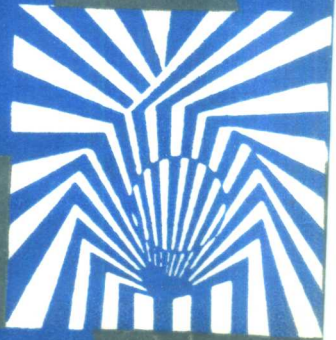
面向对象程序设计语言

[法] 克鲁德·巴依 等著

马玉书 杨立嘉 译



石油工业出版社

面向对象程序设计语言

〔法〕克鲁德·巴依　等著

马玉书　杨立嘉　译

石油工业出版社

(京) 新登字 082 号

内 容 提 要

本书全面地介绍了面向对象程序设计的思想及面向对象语言。书中第一部分阐述了面向对象语言中有关对象、类、例示、消息和继承等基本概念，并给出了对面向对象语言进行评价和比较的标准；第二部分对 SMALLTALK、ACT1、G-LOGIS、KEE、ROSS 和 ADA 等六种著名的面向对象语言进行详细介绍和综述，并对当前在美国和欧洲流行的 C++、FLAVORS、KOOL、OBJECTIVE C 等十多种面向对象语言做了概貌性的介绍；最后，在本书的第三部分介绍了面向对象程序设计在模拟仿真、专家系统和数据库等领域中的应用实例。本书可作为大专院校计算机专业及其他专业师生学习人工智能、软件工程及程序设计语言等有关课程的参考书，也可供从事计算机程序设计及有关软件研制的科技工作者、工程技术人员和管理人员阅读参考。

面向对象程序设计语言

〔法〕克鲁德·巴依 等著

马玉书 杨立嘉 译

*

石油工业出版社出版

(北京安定门外安华里二区一号楼)

石油工业出版社印刷厂排版印刷

新华书店北京发行所发行

*

850×1168 毫米 32 开本 8 印张 207 千字 印 1—2000

1993 年 11 月北京第 1 版 1993 年 11 月北京第 1 次印刷

ISBN 7-5021-0897-1 / Z · 50

定价：6.60 元

译 者 前 言

当前，世界正处于电子计算机更新换代的前夜。在新一代（人工智能）计算机尚未问世之前，计算机界都在发展自己的战略，其着眼点都放在如何在当前的冯·诺依曼（前四代）计算机上开发和研制出更理想、更实用的人工智能开发环境（工具）和其他的先进软件工具；另一方面鉴于人工智能、数据库、并行及分布式软件等技术的崛起，促使人们去寻求更丰富的表达高层概念和抽象数据结构的手段。

目前，在前四代计算机上广泛使用的 FORTRAN、PASCAL 等均属过程式语言，这些基于冯·诺依曼程序设计风格的传统计算机语言的基本设计思想，是以存放初等数据类型值的变量作为处理对象，以变量赋值改变"当前状态"作为基本操作，它们存在着"数据抽象和模块化能力差"、"信息不能隐藏"、"必须严格遵从冯·诺依曼体系的严格顺序性"等诸种缺陷。因而不能有效地在人工智能及知识工程领域中用来表示知识和处理知识，不能较真实地用来描述现实世界。

面向对象程序设计语言 OOPL（Object-Oriented Programming Language），是用基于 Parnas 的信息隐藏和 Guttag 的抽象数据类型等面向对象程序设计方法学的思想和概念，而实现的一种新型计算机语言或环境。面向对象程序设计方法简单、直观、实用、自然，十分接近于人类处理问题的自然思维方式。采用面向对象范例进行程序设计时，不是把精力用于分析和逐步求精输入／输出之间的关系（如同过程型程序设计）；也不是描述事物间的逻辑关系（如同逻辑程序设计），而是从分析问题领域中客体的属性和行为，及其相互关系入手。在面向对象程序设计中，仅有对象（object）、类（class）、方法（method）、消息

(message) 和继承 (inheritance) 五个基本概念，用它们即可完整地用来描述现实世界。在面向对象程序设计中，把客观世界中的所有事物都表示成对象，亦即客观世界是由对象所组成，每个对象是其状态和行为的封装 (encapsulation)，其中状态是该对象一系列属性值的集合，而行为是在对象状态上操作的方法 (method) 的集合。所有共享相同属性和方法集的对象构成一个类 (class)，若干本质相同的类可构成一个超类 (superclass)；与此相对应，一个类所继承的属性和方法类的特例 (specialization)，称为子类 (subclass)。所有的类被组织成一带根的有向非环图或一层次结构。一个类可以继承层次中其直接祖先或间接祖先 (超类) 的所有属性和方法。每个对象把一个数据类型和一组过程 (方法) 封装在一起，因此一个对象不能直接访问或改变另一对象的内部状态 (属性) 或操作 (方法)。对象间除了互相传递消息 (message) 的唯一一种联系之外，不再有其他的联系。一切局部于对象的信息和实现方法都被封装在相应的对象类的定义之中，在外部是不可见的。因此，在 OOPL 中，对状态和行为，只能通过显式定义和发送消息来对其进行存取。

面向对象语言的逻辑基础是反射逻辑 (reflection logic)，其操作语义为高阶逻辑。采用 OOPL 进行程序设计，就是把客观世界中的问题分解成对象；确定各对象应具有的信息 (状态特性) 和对这些信息的处理及操作方法，将相关对象综合成类和子类；通过类的例示 (instance) 生成具体的对象。程序的运行就是各个对象经过消息传递相互联系，协同完成对问题的求解。

由于面向对象程序设计方法具有模块性、信息隐藏、数据抽象、继承性、代码共享和软件重用等多种特点，它的概念和处理方法比较符合人类认识问题、分析问题和解决问题的一般规律，又能有效地用来组织和管理不同类型的数据和知识，支持多种知识表示，它将成为 90 年代计算机语言的主流和人工智能及知识工程领域的重要工具，以及主要的软件工程环境。目前，面向对象程序设计方法已在模拟仿真、软件工程、人工智能及知识工

程、专家系统及其开发工具、数据库、图形学及图象处理、窗口技术及友好用户界面等多种领域获得广泛应用。面向对象程序设计将完全改变传统的程序设计方法，它比 70 年代的结构程序设计（如 PASCAL）方法具有更多的优点。有人预言，OOPL 将为计算机程序设计语言和软件工程带来一场革命。

实际上，面向对象程序设计方法学 OOPM（object-oriented programming methodology）的研究由来已久。早在本世纪 40 年代，在对数字模拟的分析研究中就引入了“对象”的概念，随后在对模拟系统的分析中，大量的模拟仿真语言，如 Simscript、GPSS、CSL 和 Simula I，为此应运而生。在 Simula I 中的“活动（activity）”、“过程（process）”概念正是今天 OOPL 中“类”和“对象”概念的雏型。60 年代中期，随着 Simula I 中不断引入子类、模块、封装等新概念，导致了 Simula 67、Modula-2 等具有 OOP 特点的一些模拟仿真语言的出现，它们被称为 OOPL 的祖先或前身。随着 70 年代结构化程序设计、软件工程和人工智能技术的飞速发展，促使人们去寻求更丰富的能表达高层概念和抽象数据结构的手段。终于在 70 年代初期，以 Smalltalk 为代表的面向对象的程序设计语言（环境），在 Simula 67 和 Modula-2 的基础上，正式诞生了。Smalltalk 作为一种纯面向对象语言对 OOPL 的发展曾起过重要作用。但近年来，由于世界各国计算机界对 OOPM 的广泛重视和关注，在理论、实践方面取得不少进展，并已开发出数十种面向对象语言。

自 80 年代国外大规模开展面向对象程序设计方法学的研究以来，已形成多层次、多侧面的研究领域，而引起计算机科学界及人工智能领域的广泛兴趣和关注。OOPM 涉及的研究领域有：(1) 包括对继承语义、计算反射、基于约束计算、面向快速原型、OOP 语义模型等方面的 OOP 理论研究；(2) 各种不同风格（纯、混合型和并发型）的面向对象语言；(3) 从软件生命期出发，研究面向对象概念在软件的说明、分析、设计、编码、测试等阶段应用面向对象软件工程（OOPE）；(4) 面向对象数

据库系统（OODBS）；（5）面向对象系统的应用，包括CAD／CAE／CAM、CASE、动态模拟、人机界面、面向对象数据库、专家系统和基于窗口的面向对象的操作系统等等。目前，由于面向对象程序设计思想被认为是新一代的结构程序设计，因此从更高的层次和角度研究面向对象的方法，已成为90年代计算机界的热门研究课题。

本书是向国内读者最早推出的一本全面介绍面向对象程序设计思想、概念及面向对象语言的著作之一。书中第一部分阐述了面向对象语言中有关对象、类、例示、消息和继承等基本概念，并给出了对面向对象语言进行评价和比较的标准。第二部分对Smalltalk、ACT1、G-LOGIS、KEE、ROSS和ADA等六种著名的面向对象语言进行了详细介绍和综述，并对当前在美国和欧洲流行的C++、FLAVORS、KOOL、OBJECTIVE C等十多种面向对象语言做了概貌性的介绍；最后，在本书的第三部分介绍了面向对象程序设计在模拟仿真、专家系统和数据库等领域中的应用实例。本书可作为大专院校计算机专业及其他专业的师生学习人工智能、软件工程、数据库系统及程序设计语言等有关课程的参考书，也可供在人工智能、专家系统及其开发工具、数据库、软件工程等有关领域中从事计算机程序设计及有关软件研制的科技工作者、工程技术人员和管理人员阅读参考。本书在翻译过程中曾得到有关专家的指导和帮助，中国科学院计算所刘文卓同志对本书进行了认真的校对，在此表示感谢。由于我们水平有限，译文难免有错，恳请读者批评指正。

译者

1991年9月

目　录

第二部分　语言

第三部分　应用实例

绪　言

"传统"信息技术处理的是有关数据的操作问题，而这些数据大都以数值型数据或字符型数据为主。一个程序在"传统"信息技术中所承担的任务，就是以特定的方式处理一种确定类型的数据。相比之下，人工智能软件则不仅要处理数字和字符，而且主要是处理更为抽象的信息：知识。

人工智能的基本技术主要建立在两种概念之上，这两种概念能够有效地用来表示内容十分丰富而变化又多种多样的知识。第一种概念，产生式规则和逻辑程序设计，是由于专家系统的普及和推广而开发出来的工具[FARRENY]。第二种概念，同样极为常用，亦即所谓对象表示方式。

面向对象的程序设计概念始见于 1970 年左右，尔后又逐步推广到人工智能领域。它给众多的计算机语言以启示，尤其是那些试验性的、为适应局部需要或为测试某一具体概念而设计出来的语言。然而现在，由于对以往（语言）的探索、归纳和综合，这些众多的语言似乎正在朝着通用型工具的方向发展。

SIMULA 是对象语言的先驱（1967 年）。正如该语言的名称所示，它原是应用于模拟仿真领域的。在这里，对象是把程序和数据汇集在一起的一个实体，并且它遵循例示（即根据一般的模型或类建立的对象）和继承（按子类说明的类）的原则。

然而，面向对象程序设计真正得到飞跃发展，那还是 1972 年 SMALLTALK 出现之后的事。后来，在 1975 年，MINSKY 又提出和发展了"框架"的概念，这一概念对面向对象程序设计产生了极大的影响。从而使知识表示语言（框架表示语言 F.R.L＝Frame Representation Language；知识表示语言 K.R.L＝Knowledge Representation Language）在面向对象程序

设计中迅速得以采用。与此并行发展的 PLASMA 语言（1975年），引进了角色概念和用持久及筛选（方法）进行程序设计。

今天，已有众多的包括面向对象语言在内的计算机语言投入运转，并已相继商品化而大量投入市场。人工智能技术也得到广泛应用：知识表示、专家系统、窗口系统、声音合成、语音识别、机器人学……。这些技术相互结合，被纳入诸如专家系统开发工具、模拟仿真程序、数据库管理程序、C.A.D 等软件之中，并与传统的计算机语言（C、PASCAL、LISP、PROLOG）融为一体。LISP 机的操作系统也是建立在一种对象系统：FLAVORS 的基础之上的。而 Apple 的 Macintosh 只是它的首次大规模实际应用而已。

"面向对象程序设计"风格有如海底涌浪，潜力很大，它拓宽了人工智能实际应用的常规领域。届时，它将完全改变信息技术各个领域中传统程序设计的常规习俗。

我们的宗旨在于将对象语言进行分类。本书将展示对象语言家族谱系的全貌，并综合分析比较各种对象语言的相同或不同之处，其重点是放在对现存的各种面向对象的表示性语言进行细致的比较评价之上。

通常，研究程序设计语言，有两种可供选择的途径。第一种，就是认真细致地检测数种语言，对之进行比较，并把它们最显著的方面加以区别。然后通过深层原则得出结论；另一种途径则是先不管其实现方法，从语言的构造出发，然后再检验其实例和实现问题。

我们采用的是后一种途径，因为只有通过对基本概念的理解，才能在各种不同的语言之间进行有益的比较。不同的对象语言初看起来是很不同的。通过精心验证分析后，就会发现：它们彼此之间的相象，实则远远超过它们各自的外部形式所令人设想得出的近似之处。它们具有共同的基本概念。对此，我们将首先加以展示，以便更好地理解。我们还将从基本概念中，推出对比的标准，这些标准会使我们对上述基本概念的研究更为详尽具

体。

我们从现有的各种面向对象的语言中，选取了有代表性的一些样本，用它来指出"对象"方法的丰富性和多样性。最后，本书还将十分广泛地涉猎现已投入使用的各种面向对象的语言和工具，但这种浏览毕竟不能包罗万象。

本书第三部分的内容将涉及应用面向对象概念的几个实例。我们不可能描述所有一切业已进行的开发应用；尤其是现在，对象概念已越来越广泛地深入到信息技术的各个门类，这样做更不可能。但读者毕竟可从中很好地窥见并了解到，面向对象概念已深深地打入各种不同技术领域的情况。

第一部分 概 念

1. 引 言

面向对象的程序设计（Object-oriented Programming），是当今众多的计算机语言中最具有特色且别具一格的一种程序设计范例，它与其他计算机语言的程序设计风格迥然不同。面向对象程序设计同样能满足结构程序设计特点的需要：可用于设计和维护越来越庞大和越来越复杂的程序。这些程序由于通过建立一致的、易于操作的实体，即著名的对象，而更加模块化了。所谓实体，这是指实际生活中对象的信息转换。例如，一架飞机，它具有一定数量的乘客、时速、飞行高度以及相关的知识（如何起飞、如何降落等）。对象语言打破了"数据"与"程序"之间的传统的二分法。它汇集了与同一种概念相联系的所有知识，其汇聚方式有如烹调技术既包括佐料又包括加工方法一样。

这种程序设计风格使软件变得易于扩展，并可实现软件的重复使用。与传统语言（比如 PASCAL）相反，对象语言程序并不需要预先知道某个对象是哪一种类型？是什么样的内部表示？该程序使用这个对象只需简单了解对象的功能即可。对象是可用于各种不同上下文的"黑盒子"。某些对象，如果与电子元件相比，则可以毫不犹豫地称其为"软件构件"。

本书第一部分力求阐明面向对象程序设计的基本概念。然后再描述其主要优点，并提出一些实际考虑和看法。最后，建立一套广泛适用的标准。以便于同其他语言进行对比。

2. 基本概念

自面向对象语言的先驱 SIMULA [DAHL 70] [BIRTWISTLE 73]问世以来，已有多种面向对象语言(SMALLTALK、FLAVOR、ACTOR) 被研制开发出来。因此，有人就可能认为已经从对这些不同语言的研究实验中归纳出来了一系列自成体系的术语，大概以为一整套描述对象风格的概念已经确定下来了。但是，事实上这种程序设计的风格至今还在不断地演变，并呈现出多种不同的形式。许多语言都具有面向对象的概念和特点，然而它们之间却存在着很大的差异。因此，只要人们知道各种不同语言是以完全不同的方式来说明和使用"对象"这一概念的，那么最终还是可以确定这种程序设计的基本思想的。

面向对象的所谓"基本"概念的选择，主要与 SMALLTALK 这一典型、杰出的对象语言密切相关。因为，在该语言环境中已明确地提出了面向对象语言中有关类、继承以及发送消息的完整概念。

2.1 对象

传统的程序设计系统是由数据和处理这些数据的过程（程序）组成的。在面向对象的语言中，只有一种类型的实体——对象（object)。对象可同时表示为叙述性知识（局部数据库：变量或场域）和过程性知识（程序：方法)。因此，一个软件就是各种不同对象的集合。知识（数据和程序）将分布在这些实体中的每一个实体之上。

因此，一个对象就是一个基本模块，它有它自己的一些数据和操作这些数据的过程。可以把对象视为一个服务器，它能执行对自身结构所提出的请求，能够屏蔽这些请求的一部分，或者进行控制存取。这是知识的一部分，该部分拥有对自身的控制能

力，并能够独立于同一属类之外而存在。

按 LISP 的含义去理解，一个对象可以视为一种环境，亦即是（符号 — 值）和（符号 — 函数）联系的集合。两个对象可以采用相同的符号，而把各种不同的值与这个联系的集合联接起来。

在各种面向对象语言中，“对象”一词最为常用，而诸如模式图、“框架”、角色、实体或单元（Unit）……等术语则应用于每种对象语言的类似的意义中。场域（Scope）● 或者变量、“槽”（Slots）、属性、视图（Aspects）等则为对象的组成成分。有时候，对象的这些成分本身又由诸如类型、缺省值……等，也就是人们称之为侧面的那些“子成分”，来确定的（参见 KEE）。

2.2 例示化和类

具有相同结构和相同行为的对象组合在同一类（Class）中。这是一种抽象类型的表示。通常，从一种模型出发，可以建造任意多个样本。这些样本，称之为例示（Instance），这些例示具有由它们的类所定义的共同特征。例示的建立，称为例示化（Instanciation）。

类主要说明每个例示赖以建立的数据结构：亦即场域和方法表。这种说明可能更为确切具体，比如说，对场域的缺省值或侧面的定义就是这样。

例如，我们要建立一个“船舶”类，则它可由一个数据结构和一个方法来定义。

数据结构由如下一些例示变量所组成：

——船长

——设计航速

——吨位

●场域(Scope)又可译为作用域、定义域或辖域。指一个定义在程序中的变量的作用范围。