面向对象程序设计

面向对象程序设计

江明德　编著

电子工业出版社

(京)新登字 055 号

内容提要

本书介绍面向对象程序设计和 C++语言入门。在假定读者已熟识 C 语言的前提下，内容安排如下：一、面向对象程序设计的基本概念和论点(第一章)；二、C++语言的纲领性论述(第二至三章)；三、面向对象程序设计技术和 C++的各性能(第四至七章)；四、面向对象程序设计的设计方法(第八章)；此外还包括：五、作为进一步读物的各课题，其中有面向对象程序语言的现状和发展，用 Smalltalk 实行面向对象程序设计，面向对象程序设计范型与自动机理论，面向对象程序设计的思维科学基础(附录Ⅰ至Ⅳ)。

本书适合于计算机专业的本科高年级学生，低年级研究生作教材或参考书，也可以作为有意欲了解面向对象程序设计技术和 C++语言的计算机工程技术人员自学的教材或参考书。

面向对象程序设计

江明德 编著

责任编辑 王昌铭

特邀编辑 尚世清

电子工业出版社出版(北京市万寿路)

电子工业出版社发行 各地新华书店经销

顺义县天竺颖华印刷厂印刷

开本：787×1092 毫米 1/16 印张：21 字数：518 千字

1994 年 1 月第 1 版 1994 年 1 月第 1 次印刷

印数：3000 册 定价：9.70 元

ISBN7-5053-2239-7/TP·605

出 版 说 明

根据国务院关于高等学校教材工作的规定，我部承担了全国高等学校和中等专业学校工科电子类专业教材的编审、出版的组织工作。由于各有关院校及参与编审工作的广大教师共同努力，有关出版社的紧密配合，从1978～1990年，已编审、出版了三个轮次教材，及时供给高等学校和中等专业学校教学使用。

为了使工科电子类专业教材能更好地适应“三个面向”的需要，贯彻国家教委《高等教育“八五”期间教材建设规划纲要》的精神，“以全面提高教材质量水平为中心，保证重点教材，保持教材相对稳定，适当扩大教材品种，逐步完善教材配套”，作为“八五”期间工科电子类专业教材建设工作的指导思想，组织我部所属的八个高等学校教材编审委员会和四个中等专业学校专业教学指导委员会，在总结前三轮教材工作的基础上，根据教育形势的发展和教学改革的需要，制订了1991～1995年的“八五”（第四轮）教材编审出版规划。列入规划的，以主要专业主干课程教材及其辅助教材为主的教材约300余种。这批教材的评选推荐和编审工作，由各编委会或教学指导委员会组织进行。

这批教材的书稿，其一是从通过教学实践、师生反应较好的讲义中经院校推荐，由编审委员会（小组）评选择优产生出来的，其二是在认真遴选主编人的条件下进行约编的。广大编审者、各编审委员会（小组）、教学指导委员会和有关出版社，为保证教材的出版和提高教材的质量，作出了不懈的努力。

限于水平和经验，这批教材的编审、出版工作还可能有缺点和不足之处，希望使用教材的单位，广大教师和同学积极提出批评和建议，共同为不断提高工科电子类专业教材的质量而努力。

电子工业部教材办公室

前　言

本教材系按电子工业部的工科电子类专业教材 1991～1995 年编审出版规则，由计算机教材编审委员会征稿并推荐出版。责任编委为王能斌教授。

本教材由北京航空航天大学麦中凡教授担任主审。

近年来流行的“面向对象”一词，在软件工程上补充了、或许甚至会代替“结构化，而作为“良好”的高技术的一种说法。面向对象问题求解法和面向对象程序设计是计算机程序设计上一种新颖的思维方式和程序设计方法，它们与通常的过程式程序设计语言所支持的思维方式和程序设计方法很不相同。面向对象技术因其思想新颖且能满足软件工程各项原则，又在人工智能、数据库、人机界面等各别研究、开发领域里取得成功，看来它是这些领域的交汇点，并通向软件设计、实现和运行的一条颇有成功希望的途径。

面向对象程序设计语言的各种强有力性能支持面向对象的诸概念，后者使得计算机问题求解更象是人的活动。用 C++这样的程序设计语言，人们一方面能充分利用硬件的能力，另一方面以低的代价提供更强有力的软件开发工具。C++是一种混合型的语言，它把面向对象的功能与传统的而有效的过程式语言 C 融合在一起。C++向程序员提供面向对象的能力而不损害运行时间或存储效率；在普通的硬件上能产生优质的代码产品。

本书介绍面向对象程序设计和 C++语言入门。在假定读者已熟识 C 语言的前提下，内容安排如下：

一、面向对象程序设计的基本概念和论点（第一章）；

二、C++语言的纲领性论述（第二至三章）；

三、面向对象程序设计的技术和 C++的各性能（第四至七章）；

四、面向对象程序设计的设计方法（第八章）；此外，还包括：

五、作为进一步读物的各课题（附录Ⅰ至Ⅳ）。

本书适合于计算机专业的本科高年级学生、低年级研究生作教材或参考书。四则附录主要是为研究生安排的。根据授课对象、课时多少（40～60 课时），在第一至六章的基础上，教师可增删第七章、第八章、各附录来选择授课内容。本书也可作为有意欲了解面向对象程序设计技术和 C++语言的计算机工程技术人员自学的教材或参考书。

本书承蒙北京航空航天大学计算机系麦中凡教授审阅，东南大学计算机系王能斌教授关注本书的出版，他们都对书稿提出了宝贵的意见，书稿在修订中采纳了这些意见。在此一并向他们表示衷心的、诚挚的谢意。

敬请专家和读者批评指正。

江明德
电子科技大学计算机系
成都 610054

目　录

第一章　面向对象程序设计的基本概念和论点

本章介绍与面向对象程序设计这一新型的振奋人心的体裁有关的一些基本概念和论点。

1.1　导论

面向对象程序设计是一种比较新的设计和实现软件系统的方法。它的主要目标是：

●用增加软件可扩充性和可再用性的手段来改善程序员的生产率，

●控制软件维护的复杂度和费用。

所谓可扩充性指的是“软件产品能适应于规格(说明)变动的难易程度”。所谓可再用性指的是“软件产品能在新的应用中被整个地或部分地予以重新利用”的一种性能。维护是“使程序处于良好工作状态的活动，包括检查、测试、调整、更换、修理等”。

当使用面向对象程序设计时，软件开发的设计阶段是与实现阶段相紧密地联系着的。正如许多的软件设计和实现途径一样，面向对象这一方法的潜力和前程可能会超过实际上所能达到的。虽然在各个孤立的应用领域里已有面向对象程序设计取得巨大成就的报导，在软件设计和开发的一般竞争场所，评估面向对象方法学的相对成功为期尚早。纵然如此，面向对象技术的地位在编程、DBMS(数据库管理系统)分析/设计工具等领域中日趋提高，正被认为是九十年代最重要的软件技术。据有关方面对美国 1600 家单位进行调查，在导入面向对象技术方面，已经导入的单位占 21.0%，正在试验性导入的单位占 26.8%，没有打算导入的单位占 52.2%。

本书的目的在于：

●探究软件开发中这一振奋人心的问题求解的新途径——面向对象程序设计；

●引导入 C++这一新的重要的程序设计语言的门，用它来阐明面向对象程序设计。

在各种面向对象程序设计语言中，特别是在 Smalltalk 和 C++这两种当前较广为流传的面向对象程序设计语言中，我们选择了 C++语言。之所以这样选择，主要考虑到：它的表达力较强，可移植性好；与程序员的传统基础接近(虽然在 OOP(面向对象程序设计的简称)的纯度方面稍逊于 Smalltalk)；语言及其环境规模不大，较易掌握使用(虽然就语言本身的学习难度方面，稍逊于 Smalltalk)；适合开发大型程序。

面向对象程序设计环绕着几个主要的概念：抽象数据类型和类、类型层次结构(子类)、继承及多态性。本章里介绍这些基本概念。

抽象数据类型是面向对象程序设计的核心。抽象数据类型是一种模型，它包含一类型与一组相关的操作。这些操作是为“作为基础的该类型的行为”而定义的，并且刻划该行为。

在大多数面向对象程序设计语言中，用定义“对所有这些操作(它们能施行于作为基础的该类型上)的接口”的办法，类定义借此描述作为基础的抽象数据类型的行为。类定义也规定该类型的各实现细节或数据结构。通常，这些实现细节只在该类的作用域内才是可存取的。我们称这样的类型为私有类型。当数据类型的全部或一部分在该类的作用域之外是可存取的时候，我们就称该类型的这些部分为公开的。

在该类型之上定义的那些操作一般地也分成公开的或私有的。在该类的作用域之外可存

那些操作是公开的操作。私有(私用)的操作只在该类的作用域内才是可存取的。用面向对象的说法,为一个类而定义的那些操作称为方法。这些方法类似于非面向对象语言中的过程和函数。如果一个类的各公开操作在很多应用场合是足够普遍地可适用的,那么,该类就可构成一可再用软件成分的基础。

对象是一变量,该变量被说明成是一特定类的变量。这样的一个对象用包含"定义在该类定义中的(私有的和公开的)数据所有各域的复件(拷贝)"的办法,把"状态"封装起来。调用定义在该类定义中的一个或多个方法,可施行动作在此对象上。调用方法的过程称为"发送消息给此对象"。这样的消息一般地含有参数,正如在非面对象语言中过程或函数调用那样。方法的调用(发送消息给一对象)一般地要修改存储在该特定对象中的数据。

每一类变量或对象表示该类的一个实例。如果几个对象均被定义成是同一类的,那么,一般地它们所含的值组是互不相同的。

面向对象语言是可扩充的,这是因为:程序员能够创建一些新的类型,它们可赋以各特殊性质并且它们的行为被刻划在类定义中。象对待程序设计语言本身所提供的各预定义类型那样,用几乎同样的方式就能处理这些新类的各对象。

通过各子类,面向对象的"范型"提供各类型所形成的层次结构。子类定义刻划一个对象集合的行为,这些对象继承父类的某些特征,而又获得不为该父类所共享的一些特殊化的特征。容许创建各子类可以降低软件开发的费用和复杂度。

各子类能导致渐增式的问题求解。不修改,或更坏的情况,重写现有的各软件成分(假定这些成分的源代码是可利用的),一些新的子类就能从一个基准类集合创建出来。这些新子类的各对象形成软件总体结构的基层结构。

父类里定义的数据类型和操作,子类可拿来袭用。此外,子类还可以增加自己的数据类型和操作。这就是面向对象软件系统中的(类)继承(机制)。

1.2 面向对象程序设计范型

本节讨论程序设计范型这一概念,并据此对现有的程序设计语言进行分类。为了要了解 OOP 范型,论述面向对象的问题求解方法,而后再明确地定义 OOP 范型。

1.2.1 程序设计范型

程序设计范型(paradigm)这一译名含义比较笼统,具体地指的是程序设计的体裁,正如文学上有小说、散文、诗歌等体裁,程序设计体裁是用程序设计语言表达各种概念和各种结构的一套设施。它主要地是对于程序设计这一抽象级上来说的,其次才对于程序运行这一动态的具体级上来说。由此,当今的程序设计范型就分成:过程(式)程序设计范型、函数(式)程序设计范型、面向约束(逻辑)程序设计范型、面向对象程序设计范型,还有进程(式)程序设计范型、类型系统程序设计范型和事件程序设计范型。每一程序设计范型可有多种程序设计语言,例如,Pascal 和 C 均体现过程式程序设计范型,用来进行过程式程序设计。

●过程式程序设计范型:程序设计归结为选定数据结构、设计算法过程或函数。程序执行被看作各过程调用的一(偏序)序列,以此处理数据结构。此范型最为普通,它被诸如 Algol、Pascal 和 C 语言所支持。

●函数程序设计范型:程序被看作"描述输入与输出之间的关系"的一个数学函数。在函数程序

设计范型中，完全消除状态或变量这一概念。也就是说，函数程序设计范型是无变量的程序设计。它被诸如 FP 和 Lisp 语言所支持。

●面向约束(逻辑)程序设计范型：程序被看作“描述输入与输出之间的各关系”的一组方程。具体些说，程序设计归结为列举事实、定义逻辑关系(规则)、以提问方式求得(或证实)解。象在函数程序设计范型中一样，面向约束程序设计范型中也消除状态这一概念。Prolog 语言是支持面向约束程序设计范型的最主要的一种语言。

有一种观点：把进程式程序设计范型从属于面向对象范型。类型系统程序设计型是与面向对象范型有关的。至于事件程序设计范型，则与 Petri 网模型等有关。这些均不细述。

可以有兼备两种或多种范型的程序设计语言。即混合范型语言，例如，C++不是纯粹的面向对象范型的，而是过程与面向对象混合范型的语言。实质上，Lisp 并非是纯粹的函数程序设计范型的，而是过程与函数混合范型的。

1.2.2 面向对象问题求解

面向对象的软件系统的体系结构是围绕着一个类的集合而构造起来的，这些类刻划该系统中所有作为基础的数据的行为。出自每一类的各对象用调用该类的各方法来加以处理；也就是，发送消息给这些对象。这些消息表示在该对象集合上所要采取的各种动作。

面向对象程序设计对准焦点在所要处理的数据上，而不是在作此处理的各过程上。这些数据形成软件分解的基础。实际上，面向对象软件设计的主要挑战是，把一软件系统分解成各个作为基础的数据类型或者“各个类与各个子类”。并且定义每一个这些基础类与子类的性质。各对象或各类(各子类)变量对应于真实问题空间中的各物理实体或各逻辑实体。

“定义一面向对象软件系统并形成该系统的高级设计”的总体结构框架仅仅展现一(子)类集合及它们的定义和各个对象。每一类行为用各方法接口来刻划。各方法的实现细节不是该系统的高级设计的部分。

在一典型的面向对象语言中，定义在一类里各方法的接口能够与这些方法的实现细节分立地予以规定，从而容许该系统的设计与它的实现分离开来。一个概念(带有各方法接口的类定义)与它的实现(实现该类的数据结构和各算法的代码)之间的这种分离，在面向对象程序设计中以及在达成可再用性和维护费用的控制方面，是十分重要的。

在面向对象程序设计中可再用性得以提高是由于：在一类的各方法接口中提供了封装在该类中的各概念。用户只需要了解规定在各方法接口中的该类各对象的行为，而无需关心它们的实现。从用户的观点来看，这些方法实现包含在一个“黑匣子”里，是隐蔽得看不见的。

在 OOP 方法中可维护性得以提高是由于：数据结构或算法的实现(即，类实现内部的代码)上的变动能被限制在“实现该类或该类的一部分”的代码区域内。由于保持了类的接口，在该类之外的作用域内不致感生有害的效应。此接口形成依照各动作(从该类的作用域外部所能施行于该类各对象上的那些动作)来“使用”该类的基础。

面向对象软件开发的主要目标是：

●使用“呈各基准类的形态”的那些可再用软件成分，并且应用“利用各子类”的渐增问题求解法，来缩短开发时间并降低开发费用。

●通过“把变动局限于一个或多个类的实现范围内”的能力，降低软件维护的费用。

面向对象系统的可靠性能得以提高是由于在初始设计中所造成的高级整体化。组成该系统的各主要部分从一开始就配置成一定的构形并配合在一起。每一主要部分被它的各抽象性质所定义。

在作成或实现许多低级细节之前，可进行高级整体化测试。这就有助于可靠性的改善。

面向对象程序设计提供用于快速原型化的有用平台(工作场地)。在完成"一系统高级分解成各类和(出自这些类的)各对象"之后，为了要察 看该系统的这些"部分"配合得如何，能用简单而效率不高的代码快速地实现"刻划该系统的行为"的许多重要方法。往后再完善实现的细节。

1.2.3 什么是面向对象程序设计范型

面向对象程序设计范型：就程序而论，它是一个类的集合和各类之间以继承关系联系起来的结构，再加上一个主程序，其中定义各对象并规定它们之间传递消息的规律。就程序执行而论，它归结为各对象和它们之间以消息传递的方式进行着的通讯。

OOP 程序执行过程中，由主程序所规定的各对象的初始状态出发，它们通过消息传递进行通讯，从而对象的状态受到改变而达到新状态。由此，不断地进行状态的变换，达到的最终状态，即为计算结果。在对象的状态变换过程中，可能有新的对象产生，参加状态变换。

OOP 的最主要的特征是(各对象之间的)消息传递和(各类之间的)继承。

注意，为了叙述方便，在上下文自明的情况下，我们常把"程序设计"与"程序设计范型"两词交互通用。

1.3 类、对象和封装

本节中较为精细地考察类与对象以及它们的封装。

类描述涉及定义所有那些性质和性能，它们刻划"作为此类的一实例"的任一对象的行为。

类定义的私有段：

●通常定义作为基础的数据类型的(各)数据结构。

●规定对那些方法的接口，它们仅仅在该类的作用域内是可存取的。这些方法经常用来支持各公开方法的实现。

类的私有段有时可含有"出自别的一些类"的各对象，这些对象只能在该给定类的作用域内予以处理。

类的公开段通常规定对那些方法的接口，它们形成"该类的跨越很多应用领域的可再用性"的基础。用发送消息给该给定类的各对象的办法，能够从该类的作用域之外调用这些方法。

封装是用来定义各个软件对象的一种技术。封装定义：

1. 一清晰的界限，它围绕所有这些对象的内部软件的作用域。

2. 一接口，它描述此对象如何与别的对象相互作用。

3. 一受保护的内部实现，它给出该软件对象所提供的功能的细节。在定义该对象的那个类的作用域之外，这些实现细节是不能存取的。

封装这个概念本来是与类描述有关的；但是，它也提供关于一问题解法的各个不同成分是如何分组的。这样，封装的单元是对象，它具有由它的类描述所描述的那些性质。这些性质被这同一个类的别的对象所共享。封装作为对象比"类表示封装"的说法更为具体、明确。用封装的这一定义，类的每一实例就是在一问题解法中的一个分立的封装或成分。

下例引入类、对象和封装。简要地探讨二叉搜索树的抽象。(第四章里将详尽地探讨搜索树抽象。)

在最高级上，为了表示二叉搜索树，识别出两个必要的对象。这两对象的第一个被表示成"称之

为 tree 的类"的实例。因为二叉搜索树由各个结点组成，所以，第二种对象用"称之为 treenode 的类"的实例来表示。

非形式地刻划类 treenode 如下：

■类 treenode 的各实例是"形成一个二叉搜索树"的各对象。

■每个 treenode 含有一个对象，此对象用来确定结构中数据的顺序关系。

■一个二叉 treenode 含有"指向左子树和右子树"的两对象。

下列各方法用来定义"能施行于类 treenode 的各对象之上"的那些动作：

●new_node　　创建类 treenode 的一新实例

●key　　返回"存储在给定结点中的数据"的域，此域确定树的顺序。

●left　　返回给定结点的左子结点

●right　　返回给定结点的右子结点

图 1.1 描绘类 treenode.

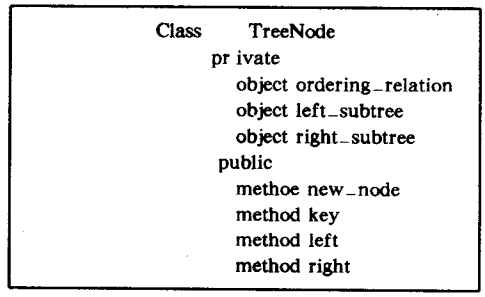

图 1.1　类 treenode 的描绘

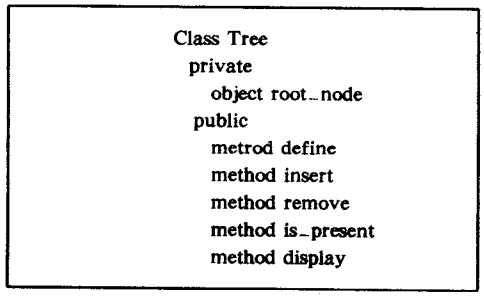

图 1.2　类 tree 的描绘

非形式地刻划类 tree 如下：

■类 treer 和各实例是一些二叉搜索树。

■一个二叉搜索树由各 treenodes 组成，其中有一特殊结点叫做根结点。对树对象的一切存取都是通过它的根结点的。

■在一个二叉搜索树中的各结点相对于某一关键对象予以排序；对于任一结点，较小的关键值在左子树中，而较大的关键值在右子树中。

■在一个二叉搜索树中，插入和删除各结点必定产生一对象，后者也是一个二叉搜索树。

下列各方法用来定义"能施行于类 rtree 的各对象之上"的那些动作：

●define　　创建一新树

●insert　　插入一新结点到该树中

●remove	从该树中移去一现有结点
●is_present	确定一特定结点是否在该树中
●display	输出该树中各结点的一个有序集

图 1.2 描绘类 tree。

1.4 子类——继承和多态性

如象前面各节中所看到的那样，对问题求解的面向对象程序设计范型用对象作为封装，而把对象定义成类的实例。各类通过类描述提供那些对象的各性质。

在本节中引入类层次结构这一概念。在这样的层次结构中，一些类从属于另一些类，称为子类，或者，象在 C++中那样，称为派生类。在层次结构中，各子类被认为是一个类的特殊情形，这些子类就在该类之下分组。在类层次结构中各较低层次通常表示增强了的特殊化，各较高层次通常表示更大的普遍化。

在大多数的面向对象程序设计语言中，如果类 P 是类 S 的一父，那么，任何“能使用类 P 的一对象 p”的地方，也能使用子类 S 的一对象 s。这就蕴含着：能够发送给类 P 和类 S 的各对象以一组公共的消息(即，操作)。当同一消息能被发送给类 P 的各对象和类 S 的各对象的时候，这就定义为“多态性”。

多态性容许每一对象以适合于那个子类(该对象取自这个子类)的方式，来对一公共消息格式起反应。例如，方法 print 可被定义成用来输出“刻划对象状态的那些数据”的某些域。多态性容许这同一个消息 print 发送给一类及其各子类内的所有各对象，使得每一对象知道如何对此消息起反应，可能以一种不同于“出自别的各子类的各对象起反应”的方式来反应。例如，对于出自每一不同子类的对象，可以输出不同的数据域。“对于施行于不同种类的对象上的一个相似的操作，使用同一个消息”这一能力是与人们在求解问题时的思维方式相符合的。为打印整数、浮点数、字符、字符串及数据记录而使用不同的术语，是不自然的。多态性是问题求解的面向对象范型中的一种关键性能。

对于“各对象能在多大程度上继承、扩充或抑制其父类的诸特征”的能力，各种不同的面向对象程序设计语言提供此能力的一个范围。有些面向对象语言支持多(重)继承，其中一子类有多于一个父类。第五章焦聚在 C++中的继承和子类上，第六章焦聚在多态性上。

1.5 面向对象程序设计中的一些特殊要求

面向对象程序设计中的一些特殊要求是：划分软件成各个类，添加功能到现有的软件系统上，构造各类型与各子类型的合适的一个层次结构。

1.5.1 划分软件成各类

对于面向对象程序员新手来说，把一种手续(过程)模拟成一个类，似乎是不自然的。例如，在编写一登记本数据库程序时，该程序存储、修改和删除学生的名字和成绩，并且打印出这些名字和成绩的报告，典型的做法是：定义一数据类型 rollbook(登记本)，接着定义各方法 insert、delete、modify 及 print 作为与基础的数据类型 rollbook 相关联的各操作。

insert、delete 和 modify 这些操作与称之为 edit(编辑)的手续相关联可能比“与称之为 rollbook 的类型相关联”要更为合适一些。这样,可以把手续 edit 说明成一类。这个 edit 类就能在别的应用(那里要求联线交互编辑数据库信息)场合中再用。类 rollbook 可含有类 edit 的一个对象来实行插入、删除和修改名字和成绩。

1.5.2 添加功能到现有的软件系统上

在面向对象程序里每一函数或过程均与一个类相关联。当欲添加新的函数到一现有的系统上的时候,设计人员或程序员必须判断“把此特定过程添加到一现有类中去,或创建一新的类,或创建一子类。”

再用性这一问题是“确定应该在哪里把新的功能添加 到面向对象系统上”的主要因素。如果该添加的功能能够被几个现有类的各对象所再用,那么,根据该函数或过程,创建一个新类可能是合理的。如果该新函数的实现需要存取一现有类型的内部细节,那么,把该新函数作为一补充的方法添加到现有类中,可能是合理的。如果该新函数表示“对一给定类中的一现有函数(方法)的一种修改”,那么,创建一子类,并且将该函数添加到此子类中作为一方法(该方法抑制其它父类的那个类似的方法),可能是合理的。

1.5.3 各类型和各子类型的层次结构

创建各类和各子类的一种方法是自顶向下数据类型分解,其中程序员识别并模拟系统中的各主要数据成分作为各类。各顶层数据成分(各类)被划分成更为特殊化的各子类,此一过程继续下去直到构造出一各类和各对象的层次结构。

作为例子,考虑一汽车作为系统中的一个主要类。类 automobile(汽车)能够被划分成各子类 engine(发动机)、transmission (变速器)、brakes(制动器)、drive_train(传动装置)、exhaust_system (排气系统)、suspension(支承系统)如此等等。子类 eangine 象别的子类一样,能被进一步划分成诸如 ignition(发火装置)、fuel_injecton(喷油器)及 starter(起动器)各成分。子类 igintion 能被进一步划分成 spark_plugs(火花塞)和 solenoid(线圈)。对于这些类和子类中的每一个类,必需定义一组相应的操作(方法)。图 1.3 上描绘一汽车的类分解的一部分。

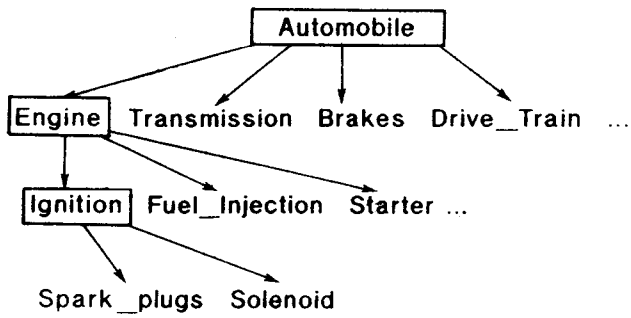

图 1.3 汽车的类分解的一部分

用了自顶向下的数据类型分解,在层次结构的顶部的那些类通过各方法通常概括“为层次结构的所有较低的各段所共有的诸特征”。

创建各类和各子类的另一种方法是自底向上分解,其中各子类扩充一父类,并且给每一相继子

类和各对象提供更多的功能。

自底向上分解的一个常用的例是类 screen_window(屏幕窗口)作为父类。这样的类概括所有屏幕窗口的各最小特征。能构造出一系列的子类,它们添加更多的功能到基本类 screen_window 中,各不同子类定义诸如前景和背景颜色、窗口边界的参数、等等。图 1.4 上描绘 screen_window 的类分解的一部分。

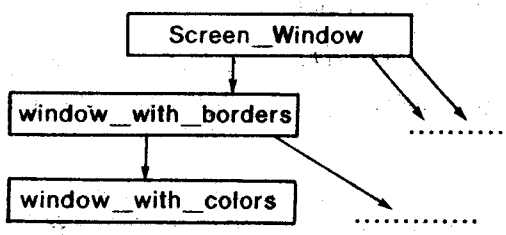

图 1.4 屏幕窗口的类分解的一部分

习 题

1.1 试将你自己选定的一软件开发问题划分成(设计的最高级上的)各类、各子类、各对象及各方法。

1.2 在一面向对象系统中对象是什么?

1.3 什么是类?

1.4 什么是消息?

1.5 什么是方法?

1.6 试描述面向对象程序设计的各主要特征。