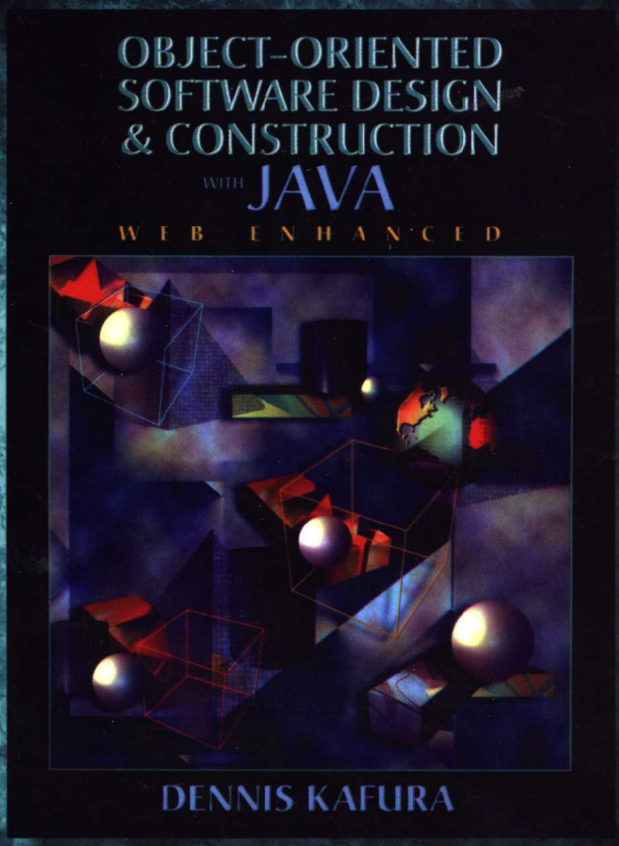
计 算 机 科 学 丛 书

Prentice Hall

Java面向对象程序设计教程

(美) Dennis Kafura 著　　袁晓华　石耀斌　等译



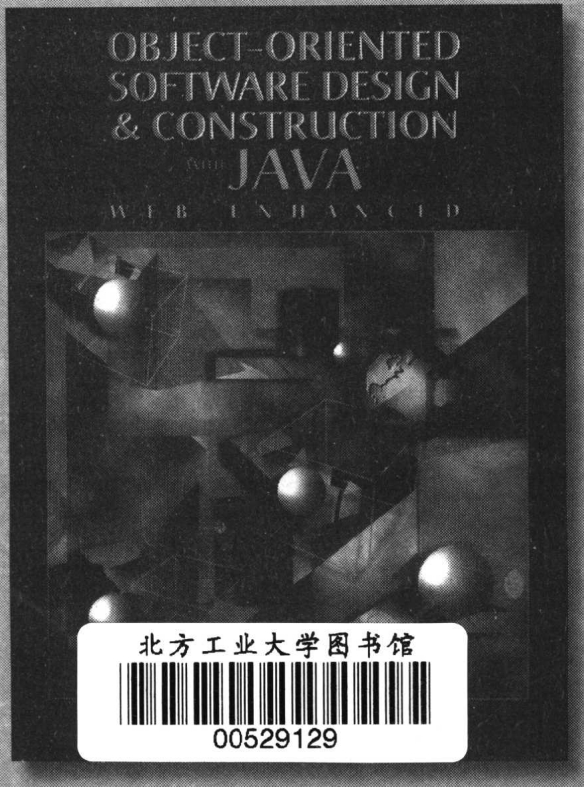
Object-Oriented Software Design and Construction with Java

机械工业出版社
China Machine Press

计 算 机 科 学 丛

Java面向对象程序设计教程

(美) Dennis Kafura 著　　袁晓华　石耀斌　等译



Object-Oriented Software Design and Construction with Java

机械工业出版社
China Machine Press

本书综合介绍了面向对象编程的各种概念、软件工程中的问题以及Java语言的特征。本书内容全面，习题丰富。更有特色的是在全书中使用了两个反复出现的例子：图形用户界面和生态仿真系统。随着讲解内容的不断深入，逐步增加了这两个例子的功能并扩展其应用。这种循序渐进，并结合具体实例的编排方式有助于读者更好地理解面向对象编程的方法和技术。

本书适合作为计算机及相关专业本科教材，也可供各类面向对象程序设计开发人员自学和参考。

本书版权登记号：图字：01-2001-3764

图书在版编目（CIP）数据

Java面向对象程序设计教程/（美）卡夫拉（Kafura, D.）著；袁晓华等译. -北京：机械工业出版社，2003.2
（计算机科学丛书）
书名原文：Object-Oriented Software Design & Construction with Java
ISBN 7-111-11581-3

I. J… II.①卡… ②袁… III. Java语言-程序设计-教材… IV. TP312

中国版本图书馆CIP数据核字（2003）第006338号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）
责任编辑：周 睿
北京昌平奔腾印刷厂印刷 · 新华书店北京发行所发行
2003年3月第1版第1次印刷
787mm × 1092mm 1/16 · 32印张
印数：0 001-5000册
定价：49.00元

译 者 序

本书详细地介绍了面向对象编程的基本概念，包括抽象、类、对象、合成、关联、聚集、概化、多态性、继承、重载、覆盖等。提出了一些好的面向对象设计的基本原则，比如一个好的抽象或者一个好的类应该具有的特征等。在描述了这些基本概念的基础上，按顺序逐步介绍了如何利用单个的对象、如何使用不同类的对象、如何实现一个新类以及如何产生一个面向对象的系统。

贯穿全书反复使用了来自两个应用领域中的例子：即图形用户界面和生态仿真系统。按照内容的不断深入，逐步增加和完善这两个例子的功能，使其更接近于实际的开发环境。通过一步一步的指导，使读者熟练掌握利用常见的工具和技术来构造复杂的系统。并且利用了专门的一章来详细介绍用Java中的Swing组件来构造一个用户界面。这样做不仅使读者可以更好地理解面向对象程序设计的实际方法和步骤，同时对于其中的代码只要做一些适当的修改就可以直接应用到实际的开发中。

书中还讨论了各种好的设计策略和软件工程问题，并利用了统一建模语言（UML）来描述各种设计问题和软件工程问题，使读者可以更好地理解和运用UML的技术和方法。书中还较详细地介绍了Java语言的语法和细节，给出的例子代码均是基于Java 2的。本书的最后两章介绍了Java中的输入输出以及并发开发中的线程。

本书英文版在美国是大学本科的教材，概念清晰、讲解透彻，每章，甚至每节都配有思考题和习题，对理解和消化书中的概念极有帮助，特别适合学生和初学者自学参考。

本书的第1、2章由袁晓华翻译，第3、4章由石耀斌翻译，第5、6章由陈虹翻译，第7章由李晓翻译，第8、9章由张睿翻译，全书由袁晓华组织并负责审稿。吴辉、袁源、张斌、冯文武、周涛、黄晓军等人做了部分工作。

由于译者水平有限，翻译中的不妥之处，敬请读者不吝批评指正。

译 者

2002年11月1日于北京

前　言

读者

本书的读者对象是大学的本科生。假定读者已经预先学过一门编程语言的课程。弗吉尼亚技术学院的学生在二年级的第一学期使用过本书的讲义稿。特别要指出的是，阅读本书只需要理解基本的数据结构（如链表、堆栈等），并且只在课程的下半部分会用到。不需要上专门的数据结构课程。

本书重点讲述有关Java中面向对象的编程方法。假定读者对基本的编程结构（判断、迭代等）的语法知识已有一个大致的了解，或者至少可以独立地阅读本书。由于Java的许多语法都来自于C语言，因此最好适当地了解一下C语言中的这些编程结构。阅读本书只要求熟悉C语言的基本知识，并不需要是C语言的专家。如果掌握其他静态类型的过程语言（不是C语言），也能够理解本书所讲解的大部分内容。当然，做编程练习至少要求掌握一些基本的Java编程结构。

书中的例子和编程练习并没有假定读者熟悉各种概念，或者具备从经验中产生的直觉（通常仅仅在三年级或四年级的学习中才会出现的）。有些问题来自于常见的图形用户界面（GUI）系统，其他的问题来自于一个关于非常简单的生态系统的模型。任何人只要使用过基于GUI的文档编写系统、电子表格、绘图工具或者类似的内容，就具备了理解这些用户界面的例子和问题的必要条件。对于理解生态仿真的例子也不要求特殊的背景知识。

目的

本书最主要的目的是帮助学习用Java做面向对象的程序设计。尽管书中只描述了Java语言，但是Java所基于的那些面向对象的概念却可以在许多其他的编程语言中实现，这些语言包括C++、Smalltalk和Eiffel。第1章描述了面向对象的主要概念，并没有特别提到Java。来自不同语言和分析方法中的术语也反映出更宽泛的编程环境。例如，下面的术语可以互换使用："成员函数"、"方法"、"操作"和"动作"。尽管可以对这些术语中做一些区分，但是这些区分对于面向对象编程的入门学生而言并没有实际意义。

本书第二个重要目的是通过强调以下各个方面来提高学生的编程能力：

- **复用**：软件复用的价值在于：在软件陈述、练习和实习中可以一次和多次复用软件。实际上，没有一个练习要求"从零开始"做程序开发。几乎所有的练习都使用了一组现成的类，靠后的章节中还会用到来自Java发布的一个扩展类库。
- **工具/技术**：除了语言特征之外，还给出了开发系统所需要的工具和技术。熟悉语言并写出代码只完成了创建真实系统所需工作的一半（有时还远少于一半）。开发人员还必须进行测试、调试和编写文档。尽管这些思想通常是在更高级的软件工程课程中学习，但这里还是给出了进行更高级学习所需要的基础内容。

- **GUI库**：通过练习和实习，学生们可以学会创建基于GUI系统的面向对象技术。这些技术和后面要介绍的Java Swing工具包将会成为学生要掌握的全部技能的一部分，这些技能在后续课程的编程实习中会用到。这些知识可以帮助你学习其他相似的类库。
- **事件驱动系统**：事件驱动系统是个亮点。初始的程序设计课程通常要处理这样的问题，即所编写的程序在运行时是完全受控的。然而，在事件驱动系统（比如，用户界面、交互式应用程序、操作系统、命令和控制系统）中，程序并没有完全受控。相反，程序是对外部事件做出反应（由外部事件驱动）。学习一些事件驱动系统可以加深体验，拓宽视野，同时提供一种感官上的直觉，有助于今后学习操作系统、计算机体系结构、网络以及类似的包含异步事件的课程。

教学

applet和在线帮助

http://www.prenhall.com/kafura

本书的站点包括了一些交互式的问题和Java applet（小应用程序）。为了使本书的内容更加全面，增加了以下几个内容：

- **动画**：applet提供了一种可视化的表达方式，这种方式比起下面这些方式能够更好地表达概念：静态图、一系列前后相关的静态图或者只是书面描述。这种情况尤其适用于涉及变化或动作的概念。以图形和动画的形式，可以更好地将教师自己头脑中的“景象”传递给学生。
- **交互**：具有主动元素（按钮、菜单等）的applet允许学生按下面的方式来获得对某个概念的（使用）经验：即控制这种经验。尤其是这种形式的applet的价值在于给出了一些关于构造性的、程序设计概念的（使用）经验，而不必关心语法和其他不必要的问题。
- **反馈**：书中包含一些简单的多项选择测试题，可以让学生测试一下自己对所学内容掌握的程度，并增强自信心。这种手段比起课堂内的测验或考试会更有效，并且使心理负担更小。在线提供的问题和applet不仅能更好地表述某些概念，而且有助于创建一个更加适合的学习环境。

jake环境

配合本书的软件包含了一个简单的可视化编程环境，取名为jake。jake环境包含了一些图标，

这些图标提供了对本书的例子和习题中用到的一组类的可视化描述。这些类的对象既可以通过直接操纵图标来创建，也可以利用jake中提供的库通过程序来创建。对象之间的关联方式通过在相应的图标之间绘制一条直线来表示。在本书中，jake环境是通过它所操作的所有类来描述的。

习题

几乎每一节的后面都配有一组习题，每道习题都只需很少的时间就能完成。理想情况下，应该在学习下一节之前完成某些习题。甚至可以鼓励教师和学生规定完成习题的时间，按定时的方式尽可能多地做一些习题。习题的设计都符合循序渐进的方式。在这个意义上，这些习题主要是用于帮助学习，而不是用于评价和分级的工具。这些习题都很重要。每道题都涉及某个新的思想。学生如果理解了小节中的内容，那么完成相应的习题就不会花费很多的时间。如果学生对章中的内容有些误解（或者只是想通过实践来更好地掌握概念），那么做习题会很有帮助。

连续的主题作为例子

贯穿全书使用了两个反复出现的例子，这两个例子的主题分别是图形用户界面和生态仿真。例子的连续性使得随着新的概念和技术的引入，可以开发出对问题更完整的解决方案。同时，这些例子也比将不相关的小例子集合在一起要更完整、更真实且更一致。这些例子和问题的设计都是有目的的，都是为了强调程序中的“对象”和“真实世界”中的“实体”之间的关系。例如，第一个编程的习题涉及到在屏幕上显示一个窗口，或者在一个仿真的生态系统中对一个简单的捕食动物的仿真。后面的几个习题涉及到移动和调整窗口大小，以及引入被捕食动物来形成一个捕食动物-被捕食动物模型。在将近3周内，学生要建立一些简单的系统，这些系统包括在图形用户界面中的按钮、定时事件和文本显示，以及在生态仿真中更复杂的场景。这类问题比起更常见的对象如Date、String和Address等更能吸引学生的兴趣。此外，真切地看到窗口如何在屏幕上移动，直接对应于在所创建窗口的类中应用`moveTo`方法，这样做就加深了理解一个对象是对其真实世界中的实体直接建模。

UML和Swing

随着引入新的面向对象的结构，给出了统一建模语言（UML）中的相应表示。由于UML是一种很大而且复杂的语言，因此本书只描述了它的基本特征。其中包括的UML框图用于类、对象、关联、聚集、界面、顺序、状态迁移和继承。这些内容是UML的核心，并且足以让学生合理地描述复杂的系统。

书中介绍了最新的Swing库（与Java 1.2版本一起引入）。由于Swing是一个丰富而且复杂的库，因此只给出了基本的Swing组件。所给出的组件足以构造出很大范围内的各种有意义的用户界面。给出的内容是进一步研究Swing的基础。利用两个简单的应用程序来举例说明Swing中提供的基本的用户界面控件：一个用于简单的图形绘制工具，另一个用于简单的文本编辑工具。许多编程习题对这两个工具进行了扩展。

独特的组织

本书中的内容是按概念和角色这两种相关的方式来组织的。概念化的组织如第1章中所示，基于下面的四个概念：抽象、分离、合成和概化，这些概念是构筑面向对象的编程语言。第1章先从很高的层次来理解所有这些概念，然后再讲解Java中的详细内容。然而，也可以交叉学习概念及如何在Java中实现这些概念，并且如何利用这些概念来创建系统。

第二种组织方式是按程序员在编程中承担的一系列不同角色进行的。这些角色包括以下几种：

- **单个现有类的用户**：在本书中，单个类表示一个图形用户界面窗口或者捕食动物。在这个简单并且直观的环境中可以自然地给出许多重要的概念（例如重载的方法、构造函数、范围以及静态与动态对象）。注意，从一开始就强调类和对象的重要性。
- **多个现有类的用户**：合成可以看成是通过将交互的对象组合在一起来创建系统。在这一阶段，学生要创建几个小的时间驱动系统，包含配套按钮和文本或者涉及捕食动物和被捕食动物的仿真。
- **单个类的实现者**：只有在这一阶段才揭示出类的内部结构。这里要强调的是，实现者的角色不只是编代码，还包括设计、编写文档、调试以及递增式测试和开发。
- **多个相关类的实现者**：在介绍这个角色的章节中引入了继承，并作为在一组相关的类中共享实现和/或界面的一种机制。

这种组织结构以一种合理并且一致的方式展现了语言特征。每一章集中介绍为了扮演好某个角色而需要学习的内容。

软件

本书例子中用到的全部代码都可以得到。此外，有一个简化的执行环境取名为jake，可以利用它来完成前面部分对图形用户界面习题的编程。这个环境允许构造出简单但有意义的界面，而不必关注于所有底层和分散注意力的细节。这个环境还提供了对程序创建的对象的图形表示以及对这些对象的直接操作。

评论

欢迎对本书提出评论、建议和批评。请发邮件至kafura@cs.vt.edu，或者按下面的地址邮寄：

Department of Computer Science
Virginia Tech
Blacksburg, VA 24061

Dennis Kafura
Blacksburg, Virginia

目　录

第1章 基本概念

1.1 引言

1.1.1 面向对象的策略

面向对象的程序设计在软件结构中体现了许多强大的设计策略，这些策略都是基于实际的和证明有效的软件工程技术。通过将对这些策略的支持合并到软件结构中，面向对象的程序设计能够构造出比以前任何时候更加复杂的、可管理的软件系统。数十年软件工程的经验总结出了这些软件结构的性质。表1-1中给出了在面向对象的程序设计中所包含的基本的设计策略。这些设计策略随着处理复杂的自然系统和人工系统的技术发展而演变。由于这些策略都是基础性的，因此它们在其他的环境和其他的程序设计语言形式中都会遇到。这里要强调的是这些策略与面向对象的软件的设计和构造之间的关系。这些策略广泛地得到现有的面向对象语言的支持，尽管在不同的语言中可能表现出不同的形式，并且某些语言可能会支持某种策略的其他变体。例如，某些面向对象的语言具有支持概化的附加方式。

表1-1 面向对象的程序设计中包含的设计策略

设计策略	定 义
抽象	将真实世界的实体简化为它的基本要素
分离	独立地处理一个实体做了“什么”以及“如何”去做
组合	构造复杂的“整个”系统，按以下两种方式之一将更简单的各个“部分”组装起来： • 关联 • 聚集
概化	按以下四种方式之一确定不同实体之间共同的要素： • 层次 • 通用性 • 多态性 • 模式

在面向对象程序设计中的这些设计策略对于构造问题域中实体的软件模型非常有效。事实上，有些人认为，软件设计在很大程度上是构造“真实世界”的软件模型，其中每一个“真实的”实体在程序中用相应的软件对象来表示；软件对象仿真其真实世界中副本的动作和条件。“像建模一样来做程序设计”的理念在三维的虚拟环境中表现得最明显，在虚拟世界中模拟真实世界的视觉和听觉特征。

1.1.2 设计策略的定义

学习计划

对主要内容的学习与各人的学习风格有关，并且通常是循序渐进式的提高或进步。有些

人在进一步学习更具体的细节之前首先了解总体的概念，这种方式对他们更有效。而有些人的学习方式却是将抽象的概念与具体的示例混杂在一起，他们认为这样更有效。不管是广度优先的学习方法还是深度优先的学习方法，有一些必要的思想（idea）却是在学习其他内容之前必须掌握的，因为这些思想建立在其他内容的基础之上，并且不是独立的。在这两种风格中，单靠阅读是不够的。经常回头复习一下前面的概念会增进对内容的理解，并获得更深层次的理解。

图1-1显示了本书中所给出内容的总体结构。图的左边给出的是设计策略。可以从上到下阅读这些设计策略，并得到对面向对象的概念和思想更广泛的了解。在任何一点都可以沿着向右的箭头进行下去。顺着箭头的方向深入到对概念更具体的表示形式，最后是在Java中的描述。在图中显示了六个重要的里程碑。每个里程碑都对应于图的底部所显示的三项任务的其中一项。每个里程碑都关联到一个重要的设计概念，并且表示在开发面向对象软件系统的实际技术中的
1~2 一个重要步骤。然而，各个里程碑之间是有顺序的。在所有前面的里程碑（更高并且靠左）完成之前，不能进入到下一个里程碑（更低并且靠右）。

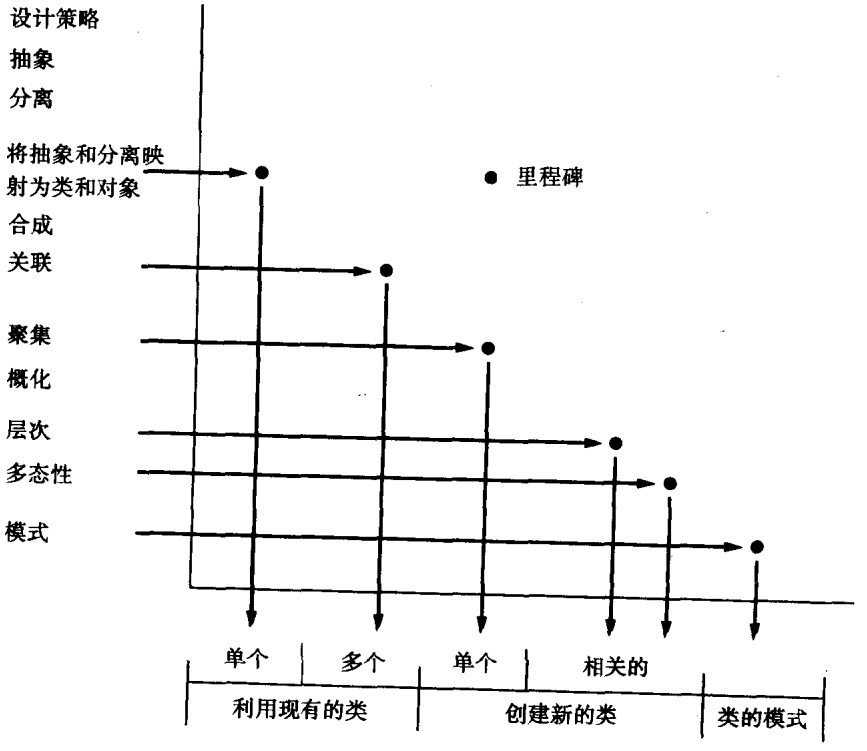


图1-1　内容指南

各个里程碑构成了一系列任务（role），如图中的底部所示。对于程序员而言，最简单的任务就是利用一个或者多个已经开发的类。在这个第一项任务中将构造出简单但有意义的系统。开始时，将利用单个类来探究并掌握创建和操作对象的基本问题。然后加入几个类，允许构造出具有交互对象的简单系统。学会如何使用现有的类来创建和操作对象是学习面向对象程序设

计的第一步。这第一项任务很重要，不仅因为它建立了面向对象程序设计的基本概念的基础，还因为这是开发实际系统中首选的任务。作为一个用户，你能够从现有类的设计者和实现者已经完成的艰苦工作中享受成果。不管是在一般的程序设计中还是在面向对象的程序设计中，利用现有类的能力是软件复用的主要好处之一。开发人员的第二项任务是创建一个或多个新的类。 3
开始时，每个类捕获一个独立的抽象。更困难、但功能更强大的是利用一种概化的形式来开发相关类的集合。可以广泛地使用前两项任务。第三项任务涉及将一些类和对象组织为通用的模式，这些类和对象被证明在解决常见的设计问题时是有用的。对这项任务将会有一些探讨，但仅限于最低限度。掌握模式的使用需要在一个或多个应用领域中创建系统方面具有相当的经验。

1.1.3 例子

纵贯全书，通过来自两个独立问题领域的一些连续的例子来说明设计的概念和编程结构。其中的一组例子是基于图形用户界面（GUI）。这方面的例子说明如何应用Java中面向对象的特性和设计策略，利用一组预先定义的GUI类来给小的应用程序构造越来越复杂的界面。第二组例子是基于对一个简单生态环境的仿真。这些例子利用了简单的文本输出。利用两组独立的例子来给出每一个概念的不同示例，并且对编程的习题有更多变化的选择。这些例子在节与节以及章与章之间都保持连续性，因此随着更多设计策略和Java特性的出现，就可以看到渐进的开发过程。在某些情况下，对早期的设计进行修改，以便利用新引入的策略或特性。

1.1.4 UML设计表示法

设计一个面向对象系统的能力对软件开发人员而言是最重要的技能，同时也是最难学会的。设计技能很重要，因为系统的设计确定了系统在多大程度上实现了大多数关键的软件工程原则，如分离、封装和信息隐藏；以及关键的设计目标，如抽象和概化。好的面向对象设计的技能是不易掌握的，因为像所有的设计活动一样，它不能被缩减为一个机械的过程。好的设计人员在完成任务时会表现出天赋，比如洞察力和灵感，还有他们在许多软件工程中实践面向对象设计艺术时所积累的经验。尽管无法清晰地说明洞察力、灵感和经验，但是可以从中引出一些启发式规则和指南供大家学习，这些启发式规则和指南刻画了一项好的设计的特征。

利用一种半形式化的、高层次的、图形化的表示形式来表示类、对象以及它们之间的关系，这种表示形式可以辅助进行面向对象系统的设计。表示形式的图形化特性使得系统的结构方面更容易可视化。在高层次上表示设计，其中只出现设计的基本要素，而与词汇细节或者实现细 4
节无关。设计表示形式（design representation）的用处表现在它们使得设计者的思想具体化，因此它们可能需要分析、评价以及可选设计方案之间的比较；对实现设计的人员来说，它们可用做蓝图；对以后执行维护和扩展的人员而言，它们记录了最终的设计。

对高层次的、图形化的设计表示形式，有三种特定的用途。首先，表示形式提供了一种语言，这种语言快速地描述了单个设计者或设计团队正在考虑的思想。由于这种表示形式可以快速地绘出并易于修改，因此它在思考设计和鼓励探索设计方案方面起到辅助的作用。第二，这种表示形式是一种用文档记录设计的手段。对于实现者和今后将会修改并维护系统的那些人来说，设计文档可以用做指南。这种图形化和高层次表示的特性对于文档设计而言是很有用的，

因为只要描述基本的结构要素，更详细的信息留给代码和其他的文档。第三，一旦将设计方案表示出来，那么这项设计本身就可以是复用的对象。确定常见的设计方案，用文档记录下来，并将它们搜集起来，以便在再次遇到它们的场合使用。学习面向对象设计的学生应该学习由好的设计人员产生的一些制品（设计），这非常类似于艺术系的学生研习大师的作品、建筑系的学生学习著名的结构和建筑物、学写作的学生学习经典的文学作品。对于学习面向对象设计的学生，最近已经出现了表示设计模式的表示法，利用这些表示法可以研究优秀的（或者至少是经常需要的和基础性的）设计方案。

在全书中都使用了统一建模语言（UML）来可视化地描述面向对象的设计。UML是一种大家熟知的图形化的表示法，表示面向对象系统的组件和组织。随着在书中不断地引入新的Java语言特性，会给出相应的UML表示法。UML是一种丰富的可视化语言，本书不能完整地包含它的内涵。因此，在这里只描述了UML表示法的核心子集。有很多其他的书籍描述了完整的UML表示法。

1.1.5 总的组织

本书的内容在总体上分为两部分。第一部分包括第1～6章，讲述面向对象程序设计的概念和Java作为一种面向对象的程序设计语言。这几章的组织如图1-1所示。第1章简要地描述了图1-1中垂直轴上列出的设计策略，这些策略也总结在表1-1中。第2～6章按照图1-1中水平轴上显示的顺序逐步讲述。第2章利用了一个现有的类来引入类和对象的基本概念。第3章中利用了几个现有的类来说明如何构造小的对象系统。在第4章中描述了实现一个新的类。第5章介绍了设计、调试和用文档记录新类的概念和技术。第6章解释了如何组织相关的类，使得利用面向对象的继承可以更容易地理解、实现和扩展这些类。第6章还描述了利用继承来创建类层次的设计策略以及设计模式。

本书的第二部分包含了有关Java的高级特性，这些特性并不是面向对象的固有部分，但包含在伴随Java发布的库中。之所以包括这些主题，是因为对于除了简单的演示示例之外的任何内容，这些主题都是在Java环境中最基本的工作。事实上，当人们谈论Java时，经常指的是由库而不是由基本的语言本身所提供的实用程序。本书中出现的工具、特性和库按如下结构组织：

- 用户界面（第7章）：利用与Java 1.2版本一起引入的新库来创建图形用户界面的类、界面和约定。
- 输入/输出（第8章）：Java中定义了一些功能性程序（facility），用来将数据传输给标准的设备和文件。这些程序允许以文本、二进制和对象形式处理数据。
- 线程（第9章）：在Java中用于并发编程的一些机制。这部分内容说明了如何创建独立、异步的活动，并使这些活动同步。

由于Java和它的库总是在变化之中，因此本书中给出的内容遵循Java 2版本。书中尽量利用了Java和它的库中核心的、不易改变的那些特性。

下面就让我们开始吧。

习题

1. 在万维网上查找各种面向对象语言的参考文献。看看你能找到多少。在它们的描述中你能识别上面提到的基本策略吗?
2. 在万维网上查找有关面向对象程序设计其他课程的位置，这些课程有哪些共同的主题?保存这些链接，以便在学习中参考。
3. 查看“免费的在线计算机词典”(Free On-Line Dictionary Computing)，找出以下术语的定义：抽象(abstraction)、聚集(aggregation)、封装(encapsulation)、层次(hierarchy)、通用性(genericity)和多态性(polymorphism)。保存这个链接，以便今后参考。
4. 查看Java虚拟库(Java Virtual Library)。浏览该库10分钟并报告所找到的三件有意义的事情。保存这个链接，以便参考。

1.2 抽象

抽象是一种设计技术，重点说明一个实体各个本质的方面，而忽略或者掩盖不很重要或非本质的方面。抽象是一种重要的工具，用来将复杂的现象简化到可以分析、实验或者可以理解的程度。例如，为了理解太阳系的力学结构，早期的数学家和天文学家将行星进行抽象，即将行星看做这样一个物体：其所有的质量都集中于单点上。这样的抽象忽略了每个行星丰富的细节：直径、空气含量、平均温度，等等。然而，这些细节都与理解太阳系基本的轨道结构以及对轨道结构建模无关。

在软件中，抽象所关心的是实体的“属性”和“行为”。属性指的是与一个实体相关的性质或特性，而行为指的是实体可能执行的动作集。在一个软件对象中，利用与该对象相关的数据来表示属性。对于一个销售跟踪系统，销售人员的相关属性(见图1-2)可能是：姓名、售出的车辆数量、售出的车辆价值、顾客清单、回扣率、总的回扣，等等。对象的动作对应于与对象相关的一项操作或功能。销售人员的动作可能包括sellCar(销售汽车)、reportIncome(报告收款)和increaseCommissionRate(增加回扣率)。

抽象对于创建易管理的软件对象是至关重要的，因为实际的对象非常复杂，难以完全用细节来捕获。考虑上面所提到的“销售人员”这个对象。一个真实的“销售人员”会有一个身份、家谱、医疗记录、遗传情况、信用记录、各种才能，等等。类似地，也会有销售人员能够采取很多的动作(sellCar、answerPhone、buyHouse、haveChild、getSick、increaseCreditLimit、payBills，等等)。在软件对象中，试图对这种非常多的细节哪怕捕获很小的一部分也是毫无意义的。重要的是只要捕获“销售人员”中与特定系统(如销售跟踪系统)的开发相关的那些方面。

人们经常想把面向对象系统中的对象直接对应于“真实世界”中的实体。在汽车经销商的跟踪系统中可能会出现如“销售人员”和“汽车”这样的对象，这些对象对应于经销商实际的雇员以及由经销商实际拥有和售出的汽车。软件对象与由它们所表示的真实实体之间的对应关系经常很直接和实际，以至基于计算机的盗窃或欺诈行为经常会窜改软件对象，