高等院校信息管理与信息系统专业参考教材

信息系统工程中的面向对象方法

陈余年　方美琪　著

清华大学出版社
http://www.tup.tsinghua.edu.cn

高等院校信息管理与信息系统专业参考教材

信息系统工程中的面向对象方法

陈余年　方美琪　著

清华大学出版社

(京)新登字 158 号

内 容 简 介

本书讲述了信息系统工程中的 00 方法。第 1 章是介绍 00 的理论;第 2 章从信息系统研制生命周期的几个阶段,详细地阐述了如何用 00 方法研制信息系统;第 3 章讲解了信息系统工程的新进展。作者收集了直到 1997 年底的美国流行的 00 软件产品,对其进行了简述。本书是在一百多篇美国最新参考文献的基础上编写的。

本书可作为高等院校信息管理与信息系统专业的高年级及研究生的教材。也可作为计算机应用系统项目的研制者、各级信息处理、管理部门的技术骨干及技术负责人的参考书。

图书在版编目(CIP)数据

信息系统工程中的面向对象方法/陈余年,方美琪著,北京:清华大学出版社,1999
高等院校信息管理与信息系统专业参考教材
ISBN 7-302-03277-7

Ⅰ. 信… Ⅱ. ①陈… ②方… Ⅲ. 信息系统 - 面向对象程序设计 - 高等学校 - 教材 Ⅳ. 6202

中国版本图书馆 CIP 数据核字(1999)第 00090 号

出版者: 清华大学出版社(北京清华大学校内,邮编 100084)
http://www.tup.tsinghua.edu.cn
印刷者: 北京市密云胶印厂
发行者: 新华书店总店北京发行所
开 本: 787 × 1092 1/16 **印张**: 20.25 **字数**: 479 千字
版 次: 1999 年 2 月第 1 版 1999 年 2 月第 1 次印刷
书 号: ISBN 7-302-03277-7/TP·1754
印 数: 0001 ~ 6000
定 价: 22.00 元

序 言

全国计算机基础教育研究会财经信息专业委员会，从20世纪90年代初期以来，对于信息管理与信息系统专业的学科建设和教材建设，投入了大量的精力。在清华大学出版社的大力支持下，专业委员会组织全国有关院校的同行们，陆续出版了一批为信息专业迫切需要的且具有一定特色的教材，这就是几年来已经陆续出版的“信息管理与信息系统专业系列教材”。1997年夏天，在烟台举行专业委员会的学术年会上，来自全国各地教学第一线的同行们，进一步讨论了信息管理与信息系统专业的学科建设。针对该专业内容新、跨度大、变化大的特点，大家一致认为，有必要再组织一套参考书，以满足这个专业本科高年级选修课和研究生课程的需要。这就是目前这一套“信息管理与信息系列专业参考教材”的由来。

最近由教育部正式颁布实行的本科专业目录中，信息管理被列为管理门类之下的一个二级学科。这表明，经过20年的成长与发展，随着信息化建设的深入，信息管理已得到社会各界的认可，成为管理学科建设与现代化管理人才培养的一个不可缺少的组成部分。按照教育部的本科专业目录，原先分散在各领域中的经济信息管理、管理信息系统、科技信息管理等，均归入“信息管理”名下，成为一个覆盖面更宽的专业。对于从事该领域工作的教师来说，是给予了充分的肯定和大力的支持，同时也意味着面临着新的、要求更高的学科建设任务。本专业委员会的全体同志决心从面向21世纪的新标准，进一步创新和探索，为信息管理与信息系统专业的进一步发展努力奋斗。

本套书与前几年出版的“信息管理与信息系统专业系列教材”不同，它不属于基本的核心课程，而是面向本科高年级的选修课和研究生的课程。按照教育部专业调整的精神，专业设置不宜过窄过细，而应当宽口径、厚基础，给学校、教师和学生以更大的发展余地。体现在课程设置中，就意味着应当增加选修课程，使学科能够在宽口径的专业设置中办出自己的特色，使学生能够在厚基础的前提下有更多的选择。而要做到这一点，就需要提供一大批供选择的课程和教材，这套书就是为此目的而组织编写的。显然，对于信息管理与信息系统这样一个内容新、发展快、综合性强的专业，这方面的需求无疑将更为迫切。

每一个专业都有自身最核心的一些内容，它包括从事本专业工作所必需的基本概念、基础知识、基本技能、基本素质，即平时所谓的“看家本领”。然而，在新技术革命的浪潮冲击下，知识与技术的更新速度大大加快，各领域知识互相渗透，综合运用的趋势不断加强，指望在大学四年中准备好一生工作所需要的知识，是不可能的。同样，由于专业分工，只靠某一狭窄的专业领域中的知识和技能，将很难适应未来多变的社会需求。因此，一方

面，拓宽视野、了解和掌握相关学科的知识对于提高素质和适应能力十分必要；另一方面，及时掌握新的技术生长点，了解学科和技术的最新发展方向，对于学生发展的后劲也是必不可少的。本书的第一批书目正是根据以上两方面的思路选定的。

信息管理与信息系统也是信息科学的一个部分，它以现代信息技术为手段和基础，同时又与信息经济学、信息社会学、信息法学、系统科学密切相关，一个称职的、高水平的信息管理人员，对一些知识都应当有一定深度的了解。对于信息管理领域和信息技术领域的一些新发展，如电子商务、数据挖掘等，信息管理与信息系统专业的学生，特别是有兴趣向这些方面发展的学生，也是应当有所了解的。毫无疑问，这些方面的具体内容发展变化是很快的，这一批选题不可能覆盖所有应当考虑的范围。目前这套书只是开头，以后必然要不断地增加、补充。同时，已经出版的几种书，也将随着技术和社会的发展，不断修订和补充，以便切实为各院校从事信息管理与信息系统专业建设的同行们提供帮助。

在清华大学出版社的大力支持下，本套教材第一批已经陆续问世，这是有关院校与老师共同努力的初步成果。由于这项工作是尝试性的，能否实现酝酿时的初衷，还需要实践的检验。因此，我们迫切希望得到各院校以及社会各界的批评指正，从选题范围到具体内容，都希望能够得到中肯的批评和建议。我们特别欢迎在信息建设第一线的同志们，从信息化人才培养的实践需要出发，对于本套书的方针和内容提出意见，并进一步参与本套书的编写工作。

全国计算机基础教育研究会
财经信息管理专业委员会
信息管理与信息系统专业参考教材编委会
主任：陈禹，　副主任：张基温
1998 年 7 月

前 言

我们在1990年由科学出版社出版了《信息系统工程》上册后，正准备出下册。但由于信息系统及其工程在猛烈发展中，原来准备的下册内容已经落后于形势了。我们是在1988年开始编写该书的，是以当时占主导地位的结构化概念和工艺为中心。该书第一篇"总论"中一开始就明确了这个主题。上册"前言"中预告的下册三篇的内容即围绕此主题而编写的。到了上册出版时，信息系统已开始向分布系统发展，而信息系统工程则逐渐由面向对象取代而占主导地位。由于这些新发展，原订下册三篇，有的已经过时了，如第五篇"结构化程序设计"；有的如第六篇"信息系统的研制工具"是讲 CASE 工具的，其内容随着信息系统研制方法与技术的发展和变化而不断在变，且有时变动很大，尚未能定型，即宜将其内容于需要时分在有关之处叙述；同时并在本书第2章中一节内具体评介一个 OO CASE 工具的商品软件，以示这类 CASE 工具的一斑。在此考虑下，这两篇的原定内容现已失去出版的迫切性和必要性。余下的第四篇"信息系统研制方法与技术"是讲功能分解方法与技术、数据分解方法与技术以及面向对象方法与技术，并把重点放在当时占主导地位的功能分解方法与技术上。同样在新形势的要求下，我们决定对这一篇作重大的改动，集中全力讲述将占主导地位的面向对象方法与技术。为了适应当时的需要，中国人民大学信息管理系曾将包括最新内容的功能分解方法与技术部分印成油印本，作为教学补充教材。至于数据分解方法与技术部分，其精华则已吸收在面向对象方法与技术之中。

这样就等于要重写这一册的内容。在考虑重写时，我们决定新内容将全部讲述面向客体及其发展。由于写作上发生这样重大的改动，参加编写者也有些更动，改由我们两人合作编写。我们经过考虑，决定将本书定名为《信息系统工程中的面向对象方法》。这一方面是为了突出反映本书的内容，另一方面也可以说是出版社改变的相应改动。

本书共分3章。第1章"面向对象说"全面系统地讲述面向对象概念的发展经过，它有别于其他说和工艺之处，以及其方法论和图示法。前两者属于基本概念，可以说现已有了一套基本定论，不会有重大的改变。后两方面，方法论和图示法仍在"百家争鸣"之中，最近出现了 UML（unitfied modeling language——通用建模语言），是作为统一的方法和图示而提出的，已被好几家软件厂商接收，但要成为软件界的统一标准，尚有待实践的考验。

第2章"面向对象信息系统的研制"是在增改的生命周期宏观模式下，依次讲述分析、设计和实现（编程）三个研制阶段。所用的方法与技术，是根据我们的观点和看法，吸取了各家之长，尤其是软件界和几个名家彼此相互公认和采用的方法与技术。为了不是空讲方法与技术，我们特选了一个简化银行存款的例子，从头到尾来具体说明这套方法与技术

的应用,而且用 Visual C ++,Smalltalk 以及 Visual Basic 三种语言分别编写出可执行的程序。

第 3 章讲面向对象信息系统的新进展,其中有些是本书的一些独到之处。我们经过广泛而深入的研究,对面向对象信息系统及其工艺的最新发展,做了一全面系统的研讨。整个研讨是在正确的理论的指引下,揭示现行最流行的商品软件中信息工艺的新潮,从实践中看动向,归纳总结而形成前三节的内容。最后一节中根据面向对象信息系统及其工艺的最新发展,提出我们对整个信息系统工程的未来方向的一点看法,指出现已进入组件软件工程时期,正向纵深发展,总有一天能像硬件用集成电路来组装一样,不需编写程序,就把软件研制出来。

我们在写作过程中曾得到中国人民大学信息学院陈禹教授的不少帮助,特此致谢。本书第 2 章中两个 C++ 语言程序是由刘玉国和于芳编写的,Visual Basic 语言程序是由熊辉编写的,在此一并向他们致谢。

最后,关于 Object Oriented 的译法,谈一下我们的看法。我们在 1990 年出版的《信息系统工程》上册中,就把此词译为“面向客体”。后来的流行译法有“面向对象”、“面向目标”以及台湾的“物件导向”。我们认为 Object 在本专业中是指一个能独立自主进行操作的实体,它的操作对象就是数据。“客体”一词反映原词所表达的主动性,而“对象”或“目标”,会令人以为是指被动的东西。我们根据编辑的意见,把书稿中的“客体”全改为“对象”。读者阅读本书时,可能遇到一些地方,“对象”一词在其中的出现,从上下文来看,显得别扭或不通顺。就请以能自主行动的客体(客观存在的实体)来理解它。

陈余年　　方美琪

1997 年 12 月

目 录

第 1 章 面向对象说 …… (1)
1.1 发展经过 …… (2)
1.2 原则 …… (12)
1.3 方法论 …… (20)
1.3.1 增改的生命周期宏观模式 …… (20)
1.3.2 问题领域和解决领域 …… (23)
1.3.3 对象和类的静态描述 …… (26)
1.3.4 对象和类的动态描述 …… (36)
1.3.5 子系统及系统的体系结构 …… (37)
1.3.6 微观的研制步骤 …… (38)
1.4 图示记号 …… (42)
第 2 章 面向对象信息系统的研制 …… (57)
2.1 分析 …… (58)
2.1.1 发现对象的一些途径 …… (59)
2.1.2 使用情节与交往图 …… (65)
2.1.3 对象类结构图与对象生命周期图 …… (71)
2.2 设计 …… (74)
2.2.1 识别接口对象和控制对象 …… (75)
2.2.2 系统的体系结构 …… (80)
2.2.3 六类子系统的设计 …… (81)
2.3 实现 …… (89)
2.3.1 OO 程序设计语言 …… (91)
2.3.2 可视速成编程环境 …… (137)
2.3.3 一个 CASE 工具 …… (148)
第 3 章 面向对象信息系统的新进展 …… (152)
3.1 交互操作对象以及可视化,组件化与速成化 …… (152)
3.1.1 交互操作对象 …… (152)
3.1.2 可视化 …… (158)
3.1.3 速成化 …… (160)

3.1.4 组件化 …… (161)
3.1.5 4GL－OO 研制环境与工具 …… (164)
3.2 分布对象系统与顾客/服务者系统 …… (167)
3.2.1 OMA 和 CORBA …… (173)
3.2.2 SOM 和 OpenDoc …… (179)
3.2.3 COM 和 OLE …… (183)
3.3 组件软件与复合文书 …… (190)
3.3.1 复合文书工艺和组件软件工艺 …… (193)
3.3.2 OLE …… (197)
3.3.3 OpenDoc …… (199)
3.3.4 交互操作对象的标准问题 …… (201)
3.3.5 几个代表性的商品软件的评介 …… (204)
3.4 信息系统工程的今后展望 …… (212)
附录 …… (222)
参考文献 …… (311)

第1章 面向对象说

面向对象说是对客观世界的一种看法，它是把客观世界从概念上看成是一个由相互配合而协作的对象所组成的系统。

人们在接触客观世界中的现象时就会产生感觉或知觉，而在人的意识中形成外界现象的一个概念，称之为知识或信息。这里提出的对象一词乃是这种知识的表达的一种顺乎自然的形式。人们在接触客观现象时，在其头脑中显示出来的总是各式各样的对象，如男人、女人、张三、汽车、房子、计算机、日、月、猫、鱼、银行、商店、工厂，如此等等。这里我们把知识的一种表达形式称之为对象，而在有些文献中则称之为数据。这两个名词在很多情况下往往是作为同义语用的，如在一本讲面向对象的软件建造和介绍面向对象语言 Eiffel 的书[1]中(第 41 页)，即把对象与数据视为同义语。再如面向对象语言 Smalltalk[2]把什么事情都看成是对象，数据和对象就会是同义的。但在混种的面向对象的语言中，如 Microsoft Visual C++，Borland C++[3]以及面向对象的 Turbo Pascal[4]都是在原来的过程语言 C 和 Pascal 的基础上发展起来的，都保留了原来数据型，如整数、浮点数、字符以及纪录等。因此，在这种混种语言中，数据和对象有区别：前者指原来的数据型，后者指新建立的对象。面向对象说是基于数据而发展起来的，其中却把对象区别于数据。第一，对象包括数据与过程；第二，数据是被动的实体，它是加工处理的对象；而对象则是自主行动的实体，能进行操作与运算以及和其他对象互通信息，要求对方作某种活动。本章要详细阐述这些内容。

面向对象说的兴起是信息系统工程发展的一个必然趋势。数据处理包括数据与处理两部分。但在信息系统的发展过程的初期却是有时偏重这一面，有时偏重那一面。在 20 世纪 70 年代和 80 年代，偏重数据处理者认识到初期的数据处理工作是计算相对复杂而数据相对简单。因此，先有结构化程序设计的发展，随后产生面向功能分解的结构化设计与结构化分析。偏重于数据方面者同时提出了面向数据结构的分析与设计。此法认为了解一个组织的最好办法，是建立它所需用的数据的模型，通过这个模型，即可看出系统的结构。从需用的数据即可知问题是什么，以及如何解决。继续这种偏重数据的趋势，到了 20 世纪 80 年代就有信息工程的兴起与发展[5],[6]。信息工程和面向数据结构法一样，也是以数据为分析和设计的基础，但有一重要的不同处：信息工程不但把数据作为组织的资产之一，而且用以建立全组织的模型，进而制定战略性计划。这是信息工程不同于其他方法(功能分解的、面向数据结构的以及面向对象的三种方法)的最重要的区别，即其他方法主要用于信息系统和软件的建模，而信息工程则强调从整个组织的建模出发，以了解组织

为完成所提出的任务所需用的数据,然后在此基础上进行信息系统和软件的建模。功能分解的及面向数据结构的方法或偏重于处理(功能),或偏重于数据,而实际上处理和数据乃是一件事情的两个方面,应作为一整体来建模。因此在20世纪80年代,在面向对象的程序设计的启迪下,面向对象说应运而生。发展到现在正在本学科中逐渐处于占上风和主导的地位。本章将从其发展经过、原则、方法论与图示记号等方面,系统地讲述此说。

1.1 发展经过

面向对象说的兴起与发展和结构化说一样,也是从面向对象的程序设计开始的,逐渐发展到面向对象的设计,最后发展到面向对象的分析。对象这个概念的形成与发展是这个发展过程中的一个中心主题。这个过程可追溯到20世纪60年代结构化程序设计的形成与发展时代,60年代末先有模拟语言SIMULA67的出现,其中体现了类和对象等两个基本观念,到了70年代,乃有Smalltalk等正式面向对象的语言的出现。在面向对象程序设计和语言的影响下,80年代初乃有面向对象设计的正式形成与发展。最后,到了80年代底与90年代初,正式出现了面向对象分析。

整个发展过程中贯穿的一条线索,是程序模块化的发展,以及由此而发展起来的抽象数据型(abstract data type——简称ADT),从模块化和ADT产生了对象概念以及面向对象的程序设计和语言,随后乃有面向对象的设计与分析的兴起与发展,直至今日,仍在演进之中。本节即顺着这条线索讲面向对象说的发展经过,见[1],Part Ⅰ、[7],第4.2节、[8]或[48],第2.1节、[9],第五章、[10]或[11]的增刊。

20世纪60年代初期流行的计算机语言有FORTRAN Ⅱ,COBOL及ALGOL 60等,在这些程序语言中,在算法(处理)上是把整个程序分为若干个子程序,而在数据组织方面,只有全局数据(global data)。这样,各个子程序就通过共用数据联系起来。因而,一处有错,就会涉及其他,造成致命的"水波"影响。程序经过几次修改,就可能会面貌全非,控制关系也会变成所谓"老鼠窝"式的,如入迷宫。这样的程序最后就会变成完全不能满足其使用要求,几乎根本无法进行维护修改,终致废弃。

20世纪60年代中叶到70年代初,程序模块化逐渐成为程序设计的一个重要原则。结构化程序设计也在这时期开始逐渐形成起来。模块化是软件和信息系统工程中分而治之原则的一个体现,它包含抽象化与信息隐藏(*information hiding*)两个概念,模块的某种结构构成整个系统,其中层次结构是最常见及最有用的。

抽象是事物或现象的简括描述,强调其性质中某些方面,而忽视其他方面,或者说,突出事物或现象之间的共性,而暂不考虑他们之间的差异。例如,人作为一个抽象概念,即抛弃了人的种族、性别、年龄及其他等的差异。在计算机程序中,则有过程抽象与数据抽象。过程抽象是软件中首先提出的一种抽象,称为子例程,或过程,或函数。逐渐出现一些程序语言支持这种过程抽象的几种参数传送方式,如形参和实参等。从此,这种过程抽象成为程序的基本构件,并在此基础上出现及发展了结构化程序设计。数据抽象的例子就是常见的整型、实型、字符型、字符串、数组及记录等。抽象数据型(ADT)以及对象,都是数据抽象的高级形式。

模块是抽象的一种表达形式，或抽象的一种包装方式，两者之间是内容与形式的关系。抽象和模块都是为了应付问题的复杂性而提出的。把程序或系统分割为其组成模块，是为了减少其中的复杂性。

20 世纪 70 年代初期提出的程序模块概念是指一组子例程或函数，这些子例程或函数可以改变程序的状态或进行计算。以模块为程序的基本构件的一套程序编写方法论，强调模块的独立性，即 1 编写一个模块时，不需要知道其他模块的内容；2 修改一个模块时，也不会影响其他模块的内容；3 可单独编译一个模块，而不影响其他。这个独立性就是信息隐藏原则的体现。因为程序模块在能隐藏其内容而不让外界知道，所以才能独立地编写。编写一个模块时，不需知其他，就能修改一模块而不影响其他，也才能单独编译一个模块。

综上所述，程序模块具有下列三个特性：

1. 有明确规定的界限；
2. 有规定好的固定接口；
3. 可分开单独编译。

抽象可有层次，例如，大学生、中学生和小学生可抽象为学生，又可再抽象为人等。图 1.1 表示顶层“库存登记”程序模块是经过两层抽象而来的：最低层的“增加库存量”、“减少库存量”、“增加一记录”以及“减少一记录”等四个程序模块抽象为上一级的“处理记录”模块。上一级的这个“处理记录”模块与同级其他三模块又进一步抽象为顶级的“库存登记”模块。在这种抽象层次中，上级模块能调用下级模块，但下级模块则不能调用上级模块。上级模块只需知下级模块能做什么而不知其如何作的。下级模块只需提供所要求的服务，而不需知上级用它去做什么。所有这些都体现了信息隐藏原则。

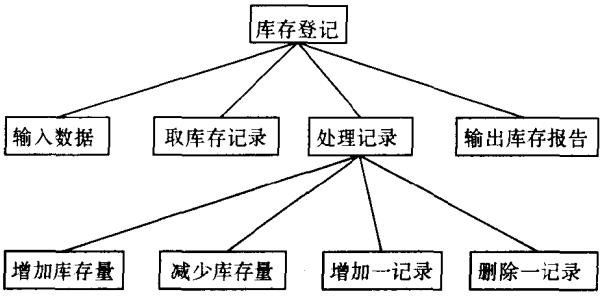

图 1.1　库存登记层次结构图

模块发展的下一阶段就是抽象数据型(abstract data type ——ADT)的出现，接着是对象概念的正式形成。先讲抽象数据型，然后再讲对象概念的正式形成。

抽象数据型是一种模块。它完全符合上列模块的三个特性，而且还具有数据型的性质，见[13]；[7]，第 4.2.2 节。型是一组实体所共有的结构或行为的性质的准确刻画。抽象数据型是程序中整型、实型及其他等原始数据的发展与扩充。原始数据型规定某一数据的格式与取值，包含了对这个数据型能进行的操作。比如，对整型数据能进行加、减、

乘、除等算术运算，对字符型数据就不能进行各种四则运算。抽象数据型将其推广，明确规定它包括数据型以及对这个型的一切正当操作等两部分，并根据信息隐藏原则，规定其中哪些能公开，哪些不能公开。用一个例子，来具体说明抽象数据型。

用一个银行存款的例子具体说明抽象数据型，并用 Ada 语言(见[14]；1，第十八章)中程序设计单元 package 来实现。package 规定其中所用的常数，型及子程序(过程或函数)。一个 package 的公开部分内载有使用该 Package 所必需的细节，其私有部分包括对用户隐藏的结构细节；Package 体则包括对说明中公开部分的如何实现内容，体本身是私有的。上述子程序是过程或函数的总称，Package 和 Object Pascal 及 C++ 等面向过程语言一样，包括说明和体两部分。说明部分规定子程序名及有关参数，体内规定该子程序的执行，体是私有的，即信息隐藏部分。由此可见，Package 完全符合实现 ADT 的要求，图1.2是这一部分程序内容，其中带下划线"-"的词都是 Ada 语言中保留字。首先列出说明部分：

```
Package Account is
type PrivItems is limited private;
type AccRec is
    record
        AccNumber   :integer
        name        :string(1..24)
        PrivPart    :PrivItems
    end record;
  procedure Deposit (amount:in float; p: in out PrivItems);
  procedure Withdraw (amount:in float; p: in out PrivItems);
  Acc - Error     : exception;
private
  type privItems is
    record
        balance     :float;
    end record;
end Account;
```

图 1.2　Package 的说明(包括公开与私有部分)

Ada 语言是从 Pascal 语言的思想演变而来，但在语法上做了较大的改变，以及在语义上做了扩充。虽然如此，这两种程序语言的基本语法与语义可说是相似的或相同的。懂得 Pascal 的人是不难看懂图 1.2 中程序段的，其中"in"及"out"两个保留字分别代表"输入"与"输出"。这里要说明的是 Ada 语言单元 package 是如何实现 ADT 的。图 1.2 中所示程序设计段是 package 的说明部分。这一部分又分为两部分：公开部分，从头到保留字"private"以前的部分；私有部分，从"private"开始直到"end"部分。公开部分中规定 AccRec 为一数据型(type)，即 record，其中有一数据项 PrivPart 是私有的，并有两个 procedure，进行涉及私有的数据项的运算。PrivPart 是规定为 PrivItems 型而成为私有的。PrivItems 作为私有的数据型，是在第一条 type 语句"type PrivItems　is　limited private" 中规定的，在这条语句中不但规定为私有(private)，而且进一步规定为加限私有(limited private)。数据型规定

为 private(私有)时,其所属数据只可由所属 package 内规定的 procedure(在本例中为 deposit 及 withdraw 两个 procedure)调用和进行运算,及对这样的数据赋值和作相等或不相等比较。进一步规定为 limited private(加限私有)时,即只能由所属 procedure 调用和运算,但不能对之赋值或作相等或不相等比较。在本例中定为加限私有,是必要的。存户余额(balance)为私有(保密)并又定其,若允许赋值或作相等或不相等比较,等于破坏保密。所有这些都是信息隐藏的体现。另外,公开部分中数据型若不规定为 private 或 limited private 时,即所谓缺省规定,视为公开的,可自由调用与运算。例如,本例中 AccRec 数据型就是公开的,其所属 record 内三个数据项中除 privPart 规定为 limited private 外,AccNumber 及 Name 等两数据项则为公开的,即可自由调用和运算。这里再重复一遍:私有的数据只有说明部分中所定义的 procedure 或 function 才能调用与运算。说明部分中最后一条语句是用 Ada 语言中保留字 exception 来定义差误标记(error flag),Acc _ error(账户差误)。它类似于语句标号,指出当差误出现时,语句转向何处。

以上所列一个 Package 的说明部分,是与用户见面的部分,包括其中明白规定的私有的部分。现在还要讲的是 Package 体,即说明部分中 procedure 或 function 是如何执行的。package 体是不与用户见面的,即私有的。下面是本例的 package 的体,如图 1.3 所示。

```
package body Account is
    procedure Deposit(amount :in float P: in out PrivItems);
    begin
        P.balance := P.balance + amount;
        put ("new balance = "); put (P.balance);
    end deposit
    procedure Withdraw(amount :in float; P:in out PrivItems);
    begin
        if a mount > P.balance
        then raise Acc - error
    end if;
    put("new balance = ");   put(P.balance);
    exception when Acc_error = = > put ("not enough fund in this account");
    end Withdraw;
```

图 1.3 Package 体——私有部分

综观图 1.2 及 1.3 中以 Ada 程序设计单元 package 来实现的 ADT,已完全体现了这个 ADT 的重要特性,即 1)其中包括有数据及过程(处理)等两部分内容;2)在明确规定的界限内,分公开部分,作为接口,与用户见面,以及私有部分,不与用户见面;3)从而实现了信息隐藏原则。这就是随后形成的面向对象程序设计中对象概念的基本内容。可见,Ada 语言时代对象概念的实质已经出现并存在了,不过尚未作为程序设计中正式概念。这时期正处于结构化方法论的成熟发展阶段。最早的面向对象程序语言 SIMULA67 已经出现和存在,见[15,1, 第 20.1 节],但在当时却未能掀起面向对象程序设计的新潮。1976 年出版的 N. Wirth 写的一部讲结构化程序设计的名著, Algorithm + Data Structure =

Programs的标题所言。其实,把这个书名等号右边的“programs”改为“Objects”(对象),即“Algorithms + Data Structures = Objcets”,它就是对象的简化且通俗的定义。但当时的主流是结构化的程序设计。结构化的程序设计是用程序模块定义处理过程。而使用数据结构,使该过程运行得更有效。程序模块发展到ADT阶段时,其内容即从处理过程扩大到包括处理过程与数据两部分。上面已讲过如何用Ada语言中程序设计单元Package来具体实现ADT。Ada是在美国国防部的要求与支持下,由一组程序设计人员研制而成的。从1975年开始研制,直到1983年才有Ada的ANSI(american national standards institute)标准文本版的发表。这个语言吸取了Algol 60的控制结构以及Algol 60和Pascal的数据结构技巧,可说是集当时几种程序语言的优点而发展起来的结构化程序设计语言,适于编写大而复杂的程序(参阅[8]中第12章所讲交通管理系统例子)。但Ada仍属面向功能(过程)语言,虽能实现ADT,还不能算是面向对象语言。到Smalltalk-76程序语言出现后,面向对象程序设计(Object-Oriented Programming,简称OOP)概念才得到正式承认。到了20世纪80年代下半叶,从C语言产生出C++[3]和objective-C[16]等两种自称为面向对象语言,从Pascal语言衍生出来Apple的Object Pascal[17]和Object-Oriented Turbo Pascal[4]。它们也叫混种的(hybrid)面向对象语言,这几种语言都在C和Pascal原有语法和语义功能的基础上而发展出来的,应称为混种的面向对象语言。纯面向对象语言除Smalltalk外,尚有Eiffel语言[1](其编译输出为C程序)。

一个程序语言至少具有下列三个基本特征时,才能称为面向对象程序设计语言:

1. 用对象而非过程(功能或算法)作为程序设计的基本的逻辑构件。

2. 每个对象属于一个类(型),并为该类的一个实例(instance)。

3. 一个类可继承其他类的性质。

一个程序若缺乏这三者中任一项,都不能称作面向对象程序。有一种分法是,把不具备其中继承性的语言叫作基于对象的(object-based)语言,而不叫做面向对象语言。如此Smalltalk,C++,Objective-C,Eiffel,Apple ObjectPascal,以及Object-Oriented TurboPascal都属于面向对象语言,而Ada则不属于对象语言。实际上,Ada只符合上列第1)条件,因其中对象(ADT)不属于一个类。

从程序模块的发展这个线索来看,从ADT发展到对象,在基本内容方面完全一样,都包括数据和处理过程两部分。但对象作为程序模块,还同时具有下列意义与作用:

1. 对象是面向对象程序设计的基本逻辑构件,整个程序是由不同的对象所组成的。这是与结构化程序设计的主要不同之处,后者的基本逻辑构件是过程模块。实际上,对象本身就是一个程序或子程序,整个面向对象程序就是这些子程序所组成的。

2. 每个对象各有专责,掌管问题领域中一部分数据以及对这些数据的处理过程。这就是我们一再反复讲的对象包括数据和处理过程两部分内容。

3. 各对象相互协作,分工执行各自在系统(程序)中负责应完成的任务。当一对象要进行不属于其掌管范围内的操作时,它可向其他具有此职能的对象传送信息,请求其进行该项操作。这就好像在建筑行业中木工和瓦工各有专责。木工不能做的,请瓦工做;瓦工不能做的,请木工做。在面向过程语言中,这种信息传送就是我们通常熟悉的过程或函数调用(invocation)。但这里有一重要的不同之处:在面向过程语言中是采用静态绑定(static

binding),即在编译时就把被调用的过程或函数的地址确定下来;而在面向对象语言中,是采用动态绑定(dynamic binding),即要等到运行时才确定被调用的过程或函数(在面向对象语言中总称之为方法——methods)的地址。

这里仍宜用程序来具体说明对象是如何实现的。以上说明 ADT 的实现,是用银行存款的例子。这里仍以银行的同一例子,分别用面向对象 TurboPascal 语言和 C++ 语言中对象程序模块来实现。

图 1.4 中所示面向对象 Turbo Pascal 程序段与 Turbo Pascal 语言中程序构件 unit 的结构相同,即分为接口(interface)与实现(implementation)两部分。接口部分是公开的,与用户

```
Type
  Account = object
      accNumber      : integer;
      Name           : string[20];
      balance          : real;
      Constructor   Init;
      Procedure   deposit(var amount : real);
      Precedure   withdraw(var amount : real);
      Procedure   check(var amont : real);
      Procedure   putItems;
      Procedure   putBal;
      Function    getAccno : integer;
      Function    getName : string ;
      Function    getBal : real;
  End;
Type
  Savings = object(Account)
  End;
Type
  Checking = object (Account);
  End;
    {以上为说明部分,以下为实现部分}
  Constructor Account.Init;
    Begin
      self.putItems;
    End;
  Pocedure Account.deposit(var amount : real);
    Begin
      balance := balance + amount;
    End;
  Procedure Account.withdraw(var amount : real);
    Begin
      If   amount > balance Then
```

```
        Begin
          writeln('Not enough fund in this account');
        End
      Else
        balance := balance - amount;
        writeln('new balance = ', balance: 6:2);
    End
  Procedure Account .check(var amount : real);
    Begin
      self.withdraw(amount);
    End;
  Procedure account.putItems;
    Begin
      write('Enter account number : ')
      readln(accNumber);
      write('Enter the name : ');
      readln(Name);
    End;
  Procedure Account.putBal
    Begin
      write('Balance = ');
      readln(Balance);
    End;
  Function Account.getAccNo : integer;
    Begin
      getAccno := accNumber;
    End;
  Function Account.getName : string;
    Begin
      getName := Name;
    End;
  Function Account.getBal :real;
    Begin
      getBal := balance;
    End;
      {以下为主程序部分}
Begin
end.
```

图 1.4　面向对象 Turbo Pascal 程序段——银行存款例

见面;实现部分是私有的,不需与用户见面。在图 1.4 中从头开始的三个 Type 的定义是公开部分,相当于 Ada 语言中 package 内说明(规定)部分。随后从过程"Constructor