视感与风格，并为应用系统的界面开发提供比一般窗口系统层次更高、功能更强的支持。
- 窗口系统和图形用户界面这两个概念迄今尚未形成统一、严格的定义。

(3) **可视化编程环境**：将窗口系统、GUI 和可视化开发工具、编程语言和类库结合为一体。

- 例如在 C++ 基础上发展起来的 Visual C++，在 Object Pascal 基础上发展起来的 Delphi 和在 Basic 基础上发展起来的 Visual Basic.



北京大学





## 2. 根据人机交互需求选择界面元素

如窗口、菜单、对话框、图符、滚动条、控制板、剪辑板、光标、按钮等。

人机界面的开发是用选定的界面支持系统所能支持的界面元素来构造系统的人机界面。在设计阶段，要根据人机交互的需求分析，选择可满足交互需求的界面元素，并策划如何用这些元素构成人机界面。**对 OOD 而言，需要用面向对象的概念和表示法来表示这些界面元素以及它们之间的关系。在实现阶段用选定的界面支持系统所提供的功能来完成上述设计。**





### 3、用 OO 概念与表示法表达所有的界面成分

1) 以窗口作为基本的类

2) 以窗口的部件作为窗口的部分对象类，  
与窗口类形成聚合关系

例：菜单，工作区，对话框……

3) 发现窗口与部件的共性，  
定义较一般的窗口类和部件类，形成继承关系

4) 用属性表示窗口或部件的静态特征

如：尺寸、位置、颜色、选项等

特别要注意表示界面对象关联和聚合关系的属性

5) 用操作表示窗口或部件的动态特征

如：选中、移动、滚屏等

6) 发现界面类之间的关系，建立关联





## 7) 建立界面类与问题域类之间的联系

- 人机界面只负责输入、输出及窗口更新这样的工作，并把所有面向问题域的请求转发给问题域部分，即在界面对象中不应该对业务领域进行处理。
- 在多数情况下，问题域部分的对象不应主动发起与界面部分对象之间的通信，而只能对界面部分对象进行响应，即只有界面部分的对象才能访问问题域部分的对象。
- 尽量减少界面部分与问题域部分的耦合。界面是易变的，从易于维护和易于复用的角度出发，问题域部分和界面部分应该是低耦合的。





# 第五部分 控制驱动部分的设计

## 一、什么是控制驱动部分

控制驱动部分——

是 **OOD 模型的组成部分之一**，该部分由系统中全部**主动类构成**。这些主动类描述了整个系统中所有的主动对象，每个主动对象是系统中的一个控制流的驱动者。

控制流是一个在处理机上顺序执行的动作序列。在目前的实现技术中，一个控制流就是一个进程或线程。

在顺序程序中，只有一个控制流，并发程序则含有多个控制流。每个控制流开始执行的源头，是一个主动对象的主动操作。





## 二、为什么需要控制驱动部分

并发行为是现实中固有的

当前大量的系统都是并发系统（多任务系统），例如：

- 设备与计算机并发工作的系统
- 有多个窗口进行人机交互的系统
- 多用户系统
- 多个子系统并发工作的系统
- 单处理机上的多任务系统
- 多处理机系统

.....

### 多任务的设置

- 描述问题域固有的并发行为
- 表达实现所需的设计决策

为了隔离硬件、操作系统、网络的变化对整个系统的影响



北京大学



## 三、如何设计控制驱动部分

### 1、识别控制流

#### 1) 以节点为单位识别控制流

不同节点上的程序之间的并发，以及同一节点上的程序间的并发。

#### 2) 从用户需求出发识别控制流

#### 3) 从 use case 认识控制流

OOA 阶段定义的每一个 use case 都描述了一项独立的系统功能。从需求角度，它描述了一项系统功能的业务处理流程；从系统构造来看，它很可能暗示需要通过一个控制流来实现其业务处理流程。

#### 4) 参照 OO 模型中的主动对象

主动对象的一个主动操作是一个控制流的源头。







## 5) 为改善性能而增设的控制流

**高优先级任务：** 把对时间要求较高的工作从其他工作中分离出来，作为独立的任务，用专门设计的控制流去实现。

**低优先级任务：** 可把这些工作分离出来，作为独立的任务，用专门设计的控制流去实现，在执行时赋予较低的优先级，使它们称为通常所说的后台进程。

**紧急任务：** 把这些工作作为单独的任务，用专门的控制流去实现。它的执行不允许其他任何任务干扰。

## 6) 实现并行计算的控制流

## 7) 实现节点间通信的控制流

某些情况下，为了实现的方便，可能要设计一些专门负责与其他节点通信的控制流。

## 8) 对其他控制流进行协调的控制流

- 可以设计一个主进程，由它负责系统的启动和初始化，其他进程的创建和撤销、资源分配、优先级的授予等工作；
- 也可以把负责协调的控制流设计成一个控制流，而把其他控制流设计成它内部的线程。





## 2、审查与筛选

- 去掉不必要的控制流
  - 多余的并发性意味着执行效率的损失
    - 每个控制流应该有以上列举的理由之一
    - 不要人为地增加控制流

## 3、控制流的表示与分类

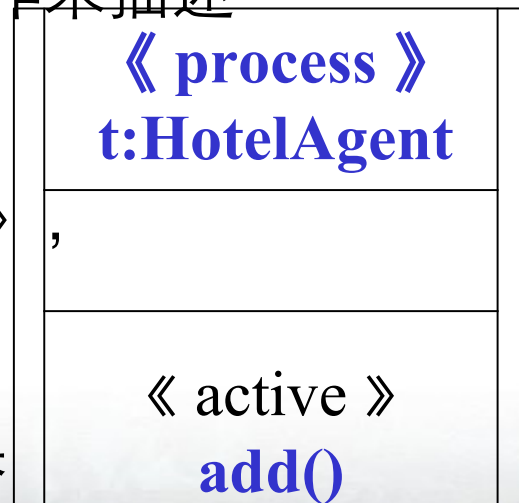
每个控制流都可以用主动对象的一个主动操作来描述

- 在主动对象的表示中区别进程和线程：

UML 中，在类的名字栏中内附加一个标准衍型 《process》 或 《thread》 以表明这个类描述的是进程还是线程。

- 主动操作的表示：

UML2.0 未提供主动操作的表示法，如果想表示主动操作，可以在操作前加 《active》



北京大学



# 第六部分 数据管理部分的设计

## 一、什么是数据管理部分

数据管理部分是 OOD 模型中负责与具体的数据管理系统衔接的外围组成部分，它为系统中需要长久存储的对象提供了在选定的数据管理系统中进行数据存储与恢复的功能。

不同的数据管理系统：

文件系统、R-DBMS、OO-DBMS

——对数据管理部分的设计有不同的影响

问题范围：

对象在永久性存储介质上的存储

只存储对象的属性部分

可能只有一部分对象需要长久存储

## 二、为什么需要数据管理部分

为了隔离数据管理系统对其它部分的影响

使选用不同的数据管理系统时，问题域部分基本相同



清华大学



## 三、如何设计数据管理部分

### 1、选择数据管理系统

文件系统， R-DBMS ， OO-DBMS

### 2、数据存放设计

针对文件系统：

#### (1) 对象存放策略

用文件系统存放对象的基本策略是：把由每个类直接定义，并需要永久存储的全部对象实例，放在一个文件中；其中每个对象实例的全部属性作为一个存储单元，占用该文件的一个记录。

具有继承关系的类的存放策略，让通过一般类直接创建的对象实例和通过特殊类创建的对象实例分别使用不同的文件，以保持文件中每个记录是等长的，并且每个记录中都没有空余不用的字节。



北京大学



## （2）设计数据管理部分的对象类：

一个最主要的对象类是为所有（需要在文件中存储数据的）其它对象提供基本保存与恢复功能的对象类，可将它命名为“对象存储器”。应用系统中各个类的对象是按关键字存取，还是按对象名称存取，还是两者兼而有之，这将对“对象存取器”类的设计提出不同的要求。

## （3）问题域部分的修改

问题域部分的对象通过请求数据管理部分提供的操作实现对象的保存 恢复。为了实现这种请求，这些对象类需要增加一些属性和操作。

对每个需要长久保存其对象实例的对象类，要增加一个属性“类名”，使它的对象在发出请求时以该属性的值作为参数，指出自己是属于哪个类的；数据管理部分通过它知道应该对哪个文件进行操作。







此外，要增加一个“请求保存”操作和一个“请求恢复”操作，它们的功能是向数据管理部分的“对象存储器”对象发消息，分别请求后者的“对象保存”操作和“对象恢复”操作，从而把自己当前的状态（属性值）保存到文件中，或者从文件中恢复以往保存的结果。

由于每个需要长久保存其对象实例的类都需要上述属性和操作，因此可以增加一个一般类来定义它们，供所有这样的类继承。







## 针对 R-DBMS :

### (1) 对象在数据库中的存放策略 :

用关系数据库存放对象的策略是：把由每个类直接定义并需要永久存储的全部对象实例存放在一个数据库表中。

—— 每个类使用一个数据库表

- 1) 列出每个类的所有属性 (包括继承来的属性)
- 2) 规范化——一般按第三范式,
- 3) 定义数据库的表按时间与空间权衡
  - 列：规范化之后的一个属性
  - 行：一个对象实例





**注：**关系数据库要求存入其中的数据符合一定的规范，并用范式来衡量规范化程度的高低。

**第一范式：**关系表中的每个属性值必须是原子的，即它恰好具有一个值而且没有内部结构。


**第二范式：**如果一个关系的所有非关键字属性都只能依赖于整个关键字（而不是依赖关键字的一部分属性），则该关系在第二范式。

**第三范式：**除第二范式的要求之外，每个非关键字属性只依赖于关键字，不能仅仅是其它关键字属性的进一步描述。（即没有传递依赖）

**Boyce-Codd 范式（BCNF）范式：**如果一个关系的每个决定因素都是侯选关键字，则该关系在 BCNF 中。

**第四范式：**除第三范式的要求之外，两个或两个以上的非关键字属性不能总是映射到另一个非关键字属性。（即没有多值依赖）

**第五范式：**除第四范式的要求之外，两个或两个以上的非关键字属性经过连接运算不能总是映射到另一个非关键字属性。



## (2) 设计数据管理部分的类 并修改问题域部分 —— 两种方案

方案 1 :

问题域部分：每个类的对象自己存储自己

数据管理部分：设立一个对象，提供两个操作——

(1) 通知问题域部分的对象自己存储自己

(2) 检索被存储的对象

为了存储自己，对象要知道什么？

本类对象的属性数据结构

本类对象对应哪个数据库表

对象实例对应数据库表的哪一行



北京大学



## 方案 2 :

数据管理部分设立一个对象，负责问题域部分所有对象的存储与检索

问题域部分的对象通过消息使用数据管理部分对象的操作

为了存储各个类的对象，数据管理部分的对象要知道什么？

每个要求存储、检索的类的属性数据结构

每个要求存储、检索的类的对象存放在哪个数据库表

当前要求存储或检索的对象属于哪个类，对应数据库表的哪一行

