



面向对象的编程（OOP）和 面向对象的编程语言（OOPL）



北京大学



主要内容：

程序设计范型

不同的程序设计范型

面向对象的程序设计范型主要特点

面向对象的编程语言

基本特点、历史、类别

语言、类库及编程环境的结合

为实现 OOD 模型，如何选择编程语言

从哪些方面评价编程语言？

简单介绍几种典型的 OOPL

C++——Visual C++

Object Pascal——Delphi

Smalltalk

Objective-C

Eiffel

Java



北京大学



一、程序设计范型（programming paradigm）

关于计算机系统的思考方法。它体现了一类语言的主要特点。
（蔡希尧）

人们在程序设计时所采用的基本方式模型。（Tello. E. R）

程序设计范型（编程范型或编程范式），是一类典型的编程风格，如过程化编程、面向对象编程、指令式编程等等为不同的编程范型。编程范型提供了（同时决定了）程序员对程序执行的看法。例如，在面向对象编程中，程序员认为程序是一系列相互作用的对象，而在函数式编程中一个程序会被看作是一个无状态的函数计算的序列。（维基百科）



北京大学



面向过程的程序设计范型：

中心思想——程序设计主要是过程设计

决定所需的过程，设计过程的算法

关键：过程调用

语言提供向过程传送变元和返回值的设施

模块化程序设计范型：

基本思想——信息隐蔽，需求与求解方法分离，相关的数据结构与算法结合在一个模块中，与其它模块隔离，使其它模块不能随便访问——有了封装的思想

例如：Modula-2

其它程序设计范型：

结构化程序设计，函数式程序设计，逻辑程序设计等



北京大学



面向对象是一种新的程序设计范型

是在上述范型基础上发展起来的

增加了类和继承，用类创建对象实例

思想方法

从客观存在的事物出发构造软件系统

运用人类日常思维方式

主要特点

使用对象、类、继承、封装、聚合、关联、消息、多态性等基本概念来进行程序设计。



北京大学



二、面向对象的编程语言（OOP）

1、基本特性：

语言元素能够支持——

- 类的定义

- 对象的静态声明或动态创建

- 属性和操作的定义

- 继承、聚合、关联和消息的表示

语言机制——

- 类机制

- 封装机制

- 继承机制

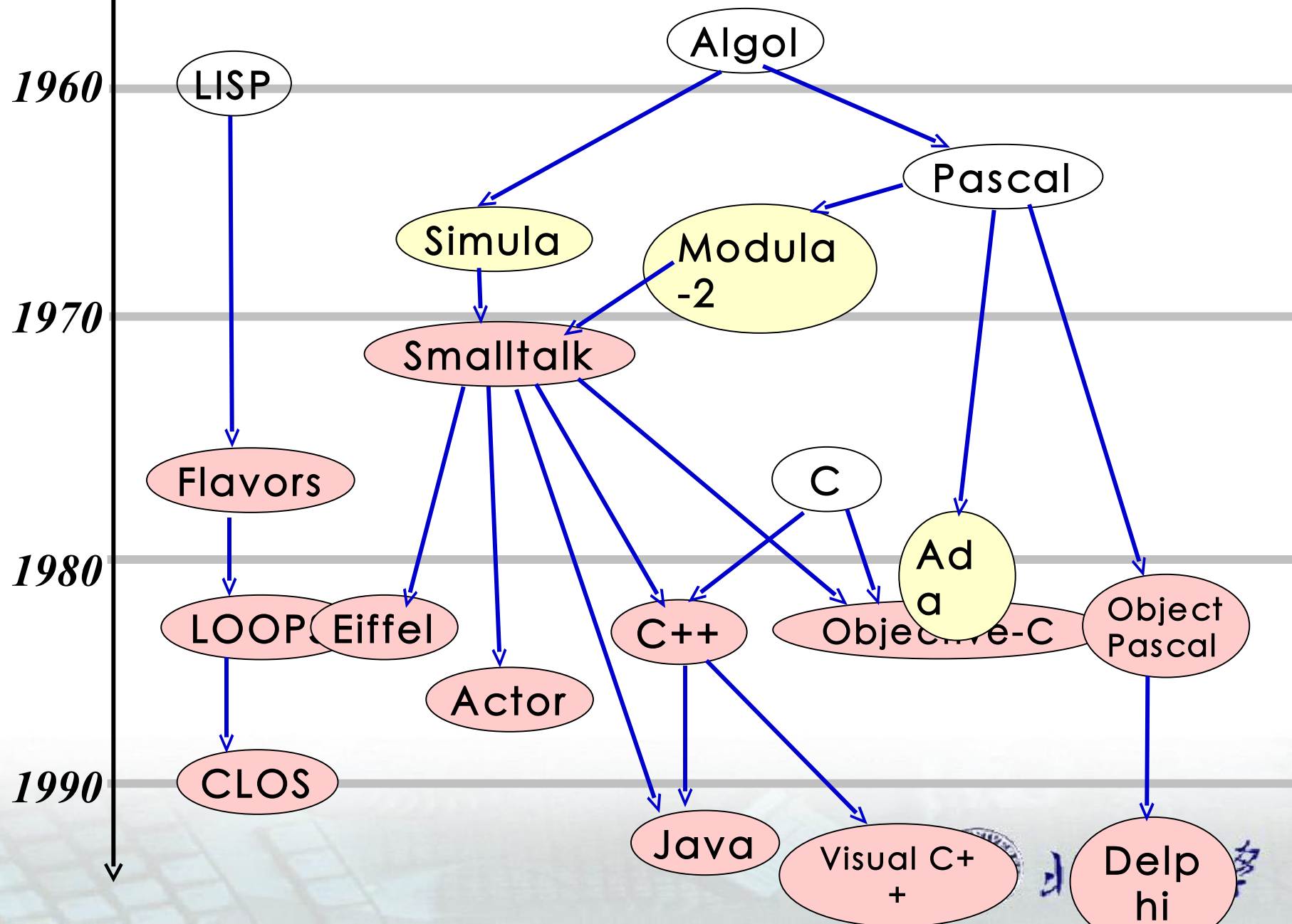
高级特性：

多态、多继承的表示和支持机制



北京大学

2、发展历史及语言谱系





3、类别

纯面向对象语言

例如：Smalltalk、Eiffel

较全面地支持 OO 概念
强调严格的封装

混合型面向对象语言

例如：C++、Objective-C、Object Pascal

在一种非 OO 语言基础上扩充 OO 成分
对封装采取灵活策略

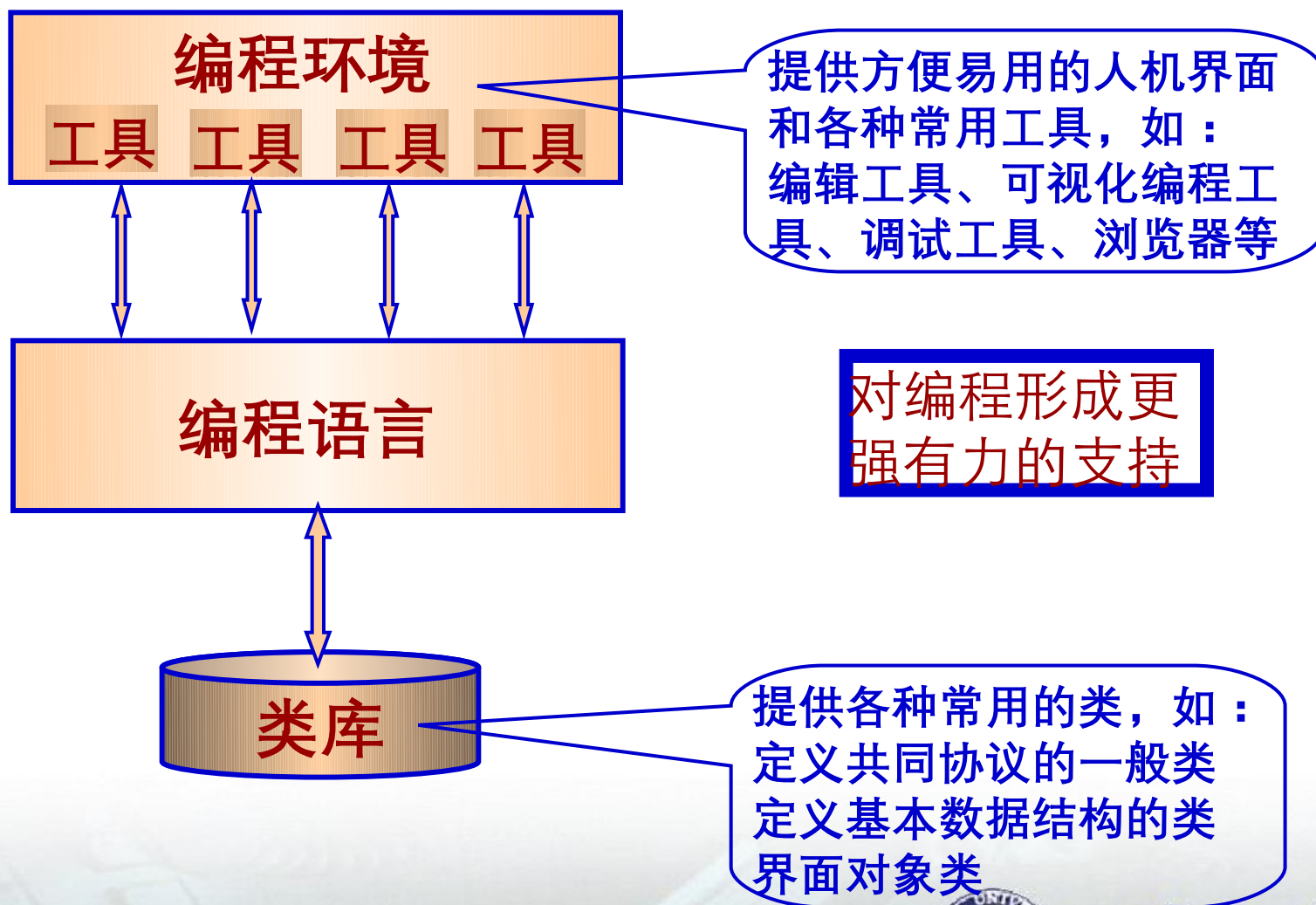
结合人工智能的面向对象语言

例如：Flavors、LOOPS、CLOS



北京大学

4、语言 + 类库 + 编程环境





三、为实现 OOD 模型选择编程语言

在 OOD 完成之后，选择什么编程语言实现 OOD 模型？

1、一般原则

- * 基本原则——语言的选择完全从实际出发
主要考虑成本、进度、效率等实际因素
- * OOPL 是实现 OOD 的理想语言
它使源程序能很好的对应 OOD 模型。
- * 带有类库、编程环境、权限管理的 OOPL 更好。
- * 用非 OO 语言也能实现 OOD 模型
缺乏 OO 机制的保证和支持，
但若自觉遵循一定的原则，可以保持某些 OO 风格。
。





着眼点：

语言捕捉问题域语义的能力，
即对 OO 概念的表达能力
对 OOD 模型的实现能力

目标：

一致性：分析、设计和编程
各个阶段都能采用一致的基本表示
概念、术语、风格都一致，形成良好的映射

可复用性：

可维护性：



北京大学

2、编程语言的评价标准

(1) 能否描述类和对象

是否提供封装机制？

对封装有无可见性控制？

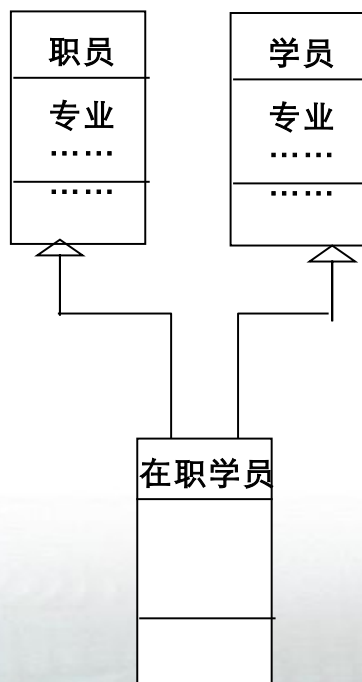
(2) 能否实现一般 - 特殊结构

支持多继承、单继承还是不支持继承？

支持多继承时，是否能解决命名冲突？

是否支持多态？

什么是命名冲突



- 职员类的“专业”是该职员从事的专业；
- 学员类的“专业”是该学员学习的专业。

问题：

“在职学员”类同时继承了两个“专业”属性，引用时无法判断指的是哪一个。

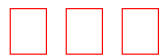


北京大学



- (3) 如何实现整体 - 部分结构
用什么实现？如何表示多重性？
- (4) 如何实现属性和操作
用什么表示属性？用什么描述操作？
有无可见性控制？
能否描述约束？
是否支持动态绑定(dynamic binding) ？
- (5) 如何实现关联和消息通讯
用什么实现关联？如何表示多重性？
如何实现消息通讯？
- (6) 其它可考虑的因素（反映于具体的语言版本）
是否带有可视化编程环境
是否带有类库
能否支持对象的永久存储





绑定：一个对象（或事物）与其某种属性建立某种联系的过程。如：一个变量与其类型或值建立联系，一个进程与一个处理器建立联系等。——《计算机科学技术百科全书（第二版）》

在计算机语言中有两种主要的绑定方式：

- **静态绑定**发生于数据结构和数据结构间，程序执行之前。静态绑定发生于编译期，因此不能利用任何运行期的信息。它针对函数调用与函数的主体，或变量与内存中的区块。
- **动态绑定**则针对运行期产生的访问请求，只用到运行期的可用信息。在面向对象的代码中，动态绑定意味着决定哪个方法被调用或哪个属性被访问，将基于这个类本身而不基于访问范围。



四、几种典型的面向对象的编程语言简介

C++——Visual C++

Object Pascal——Delphi

Smalltalk

Objective-C

Eiffel

Java

**重点： C++， Smalltalk， Eiffel
， Java**





1、C++

由 AT&T 的 Bell 试验室开发， 1988 年推出产品

是在 C 语言的基础上扩充 OO 特征而得到的
是 C 语言的超集

是一种混合型的 OOPL

保持 C 语言的高效率、可移植，与 C 兼容
使广大程序员容易接受

采用强类型机制

支持动态绑定

是目前使用最广的 OOPL



北京大学



(1) 类和对象

类 \Rightarrow class

对象 \Rightarrow object

封装机制：有

可见性控制：

private 、 protected 、 public 、 friend

对象的创建和删除:

提供构造函数（ constructor ）—— 类名（ ）

(C++)

析构函数（destructor）—— ~类名（）

:

静态对象（从程序开始执行到退出）

创建：类名 对象名

删除：程序退出时

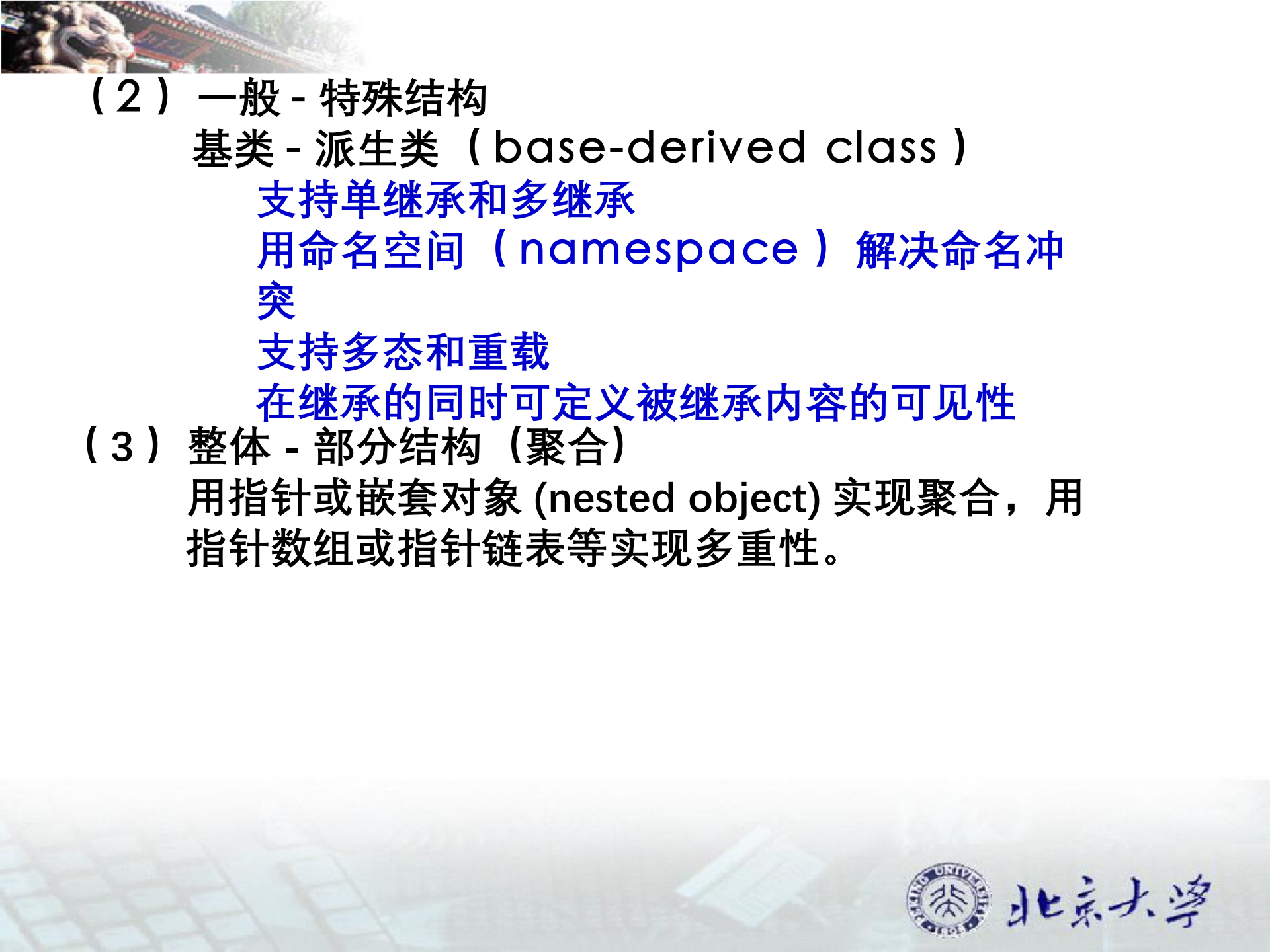
动态对象（显式地创建和删除）

创建：对象指针 = new 类名（参数）

删除: delete 对象指针



北京大学



(2) 一般 - 特殊结构

基类 - 派生类 (base-derived class)

支持单继承和多继承

用命名空间 (namespace) 解决命名冲突

支持多态和重载

在继承的同时可定义被继承内容的可见性

(3) 整体 - 部分结构 (聚合)

用指针或嵌套对象 (nested object) 实现聚合, 用指针数组或指针链表等实现多重性。





(4) 属性、操作

属性⇒成员变量

可见性：private，protected，public，friend

类中的变量可说明为静态的（即为其所有的对象共享）

操作⇒成员函数

构造函数：与类名相同，可以有多个

析构函数：~类名

可见性：同属性

没有显式的约束

动态绑定：声明为 virtual 的函数是动态绑定的，

在基类中定义的 virtual 的函数，可以在派生类中改写

(5) 关联和消息

用指针实现关联

多重关联可以用指针数组

消息：用函数调用实现，采用强类型的参数



北京大学

术语对照

OOA 和 OOD

C++

对象 object

对象 object

类 class

类 class

属性 attribute

成员变量 member variable

操作 operation

成员函数 member function

一般 / 特殊
generalization/specialization

基类 / 派生类
base class/derived class

整体 / 部分 whole/part

嵌套对象 nested object
嵌入指针 embedded
pointer

消息 message

函数调用 function call

关联 association

对象指针 object pointer



北京大学



2、Smalltalk

是由 Xerox 公司 PARC 研究中心在 Alan Kay 的研究工作基础上开发的，先后发布了 Smalltalk-72、76、78、80 等版本。

Smalltalk-72 其中正式使用了“面向对象”的术语
Smalltalk-80 成为完善的面向对象语言

Smalltalk 语言是最早的、最有代表性的 OOPL 之一
迄今为止大部分 OO 基本概念 Smalltalk 都已具备

是一种纯 OO 的语言

除对象之外，没有其它形式的数据

全部属性都是严格封装的，全部方法对外可见

是一种弱类型的语言


不声明变量类型，不作类型检查

伪编译，生成虚拟机代码，然后解释执行

支持动态绑定



北京大学



Smalltalk - 80 不仅是一个编程语言，而且带有：

类库

类库中的类不仅支持用户的编程语言的编译、排错、编辑等程序的实现也基于该类库

多窗口的图形用户界面 和 一组编程工具——形成集成化、交互式的编程环境

用户可以在浏览器上通过填写类的模板来定义类，
并在相关的窗口中定义它的方法和消息



北京大学



(1) 类和对象

类 \Rightarrow class 对象 \Rightarrow instance

封装机制：有

类的定义：用 Smalltalk 浏览器填写，供填写的类定义模板格式如下：

Class name 类名

Superclass 指出父类是谁（最上层是 object）

Instance variables 实例变量

Class variables 类变量（大写开头）

Instance methods 实例方法

Class methods 类方法

对象的创建：

对象名： = 类名 new

(2) 一般 - 特殊 \Rightarrow 超类 - 子类

特殊类的定义：只要在模板的 superclass 一栏中填写超类名
单继承

以 Object 为最上层的类（根）





(3) 整体 - 部分 \Rightarrow 嵌套对象

利用 Smalltalk 库中的集合 (collection) 类,
可使一个整体对象中包括多个部分对象

(4) 属性、操作

属性 \Rightarrow 实例变量

可见性: 本类

没有显式约束

操作 \Rightarrow 方法

可见性: 都可见

没有显式约束

支持动态绑定

(5) 关联和消息

关联 \Rightarrow 用对象标识聚集实现

消息 \Rightarrow 消息 (通过协议)





术语对照

OOA 和 OOD

Smalltalk

对象 object

对象 object (一般讨论)
实例 instance(编程语言)

类 class

类 class

属性 attribute

实例变量 instance variable

操作 operation

方法 method

一般 / 特殊

generalization/specialization

超类 / 子类

superclass/subclass

整体 / 部分 whole/part

嵌套对象 nested object

消息 message

消息 message

关联 association

对象标识集合

object identifier collection



北京大学



3、Eiffel

是由 B. Meyer 等人于 1985 年在美国交互软件工程公司（ISE）开发的

是一种纯 OO 的语言

全面地支持 OO 概念

支持多继承（是 OOP 中较早的）

解决命名冲突

强类型

全面地静态类型化

支持参数化类型（类属）

支持断言和不变式

带有编程环境和大量的开发工具

是一种较成熟的 OOP

在语言设计上很受称赞

80 年代后期已成功的应用于工业界和军界



北京大学

(1) 类和对象

类 \Rightarrow class 对象 \Rightarrow object

类的定义:

class 类名 export

移出特征表

——列出允许外部访问的变量及例程名

inherit

父类名

rename 特征名 as 新特征名,

——列出重新命名的变量及例程名

redefine 特征名,

——列出要在本类重新定义的变量及例

程名

父类名.....

feature

变量名: 类型;

.....

例程名 (参数表) is

do

语句

end

.....

end



北京大学



(2) 一般 - 特殊 \Rightarrow 祖先 - 子孙

支持多继承，用 inherit 列出所有父类

实现多态性，允许特征重定义（用 redefine）

通过重命名解决命名冲突（用 rename）

(3) 整体 - 部分

嵌套对象或嵌入指针





(4) 属性、操作

属性 \Rightarrow 变量

可见性：由移出表控制

移出——外部可见

不移出——本类可见

操作 \Rightarrow 例程 (routine)

可见性：由移出表控制

约束：通过断言表示操作约束

先决条件 (require)

后续条件 (ensure)

循环不变式 (invariant)

支持动态操作

(5) 关联和消息

关联 \Rightarrow 嵌入对象指针 (集合)

消息 \Rightarrow 例程调用



北京大学



术语对照

OOA 和 OOD

Eiffel

对象 object

对象 object

类 class

类 class

属性 attribute

变量 variable

操作 operation

例程 routine

一般 / 特殊
generalization/specialization

祖先 / 子孙
ancestor/descendant

整体 / 部分 whole/part

嵌套对象 nested object

消息 message

例程调用 routine call

关联 association

嵌入对象指针
nested object pointer




北京大学



4、Java

- ❖ Java 是由 Sun 公司于 1995 年推出的，适合于分布式环境，独立于平台。
- ❖ Java 是纯面向对象，语法与 C++ 基本一致，但去掉了 C++ 的非面向对象成分。
- ❖ Java 不使用指针，解释执行。
- ❖ Java 对分布式和客户 / 服务器结构的支持，提供了丰富的类库和方便有效的开发环境，并提供了语言级的多线程、同步原语和并发控制机制。





(1) 类和对象

定义类的关键字: `class`

封装机制: 有

对象的创建和删除:

构造函数: 类名 ()

终止函数: `finalize()`

对象的静态声明和动态创建类似于 C++

(2) 属性和操作

属性: 实例变量、类变量

可见性: `private`、`protected`、`public` 和 `friendly`

操作: 实例方法、类方法

可见性: 同属性

(3) 继承

超类 / 子类

支持重载和多态

(4) 关联、聚合

可用对象引用实现关联或聚合



北京大学



五、用非 OO 编程语言实现 OOD 模型

1、基于对象的语言——Ada

Ada 是面向程序包的语言

有把变量和过程打包的结构

但不是 OOPL，主要因为不支持继承

(1) 类，对象 \Rightarrow 无

但可以把变量和过程打包，在形式上描述类和对象

(2) 一般 - 特殊 \Rightarrow 无

Ada 不支持继承

(3) 整体 - 部分 \Rightarrow 集合或指针



北京大学



(4) 属性与操作

属性⇒变量

以“私有”、“有限私有” (limited private) 控制可见性

支持显式的约束

操作⇒过程，函数

参量的类型可以是参数化的（类属）

例如： $x : \text{ArgType}$

其中 x 是参量， ArgType 是未定的类型，调用时指定。
这使过程、函数可适应多种类型的参量。

程序包说明部分声明的过程函数外部可见
实体部分声明的为私有

(5) 支持异步执行的任务

关联⇒类型指针

消息⇒过程 / 函数调用



北京大学



2、过程语言——以 C 为例

(1) 类，对象 \Rightarrow 无

可以用结构定义对象，可通过指针说明应该有哪些函数（不封装）

(2) 一般 - 特殊 \Rightarrow 无

可把一般结构嵌入特殊结构

(3) 整体 - 部分 \Rightarrow 指针、嵌套的结构

(4) 属性与操作 \Rightarrow 变量、函数

(5) 关联 \Rightarrow 指针；消息 \Rightarrow 函数调用





附：参见《计算机科学技术百科全书》第二版，清华大学出版社，2005年11月。

(1) 绑定：一个对象（或事物）与其某种属性建立联系的过程。

绑定可分为静态绑定和动态绑定。静态绑定只需检查程序正文就可以判定绑定出现与给定的应用出现是否对应。至于动态绑定，出现与给定的应用出现是否相对应要取决于程序的动态控制流，LISP和smalltalk等语言有动态绑定，即有动态类型，大多数程序设计语言选择了静态绑定。

(2) 前后断言方法：在语句前后分别加上前提条件（即前断言）和结果断言（即后断言），用程序设计逻辑证明程序正确性的方法。前提条件是语句可执行的充分条件。结果断言指语句执行中止后应具有的性质。前提条件又称为前置断言，结果断言又称为后置断言或后置条件等。



北京大学