普.通.高.等.学.校
计算机教育“十二五”规划教材

Java 面向对象程序设计

（第2版）

OBJECT-ORIENTED PROGRAMMING WITH JAVA (2nd edition)

韩雪 ◆ 主编
王维虎 ◆ 副主编

人民邮电出版社
POSTS & TELECOM PRESS

普.通.高.等.学.校
计算机教育“十二五”规划教材

Java 面向对象程序设计

（第 2 版）

OBJECT-ORIENTED PROGRAMMING WITH JAVA
(2^{nd} edition)

韩雪 ◆ 主编
王维虎 ◆ 副主编

人 民 邮 电 出 版 社
北 京

图书在版编目（CIP）数据

Java面向对象程序设计 / 韩雪主编. -- 2版. -- 北京 : 人民邮电出版社, 2012.9
普通高等学校计算机教育“十二五”规划教材
ISBN 978-7-115-29041-0

Ⅰ. ①J… Ⅱ. ①韩… 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(2012)第174333号

内 容 提 要

本书根据 Java 语言面向对象的本质特征以及面向对象程序设计课程的基本教学要求，在详细阐述面向对象程序设计基本理论和方法的基础上，详细介绍了 Java 语言及其面向对象的基本特性、基本技术。全书共分为 10 章，首先介绍了面向对象程序设计、Java 语言的基础知识，而后详细讲述 Java 语言中面向对象思想的实现以及使用，最后介绍了 Java 图形用户界面、Applet、数据库等相关知识。

书中采用大量的实例进行讲解，力求通过实例使读者更形象地理解面向对象思想，快速掌握 Java 编程技术。本书难度适中，内容由浅入深，实用性强，覆盖面广，条理清晰。每章附有精心编写的实验和习题，便于读者实践和巩固所学知识。本书可作为普通高等院校 Java 程序设计课程的教材，也可作为读者的自学用书。

普通高等学校计算机教育“十二五”规划教材

Java 面向对象程序设计（第 2 版）

◆ 主　　编　韩　雪
副 主 编　王维虎
责任编辑　刘　博

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京新华印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：22　　2012 年 9 月第 2 版
字数：576 千字　　2012 年 9 月北京第 1 次印刷

ISBN 978-7-115-29041-0

定价：42.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

前言

面向对象程序设计已经成为软件编程技术中一项非常关键的技术。相比过程化程序设计技术，面向对象程序设计中的继承、封装、多态等特性更接近于人的语言和思维，从而更容易理解和使用。与此同时，面向对象程序设计更加符合现代软件大规模开发的需求，有利于软件复用。

Java 语言是面向对象程序设计语言中的代表，相比 C++，Java 语言更全面地体现了面向对象的思想。Java 语言诞生于 1995 年，短短十几年，Java 语言已经遍布软件编程的各个领域。随着 Internet 的飞速发展，Web 得到广泛的应用，而 Java 语言在 Web 应用方面所表现出的强大特性，使得 Java 语言成为 Web 开发的主流技术。

由于 Java 语言具有简单易学、面向对象、使用范围广等特征，因此，非常适合于普通高等院校程序设计课程，尤其是面向对象程序设计课程。本书采用循序渐进、由浅入深、概念与例子相结合的编写方式，对内容的安排、例程的选择、习题的编写都进行了严格控制，确保难度适中，更贴近于实用。

在学习本书之前，读者应具有基本的计算机操作基础，但不必具有编程基础。掌握一门语言最好的方式就是实践，本书的着眼点是将基础的理论知识讲解和实践应用相结合，使读者在理解面向对象的思想上，快速掌握 Java 编程技术。

全书共分 10 章，在大多数章节中，首先对相关的基础知识进行介绍，然后重点讲解相关的实例。其中，第 1 章对面向对象程序设计和 Java 语言进行简要介绍。第 2 章介绍了 Java 语言的基本语法。第 3 章～第 5 章是本书的重点，详细讲述 Java 语言的面向对象特性，包括 Java 语言中类、对象、继承、多态、接口和内部类等重要概念及其应用实践。第 6 章介绍 Java 中的输入/输出以及异常机制。第 7 章讲述如何利用 Java 编写图形用户界面。第 8 章讲述 Applet 的使用，包含如何编写 Applet 以及如何在浏览器中运行 Applet。第 9 章在简要讲述 TCP/IP、UDP、Socket 协议的基础上，介绍如何利用 Java 语言编写网络应用。第 10 章为 Java 的高级应用，包含 Java 的多线程技术、JSP 和 Servlet 及数据库技术。

本书在每章之后附有习题和上机指导，供读者练习实践以检验学习效果。

本书中所有例题和相关代码已调试通过，并根据本书内容制作了电子课件，供老师教学时参考使用。

最后感谢读者选择本书，由于时间仓促和作者的水平有限，书中错误和不妥之处在所难免，敬请批评指正。

编　者

2012 年 7 月

目 录

第 1 章 Java 语言概述

Java 是 Sun Microsystem 公司研究开发的一种新型的程序设计语言。在众多高级语言中，Java 语言脱颖而出。它不仅成为最为流行的计算机语言之一，而且形成一种专门的技术。

1.1 面向对象程序设计

面向过程和面向对象是两种主要的程序设计理念。面向过程是早期程序设计的主要方式，近些年来，面向对象逐渐成为程序设计的主要方式。面向对象的编程思想由来已久，但真正意义上的纯面向对象编程语言目前只有 Java。

本节将对面向对象的基础知识进行简单的介绍，主要包括面向对象程序设计思想的诞生、面向对象与面向过程程序设计思想的对比、面向对象技术的背景和特点等三方面的内容。

1.1.1 面向对象程序设计思想的诞生

随着软件复杂度的提高，以及 Internet 的迅猛发展，原先面向过程的软件开发方式已经很难满足软件开发的需要。针对日趋复杂的软件需求挑战，面向对象的软件开发模式诞生了。目前作为针对软件危机的最佳对策，面向对象（Object Oriented，OO）技术已经引起人们的普遍关注。许多编程语言都推出了面向对象的新版本，一些软件开发合同甚至也指明了必须使用基于 OO 的技术和语言。下面简要列出了 OO 技术的发展历程。

（1）诸如“对象”和“对象的属性”这样的概念，可以一直追溯到 20 世纪 50 年代初，首先出现于关于人工智能的早期著作中。然而，OO 的实际发展开始于 1966 年，开发了具有当时更高级抽象机制的 Simula 语言。

（2）Simula 语言提供了比子程序更高一级的抽象和封装，并且为仿真一个实际问题，引入了数据抽象和类的概念。后来一些科学家吸取了 Simula 类的概念，开发出了 Smalltalk 语言。

（3）几乎在同时，“面向对象”这一术语被正式确定。在 Smalltalk 中一切都是对象——即某个类的实例。最初的 Smalltalk 世界中，对象与名词紧紧相连。

（4）Smalltalk 语言还影响了 20 世纪 80 年代早期和中期的很多面向对象语言，如 Objective-C（1986 年）、C++（1986 年）、Self（1987 年）、Eiffel（1987 年）、Flavors（1986 年）。同时，面向对象的应用领域也被进一步拓宽，对象不再仅仅与名词相联系，还包括事件和过程。

（5）到了 20 世纪 90 年代，随着 Internet 的迅猛发展，Sun 公司于 1995 年推出了纯面向对象的 Java 语言，自此之后，OO 技术在开发中越来越占主导地位。

Smalltalk 被认为是第一个真正面向对象的语言，现在国外还有一些开发人员在使用，对该语言感兴趣的同学可以参阅其他相关资料做更详细的了解。

1.1.2 面向对象与面向过程的对比

传统的过程化程序设计通过设计一系列的过程（即算法）来求解问题。这些过程一旦被确定，下一步就要开始寻找存储数据的方式，即“程序=算法+数据结构”。其中，算法是第一位，而数据结构是第二位。而面向对象的程序设计（Object Oriented Programming，OOP）调换了这个次序，将数据放在第一位，然后再考虑操纵数据的算法。

在 OOP 中，程序被看作是相互协作的对象集合，每个对象都是某个类的实例，所有类构成一个通过继承关系相联系的层次结构。面向对象的语言通常具有以下特征。

- 对象生成功能
- 消息传递机制
- 类和遗传机制

这些概念可以并且也已经在其他编程语言中单独出现，但只有在面向对象语言中，它们才共同出现，并以一种独特的合作方式互相协作，互相补充。实际上，软件开发的过程就是人们使用各种计算机语言将自身关心的现实世界（问题域）映射到计算机世界的过程，这个过程通常如图 1-1 所示。

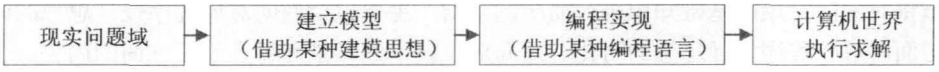

图 1-1 问题域映射过程

面向对象方法的软件开发主要经历如下 3 个阶段。

- 面向对象的系统分析（Object-Oriented System Analysis，OOA）。系统分析的主要任务是通过对用户需求进行分析确定系统的整体功能，即系统要做什么。
- 面向对象的系统设计（Object-Oriented System Design，OOD）。系统设计的核心是确定系统应怎样做。
- 面向对象的编程（Object-Oriented Programming，OOP）。

OOA 强调直接针对要开发的系统、客观存在的各种事物建立 OOA 模型。系统中有哪些值得考虑的事物，OOA 模型就有哪些对象；即客观世界与面向对象存在着一一对应的映射关系。面向对象分析方法用属性描述事物的静态（状态）特性；用方法描述事物的动态行为。OOA 分析方法的核心思想是利用面向对象的概念和方法为软件分析建造模型，从而将用户需求逐步细化、完整、精确。OOA 分析方法的大致步骤为：识别对象、属性及外部服务，识别类及其结构，定义对象之间的消息传递等。

OOD 是对 OOA 产生的结果增添实际的计算机系统中所需的细节，如人机交互、任务管理和数据管理的细节。从 OOA 到 OOD 是一个模型扩充过程。OOD 包括两种：一是将 OOA 模型直接引入而不必转换，只作细节修正与补充；二是针对具体实现中的人机界面、数据存储、任务管理等因素运用面向对象的方法进行模型扩充。OOD 可以分为四部分：问题空间部分的设计（Problem Domlain Component，PDC）、人机交互部分的设计（Human Interface Component，HIC）、任务管

理的设计（Task Management Component，TMC）、数据部分的设计（Data Management Component，DMC）。

OOP 是与面向过程编程完全相反的思考模式，利用面向对象的特性，把问题中的所有数据及操作过程，一一封装成独立的对象，而对象之间的关系即是对象彼此之间如何传递消息（Message）。就像日常生活中每个人之间的关系，如师生关系、亲子关系、朋友关系等。即，面向对象是以人类角度观察世界的思维方式。OOP 的目标在于创建软件重用代码，具备更好地模拟现实世界环境的能力。

本小节将分别介绍面向对象与面向过程的编程模式，进而帮助读者更加清楚地了解两者之间的区别。

1. 面向过程的编程模式

在这种编程模式中，数据和函数（过程）是分开的，即开发人员看到的是函数或过程的集合以及单独的一批数据。程序的处理过程为：参数输入→函数/过程代码→结果输出，其编程模式如图 1-2 所示。

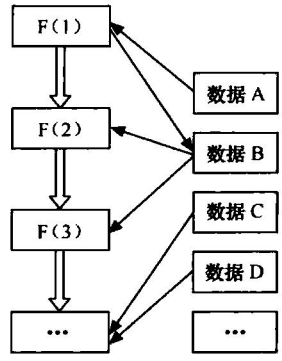

图 1-2　面向过程的编程模式

图 1-2 中的“F（1）”、“F（2）”、“F（3）”表示函数（过程），细线箭头表示函数（过程）对数据的访问。

从图 1-2 可以看出，对于软件维护人员来说，无论是函数还是数据结构的改动，都会使整个程序受到干扰，进而可能引发软件系统的崩溃，可以说是“牵一发而动全身”，程序的维护和扩展几乎难以进行。

通过前面的分析还可以看出，在面向过程的编程模式中，开发人员分析了问题之后，得到一个面向过程的模型，其过程如图 1-3 所示。

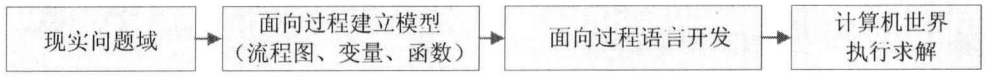

图 1-3　面向过程的基本开发过程

2. 面向对象的编程模式

在这种模式中，函数和它需要存取的数据封装在称为对象的单元中，对象之间的数据访问是间接的，是通过接口进行的。这里所说的接口是指为其他代码调用提供的一套访问方法，即代码的 API，其编程模式如图 1-4 所示。

- 图 1-4 中每一个圆表示一个对象，细线箭头表示对象间的消息（调用）。

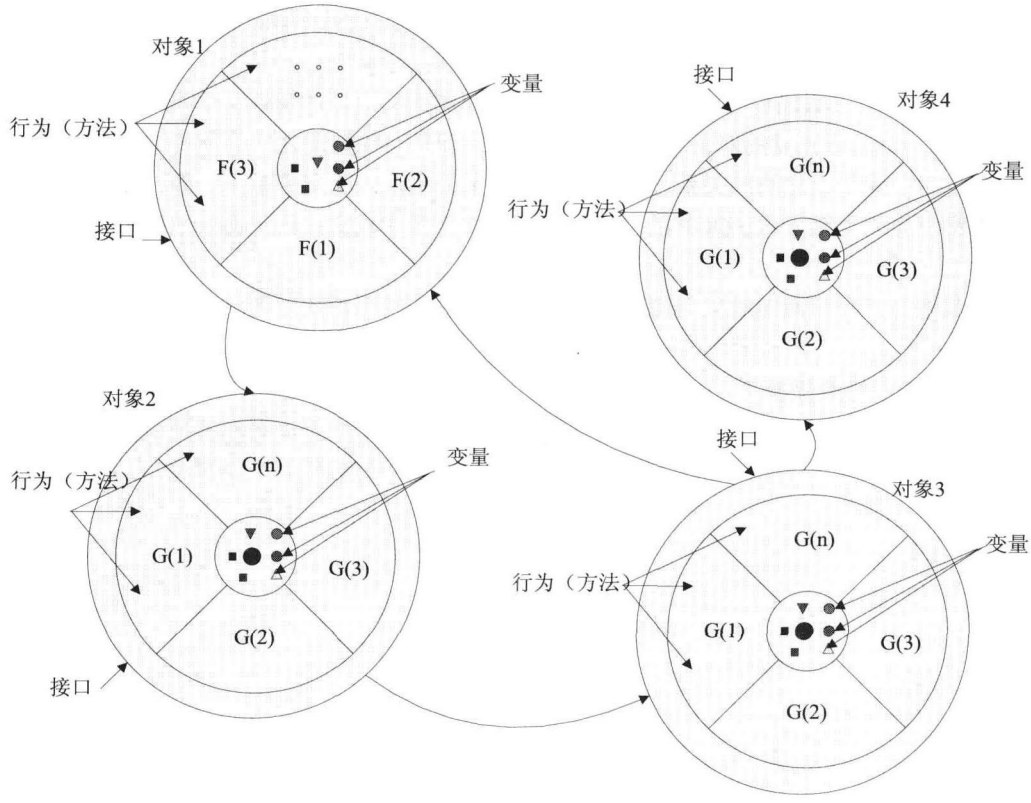

图 1-4　编程模式

- 通常将对象比作鸡蛋，蛋黄是数据，蛋清是访问数据的函数，蛋壳代表接口（即那些公开或公布的方法和属性）。
- 蛋壳接口隐匿了函数和数据结构的实现，当数据结构和内部函数变化时，这种变化被限制在内部的局部范围内。由于接口的相对稳定性，使得这种内部变化的影响不会波及到其他对象，除非蛋壳破裂（即接口发生变化）。

从上面的分析可以看出，通过使用面向对象的开发模式，可以解决面向过程中“牵一发而动全身”的问题，大大提高了程序的灵活性和可维护性。

在面向对象的编程模式中，程序的功能是通过对象间的通信获得的。对象被定义为一个封装了状态（数据）和行为（操作）的实体。

- 状态包含了执行行为的信息，以数据形式存储于对象之中。
- 消息是对象通信的方式，也是获得功能的方式。对象收到发给它的消息后，或者执行一个内部操作，或者再去调用其他对象的操作。

在面向对象编程模式中，开发人员先得到一个面向对象的模型，其中常见的词语是类、对象、方法、消息等，其基本过程如图 1-5 所示。

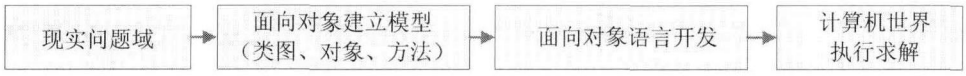

图 1-5　面向对象的基本开发过程

从上面的对比可以看出，面向过程更接近于计算机世界的物理实现，而面向对象的思想则更

符合人们的认知习惯。面向对象作为一种开发模式，为软件开发的整个过程（从分析设计到编码）提供了一套完整的解决方案。

1.1.3　面向对象技术的背景和特点

OO 是一种方法，一种思想，同时又是一种技术。OO 力求更客观自然地描述现实世界，使分析、设计和实现系统的方法同认识客观世界的过程尽可能一致。

对象是客观世界中的事物在人脑中的映像，这种映像通过对同一类对象的抽象反映成为人的意识，并作为一种概念而存在。例如，当人们认识到一种新的事物——苹果，于是人们的意识当中就形成了苹果的概念。这个概念会一直存在于人们的思维当中，并不会因为这个苹果被吃掉而消失。

这个概念就是现实世界中的事物在人们意识中的抽象。只要这个对象存在于人们的思维意识当中，人们就可以藉此判断同类的东西。下面将详细介绍面向对象技术的背景和特点。

1. 面向对象技术的背景

客观世界是由许多不同种类的对象构成的，每一个对象都有自己的运动规律和内部状态，不同对象之间相互联系、相互作用。例如，一辆自行车具有的状态是轮胎个数、车身颜色、车灯个数、车速等；具有的行为是加速、刹车、减速等。所有东西都是对象，可将对象理解为一种变量，对象保存着数据，并且可对自身进行操作。

面向对象技术是一种从组织结构上模拟客观世界的方法，它从组成客观世界的对象着眼，通过抽象，将对象映射到计算机系统中，又通过模拟对象之间的相互作用、相互联系来模拟客观世界，描述客观世界的运动规律。面向对象技术以基本对象模型为单位，将对象内部处理细节封装在模型内部，并且重视对象模块间的接口联系和对象与外部环境间的联系，能层次清晰地表示对象模型。

2. 面向对象技术的特点

通过前面的讨论，可以看出对象有以下几个特点。

- 某类对象是对现实世界具有共同特性的某类事物的抽象。
- 对象蕴涵许多信息，可以用一组属性来表征。
- 对象内部含有数据和对数据的操作。
- 对象之间是相互关联和作用的。

面向对象技术，正是利用对现实世界中对象的抽象和对象之间相互关联和作用的描述来对现实世界进行模拟，并且使其映射到目标系统中。所以面向对象的主要特点概括为：抽象性、继承性、封装性和多态性，本章后面的小节将详细讲解。

从编程开发的角度来看，所谓对象就是协调数据存储以及作用于数据之上操作的独立实体。用户可以通过定义一个对象集合以及它们之间的相互作用来创建一个面向对象的程序，许多对象协同工作来定义一个完成用户需要的软件系统。

1.2　Java 概述

本节首先对 Java 语言进行简述，包含 Java 的起源和发展，Java 的特性和 Java 的体系结构，使读者对 Java 有一个初步的认识。

1.2.1 Java 的起源和发展

Java 起源于 1994 年，美国 Sun Microsystem 的 Patrick Nawghton、Jame Gosling 和 Mike Sheridan 等人组成的开发小组，开始了代号为 Green 的项目的研制，其目标是研制一种开发家用电器的逻辑控制系统，产品名称为 Oak。1995 年 1 月，Oak 被更名为 Java。这个名字来自于印度尼西亚的一个盛产咖啡的小岛的名字，小岛的中文名叫爪哇。正是因为许多程序设计师从钟爱的热腾腾的香浓咖啡中得到灵感，因而热腾腾的香浓咖啡也就成为 Java 语言的标志。

2006 年底，Sun 公司发布了 Java Standard Edition 6（Java SE 6）的最终正式版，代号“Mustang（野马）”，跟“Tiger（Java SE 5）”相比，“Mustang”在性能方面有了不错的提升。从与“Tiger”在 API 类库的比较来讲，有了大幅加强，虽然“Mustang”在 API 库方面的新特性显得不太多，但其提供了许多实用和方便的功能：在脚本、Web Service、XML、编译器 API 数据库、JMX、网络方面都有不错的新特性和功能加强。

2010 年底，Sun 公司发布了 Java Standard Edition 7（Java SE 7）的相关版本。该版本的 JDK 主要在模块化和运行性能上做了一些重要改变。

1.2.2 Java 特点

随着 Internet 的飞速发展，人们的学习、工作、科研、商业和生活方式随之发生了巨大变化。人们不仅需要具有声音、图像和动画等多媒体信息的 Web 页面，以及实时视频、多用户网络游戏等，而且要求能向用户提供更好的实时交互性，并具有平台无关性。Java 的出现，使人们看到了解决以上问题的希望。Java 之所以能解决这些问题是与其自身的特点分不开的。Java 具有如下特点。

1. 简单性

由于 Java 语言与 C 语言类似，只要学过 C 语言的人，就可以很容易地掌握。而 Java 语言本身编写容易，语法简单，稍有程序设计经验的人，很快就能上手。

2. 平台无关性

Java 执行环境与使用平台无关，可以方便地将 Java 部署到任何不同平台的机器上。同时，Java 的类库封装了不同平台上的实现，为其提供统一的接口，这使得同样的类库可以在不同的平台上使用。这也就意味着用 Java 开发的应用可以“一次开发，随处运行”。

3. 分布式

Java 在网络方面的强大易用是其他任何语言无法比拟的，可以说 Java 是面向网络的语言。通过其提供的类库可以方便地处理各种网络协议，方便地进行传统的套接字网络开发，如 RMI、CORBA、Web 服务等现在流行的网络开发。

4. 健壮性

Java 在编译和运行时，都会对程序可能出现的问题进行检查，并将出错信息报告给程序员。同时，其提供垃圾收集机制来自动管理内存，避免了程序员无心的错误和恶意的攻击。

5. 安全性

在安全性方面，Java 表现得非常出色，从一开始，Java 就被设计为有防范各种病毒、袭击的能力。例如，一切对内存的访问都必须通过对象的实例引用来实现；禁止破坏自己处理空间之外的内存；禁止运行时堆栈溢出；未经授权禁止读写文件等。

6. 浏览器应用

Applet 是只能运行在 Web 浏览器里的小程序。Applet 作为 Web 页面的一部分自动下载，就

像图片一样自动下载，程序员只需要创建一个简单的程序，就能让它自动地运行于任何机器，只要这台机器装上了内置有 Java 解释器的浏览器就行了。

1.2.3　Java 7 的新特性

Java 7 是最新发布的版本，它主要有如下特性。

1. 模块化（Modularization）

在具体打造 Java SE 7 平台时，把以前的平台打散成更小的、独立的若干模块。各个独立的模块能够根据 Java VM 或者 Java 应用的需要被分别下载。这极大地减小了用户机器上 Java Runtime 的大小。模块化的另一个好处是 Java SE 7 平台的下载更小了，这样就无形中加快了启动速度。更小的内存需求也使得执行性能得到极大的提高，特别是对桌面应用程序。一个更小的平台也意味着它可以适用在内存不多的设备上。

2. 多语言支持（Multi-Language Support）

所谓多语言不是指中文英文之类的语言，而是说 JDK 7 的虚拟机对多种动态程序语言增加了支持，如 Rubby、Python 等。对这些动态语言的支持极大地扩展了 Java 虚拟机的能力。对于那些熟悉这些动态语言的程序员而言，在使用 Java 虚拟机的过程中同样可以使用它们熟悉的语言进行功能的编写，而这些语言是跑在功能强大的 JVM 之上的。

3. 开发者生产力（Developer Productivity）

所谓开发者生产力，就是开发效率。JDK 7 通过多种特性来增强开发效率。如对语言本身做了一些细小的改变来简化程序的编写；在多线程并发与控制方面——轻量级的分离与合并框架，一个支持并发访问的 HashMap 等。通过注解增强程序的静态检查。提供了一些新的 API 用于文件系统的访问、异步的输入输出操作、Socket 通道的配置与绑定、多点数据包的传送等。

4. 性能（Performance）

这里的性能是指程序的执行效率，JDK 7 的最显著特性是压缩了 64 位的对象指针，即通过对对象指针由 64 位压缩到与 32 位指针相匹配的技术使得内存和内存带块的消耗得到了很大的降低，因而提高了执行效率。此外还提供了新的垃圾回收机制（G1）来降低垃圾回收的负载和增强垃圾回收的效果。G1 垃圾回收机制拥有更低的暂停率和更好的可预测性。

1.2.4　Java 体系结构

虽然 Java 常用作开发各种应用程序的编程语言，但是作为编程语言只是 Java 的众多用途之一，而用途更广泛的是其底层架构。完整的 Java 体系结构实际上由 4 个组件组合而成：Java 编程语言、Java 类文件格式、Java API 和 JVM。

其中，JVM 是 Java Virtual Machine（Java 虚拟机）的缩写，它是一个虚构出来的计算机。它是通过在实际的计算机上仿真模拟各种计算机功能来实现的。JVM 有自己完善的硬件架构，如处理器、堆栈、寄存器等，还具有相应的指令系统。因此，使用 Java 开发时，本质就是用 Java 编程语言编写代码，然后将代码编译为 Java 类文件，接着在 JVM 中执行类文件。

Java API 是预先编写的代码，并按相似主题分成多个包。现在，Java API 主要分为以下 3 大平台。

- Java SE（Java Standard Edition）：该平台中包含核心 Java 类和 GUI 类。
- Java EE（Java Enterprise Edition）：该平台中包含开发 Web 应用程序所需的类和接口，有 Servlet、Java Server Pages 以及 Enterprise JavaBean 类等。
- Java ME（Java Micro Edition）：该平台体现了 Java 的传统优势，为消费类产品提供了一

个已优化的运行时环境，用于传呼机、手机或汽车导航系统等。

在 Java SE 5.0 以前，这 3 个平台分别称为 J2SE（Java 2 Standard Edition）、J2EE（Java 2 Enterprise Edition）、J2ME（Java 2 Micro Edition）。

图 1-6 所示为 Java 不同功能模块之间的相互关系，以及它们与应用程序、操作系统之间的关系。

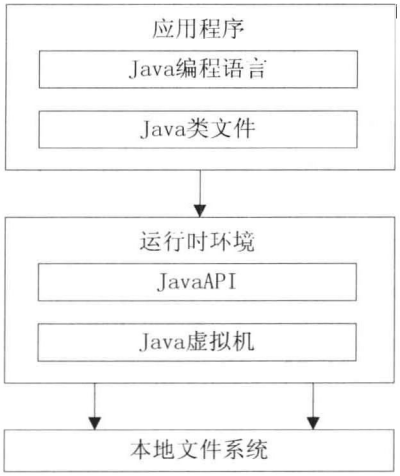

图 1-6　Java 功能模块及模块间关系图

从图 1-6 中可以看出，JVM 与核心类共同构成了 Java 平台，也称为 Java 运行时环境（Java Runtime Environment，JRE）。JRE 是 Java 体系结构的核心，上层是应用程序包括 Java 语言源程序和类文件，底层是异构的操作系统平台。JRE 可以建立在任意操作系统上。

1.3　Java 运行机制与 JVM

在 1.2 节中讲到，在 Java 体系结构中，JVM 处在核心的位置，是程序与底层操作系统和硬件无关的关键。本节将从 JVM 的体系结构和运行过程这两个方面来对 JVM 进行深入介绍。

1.3.1　JVM 的体系结构

Java 语言的一个非常重要的特点就是与平台的无关性，而使用 JVM 是实现这一特点的关键。一般的高级语言如果要在不同的平台上运行，至少需要编译成不同的目标代码。而引入 JVM 后，Java 语言在不同平台上运行时不需要重新编译即可运行。

JVM 屏蔽了与具体平台相关的信息，使得 Java 语言编译程序只需生成在 JVM 上运行的目标代码（字节码），就可以在多种平台上不加修改地运行。JVM 在执行字节码时，把字节码解释成具体平台上的机器指令执行。

在介绍 JVM 体系结构之前首先要明确一个概念，什么是 JVM？JVM 是一个想象中的机器，在实际的计算机上是通过软件模拟来实现。JVM 有自己虚拟的硬件，如处理器、堆栈、寄存器等，还具有相应的指令系统。

每个 JVM 包括方法区、堆、Java 栈、程序计数器和本地方法栈这五个部分，这几个部分和类装载机制与运行引擎机制一起组成的体系结构如图 1-7 所示。