普通高等教育计算机规划教材

Java 面向对象程序设计

邹蓉　等编著

提供电子教案

下载网址 http://www.cmpedu.com

机械工业出版社
CHINA MACHINE PRESS

普通高等教育计算机规划教材

Java 面向对象程序设计

邹　蓉　等编著

机 械 工 业 出 版 社

本书以 Java 语言为平台，通过大量例题和综合实例由浅入深、全面详尽地介绍了面向对象程序设计的基本要素和必备内容。全书共 12 章，分别是：面向对象程序设计基础、Java 概述、Java 基本语法、类和对象、数组和字符串、继承与多态、类设计原则和规范、异常处理、多线程程序设计、输入/输出机制、集合框架，以及数据库访问技术。每章均配有本章小结，并提供了难易程度结合的习题，供读者参考和练习。

本书内容丰富，理论与实际相结合，不仅可以作为高等学校计算机、信息管理及相关专业本专科学生 Java 语言和面向对象程序设计课程的教材，也同样适合自学者和软件开发人员参考使用。

本书配有电子教案，需要的教师可登录 www. cmpedu. com 免费注册、审核通过后下载，或联系编辑索取（QQ：2399929378，电话：010 - 88379753）。

图书在版编目（CIP）数据

Java 面向对象程序设计/邹蓉等编著. —北京：机械工业出版社，2014.2
普通高等教育计算机规划教材
ISBN 978-7-111-45425-0

Ⅰ. ①J… Ⅱ. ①邹… Ⅲ. ①JAVA 语言 - 程序设计 - 高等学校 - 教材
Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2014）第 004504 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
责任编辑：和庆娣 孙文妮
责任印制：李 洋
北京宝昌彩色印刷有限公司印刷

2014 年 2 月第 1 版 · 第 1 次
184mm × 260mm · 19 印张 · 470 千字
0001 - 3000 册
标准书号：ISBN 978-7-111-45425-0
定价：39.90 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
社服务中心：（010）88361066
销售一部：（010）68326294
销售二部：（010）88379649
读者购书热线：（010）88379203

网络服务
教材网：http://www.cmpedu.com
机工官网：http://www.cmpbook.com
机工官博：http://weibo.com/cmp1952

出 版 说 明

信息技术是当今世界发展最快、渗透性最强、应用最广的关键技术，是推动经济增长和知识传播的重要引擎。在我国，随着国家信息化发展战略的贯彻实施，信息化建设已进入了全方位、多层次推进应用的新阶段。现在，掌握计算机技术已成为21世纪人才应具备的基础素质之一。

为了进一步推动计算机技术的发展，满足计算机学科教育的需求，机械工业出版社聘请了全国多所高等院校的一线教师，进行了充分的调研和讨论，针对计算机相关课程的特点，总结教学中的实践经验，组织出版了这套“普通高等教育计算机规划教材”。

本套教材具有以下特点：

1）反映计算机技术领域的新发展和新应用。

2）为了体现建设“立体化”精品教材的宗旨，本套教材为主干课程配备了电子教案、学习与上机指导、习题解答、多媒体光盘、课程设计和毕业设计指导等内容。

3）针对多数学生的学习特点，采用通俗易懂的方法讲解知识，逻辑性能、层次分明、叙述准确而精炼、图文并茂，使学生可以快速掌握，学以致用。

4）符合高等院校各专业人才的培养目标及课程体系的设置，注重培养学生的应用能力，强调知识、能力与素质的综合训练。

5）注重教材的实用性、通用性，适合各类高等院校、高等职业学校及相关院校的教学，也可作为各类培训班和自学用书。

希望计算机教育界的专家和老师能提出宝贵的意见和建议。衷心感谢计算机教育工作者和广大读者的支持与帮助！

机械工业出版社

前　言

面向对象技术是目前较为流行的系统开发技术，面向对象程序设计也已成为软件开发的主流程序设计方法之一。面向对象的程序设计语言 Java 具有与平台无关、简单高效、多线程、安全和健壮等特点，因此被广泛运用于企业级 Web 应用开发和移动应用开发，是当代最重要的编程语言之一。

要学好程序设计，首先要有兴趣。对此，本书在结构上作了精心安排，由简单实例引出问题和知识点，让读者可以对所学内容有初步的认识。随后在对细节的逐步深入过程中，通过对问题的求解提升读者的学习兴趣。

本书在对知识点进行分析和归纳的同时，对引例作扩展或改变，逐步形成更全面、复杂的实例，让读者通过对比加强对概念的理解，从而达到举一反三的学习效果。考虑到知识的连贯性，各章节之间也会在内容和实例上有所联系，并根据需要补充例题，以涵盖各知识点并拓宽读者思路。

为了进一步深化读者对基本概念的理解，提高读者综合应用能力，本书大部分章节的最后都设有综合性较强的应用实例。实例的选择不仅兼顾基本理论，且设计实现过程紧密围绕面向对象技术，力求使读者在掌握 Java 面向对象程序设计核心理论与编程思想、技巧的同时，养成良好的编程习惯。本书所配备的例题，清晰直观、循序渐进，并通过通俗易懂且逻辑性强的讲解巩固知识点。每章后的习题也供读者参考。

本书内容上主要分两大部分，共 12 章。

第一部分包括第 1 章～第 7 章，内容主要涉及 Java 语言及面向对象程序设计技术的理论基础知识，详细阐述了面向对象程序设计基础、Java 概述、Java 基本语法、类和对象、数组和字符串、继承与多态、类设计原则和规范等内容。

第二部分包括第 8 章～第 12 章，是对第一部分内容的扩展和实际运用，以使用 Java 面向对象程序设计技术实现各类应用为主，介绍了异常处理、多线程程序设计、输入/输出机制、集合框架，以及数据库访问技术等内容。

本书由邹蓉、张媛、林云、李留青编著。其中，邹蓉编写第 1、2、4、7、8、9 章，张媛编写第 5、6、11 章，林云编写第 10、12 章，黄淮学院的李留青编写第 3 章。全书由邹蓉统稿。此外，本书编写过程中还得到了中国石油大学（北京）陈明老师的指导和帮助，在此表示感谢。

由于作者水平有限，书中难免有不足之处，恳请读者批评指正。

编　者

目　录

第1章　面向对象程序设计基础

自计算机诞生以来，软件的发展始终滞后于硬件的发展。随着程序规模的不断扩大，复杂度的加深，人们也在不断地寻找更科学、更规范的软件开发方法。面向对象方法便是目前非常流行的软件开发方法之一。

1.1　结构化方法与结构化程序设计

在面向对象方法大行其道之前，人们普遍采用结构化方法来进行软件开发。结构化方法源自迪克斯特拉（E. W. Dijkstra）提出的结构化程序设计概念。它采取自顶向下、逐步求精的程序设计方法，以模块化设计为中心，把工作分解成若干个相互独立的模块，使用顺序、分支和循环三种流程控制结构来构造程序。如图1-1所示展示了结构化程序设计的三种流程控制结构。

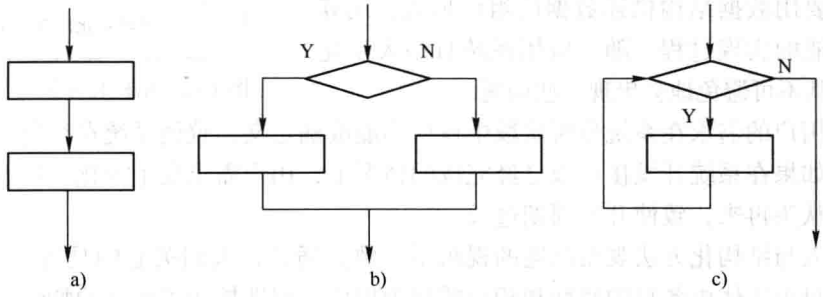

图1-1　三种流程控制语句结构

a）顺序结构　b）分支结构　c）循环结构

常见的支持结构化程序设计的语言有Pascal和C等。它们以过程（函数）作为程序的基本单元。每个过程中仅使用顺序、分支和循环三种流程控制语句，不同过程之间通过语句调用来共同完成整个程序的功能。概括地说，程序内容包括过程和过程调用。因此，结构化程序设计语言也被称为过程式语言。

结构化程序设计是软件发展史上一个重要的里程碑，由其发展成了软件开发的结构化方法（Structured Method），也称为面向过程方法（Procedure Oriented Method）。结构化方法把一个复杂问题的求解过程分成结构化分析（Structured Analysis，SA）、结构化设计（Structured Design，SD）和结构化程序设计（Structured Programming，SP）三阶段进行，从而使每个阶段解决的问题都被控制在人们容易理解和处理的范围内，以增强软件的可读性，规范软件的开发。

例如，用结构化方法设计实现模拟剪刀石头布的游戏过程。

（1）对问题进行功能分析

1）两个玩家先各自握紧拳头，其中一人或者两人一起念出口令。

2）在说完口令后，两个玩家需立即出示“剪子”、“石头”或“布”的手势。

3）根据规则判定输赢。

（2）设计问题的解决方案

1）将得到的结果按相互独立的原则划分成若干个功能模块，包括准备、出拳、判定输赢和输出结果等。

2）再设计一个顶层模块，负责按照操作顺序将上述功能模块组装起来。

（3）进行结构化程序设计

结构化程序设计阶段，根据设计结果使用结构化语言对各模块实施编码。

如图 1-2 所示的是用结构化方法设计实现模拟剪刀石头布游戏过程三个阶段的示意图。

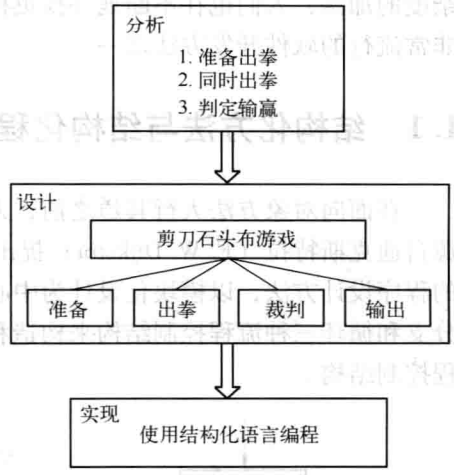

图 1-2　结构化方法的三个阶段

结构化方法强调整体设计思路，工作中阶段性非常强，这样有利于系统开发的总体管理和控制。然而，结构化方法求解问题的出发点是功能。整个程序被看成是完成某种功能的功能模块，其又由若干完成特定任务的子功能模块组成。在程序设计阶段，每个功能模块都要用数据结构描述数据的组织形式，用算法来描述功能的实现过程。随着应用系统日趋大型化和复杂化，其不可避免地会出现一些问题。

首先，用户的需求在系统分析阶段中往往不能准确定义，致使系统在交付使用时会产生许多问题。如果在系统开发接近或已经完成的情况下，用户需求发生变化，则系统须做大量修改，甚至从头再来，致使开发周期过长。

其次，人与结构化方法观察问题的视角不一致。通常，人们关心的是客观世界的实体。自然界中，每个实体由各自的属性和相应的行为构成，属性描述了实体的特性，行为是实体的动作。例如，狗有出生年月、性别、品种等属性，吃食、睡觉、吠叫等行为，它们共同构成了狗这个实体。结构化方法关心的是实体中的行为，程序被按功能分解为若干个子过程，因此对客观实体的映射就变得不那么直接。这既增加了程序设计的复杂度，又提高了后期维护和扩展的难度。

最后，结构化程序设计实质上是围绕实现处理功能的“过程”来构造系统的。模块是程序的基本单元，也是重用的基本单位。在越来越强调软件通用性的今天，模块的抽象级别太低，重用力度微不足道。而且，对于由算法和数据结构组成的模块来说，一旦参与运算的数据结构发生变化，模块便不能再被使用。

正是由于结构化方法开发软件存在开发的周期长，程序设计的复杂度高，又具有可维护性、可扩展性和可重用性较差等问题，因此，伴随着计算机应用范围的日益扩大，人们逐渐开始转向使用面向对象方法求解问题。

1.2　面向对象方法与面向对象程序设计

“对象”概念最早出现在 20 世纪 50 年代初关于人工智能的早期著作中。1967 年挪威计

算中心的克利斯登·奈加特（Kristen Nygaard）和奥利-约翰·达尔（Ole-Johan Dahl）发布了 Simula 语言，又称为 Simula67，第一次引入了“类”的概念。20 世纪 70 年代初，美国 Xerox 公司研究中心（PARC）的艾伦·凯（Alan Kay）等人发明 Smalltalk 语言，第一次采用了“面向对象”一词。1981 年推出的 Smalltalk80 被认为是最纯的面向对象语言，成为面向对象发展史上的里程碑。之后，面向对象思想迅速发展，最终形成了面向对象分析（Object Oriented Analysis，OOA）、面向对象设计（Object Oriented Design，OOD）和面向对象程序设计（Object Oriented Programming，OOP）的面向对象方法（Object Oriented Method）。

与传统的将程序看作一系列功能模块的集合不同，面向对象的核心是构成问题域的现实世界中存在的各个实体。这些实体被描述成对象，它们之间的联系被描述成对象之间的关系。通过消息传递实现对象之间的交互，进而完成相应的功能。

例如，同样针对剪刀石头布游戏，用面向对象方法设计实现其游戏过程。

（1）对问题进行分析

1）参与游戏的包括两位行为一模一样的玩家和负责判定输赢的裁判。

2）两位玩家首先根据“开始”命令同时完成出拳的动作。

3）裁判按照游戏规则对结果进行输赢的判定。

（2）设计问题的解决方案

1）根据分析结果设计玩家和裁判两个类，并由此产生两个对等的玩家对象和一个裁判对象。

2）玩家对象拥有“参赛者编号”“输赢次数”等属性和“出拳”等方法，并将出拳消息发送给裁判对象。

3）裁判对象拥有“判定输赢”及“确定最终结果”等方法，对两个玩家传递的出拳消息进行处理。

4）游戏开始时，先由用户对玩家发出开始的命令，然后玩家出拳并由裁判根据游戏规则对输赢进行判定。

（3）进行面向对象程序设计

在面向对象程序设计阶段，根据设计结果使用面向对象语言进行编码。

图 1-3 是用面向对象方法设计实现模拟剪刀石头布游戏过程三个阶段的示意图。

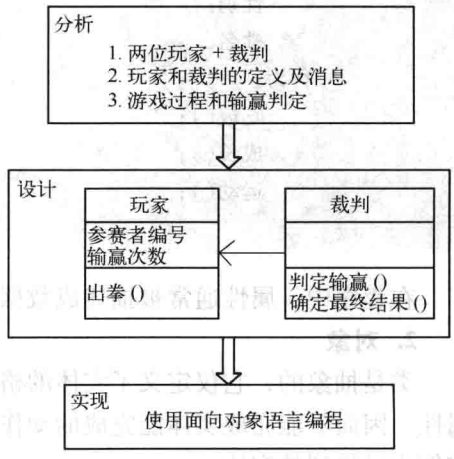

图 1-3　面向对象方法的三个阶段

显然，面向对象从构成问题域的现实世界中存在的实体出发，实现了对客观实体的直接映射，使得程序开发过程更加自然且容易理解。同时大大降低了程序的复杂度，提高了程序的独立性和稳定性。

面向对象程序中包含的各种对象既相互独立又可相互调用，每个对象都可以接收数据、处理数据并将数据传给其他对象，使程序变得灵活且可靠。如果需求发生变化（例如上面例子中玩家的出拳动作有改变），程序也不用大范围地调整，仅需通过继承等方式对程序作局部改动即可。这既保证了程序的可扩展性，也降低了后期维护的难度。

对象是对现实世界实体的直接映射，而类通过描述对象的属性和方法实现了对象格式的声明。相较于结构化程序设计中的模块，类具有更大的力度，可以供其他程序直接使用，从而提高程序的可重用性。按照面向对象的要求，类提供了信息隐藏和封装机制，既可以防止外界对对象的非法操作，保障程序的安全，也使对程序的修改变得更加容易。

在计算机应用范围日渐扩大的今天，正是因为面向对象相较于面向过程在程序设计的复杂度、灵活性、可靠性、可维护性、可扩展性和可重用性等方面的优势，从而使得面向对象方法逐渐成为主流程序开发方法之一。

1.3 面向对象程序设计的基本概念和特征

要理解面向对象程序设计，首先必须了解它的基本概念。

1. 类

因为面向对象实现对客观实体的直接映射，所以先要将具有相同类型特性的客观实体分类，抽象出同一类实体共有的特征和行为，进行一般性描述，这就是类（Class）。从概念上说，类是对客观实体的抽象，通过定义实体的属性（Attribute）和行为（Behavior）描述了对象的格式。其中，属性是类实体的特性，行为是类实体能完成的动作或操作。

例如，无论是司机、职员、农民等从业者，还是张三、李四等个体，都可以概括成“人”这个类。“人”类包含了人的一切基本特征和行为，如出生年月、性别、姓名等属性，吃饭、成长、运动等行为。下面一段伪代码对“人”这一类进行了描述，所有属于“人”范畴的客观实体都可以按照这个描述来生成：

```
类 人{
    属性:
        出生年月;
        性别;
        姓名;
    行为:
        吃饭();
        成长();
        运动();
}
```

在程序中，属性通常被描述成数据结构，行为则由方法（Method）来实现。

2. 对象

类是抽象的，它仅定义了实体的格式，但其却不是一个真正的实体，不具有实体应有的属性，因而不能完成实体能完成的动作。对于面向对象技术来说，对象（Object）才是对类实例化后得到的实体。

例如，“人”这个类定义了人的结构，它适用于世界上所有的人。但“人”类是抽象的，本身没有任何具体的属性。而“张三”这个对象是一个具体的人，它的属性才是具体的：出生年月 1980 年 8 月，性别男，姓名张三等。因此，“张三”就是“人”这个类的一

个实例，可以用“人”类来定义张三对象：

```
人 张三;
构造张三对象(出生年月 = 1980 年 8 月;性别 = 男;姓名 = 张三);
张三. 运动();
```

“人”这个类无法运动，但是对象“张三”却可以运动。

3. 消息

对象间的互动是通过消息（Message）传递来实现的。消息中包含了消息的接收者（对象）和要求接收者完成的操作请求。例如：

```
张三. 运动();
```

就是一条消息。它要求张三这一对象完成“运动”这种行为。

对象间通过发送消息、接收消息、处理消息（调用相应的方法）的过程，并不断地重复该过程来实现程序的有效运行，最终得到相应的结果。

4. 抽象

所谓抽象（Abstraction），是指从众多事物中抽取出共同的、本质的特征，舍弃其非本质的特征。例如，猫、狗、兔、鸡、鸭等，它们共同的特性就是动物。获得动物概念的过程，就是一个抽象的过程。共同特征是指那些能把一类事物与其他类事物区分开的特征。这些具有区分作用的特征又被称为本质特征。

抽象又分为功能抽象和数据抽象两部分。功能抽象提取出一类事物共同拥有的功能。数据抽象提取事物的属性特征和行为特征，这恰恰是面向对象程序设计的核心。类正是数据抽象的具体表现。数据抽象过程中，有两个概念很重要，一个是模块化（Modularity），另一个是信息隐藏（Information hiding）。

模块化指的是将复杂问题逐步分解成若干个相对简单的子问题，每个子问题便是一个模块。模块化的过程类似于结构化“自顶向下、逐步求精”的方法。但与结构化的功能模块相比，面向对象方法中的模块拥有更大的粒度和抽象级别。它以类为单位，封装了对象的属性和方法。

信息隐藏是指将模块的某些内容隐藏起来，令外界不可见。被隐藏的内容既包含模块的实现细节，也包含模块的执行机制和相关数据。如果一个模块想访问另一个模块的信息，则必须通过特定的接口来完成。信息隐藏提高了系统的安全性和可靠性，有利于代码的重用，也使对象具有了封装的特性。

5. 封装

封装（Eneapsulation）机制在现实生活中随处可见。举例来说，电视机内部有很多元器件。若干元器件按设计要求组合在一起完成特定的功能，比如开关机、调台、调节音量大小等。除了专业人士以外，普通用户不需要了解电视机内部的构造。看电视时，用户不直接操作这些元器件，而是通过遥控器或者机上面板来发出指令。电视机则根据指令，按照事先设计好的方法实现相关功能，改变自身状态。

面向对象程序设计中，对象由属性和方法组成。属性代表了对象的状态，它的改变意味着对象状态的改变。封装则是将对象的属性和方法组合成一个可供访问的基本逻辑单元。外

界欲修改获取对象的属性时，只能通过对象提供的方法来实施。除了这些方法外，对象中还有大量隐藏起来的属性和方法，它们被用来支持访问接口的实现和对象内部的操作。

例如，“张三”对象拥有出生年月、性别、姓名等属性，并在构造对象时就有了确定的值。按照常理，出生年月和性别属性一旦生成，便不能修改，只能查看；成长方法也不能通过外界的干预而执行，只能是对象内部的动作；运动方法却可以借由外界的消息而调用。程序中，各属性和成长方法变成私有访问权限被隐藏起来，外界可见的仅有“吃饭”和“运动”等方法。另外还可以根据需要，提供外界可见的查看各属性的方法和修改姓名属性的方法，但不提供修改出生年月和性别属性的方法。这样可以使外界无法对其加以改变，从而保证数据的安全。由此，“张三”对象所属的“人”类可以修改为如下描述：

```
类 人 {
    私有属性：
        出生年月；
        性别；
        姓名；
    私有方法：
        成长()；
    公有方法：
        吃饭()；
        运动()；
}
```

封装使得软件具有良好的模块性，防止程序间相互依赖带来的影响，让系统拥有更高的可维护性和可移植性。然而，当问题域很复杂时，对象中需要封装的属性和方法会急剧膨胀。因此，为控制类的规模，面向对象技术引入了“继承”的概念。

6. 继承

继承（Inheritance）是类的一种层次结构：一个类的上层可以有“父类”（基类），下层可以有“子类”（派生类）。子类可以继承父类的全部特征，并对其加以扩展或覆盖。由此可知，子类比父类更具体。

继承具有传递性，子类继承层次结构处于它之上所有类的全部特征。继承还有单重继承和多重继承之分：单重继承指一个类仅有一个直接父类；当一个类有多个直接父类时，就是多重继承。

例如，“人”这个类有个子类“学生”，李四是学生类的一个实例。基于继承的特性，李四对象既有“人”类的属性和方法，还会有自己的属性和方法。

假设学生类的描述如下：

```
类 学生 继承 人 {
    属性：
        班级；
    方法：
        上课()；
}
```

则学生类的实例将拥有出生年月、性别、姓名和班级等属性，以及吃饭、成长、运动和上课等方法。对于“人”类的实例张三对象来说，它没有班级属性和上课方法。

继承体现了客观世界中一般与特殊的关系，很大程度上解决了软件的可重用性问题。可是它仍然有不足之处，例如，父类定义的某个方法被子类所继承，但是不同子类针对此同名方法却有各自不同的动作。对此，面向对象需要使用多态来解决这个问题。

7. 多态

多态（Polymorphism）是指不同类的对象对同一消息做出完全不同的响应。例如，司机和职员都要工作，但是他们工作的具体行为却完全不同。

面向对象程序设计中，对象的多态依赖于对象的继承。例如，“人”类拥有工作方法，它有两个子类，即司机和职员，它们继承了父类的工作方法。当对司机和职员对象都发出工作请求时，它们分别完成如开车和写报告的工作，因此会产生完全不同的结果。可以用下列伪代码描述这个现象。首先定义类及其继承关系：

```
类 人 {
    ...
    方法:
        工作();
}
类 司机 继承 人 {
    ...
    方法:
        工作(){
            开车;
        }
}
类 职员 继承 人 {
    ...
    方法:
        工作(){
            写报告;
        }
}
```

工作方法的调用如下所示：

```
司机 张三;
职员 李四;
张三. 工作();
李四. 工作();
```

这里可以看出，同样是工作，张三和李四完成的动作完全不同。

常见的多态形式有参数多态（重载，Overload）和包含多态（覆盖，Override）。后面的

章节将对它们进行详细的介绍。

多态提高了程序的灵活性、可维护性和可扩展性。它与抽象、封装以及继承一起被称为面向对象技术的四个基本特征。

1.4 统一建模语言 UML

用文字来说明类和对象并不直观，目前人们更多地使用图形表示方式。统一建模语言 UML（Unified Modeling Language）以图形化的形式对系统的不同侧面做出面向对象的描述，为软件开发的各个阶段提供了模型化和可视化支持。最常用的 UML 图是描述类、接口以及它们之间静态关系的类图。本书也将使用类图来描述类及类之间的关系。

1. 类

类在 UML 中通常以实线矩形框表示，矩形框主要分类名、属性、方法等部分，如图 1-4 所示。

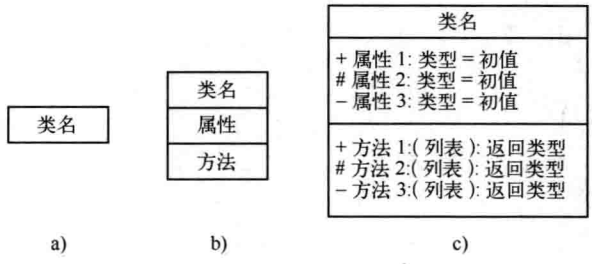

图 1-4 类的 UML 图形

a）简易表示法 b）设计表示法 c）实现表示法

其中，图 1-4a 仅表示类的存在；图 1-4b 是分析设计阶段的表示法；图 1-4c 可以反映类的更多细节，如“+”“#”和“-”分别表示 public、protected 和 private 访问属性。

2. 抽象类

抽象类是因含有抽象方法而无法实例化的类。类名和抽象方法用斜体表示，其他与类的说明相同，如图 1-5 所示。

3. 接口

接口是特殊的类。图中没有属性说明，接口名和方法用斜体表示，如图 1-6 所示。

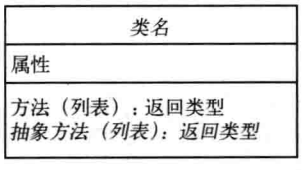

图 1-5 抽象类的 UML 图形

图 1-6 接口的 UML 图形

4. 关联

关联（Association）是对象间的一种引用关系，例如学生类与课程类之间的关系，使用带箭头的实线表示。箭头从使用类指向被关联的类，可以是单向，也可以是双向，如图 1-7

所示。双向关系可以不使用箭头。

5. 聚合

聚合（Aggregation）也是关联关系中的一种，表示整体与部分的关系，即“has-a”关系，使用空心的菱形表示，菱形那端为整体，其中包含了另一端所指的部分，如图 1-8 所示。

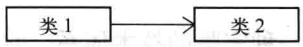

图 1-7　关联关系

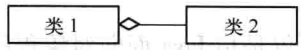

图 1-8　聚合关系

6. 泛化

泛化（Generalization）表示“is-a”关系，即继承关系，使用带空心三角箭头的实线表示，箭头指向父类，如图 1-9 所示。

7. 实现

实现（Realization）是接口和实现类的关系，使用带三角箭头的虚线表示，箭头从实现类指向接口，如图 1-10 所示。

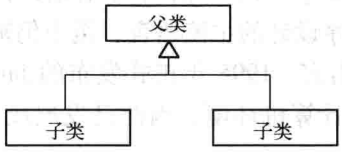

图 1-9　泛化关系

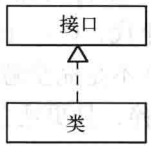

图 1-10　实现关系

除了类图外，UML 还有其他类型的图。有兴趣的读者可以参考相关资料，本书不再赘述。

1.5　本章小结

本章首先介绍了结构化方法的基本特征和现代应用环境下存在的问题，通过实例的对比引出面向对象方法。在对面向对象程序设计的基本概念和特征进行详细的阐述之后，对 UML 中类图的基本符号也做了大致讲解，为后面章节的学习打下一定基础。

1.6　习题

1. 面向对象与面向过程的主要区别表现在哪些方面？
2. 简述面向对象的基本特征，并说明它们的作用分别是什么。
3. 举几个现实中封装和继承的例子。
4. 什么是 OOA、OOD 和 OOP？它们之间有何关系？
5. 用面向对象方法设计顾客与销售员讲价成交的交易过程。
6. 考虑学生和课程，对它们抽象出应有的属性和行为，以及两者的关系，并用 UML 类图描述。