中国电子教育学会高教分会推荐

普通高等教育电子信息类“十三五”课改规划教材

Java实用程序设计

PRACTICAL JAVA PROGRAMMING

孙聪　李金库　李辉　马建峰　编著

西安电子科技大学出版社
http://www.xduph.com

中国电子教育学会高教分会推荐
普通高等教育电子信息类“十三五”课改规划教材

Java 实用程序设计

孙　聪　李金库　李　辉　马建峰　编著

西安电子科技大学出版社

内容简介

本书强调Java语言核心原理和常见技术的应用，突出实用性和实践性，尽可能采用简洁的实例来介绍语言特征。本书的主要内容包括：面向对象程序设计基础，Java的基本语法及面向对象程序设计方法，Java的高级特性，容器类及常用预定义类，异常处理，输入输出，GUI程序设计，网络与数据库程序设计等。本书的附录对Java语言的编程风格及Java虚拟机架构做了简要介绍。

本书适合作为高等学校计算机、通信和电子等相关专业的本科生教材，也可供计算机软件相关技术领域的研究人员和工程人员参考使用。

图书在版编目(CIP)数据

Java实用程序设计/孙聪等编著. —西安：西安电子科技大学出版社，2015.8
高等学校计算机专业“十三五”规划教材

ISBN 978-7-5606-3753-2

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

中国版本图书馆CIP数据核字(2015)第165311号

策划编辑 刘玉芳
责任编辑 刘玉芳
出版发行 西安电子科技大学出版社(西安市太白南路2号)
电 话 (029)88242885 88201467 邮 编 710071
网 址 www.xduph.com 电子邮箱 xdupfxb001@163.com
经 销 新华书店
印刷单位 陕西天意印务有限责任公司
版 次 2015年8月第1版 2015年8月第1次印刷
开 本 787毫米×1092毫米 1/16 印张 20.5
字 数 487千字
印 数 1～2000册
定 价 36.00元
ISBN 978-7-5606-3753-2/TP

XDUP 4045001-1

前　言

Java是一种优秀的面向对象程序设计语言，历经20年的发展，已经成为了目前使用范围最广的通用编程语言。在TIOBE编程语言排行榜中，Java语言和C语言长期牢牢占据着排行榜的前两位。

作为大学计算机本科必修的一门语言课程，Java语言受到了广泛的重视，与其相关的教材和教学方法也很多，其中有的从语言机制和语法机制的角度进行讲解，有的则以Java语言为载体介绍面向对象的程序设计思想。然而无论如何编排，想要在30～40个学时的课程学习中掌握一门特征如此丰富和实用的语言都是不容易的。国外一些先进教材的内容往往过于庞杂，用于教学时，我们经常难以切断其中的一些关联性，而使得教学内容的选择较为困难。而国内许多教材则偏重于知识点的罗列，对于一些已经过时的技术仍用大量篇幅进行介绍，而对于一些新的前沿技术则一笔带过，无法反映Java技术的发展及应用。

本书在深入介绍Java语言基本机制和技术特性的基础上，具有一些明显的特点：

(1) 本书侧重介绍Java的实用特征，用专门的章节介绍了Java类库中的集合类和常用预定义类，引导读者实现一些实用程序，以此联系读者此前所学的数据结构、算法设计与分析等方面的知识，同时让读者体会Java在复杂程序开发过程中的优势。

(2) 选择的实例尽可能简洁，用尽可能简单的例子揭示出尽可能多的Java特性，一方面使得课堂教学比较容易，另一方面也减少了学生理解程序的时间，容易引起学生的兴趣。

(3) 针对Java语言的发展介绍了一些新机制，如NIO、枚举类型、正则表达式、线程执行器Executor、SWT与Eclipse插件开发、自动装箱与拆箱、XML、大数据处理等。

(4) 提及了一些其他教材中不常涉及的容易引起错误的细节，如隐式和显式的类型转换，字符串操作和比较，以及类加载、连接和初始化顺序等。

(5) 附录简要介绍了Java虚拟机的规范，将Java语言概念与Java虚拟机规范、class文件格式及虚拟机实现更紧密地结合起来，使学生能够从实现机理上理解Java语言中一些较难掌握的概念，如static关键字、final关键字、this引用、对象的生命周期、子类的初始化、类的加载过程与加载时间等。

此外，本书的练习题强调实践性，其中的一些已经作为实际课程教学中的上机习题。读者在学习本书的过程中，应尽量动手去实现这些练习程序，以便切实提高自己的编程能力。事实上，没有任何课堂讲授或书籍阅读能够代替实际编程来教会你一种程序设计语言。本书的所有实例均在Java SE 7下进行编译并能正常运行，读者可以通过运行和更改它们加快自己对Java语言的理解。

在本书的编写中，对Java技术的介绍是有侧重点和选择性的。我们并没有提及Applet、断言等Java特性，原因是这些特性要么有些过时，要么在当前集成开发环境占据主流地位的背景下难有用武之地。我们也没有介绍JDK的bin目录下众多的开发工具，例

如 javah. exe、javadoc. exe、jdb. exe、jar. exe 等，它们实际上已可以被 Ant、Maven 和集成开发环境取代了。如果读者希望了解如何生成一个 jar 包或如何生成 Java 文档，可以首先去阅读与具体集成开发环境相关的技术文档。

参与本书编写的老师均常年在教学第一线担任相关的教学工作，对 Java 程序设计、C 语言、程序设计基础、网络程序设计、数据库与数据挖掘、计算机网络和信息安全等课程有着非常丰富的教学经验。不仅如此，他们还从事计算机软件理论、系统和网络安全领域的科研工作，对于 Java 语言在计算机学科中的作用以及在相关研究领域的使用方法有较深的理解，能够利用在科研和工程实践中的实际经验对 Java 语言的诸多特性进行有针对性的介绍。

根据本书的技术特点，编写人员的分工如下：孙聪(主编，编写第 1 章～第 8 章、第 10 章～第 11 章)，李金库(编写第 9 章)，李辉(编写第 12 章)，马建峰(主审并指导全书编写)。全书由马建峰、孙聪、万波审定。

本书的编写得到了西安电子科技大学教材基金资助。

由于 Java 技术发展很快，而本书作者的水平有限，在编写中难免会有不足之处，恳请读者提出宝贵的建议。

编　者

2015 年 2 月

目　录

第1章 Java概述

1.1 Java的发展历史与地位

1. Java的诞生过程

从1972到1991年的20年间，计算机硬件的性能有了很大的提升，同时价格稳步下降，对复杂软件的需求呈现指数级增长。为了满足这一需求，人们发明了越来越多的软件开发技术。当时，Dennis Ritchie于1972年开发的C语言仍是很多程序员的首选语言，在面向对象程序语言出现之前，C语言成功地流行了很长一段时间。然而，程序员们渐渐发现C语言的结构化语法十分冗繁，这促使了面向对象程序设计(Object-Oriented Programming，OOP)这一方法学的提出，该方法学实现了更好的重用性，大大提高了开发效率。早期面向对象程序语言的代表是由Alan Kay研制的Smalltalk，并于1980年首度对外发布，标志着OOP思想的确立。20世纪80年代初，Bjarne Stroustrup研发了C++语言，该语言在C语言的基础上加入了OOP的基本要素与特征。在Java语言出现并流行之前，C++是程序设计语言领域的主导者。

1990年12月，Sun启动了一个称做Stealth的内部项目，该项目的最初目的是创建一种能够替代C和C++的新程序语言工具，Patrick Naughton是这个项目的主要发起者和参与者之一。作为Sun公司的工程师，Naughton一直认为Sun公司的C/C++应用程序接口和工具难以使用。后来，项目团队更名为Green并由James Gosling领导。最初Gosling希望对C++进行改进和扩充，提出了一种叫“C++ ++ --”的语言，意指对C++加入一些特性并去掉一些特性。但很快他们放弃了扩充C++的想法。理由是：

(1) C++需要的内存过多且过于复杂；

(2) C++没有垃圾收集机制，需要程序员人工管理内存；

(3) C++没有可移植的安全机制和多线程机制；

(4) 没有一种能够方便地将C++程序移植到各类型设备上的平台，随着设备硬件的频繁更新换代，相应的C++控制代码也需要重新编译和调试，给设备更新带来很大负担。

基于以上原因，Gosling决定实现一种全新的语言，他将这种语言命名为Oak，此灵感来自Gosling办公室外的一棵橡树。

1992年9月3日，Green团队展示了一个称为StarSeven(*7)的PDA设备，该设备将GreenOS操作系统、Oak语言及程序库、用户接口等集成于硬件之上，这是Oak语言的首次对外呈现。1994年6月和7月，Gosling、Naughton、Bill Joy(Sun公司联合创始人之一)、John Gage(时任Sun公司科学办公室主任)等人研究决定，将团队的目标锁定在万维网(World Wide Web，WWW)。他们认为互联网正在朝着高度交互化的方向演变，这一趋

势与他们曾经对电视机顶盒的预测一致。Naughton 还实现了一个浏览器原型系统，称为 WebRunner，这一浏览器原型系统成为后来 HotJava 浏览器的雏形。这一年，Oak 语言被改名为 Java。

1995 年 5 月 23 日，Java 的第一个发布版本(Java 1.0a 和 HotJava 浏览器)由 John Gage 在 SunWorld 大会上发布，这标志着 Java 的正式诞生。当天，John Gage 还与 Netscape 公司联合创始人兼执行副总裁 Marc Andreessen 宣布了将 Java 技术应用于网景浏览器(Netscape Navigator)的消息。第一个支持 Java 的 Navigator 版本是 Navigator 2.0 的 32 位版，而在现在，几乎所有的浏览器都能支持 Java，由 Java 开发的 Internet 应用已成为目前互联网的主流应用模式之一。

2. Java 的版本历史

1996 年 1 月 9 日，Sun 公司成立了 JavaSoft 小组，负责 Java 相关技术的开发。1996 年 1 月 23 日，Sun 发布了第一个 Java 开发工具包 JDK 1.0。自此以后 Java 语言经历了多次更改和标准库增补。

1997 年 2 月 19 日，JDK 1.1 发布，主要增加了 AWT 事件模型、内部类、JDBC 等内容。

1998 年 12 月 8 日，Sun 发布了 Java 2 平台和 JDK 1.2，主要更改包括引入了 JIT 编译器、重写了事件处理机制、更改了线程同步机制、引入了 Swing 图形 API 等。JDK 1.2 在类与接口数量、方法与成员变量数量等关键指标上较 JDK 1.1 都有了巨大的发展，因此 Java 2 平台的诞生标志着 Java 技术发展进入了一个新阶段。

1999 年 6 月，Sun 重新定义了 Java 的技术架构，针对不同的市场目标，将 Java 2 平台分为三个版本：Java 2 标准版(J2SE)、Java 2 企业版(Java EE)和 Java 2 微缩版(J2ME)。J2SE 主要针对桌面电脑与工作站设备，Java EE 主要针对高端服务器系统，J2ME 主要针对消费类设备和嵌入式产品。自此开始，“J2SE”就代替“JDK”用于表示 Java 的标准平台，这一表示方式一直沿用到 J2SE 5.0。

2004 年 9 月 30 日，Sun 发布了 J2SE 1.5，这时 Sun 将 J2SE 的版本号由 1.5 变更为 5.0。J2SE 5.0 包含了泛型、可变数目参数、断言、自动装箱和拆箱等功能。2006 年 12 月 11 日，Sun 发布了 J2SE 6.0，在这一版本中，Sun 将 J2SE、J2EE、J2ME 分别改称 Java 平台标准版(Java SE)、Java 平台企业版(Java EE)和 Java 平台微缩版(Java ME)，同时去掉了版本号中的“.0”，因而 J2SE 6.0 改称 Java SE 6。目前 Java 的最新可下载版本是 Java SE 8，该版本于 2014 年 3 月发布，沿用至今。

1.2 Java 的技术体系

Java 技术既是一种语言也是一种平台。Java 编程语言(Java Language)是一种通用、并发、强类型、基于类的面向对象程序设计语言。Java 语言规范(Java Language Specification)是对 Java 编程语言的技术定义，包括 Java 编程语言的语法和语义。与 C/C++相比，Java 语言的特殊之处在于，程序的运行既要通过编译，又要通过解释，如图 1-1 所示。Java 源程序通过编译器进行编译，得到一种中间代码，称为 Java 字节码(bytecode)。Java 字节码是

一种与平台无关的目标文件格式，是运行于 Java 虚拟机(Java Virtual Machine，JVM)上的代码。Java 虚拟机中的字节码解释器将字节码依次转换为具体底层平台的机器码，使 Java 程序得以运行。

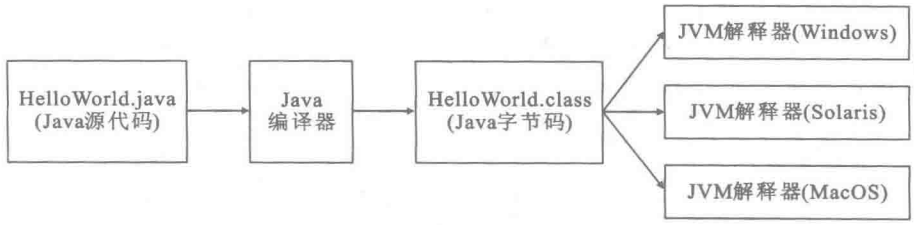

图 1-1　Java 程序的编译和执行过程

Java 字节码具有自己的规范，与具体的硬件和操作系统平台无关，因而可以在任何操作系统的 Java 虚拟机上运行。Java 技术体系通过实现与底层平台相关的 Java 虚拟机，为上层的 Java 应用程序提供平台无关性。在任意一个平台上编译得到的 Java 字节码，都可以在其他平台的 Java 虚拟机上运行，从而实现了“一次编程(编译)，到处运行”。

Java 平台是 Java 语言应用运行的特定环境，完全由软件构成并运行于不同的操作系统之上，使得 Java 应用程序与底层操作系统及硬件平台隔离。在现实中存在多种 Java 平台，很多开发者包括一些长期应用 Java 语言的开发者并不清楚这些平台之间的区别。目前，Java 平台的技术体系主要分为以下 3 个分支：

(1) Java 平台标准版(Java SE)；

(2) Java 平台企业版(Java EE)；

(3) Java 平台微缩版(Java ME)。

以上 3 种 Java 平台的主体均由 Java 虚拟机和应用程序接口(API)组成。Java 虚拟机是一个程序，能够针对特定的软件和硬件平台运行 Java 应用程序。API 是一系列软件组件，包括为开发 Java 应用程序而预定义的类和接口，可以用它们来创建其他软件组件或应用。Java 平台使得应用程序能够在任意兼容系统上运行并获得 Java 语言所具有的平台无关、稳定、易开发、安全等特点。下面具体介绍这些平台及其特点。

1. Java 平台标准版(Java SE)

Java SE 平台提供了开发和部署 PC 级和服务器级 Java 应用程序的开发工具、运行环境(虚拟机)与核心 API。图 1-2 中给出了 Java SE 平台中包括 Java 语言在内的组成部分。Java SE 平台的实现又称为 Java SE 开发工具集(JDK)。Java SE 运行时环境(JRE)是 JDK 的一个子集，然而在很多场景下，也可将 JRE 本身看做 Java SE 平台的实现。JDK 包括 JRE 和其他一系列必要的开发工具，JRE 提供 API 库、Java 虚拟机，以及其他运行应用程序所需的组件，这些组件一部分是 Java SE 规范所要求的，另一部分则不是。Java SE 的 API 提供了 Java 的核心功能。在 Java SE 8 版本中，引入了紧凑轮廓(Compact Profile)的概念，紧凑轮廓是 Java SE API 的子集，这些子集可以将 Java 运行时的静态存储开销降低到资源受限设备所能接受的程度。下面对 Java SE 中的概念进行分类简述。

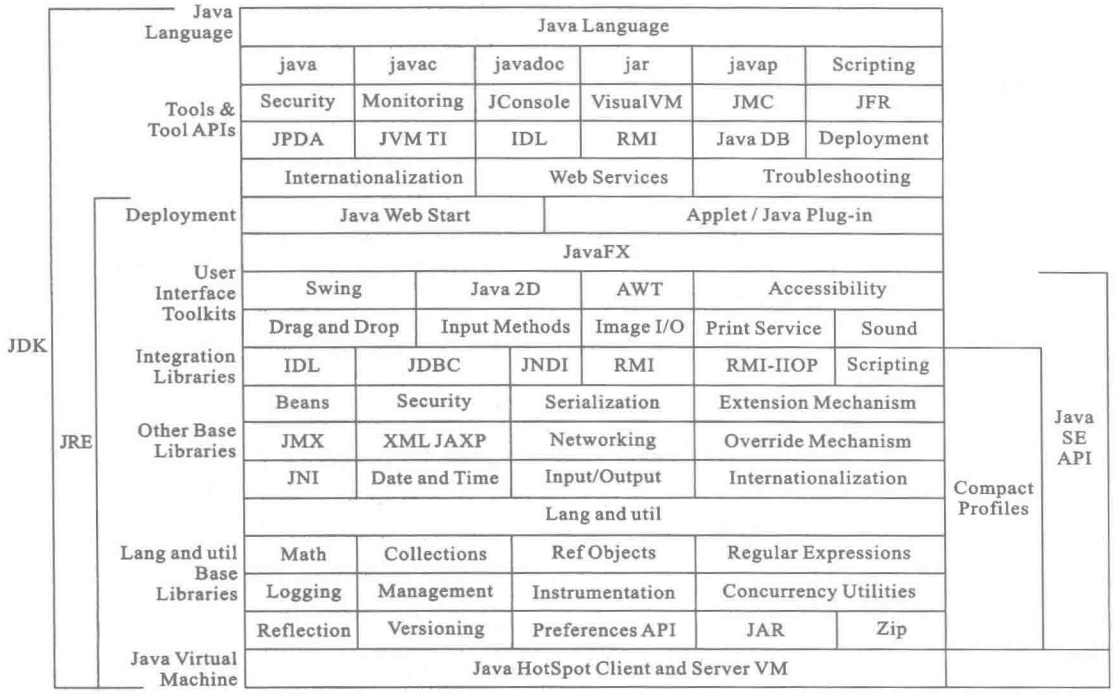

图 1-2 Java SE 概念图

1) Java 虚拟机(Java Virtual Machine)

Java 虚拟机是一个包含指令集并在运行时操作内存的抽象计算机。Java 虚拟机可被移植到不同的平台上，提供硬件和操作系统无关性。Java SE 平台提供两种不同的虚拟机实现：Java HotSpot Client VM 和 Java HotSpot Server VM。客户端虚拟机 Java HotSpot Client VM 的启动时间更短，内存消耗更小；而服务器虚拟机 Java HotSpot Server VM 则会最大化程序的执行速度。

2) 基本库(Lang and util Base Libraries，Other Base Libraries)

基本库包含了实现 Java SE 平台基本特征和功能的类和接口。其中，java. lang 包和 java. util包提供了几乎所有应用程序均需使用的基本功能。其他基本库还提供了输入/输出、对象串行化、网络通信、安全、国际化、JavaBeans 组件 API、Java 管理扩展(JMX)、Java XML 处理 API(JAXP)、Java 本地接口(JNI)等功能。

3) 集成库(Integration Libraries)

Java SE 平台的集成 API 提供以下主要功能：Java 数据库连接(JDBC)API、远程方法调用(RMI)、Java 接口定义语言(CORBA)、互联网 ORB 间通信协议上的远程方法调用(RMI-IIOP)、Java 平台脚本、Java 命名与目录接口(JNDI)API 等。

4) 用户接口工具集(User Interface Toolkits)

用户接口工具集包括：支持多种语言输入的输入方法框架；用于实现非常规输入/输出的高级访问工具 Accessibility；打印服务 API；用于声音捕获、处理与播放的 Java Sound API；拖放数据传递的 Drag & Drop；用于二维图形与成像的 Java 2D；图形用户界面库

AWT 和 Swing；支持对网络或文件中的图像进行操作的图像输入/输出 API；用于客户端应用开发的 JavaFX。其中 JavaFX 是在 Java SE 7 Update 2 版本之后引入的，JavaFX 平台从 Java 客户端平台演化而来，包含一系列图形和媒体包，使得应用程序开发者能够方便地创建和配置跨平台行为一致的富互联网应用(Rich Internet Applications，RIAs)。

5) Java 部署技术(Deployment)

Java 部署技术提供了对 Java 应用程序部署与运行的支持，包括以 Web 方式装载与运行 Java 应用程序的 Java Web Start 和支持 Applet 在浏览器中运行的 Java Plug-in 等。

6) Java 开发工具(Tools & Tools APIs)

标准的 JDK 工具包括以下几类：

(1) 基本工具(appletviewer，extcheck，jar，java，javac，javadoc，javah，javap，jdb)；

(2) 安全工具(keytool，jarsigner，policytool，kinit，klist，ktab)；

(3) 国际化工具(native2ascii)；

(4) 远程方法调用(RMI)工具(rmic，rmiregistry，rmid，serialver)；

(5) Java IDL 与 RMI-IIOP 工具(tnameserv，idlj，orbd，servertool)；

(6) Java 部署工具(javafxpackager，pack200，unpack200)；

(7) Java Web Start 工具(javaws)；

(8) Java 检修、轮廓、监控和管理工具(JConsole)；

(9) Java Web Services 工具(schemagen，wsgen，wsimport，xjc)。

7) Java 语言(Java Language)

一种通用、并发、强类型的面向对象程序设计语言。通常被编译成字节码指令序列，指令序列具有 Java 虚拟机规范所定义的二进制格式，并被虚拟机解释执行。

2. Java 平台企业版(Java EE)

Java EE 早期又称 J2EE，是一种利用 Java 2 平台来简化企业解决方案中开发、部署和管理相关复杂问题的体系结构。Java EE 技术的基础就是 Java SE，Java EE 不仅巩固了标准版中的许多优点，如“一次编程、到处运行”的特性、JDBC 数据库存取 API、CORBA 技术以及能够在 Internet 应用中保护数据的安全模式等，同时还提供了对 EJB(Enterprise Java Beans)、Java Servlets API、JSP(Java Server Pages)以及 XML 技术的全面支持。其最终目的是成为一个能够大幅缩短企业级应用的开发和投放市场时间的体系结构。

Java EE 体系结构提供中间层集成框架，用来满足用户的高可用性、高可靠性以及可扩展性的应用需求。通过提供统一的开发平台，Java EE 降低了应用的复杂性和开发成本，同时能够在一定程度上对现有应用程序集成进行支持，对应用的打包和部署有向导支持，有相应的安全机制。

Java EE 通常使用多层的分布式应用程序模型，应用程序的逻辑根据其实现的不同功能被封装到组件中，组成 Java EE 应用程序的大量应用程序组件，并根据其所属的层被安装到不同的机器中。典型的分布式 Java EE 应用程序可分为如下四层：

(1) 运行在客户端机器上的客户层组件；

(2) 运行在 Java EE 服务器中的 Web 层组件；

(3) 运行在 Java EE 服务器中的业务逻辑层组件；

(4) 运行在 EIS 服务器中的企业信息系统层软件。

Java EE 提供了一个框架，对框架的实现交由具体的应用服务器厂商完成，如 Apache 的 Tomcat 提供了对 JSP 和 Servlets 的支持，而 WebLogic 则能够为整个 Java EE 规范提供较完整的支持。Java EE 的核心技术随着 Java EE 版本的演化而演化，在 Java EE 7.0 版本中包含的核心技术主要包括以下几个。

(1) Java 事务(JTA/JTS)。事务是具有原子性、一致性、孤立性和持久性的工作单元，通常有“提交”和“回滚”两种终止方式。Java 事务服务(JTS)对事务分界和事务环境传播之类的服务提供支持，JTS 功能由应用程序通过 Java 事务 API(JTA)访问。

(2) Java Servlet API。Java Servlet API 提供了一种通信机制，可扩展任何使用了基于请求-响应协议(通常建立在 HTTP 行为基础之上)的服务器的功能。Java Servlet 实质上是一种 Java 类，该类的字节码被 Servlet 容器管理，通过一种由 Servlet 容器实现的请求-响应模型与 Web 客户机进行交互。通常 Servlet 可用于替代 CGI 脚本。

(3) Java 服务器页面(JSP)。JSP 是对 HTML 的一种扩展，它可以通过一些特殊的标签向静态 HTML 页面中插入动态的信息，使用不同的标签添加 Java 代码段、向页面写入表达式的值、引用 Java Bean 等。除此之外，还允许开发者创建自己的标签和相应的实现方法。JSP 提供的大多数功能与 Java Servlet 类似，但实现方式不同。Servlet 完全由 Java 编写并生成 HTML，而 JSP 则是在 HTML 中嵌入少量 Java 代码。

(4) 企业级 Java Bean(EJB)。EJB 为分布式商务应用提供了一个开发框架，简化了高度复杂的企业级应用的开发。EJB 服务器负责管理 EJB 容器，提供对操作系统服务的存取和 Java 相关的服务；EJB 容器负责管理部署于其中的 EJB，客户端应用程序通过由 EJB 容器生成的封装接口与具体 EJB 进行交互；EJB 客户端可以是 Servlet、JSP 或其他应用程序。

(5) Java 消息服务(Java Message Service，JMS)。JMS 是用于企业消息传递系统相互通信的应用程序接口。企业消息传递系统又称为面向消息的中间件(Message Oriented Middleware，MOM)，用 JMS 编写的应用程序能够在任何实现了 JMS 标准的 MOM 上运行。

(6) Java XML。XML 是一种可用于定义其他标记语言的元语言，还可用于在不同的主体之间共享数据。支持 XML 的 Java API 主要包括以下几类：用于 XML 解析的 JAXP、用于基于 XML 的 Web 服务的 JAX-WS、用于基于 XML 的远程方法调用的 JAX-RPC、用于 XML 消息传递的 JAXM、用于 XML 注册的 JAXR、用于 XML 绑定体系结构的 JAXB。

(7) Java 管理扩展(JMX)。Java 管理扩展的前身是 Java 管理 API。JMX 是一种 API、可扩展对象和方法的集合体，可以跨越异构的平台、体系结构和网络协议，开发无缝集成的面向系统、网络和服务的管理应用。

(8) 安全服务。Java 认证与授权服务(JAAS)是标准可插入式认证模块(PAM)架构的 Java 版本实现，使得服务能够对用户提供认证授权和访问控制。Java 容器认证服务提供者接口(JASPIC)定义一个服务程序接口，通过这一接口，实现消息认证机制的认证提供者可以被集成到客户端或服务器消息处理容器或运行时环境中。Java 容器授权契约(JACC)在 Java EE 应用服务器与授权服务提供者之间定义一种契约，使得传统授权服务提供者能够嵌入到任意 Java EE 应用中。

3. Java 平台微缩版(Java ME)

Java ME 为运行于嵌入式设备和移动设备上的应用程序提供了一种健壮、灵活的运行时环境。这些设备包括微控制器、传感器、手机、个人数字助理(PDA)、电视机顶盒、打印机等。作为一系列技术和规范的集合，Java ME 包含了灵活的用户接口、可靠的安全性、内建的网络协议，并支持在线应用程序和离线动态下载的应用程序。基于 Java ME 的应用可以兼容多种设备并充分使用每种设备的本地能力。

Java ME 体系结构自下而上分为主机操作系统、Java 虚拟机、配置(Configuration)、轮廓(Profile)四层。配置是 Java ME 体系结构中的一组规约，从可用内存类型与大小、处理器类型与速度、可用的网络连接类型等方面将设备分类，并对每类设备的软件环境进行定义。典型的配置中包括有限连接设备配置(CLDC)和连接设备配置(CDC)。轮廓则针对具体的设备类型对配置进行扩充，加入额外的类，形成特定类型设备上可用的 API 集合。应用程序针对特定的轮廓编写，可移植到支持该轮廓的任一设备上。典型的轮廓包括移动信息设备轮廓(MIDP)、PDA 轮廓(PDAP)等。

1.3　Java 的特征

Sun 公司在其 Java 语言白皮书中将 Java 语言描述为一种简单(simple)、面向对象(object oriented)、分布式(distributed)、解释型(interpreted)、健壮(robust)、安全(secure)、体系结构中立(architecture neutral)、可移植(portable)、高性能(high performance)、多线程(multithreaded)和动态(dynamic)的编程语言。

1. 简单

Java 的语法和语义都比较单纯，一些基本的语言特征继承自 C 语言和 C++语言，因而更容易学习和使用。Java 中取消了很多 C++中极少使用且难以理解的特性，如运算符重载、多重继承、自动强制转换等，还取消了 C 与 C++中普遍存在的指针类型，同时取消了 goto 语句，转而使用 break 和 continue 语句。Java 中不再有结构体、联合体、头文件、预处理程序等概念。与此同时，Java 实现了自动垃圾收集机制，避免了 C++要求程序员进行内存管理所可能导致的错误。正是基于以上特点，一些人认为 Java 可以看做是“C+++−−”，即类似 C++并取消了 C++的一些负面特性。此外，Java 还提供大量功能丰富的可重用类库，简化了编程工作量。

2. 面向对象

Java 是一种面向对象语言。这意味着程序开发主要关注“数据”以及操作数据的“方法”，而不是直接考虑程序执行过程。与 C++等源自面向过程语言不同，Java 从诞生伊始就是面向对象的。在面向对象系统中，现实世界中的事物由对象(object)进行建模，编程即创建对象、操作对象并使得对象共同工作。为此，Java 基于“类”(class)的概念建立对对象的抽象和实例化，一个类是一个由数据和操作数据的方法所组成的聚合体，数据和方法分别描述对象的状态与行为。Java 通过封装、继承和多态来提供很大程度的灵活性、模块化和可重用性。Java 类的访问权限限定能够实现封装和信息隐藏。所有的类均按照层次化的方式进行组织，父类的行为可以被子类继承。Java 的抽象类和接口能够实现面向对象的多态性要求，即对外一个接口，内部多种实现。

3. 分布式

分布式计算通常涉及网络上共同工作的多台计算机。Java 对分布式的支持分为两个层面：一是数据分布式，即通过 java. net 包中的类，Java 支持多个层面的网络连接，例如 URL 类支持 Java 应用程序打开并访问互联网上的远程对象，且这种对网络资源的访问与对本地文件的访问完全类似；二是操作分布式，即 Java 程序本身可以被传输，并在互联网的客户机上运行。

4. 解释型

Java 编译器生成字节码而非本地机器码，故为了运行 Java 程序，必须使用 Java 解释器，Java 解释器负责将字节码翻译为目标机器的机器码，Java 解释器通常是 Java 虚拟机的一部分。字节码具有跨平台特性，可以在任何平台的 Java 虚拟机上解释执行。然而与传统脚本语言的直接解释执行不同，Java 的解释执行不是以源代码作为输入，而是以编译生成的字节码作为输入，因而也可以将整个执行过程看做是“半编译，半解释”，这种方式综合考虑了编译执行的效率优势和解释执行的灵活性。

5. 健壮性

健壮性意味着可靠，而可靠通常是相对的。由于我们要求 Java 字节码能够在多样的系统环境下可靠地运行，因而就对 Java 语言的健壮性提出了更高的要求。首先，Java 提供自动垃圾收集机制，防止由程序员进行内存管理所可能产生的错误，避免了忘记释放内存、错误释放了仍在使用的内存等多种情况；其次，Java 提供了大量的编译时检查，帮助程序员在编写代码时及早发现错误和一些可检测的异常情况，同时 Java 支持运行时的异常处理机制，对程序运行时发生的异常进行处理；第三，Java 是强类型语言，程序编译时的类型检查更严格，防止程序在运行时出现类型不匹配等问题。

6. 安全

安全性是现代程序设计语言需要考虑的重要问题。特别是在分布式应用环境下，如果不考虑安全性，会影响到用户在自己的计算机上使用互联网 Java 应用程序的意愿。Java 内建的安全机制分为以下两部分。

(1) Java 内存分配模型。Java 的内存布局不是由编译器决定的，而是由运行系统决定的。内存布局依赖于运行系统所在的软硬件平台特性，同时对程序员是透明的。Java 没有通常意义上的内存单元指针，而是通过符号指针来引用内存，其运行系统在运行时将符号指针解释为实际的内存地址，因而避免了通过将整型值强制转化为地址或将地址作为整型值进行运算所带来的错误内存访问。

(2) Java 安全控制模型。Java 运行系统通过类装载器、字节码验证器等保证被执行的 Java 程序对主机内存的安全。在字节码执行前，Java 运行系统将待执行代码输入到字节码验证器，字节码验证器静态地分析字节码，保证不存在伪造指针、没有栈溢出、方法调用的参数合法、未违反对象的访问权限等。Java 还通过建立保护域、访问控制策略与机制，对代码的来源进行确认并对代码所能操作的本地资源(如本地文件、网络端口等)的访问进行严格的权限控制，保证主机本地资源的安全。

7. 体系结构中立

“体系结构中立”又称为“平台无关”，指编译器生成的字节码具有其自身的规范，与具体的计算机系统结构无关。例如，Java 基本数据类型的长度不会随着目标机器的变化而变化，int 类型的值总是 32 位，long 类型的值总是 64 位，算术运算的执行方式也是确定的。

也正因为如此，同一段字节码程序可以在不同的Java虚拟机和硬件平台上执行，如图1-1所示。另外，Java平台API还提供了对底层操作系统资源的访问方法，如java.io包、java.net包等，当应用程序使用这些API时，也保证它们能够正常地运行于支持Java的平台上。因此，Java平台API也在一定程度上隐藏了系统差别。

8. 可移植性

由于Java的解释执行特性和体系结构中立特性，Java程序具有可移植性，可以运行于不同的平台上而不需要重新编译。在Java语言中，一方面，没有任何与平台相关的特性。另一方面，Java环境本身也可以移植到新的硬件和操作系统平台，例如，Java编译器本身可以由Java语言实现。

9. 高性能

实际上，多数语言的平台无关性都是以牺牲性能为代价的，例如Tcl、Perl等。Java是解释执行的，即字节码由解释器执行而不是直接由系统执行，因而Java不会比编译型语言(C/C++)更快。据统计，Java的执行速度约是C语言的1/20。然而这一速度对于运行交互式、基于图形用户界面和网络的应用程序来说，已经足够快了。Java字节码的设计与机器码十分相似，因而从字节码到机器码的转换十分快捷。另外，新的Java虚拟机都使用了即时编译技术(just-in-time compilation)，通过这一技术，虚拟机能够将字节码编译为本地机器码并保存这些机器码，当其对应的字节码再次被执行时，可重新调用这些本地机器码。这就在保留了可移植性的同时，也提高了Java的性能。

10. 多线程

所谓多线程，指的是程序同时执行多项任务的能力，例如程序在下载视频文件的同时播放该文件。Java的设计本身是要满足现实中创建交互式网络化程序的需求，为实现这一目标，Java在语言级嵌入了多线程机制。如果底层操作系统支持多线程，那么Java的线程通常会被运行系统映射到实际的操作系统线程中，但这一映射过程对程序员来说是透明的。相比而言，其他一些程序设计语言可能需要调用操作系统相关的库才能实现多线程。多线程机制对于图形用户界面编程和网络编程尤其必要。多线程程序设计的关键是进行正确的线程调度和线程同步，Java线程的调度和控制主要基于java.lang.Thread类的内建机制，而Java线程的同步沿用了操作系统中广泛使用的Hoare临界区保护规则，一般使用语言内建的同步原语。

11. 动态特性

Java的设计目标之一是能够适应执行环境的演化，而程序的执行环境主要依赖于支持程序执行的类。Java程序携带了大量运行时的类型信息，使用这些信息，可以在运行时验证和解析那些对对象的访问，这使得对代码的动态链接相对安全。Java程序只在运行时动态地装载必需的类，这一动态性使得支持类库的演化对程序执行的影响达到最小。相较而言，C++应用程序在类库升级后往往需要重新编译。

1.4 Java虚拟机简介

Sun公司在Java虚拟机规范中指出，Java虚拟机是一种在真实的计算机上通过软件仿真实现的虚构机器，虚拟机代码(即字节码)存储于.class文件中，每个.class文件最多包含一个public class类的代码。Java虚拟机是Java平台的基石，是Java技术用以实现硬件