



高等学校电子信息类“十三五”规划教材  
应用型网络与信息安全工程技术人才培养系列教材

# 面向对象程序设计 (Java)

主 编 何林波 昌 燕 索 望  
副主编 熊 熙 陈 丁 黄源源 刘 丽



西安电子科技大学出版社  
<http://www.xduph.com>

高等学校电子信息类“十三五”规划教材

应用型网络与信息安全工程技术人才培养系列教材

# 面向对象程序设计(Java)

主 编 何林波 昌 燕 索 望

副主编 熊 熙 陈 丁 黄源源 刘 丽

西安电子科技大学出版社

## 内 容 简 介

本书主要讲述面向对象编程语言 Java。从程序设计语言及 Java 语言产生的背景和发展过程入手,以什么是“面向对象”这一问题引出后续章节,循序渐进地讲述了 Java 基础语法到 Java 网络编程。

本书主编及参编作者有多年的 Java 系列课程教学经验,编撰过程中结合了编者历年来的 Java 教学和信息系统开发经验,比较详细地介绍了 Java 的基础语法知识和相关技术。全书共 13 章,内容包括 Java 概述、面向对象的基本思想、类与对象、Java 语言基础、继承、接口、内部类、异常处理、Java 中的 I/O 系统、Java 多线程、数组和集合类、Java 网络编程、Java 的常用类。由于目前 Java 在视窗方面的应用场景较少,因此本书对 Java 中有关 UI 编程的部分,如 awt、swing 进行了省略,读者可以参考其他教材。

本书内容丰富、注重实用,理论知识点之后一般都给出了示范代码,部分代码有一定的实际设计意义。另外每章都附有思考与练习,引导读者回顾、总结所学知识并进行进一步的学习。

本书可作为高等院校计算机类、信息类、工程类、电子商务类和管理类各专业本、专科生的教材,也可作为普通程序开发人员的自学教材或参考书。

## 图书在版编目(CIP)数据

面向对象程序设计:Java/何林波,昌燕,索望主编. —西安:

西安电子科技大学出版社,2016.8

高等学校电子信息类“十三五”规划教材

ISBN 978-7-5606-4159-1

I. ① 面… II. ① 何… ② 昌… ③ 索… 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(2016)第 157183 号

策 划 李惠萍

责任编辑 杨璠

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 2016 年 8 月第 1 版 2016 年 8 月第 1 次印刷

开 本 787 毫米×1092 毫米 1/16 印张 17.5

字 数 408 千字

印 数 1~3000 册

定 价 32.00 元

ISBN 978-7-5606-4159-1/TP

XDUP 4451001-1

\*\*\* 如有印装问题可调换 \*\*\*



## 序

进入 21 世纪以来,信息技术迅速改变着人们传统的生产和生活方式,社会的信息化已经成为当今世界发展不可逆转的趋势和潮流。信息作为一种重要的战略资源,与物资、能源、人为一起被视为现代社会生产力的主要因素。目前,围绕着信息的获取、利用和控制,世界各国间的竞争日趋激烈,网络与信息安全问题已成为一个世纪性、全球性的课题。党的十八大报告明确指出,要“高度关注海洋、太空、网络空间安全”。党的十八届三中全会决定设立国家安全委员会,成立中央网络安全和信息化领导小组,并把网络与信息安全列入了国家发展的最高战略方向之一。这为包含网络空间安全在内的非传统安全领域问题的有效治理提供了重要的体制机制保障,是我国国家安全体制机制的一个重大创新性举措,彰显了我国政府治国理政的战略新思维和“大安全观”。

人才资源是确保我国网络与信息安全第一位的资源,信息安全人才培养是国家信息安全保障体系建设的基础和必备条件。随着我国信息化和信息安全产业的快速发展,社会对信息安全人才的需求不断增加。2015 年 6 月 11 日,国务院学位委员会和教育部联合发出“学位[2015]11 号”通知,决定在“工学”门类下增设“网络空间安全”一级学科,代码为“0839”,授予工学学位。这是国家推进专业化教育,在信息安全领域掌握自主权、抢占先机的重要举措。

新中国成立以来,我国高等工科院校一直是培养各类高级应用型专门人才的主力。培养网络与信息安全高级应用型专门人才也是高等院校责无旁贷的责任。目前,许多高等院校和科研院所已经开办了信息安全专业或开设了相关课程。作为国家首批 61 所“卓越工程师教育培养计划”试点院校之一,成都信息工程大学以《国家中长期教育改革和规划纲要(2010—2020 年)》、《国家中长期人才发展规划纲要(2010—2020 年)》、《卓越工程师教育培养计划通用标准》为指导,以专业建设和工程技术为主线,始终贯彻“面向工业界、面向未来、面向世界”的工程教育理念,按照“育人为本、崇尚应用”“一切为了学生”的教学教育理念和“夯实基础、强化实践、注重创新、突出特色”的人才培养思路,遵循“行业指导、校企合作、分类实施、形式多样”的原则,实施了一系列教育教学改革。令人欣喜的是,该校信息安全工程学院与西安电子科技大学出版社近期联合组织了一系列网络与信息安全专业教育教学改革的研讨活动,共同研讨培养应用型高级网络与信息安全工程技术人才的教育教学方法和课程体系,并在总结近年来该校信息安全专业实施“卓越工程师教育培养计划”教育教学改革成果和经验的基础上,组织编写了“应用型网络与信息安全工程技术人才培养系列教材”。本套教材总结了该

校信息安全专业教育教学改革成果和经验，相关课程有配套的课程过程化考核系统，是培养应用型网络与信息安全工程技术人才的一套比较完整、实用的教材，相信可以对我国高等院校网络与信息安全专业的建设起到很好的促进作用。该套教材为中国电子教育学会高教分会推荐教材。

信息安全是相对的，信息安全领域的对抗永无止境。国家对信息安全人才的需求是长期的、旺盛的。衷心希望本套教材在培养我国合格的应用型网络与信息安全工程技术人才的过程中取得成功并不断完善，为我国信息安全事业做出自己的贡献。

高等学校电子信息类“十三五”规划教材  
应用型网络与信息安全工程技术人才培养系列教材  
名誉主编(中国密码学会常务理事)

何大可

2015年12月

中国电子教育学会高教分会推荐  
高等学校电子信息类“十三五”规划教材  
应用型网络与信息安全工程技术人才培养系列教材

编审专家委员会名单

- 名誉主任：何大可（中国密码学会常务理事）
- 主任：张仕斌（成都信息工程大学信息安全学院副院长、教授）
- 副主任：李 飞（成都信息工程大学信息安全学院院长、教授）
- 何明星（西华大学计算机与软件工程学院院长、教授）
- 苗 放（成都大学计算机学院院长、教授）
- 赵 刚（西南石油大学计算机学院院长、教授）
- 李成大（成都工业学院教务处处长、教授）
- 宋文强（重庆邮电大学移通学院计算机科学系主任、教授）
- 梁金明（四川理工学院计算机学院副院长、教授）
- 易 勇（四川大学锦江学院计算机学院副院长、成都大学计算机学院教授）
- 杨瑞良（成都东软学院计算机科学与技术系主任、教授）

编审专家委员：（排名不分先后）

|     |     |     |     |     |     |     |
|-----|-----|-----|-----|-----|-----|-----|
| 范太华 | 叶安胜 | 黄晓芳 | 黎忠文 | 张 洪 | 张 蕾 | 贾 浩 |
| 赵 攀 | 陈 雁 | 韩 斌 | 李享梅 | 曾令明 | 何林波 | 盛志伟 |
| 林宏刚 | 王海春 | 索 望 | 吴春旺 | 韩桂华 | 赵 军 | 陈 丁 |
| 秦 智 | 王中科 | 林春蕾 | 张金全 | 王祖俪 | 蔺 冰 | 王 敏 |
| 万武南 | 甘 刚 | 王 焱 | 闫丽丽 | 昌 燕 | 黄源源 | 张仕斌 |
| 李 飞 | 王海春 | 何明星 | 苗 放 | 李成大 | 宋文强 | 梁金明 |
| 万国根 | 易 勇 | 杨瑞良 |     |     |     |     |





## 前言

在计算机刚投入实际使用的 20 世纪 60 年代,软件设计往往只是为了一个特定的应用而在指定的计算机上设计和编制,采用密切依赖于计算机的机器代码或汇编语言,而且软件的规模比较小,从设计、使用、操作方面来说都是私人化的生产方式。

20 世纪 60 年代中后期,随着计算机应用范围的迅速扩大,软件开发需求急剧增长。高级语言的出现与操作系统的发展变化引起了软件在设计、开发上的改变。同时软件的需求越来越复杂,软件的可靠性、扩展性方面的问题也越来越突出,原来的个人设计、个人使用的方式已经不再满足大规模生产的要求,迫切需要改变软件生产方式,提高软件生产率,这就是软件开发历史上著名的“软件危机”现象。1968 年,北大西洋公约组织(NATO)在联邦德国的国际学术会议上创造“软件危机(Software Crisis)”一词。为解决问题,NATO 在 1968、1969 年连续召开两次著名的 NATO 会议,并同时提出“软件工程”的概念。软件工程从其他人类工程中吸收了许多成功的经验,通过研究软件生产的客观规律性,建立与系统化软件生产有关的概念、原则、方法、技术和工具,指导和支持软件系统的生产活动。从管理上结合工程化思想,明确提出软件生命周期的模型;从软件开发技术上发展了许多软件开发与维护阶段的适用方法,包括面向过程(PO)或者结构化程序设计、面向对象(OOP)的软件设计与开发方法等。

1967 年,挪威计算中心的 Kristen Nygaard 和 Ole Johan Dahl 开发了 Simula67 语言,它提供了比子程序更高一级的抽象和封装,引入了数据抽象和类的概念,被认为是第一个面向对象语言。20 世纪 70 年代初,美国 Xerox(施乐)公司 Palo Alto 研究中心的 Alan Kay 所在的研究小组开发出 Smalltalk 语言,之后又开发出 Smalltalk-80 版本。Smalltalk 系列语言被认为是最纯正的面向对象语言,它对后来出现的面向对象语言,如 Object-C、C++、Self, Eiffel 等都产生了深远的影响。随着这一系列面向对象语言的出现,面向对象程序设计方法也就应运而生且得到快速的发展。面向对象这一方法不断向其他阶段渗透,1980 年 Grady Booch 提出了面向对象设计的概念。从 80 年代后期开始的面向对象分析(OOA)、面向对象设计(OOD)与面向对象程序编程(OOP)等新的系统开发方式模型的研究被认为是软件发展史上的重要里程碑。在过去的十几年中,Java 语言成为了广为应用的面向对象语言,除了与 C 和 C++语法上的近似性,Java 的跨平台及可移植性成为了它成功中不可磨灭的重要特性,因为这一特性,吸引了庞大的程序员的加入。尽管目前 Java 语言所诞生的 Sun 公司(Sun Microsystems)早在 2009 年 4 月 20 日被 Oracle 公司(甲骨文)收购,但它已经成为当今推广最快的、最为流

行的面向对象编程语言。Java 的出现引起了软件开发的重大变革，成为推动 IT 业蓬勃发展的最新动力，它的出现对整个计算机软件业的发展产生了重大而深远的影响。

本书在编写过程中，章节安排参考了一些经典教材，内容、知识点上参考了部分 OCJP(Oracle Certified Java Programmer)的技术资料。全书面向有 C 语言基础的读者，尽量用简单易懂的语言来描述相关的知识点，全部示例程序在 Eclipse 上调试运行，同时编者参考了互联网上一些技术文档和相关资源，在此向这些资料的作者深表谢意。本书在编撰过程中，还得到了很多同事和西安电子科技大学出版社李惠萍编辑的建议、关心和帮助，在此表示深深的感谢。

本书第 1、2、9 章为何林波老师编写，第 3、4 章为索望老师编写，第 5、6、7 章为昌燕老师编写，第 8、11 章为熊熙老师编写，第 10 章由陈丁老师编写，第 12、13 章为何林波与黄源源老师合编。全书由黄源源和刘丽老师参与英文资料的翻译和校对工作。

由于编者水平有限，书中如有不妥之处，敬请第一时间联系编者，我们将虚心接受您的批评、建议和意见，并请见谅！

编 者

2016 年 4 月





|                                   |    |                               |    |
|-----------------------------------|----|-------------------------------|----|
| <b>第1章 Java 概述</b> .....          | 1  | <b>第3章 类与对象</b> .....         | 46 |
| 1.1 程序设计语言的发展 .....               | 1  | 3.1 类的基本概念 .....              | 46 |
| 1.2 Java 语言概述 .....               | 2  | 3.1.1 类的定义 .....              | 46 |
| 1.2.1 Java 语言的产生及发展 .....         | 2  | 3.1.2 类与对象的辨析 .....           | 46 |
| 1.2.2 Java 语言的特性 .....            | 3  | 3.2 类与对象 .....                | 47 |
| 1.2.3 Java 虚拟机及 Java 的跨平台原理 ..... | 4  | 3.2.1 类的声明 .....              | 47 |
| 1.2.4 Java 的平台版本 .....            | 6  | 3.2.2 成员变量 .....              | 49 |
| 1.3 JDK 的安装与使用 .....              | 7  | 3.2.3 成员方法 .....              | 52 |
| 1.3.1 JDK 的下载与安装 .....            | 7  | 3.2.4 构造器(Constructor) .....  | 55 |
| 1.3.2 Path 与 classpath 环境变量 ..... | 10 | 3.2.5 创建对象 .....              | 58 |
| 1.3.3 第一个 Java 程序 .....           | 12 | 3.2.6 访问对象的成员 .....           | 59 |
| 1.3.4 Java 的反编译 .....             | 17 | 3.2.7 main 方法 .....           | 60 |
| 1.4 集成开发工具介绍 .....                | 18 | 3.2.8 关键字 .....               | 60 |
| 思考与练习 .....                       | 28 | 3.2.9 标识符 .....               | 61 |
| <b>第2章 面向对象的基本思想</b> .....        | 29 | 3.2.10 static 关键字 .....       | 61 |
| 2.1 结构化程序设计方法的缺点 .....            | 29 | 3.2.11 this 关键字 .....         | 63 |
| 2.2 面向对象的基本概念 .....               | 30 | 3.2.12 package 与 import ..... | 65 |
| 2.2.1 对象的基本概念 .....               | 31 | 3.2.13 访问控制修饰符 .....          | 69 |
| 2.2.2 面向对象中的抽象 .....              | 32 | 3.2.14 完整的范例程序 .....          | 70 |
| 2.3 面向对象核心思想 .....                | 34 | 3.3 Java 虚拟机运行数据区 .....       | 72 |
| 2.3.1 封装与透明 .....                 | 34 | 思考与练习 .....                   | 74 |
| 2.3.2 消息与服务 .....                 | 36 | <b>第4章 Java 语言基础</b> .....    | 75 |
| 2.3.3 继承 .....                    | 37 | 4.1 Java 的数据类型 .....          | 75 |
| 2.3.4 接口 .....                    | 38 | 4.1.1 基本数据类型 .....            | 75 |
| 2.3.5 多态 .....                    | 39 | 4.1.2 布尔型 .....               | 76 |
| 2.4 类之间的关系 .....                  | 40 | 4.1.3 整数类型 .....              | 76 |
| 2.4.1 UML 简介 .....                | 40 | 4.1.4 字符型 .....               | 80 |
| 2.4.2 依赖 .....                    | 41 | 4.1.5 浮点数类型 .....             | 81 |
| 2.4.3 关联 .....                    | 41 | 4.1.6 基本数据类型之间的转换 .....       | 82 |
| 2.4.4 聚合与组合 .....                 | 42 | 4.1.7 引用类型 .....              | 85 |
| 2.4.5 泛化 .....                    | 43 | 4.2 运算符 .....                 | 86 |
| 2.4.6 实现 .....                    | 43 | 4.2.1 算术运算符 .....             | 86 |
| 思考与练习 .....                       | 44 | 4.2.2 关系运算符 .....             | 88 |
|                                   |    | 4.2.3 位运算符 .....              | 89 |
|                                   |    | 4.2.4 逻辑运算符 .....             | 93 |

|                          |     |
|--------------------------|-----|
| 4.2.5 赋值运算符              | 94  |
| 4.2.6 其他运算符              | 96  |
| 4.2.7 运算符的优先级            | 99  |
| 4.3 表达式与语句               | 99  |
| 4.3.1 分支语句               | 100 |
| 4.3.2 循环语句               | 104 |
| 4.3.3 break 和 continue   | 107 |
| 思考与练习                    | 108 |
| <b>第5章 继承</b>            | 110 |
| 5.1 继承的基本概念              | 110 |
| 5.2 Java 继承的语法           | 110 |
| 5.2.1 子类与超类              | 110 |
| 5.2.2 子类能继承的属性及方法        | 111 |
| 5.2.3 构造方法的继承(super 关键字) | 111 |
| 5.2.4 方法的重载              | 113 |
| 5.2.5 方法的覆盖              | 114 |
| 5.2.6 方法覆盖与重载的区别         | 116 |
| 5.2.7 Java 的上下转型         | 118 |
| 5.2.8 继承的利弊与使用原则         | 116 |
| 5.3 终止继承                 | 118 |
| 5.4 抽象类                  | 119 |
| 5.5 多态                   | 120 |
| 5.6 Object 类             | 125 |
| 思考与练习                    | 126 |
| <b>第6章 接口</b>            | 128 |
| 6.1 接口的概念与特性             | 128 |
| 6.2 接口的定义与使用             | 128 |
| 6.2.1 接口定义的语法            | 128 |
| 6.2.2 接口实现的语法            | 129 |
| 6.3 比较接口与抽象类             | 131 |
| 6.4 基于接口的设计模式            | 135 |
| 6.4.1 定制服务模式             | 135 |
| 6.4.2 适配器模式              | 136 |
| 6.4.3 默认适配器模式            | 138 |
| 6.4.4 代理模式               | 139 |
| 思考与练习                    | 142 |
| <b>第7章 内部类</b>           | 145 |
| 7.1 内部类                  | 145 |
| 7.1.1 内部类概述              | 145 |
| 7.1.2 成员内部类              | 145 |
| 7.1.3 局部内部类              | 146 |
| 7.1.4 匿名内部类              | 146 |

|                                              |     |
|----------------------------------------------|-----|
| 7.1.5 静态内部类                                  | 147 |
| 7.2 内部类的使用                                   | 147 |
| 思考与练习                                        | 149 |
| <b>第8章 异常处理</b>                              | 150 |
| 8.1 异常处理机制基础                                 | 150 |
| 8.1.1 什么是异常                                  | 150 |
| 8.1.2 Java 异常处理机制的优点                         | 150 |
| 8.2 异常的处理                                    | 153 |
| 8.2.1 try... catch 捕获异常                      | 153 |
| 8.2.2 finally 子语句                            | 154 |
| 8.2.3 throws 和 throw 子语句                     | 155 |
| 8.2.4 异常处理语句的语法规则                            | 155 |
| 8.3 Java 的异常类                                | 158 |
| 8.3.1 异常的分类                                  | 158 |
| 8.3.2 运行时异常与受检查异常的区别                         | 158 |
| 8.3.3 异常与错误的区别                               | 159 |
| 8.4 自定义异常类                                   | 159 |
| 8.5 异常处理原则                                   | 160 |
| 思考与练习                                        | 161 |
| <b>第9章 Java 中的 I/O 系统</b>                    | 162 |
| 9.1 认识输入流与输出流                                | 162 |
| 9.2 输入流                                      | 165 |
| 9.2.1 字节数组输入流                                | 166 |
| 9.2.2 文件输入流                                  | 167 |
| 9.2.3 文件字符输入流                                | 169 |
| 9.2.4 Java 管道流                               | 170 |
| 9.3 过滤器输入流                                   | 172 |
| 9.3.1 DataInputStream 的使用                    | 174 |
| 9.3.2 BufferedInputStream 的使用                | 175 |
| 9.4 输出流                                      | 176 |
| 9.4.1 字节数组输出流                                | 176 |
| 9.4.2 文件输出流                                  | 177 |
| 9.5 过滤器输出流                                   | 178 |
| 9.5.1 FilterOutputStream                     | 178 |
| 9.5.2 DataOutputStream                       | 178 |
| 9.5.3 BufferedOutputStream                   | 178 |
| 9.5.4 PrintStream                            | 179 |
| 9.6 Reader 与 Writer                          | 180 |
| 9.6.1 InputStreamReader 和 OutputStreamWriter | 180 |

|        |                                 |     |        |                                |     |
|--------|---------------------------------|-----|--------|--------------------------------|-----|
| 9.6.2  | BufferedReader 和 BufferedWriter | 181 | 11.3.1 | Set 集合概述                       | 222 |
| 9.7    | 标准 I/O                          | 182 | 11.3.2 | HashSet                        | 223 |
| 9.8    | File 处理                         | 183 | 11.3.3 | TreeSet                        | 223 |
| 9.8.1  | 创建文件与目录                         | 184 | 11.4   | List 列表                        | 225 |
| 9.8.2  | 随机文件访问                          | 186 | 11.4.1 | List 列表概述                      | 225 |
| 9.9    | 对象的序列化与反序列化                     | 187 | 11.4.2 | List 的实现类                      | 226 |
| 思考与练习  |                                 | 190 | 11.4.3 | List 的 ListIterator 接口         | 226 |
| 第 10 章 | Java 多线程                        | 192 | 11.5   | Map 映射                         | 228 |
| 10.1   | 线程的基本概念                         | 192 | 11.5.1 | Map 映射概述                       | 228 |
| 10.1.1 | 进程与线程                           | 192 | 11.5.2 | Map 的实现类                       | 229 |
| 10.1.2 | 线程的运行机制                         | 193 | 思考与练习  |                                | 232 |
| 10.2   | 线程的创建与启动                        | 194 | 第 12 章 | Java 网络编程                      | 233 |
| 10.2.1 | 继承 Thread 类                     | 194 | 12.1   | 网络编程基础                         | 233 |
| 10.2.2 | 实现 Runnable 接口                  | 195 | 12.1.1 | 网络的基本概念                        | 233 |
| 10.3   | 线程中常见的方法                        | 196 | 12.1.2 | IP 地址与端口                       | 233 |
| 10.3.1 | start()方法                       | 197 | 12.1.3 | TCP/IP 的传输层协议                  | 234 |
| 10.3.2 | sleep()方法                       | 197 | 12.2   | URL 应用                         | 235 |
| 10.3.3 | yield()方法                       | 198 | 12.2.1 | 统一资源定位器                        | 235 |
| 10.3.4 | join()方法                        | 198 | 12.2.3 | URL 应用示例                       | 236 |
| 10.4   | 线程的状态转换                         | 201 | 12.3   | TCP 编程                         | 238 |
| 10.5   | 线程同步                            | 202 | 12.3.1 | Socket 的基本概念                   | 238 |
| 10.5.1 | 临界资源问题                          | 202 | 12.3.2 | Socket 简单编程应用                  | 239 |
| 10.5.2 | 互斥锁                             | 202 | 12.3.3 | 支持多客户的 Client/Server 应用        | 242 |
| 10.5.3 | 多线程的同步                          | 202 | 12.4   | UDP 编程                         | 246 |
| 10.5.4 | 同步与并发                           | 206 | 12.4.1 | DatagramSocket 类               | 246 |
| 10.5.5 | 对象锁与线程通信                        | 206 | 12.4.2 | 基于 UDP 的简单的 Client/Server 程序设计 | 247 |
| 10.5.6 | 死锁                              | 209 | 思考与练习  |                                | 251 |
| 10.6   | Daemon 线程                       | 212 | 第 13 章 | Java 的常用类                      | 252 |
| 思考与练习  |                                 | 214 | 13.1   | String 类和 StringBuffer 类       | 252 |
| 第 11 章 | 数组和集合类                          | 215 | 13.1.1 | String 类                       | 252 |
| 11.1   | Java 数组                         | 215 | 13.1.2 | StringBuffer 类                 | 254 |
| 11.1.1 | 数组的声明、创建与初始化                    | 215 | 13.2   | 基本数据类型封装类                      | 256 |
| 11.1.2 | 多维数组                            | 217 | 13.3   | Properties 类                   | 260 |
| 11.1.3 | 数组实用类 Arrays                    | 218 | 13.4   | Date 与 Calendar 类              | 261 |
| 11.2   | Java 集合                         | 219 | 13.4.1 | Date 类                         | 262 |
| 11.2.1 | Java 中的集合概述                     | 219 | 13.4.2 | Calendar 类                     | 263 |
| 11.2.2 | Collection 接口                   | 220 | 13.5   | Math 与 Random 类                | 266 |
| 11.2.3 | Iterator 接口                     | 221 | 思考与练习  |                                | 267 |
| 11.3   | Set 集合                          | 222 |        |                                |     |

# 第1章 Java 概述

关于 Java,也许读者的第一个问题就是“Java 是什么,可以用来做什么?”如果读者对计算机编程技术有较多了解的话,那么也许会有类似这样的问题:“Java 跟 C++、C# 有区别吗,学哪一种更容易,哪一种能更好适应社会的需求?”。

首先来回答第一个问题,Java 是什么,可以用来做什么?Java 如同 C 语言、C++ 语言一样,也是一种编程语言。因此,Java 有自己相关的编程语言规则、程序编译方法以及程序的运行方法。同时,Java 也作为一种开发平台存在,提供了相应的软件开发工具包(SDK, Software Development Kit)供开发者使用,SDK 中有用于辅助开发的相关文档、范例和工具的集合。使用 Java 提供的相关 SDK 及开发工具,并依赖正确的编程及良好的设计,从理论上来说,就可以利用 Java 来开发任何计算机应用软件。其次,关于第二个问题,其实也一直是各种阵营的程序设计人员讨论或争论的热点,站在不同的角度,每个人都有不同的看法和理解,而且都有足够支撑的依据。

对于此类问题,作者认为没有必要过于纠结,任何编程语言都只是程序设计的工具,而利用工具设计开发应用软件才是最终的目的。在某些场合下,一种语言可能更适合某一种类型的应用开发而不适合另外一种应用类型,但也许这种类型的应用正好有一种别的语言恰能胜任,那么毫无疑问,我们总是利用合适的工具去做合适的事情。简单来讲,斧子和电锯都能砍倒一棵树,那么你会选择什么?读者肯定回答,“当然是电锯!”但是这种回答并不一定正确,如果使用程序设计的思维来回答,应该是:当斧子和电锯都存在时,优先选择电锯;当仅有一种工具时,就只能选择仅有的工具。如果我们发现不同的程序设计语言在解决同一问题时,好像并没有谁有比较突出的优点,那么,就应该选择自己熟悉的编程语言,这是一个很简单的道理。

没有做不了事的程序设计语言,只有做不了事的程序员。

软件的设计和开发,并不是比较语言或者开发工具之间的孰优孰劣,更重要的是,充分了解需要开发的应用本身,关注其业务逻辑;并且懂得选用合适的编程语言去解决这类应用,使用科学的、经过实践论证的正确的软件设计思想与方法去解决实际问题,这才是软件设计及开发工程师应该关注的问题本质,而绝非是语言或者开发工具之间无谓的争论。

## 1.1 程序设计语言的发展

语言是人类最重要的交流工具,是人们进行沟通交流的主要表达方式,结合了各种表达符号或者手势、动作等。人与计算机之间应当如何沟通呢?显然,必须以一种计算机能够理解的形式来下发各种运算指令。因此,计算机语言是人机沟通交互的最重要的桥梁和手段。从计算机问世至今,人们一直为研制更新更好的程序设计语言而努力。目前已问世的

各种程序设计语言有成千上万种，但这其中只有少数得到了人们的广泛认可和使用。

最早的一代程序设计语言是机器语言。从有关的计算机基础科学的学习中，我们应当已经对这种语言有了概念。机器语言是用二进制代码表示的计算机能直接识别和执行的一种机器指令的集合。它是计算机的设计者通过计算机的硬件结构赋予计算机的操作功能。机器语言具有灵活、直接执行和速度快等特点。用机器语言编写程序，编程人员要首先熟悉所用计算机的全部指令代码和代码的含义。程序员编写程序时得自己处理每条指令和每一数据的存储分配及输入、输出，还需要记住编程过程中每步所使用的工作单元处在何种状态。这是一件十分繁琐的工作，编写程序花费的时间往往是实际运行时间的几十倍或几百倍。编出的程序全是 0 和 1 的指令代码，直观性差，容易出错。除了计算机生产厂家的专业人员外，绝大多数的程序员已经不再去学习机器语言了。

由于机器语言的可读性差、编程困难，因此，第二代语言——汇编语言很快就发展起来了。汇编语言是汇编指令集、伪指令集和使用它们规则的统称。具有一定含义的符号为助记符，用指令助记符、符号地址等组成的符号指令称为汇编格式指令。不同型号的计算机其机器语言是不相通的，按一种计算机机器指令编制的程序，不能在另一种计算机上执行。

第三代语言是高级语言，它主要有四种范型：命令式语言、函数式语言、逻辑式语言与对象式语言。第四代语言是面向数据库的，实际上它不只是语言而且也是交互式程序设计环境。程序设计语言的发展主要经历了这四代，也有人把逻辑式语言、函数式语言，甚至于对象式语言称为第五代语言。这五代语言中，前三代语言的发展有明确的先后界限，而后三代语言之间则没有这些界限，它们仍然在同时发展。

## 1.2 Java 语言概述

Java 是 Sun 公司(Sun Microsystems, 2009 年 4 月 20 日被 Oracle(甲骨文)收购，交易价格达 74 亿美元)推出的新一代面向对象的编程语言。1996 年初，Java 1.0 版的正式发表就迅速引起了整个计算机界的高度关注。由于 Java 提供了强大的图形、图像、音频、视频、多线程和网络交互能力，它已经成为当今推广最快的最为流行的网络编程语言。Java 的出现引起了软件开发的重大变革，成为推动 IT 业蓬勃发展的最新动力。它的出现对整个计算机软件业的发展产生了重大而深远的影响。

### 1.2.1 Java 语言的产生及发展

Java 语言诞生于 1991 年 Sun 公司一个被称之为 Green 的项目(Green Project, 绿色计划)，该项目的目的是开拓消费类嵌入式电子产品市场，用以开发如交互式电视、烤面包机等家用电器的控制软件。James Gosling 是该小组的领导人(后被称为 Java 之父，是一位非常杰出的程序员)。Green Project 所使用的语言是 C、C++ 以及 Oak (橡树，为 Java 语言的前身)，后因语言本身和市场的问题，使得该项目的发展无法达到当初预期的目标，再加上网络的兴起，绿色计划也因此于 1994 年改变发展的方向。“Java”一词是 Sun 公司 Java 发展小组历经无数次的激烈讨论之后确定的。“Java”这个词本身来自于太平洋上一个盛产咖啡的岛屿名字，是从许多程序设计师钟爱的热气腾腾、浓香四溢的咖啡中获得的灵感，因

此, Java 的 Logo 看上去就是一杯冒着热气的咖啡, 如图 1.1 所示。



图 1.1 Java 图标

1995 年 5 月 23 日, Java 语言诞生。

1996 年 1 月, 第一个 JDK——JDK 1.0 诞生。

1996 年 4 月, 10 个最主要的操作系统供应商申明将在其产品中嵌入 Java 技术。

1996 年 9 月, 约 8.3 万个网页应用了 Java 技术来制作。

1997 年 2 月 18 日, JDK 1.1 发布。

1997 年 4 月 2 日, JavaOne 会议召开, 参与者逾一万人, 创当时全球同类会议规模之纪录。

1997 年 9 月, Java Developer Connection 社区成员超过十万。

1998 年 2 月, JDK 1.1 被下载超过 200 万次。

1998 年 12 月 8 日, Java2 企业平台 J2EE 发布。

1999 年 6 月, SUN 公司发布 Java 的三个版本: 标准版、企业版和微型版。

2000 年 5 月 8 日, JDK 1.3 发布。

2000 年 5 月 29 日, JDK 1.4 发布。

2001 年 9 月 24 日, J2EE 1.3 发布。

2002 年 2 月 26 日, J2SE 1.4 发布, 自此 Java 的计算能力有了大幅提升。

2004 年 9 月 30 日下午 6 点, J2SE 1.5 发布, 成为 Java 语言发展史上的又一里程碑。

为了表示该版本的重要性, J2SE 1.5 更名为 Java SE 5.0。

2005 年 6 月, JavaOne 大会召开, SUN 公司公开 Java SE 6.0。此时, Java 的各种版本已经更名, 以取消其中的数字“2”: J2EE 更名为 Java EE, J2SE 更名为 Java SE, J2ME 更名为 Java ME。

2006 年 12 月, SUN 公司发布 JRE 6.0。

2009 年 4 月 7 日, Google App Engine 开始支持 Java。

2009 年 04 月 20 日, 甲骨文公司以 74 亿美元收购 Sun 公司, 取得 Java 的版权。

2011 年 7 月 28 日, 甲骨文公司发布 Java 7.0 的正式版。

2014 年 3 月 19 日, 甲骨文公司发布 Java 8.0 的正式版。

2014 年 11 月, 甲骨文公司发布了 Java 9.0 版本。

### 1.2.2 Java 语言的特性

Java 语言编程风格十分接近 C 语言以及 C++ 语言。因此, 只要具备 C 语言基础, 学习 Java 语言是一件比较简单的事。同 C++ 语言一样, Java 也是一种纯粹的面向对象 (Objected-Oriented) 的程序设计语言 (关于什么是面向对象, 将在第 2 章进行学习)。对初学者来说, Java 语言给他们最大的惊喜可能是发现在 Java 编程中不再有像 C 语言中那样对

指针操作的内容,于是也就没有了像指针数组、数组指针这些晦涩难懂的问题。同时,Java特有的内存管理机制使得程序员也不再像C语言中那样需要对于开辟出来的内存空间进行释放及管理,从而程序员可以更专注于程序的逻辑设计而不是内存溢出错误的修正。Java的这种垃圾回收机制在若干年后微软的C#语言中也有类似的应用。下面列举关于Java语言的一些主要特性。

(1) **Java语言是跨平台的。**简单来说,Java的跨平台指的是“一次编译,多次运行”,即成功编译之后,其编译的程序可以在不同的平台(OS,操作系统)上运行,比如在Windows环境下编译成功的Java程序,也可以直接在Linux系统中运行。熟悉C语言开发的读者可能清楚,在Windows环境下编译的C语言程序是无法在Linux或者UNIX系统下直接运行的。关于Java语言的跨平台特性,将在1.2.3小节详细阐述。

(2) **Java语言的学习是比较容易的。**一方面,Java语言的语法与C语言和C++语言很接近,使得大多数程序员很容易学习和使用。另一方面,跟同属于面向对象程序设计语言的C++相比,Java舍弃了C++中令人迷惑、难以理解的一些特性,如多继承、操作符重载等。

(3) **Java语言是面向网络、面向分布式的。**Java语言之所以成功,其实并非来源于Green Project的发展,而是Java对于Internet应用的良好支持。在基本的Java应用编程接口中有一个网络应用编程接口(Java Net),它提供了用于网络应用编程的类库,包括URL、URLConnection、Socket、ServerSocket等。Java的RMI(远程方法激活)机制也是开发分布式应用的重要手段。

(4) **Java语言是健壮的。**Java的强类型机制、异常处理、垃圾的自动收集等是Java程序健壮性的重要保证。对指针的丢弃是Java的明智选择。Java的安全检查机制以及异常处理方法使得Java程序更具健壮性。

(5) **Java语言是安全的。**Java通常被用在网络环境中,为此,Java提供了一个安全机制以防恶意代码的攻击。除了Java语言本身具有的许多安全特性以外,Java的类加载器(ClassLoader)对通过网络下载类具有一定的安全防范机制,如分配不同的名字空间以防替代本地的同名类、可进行字节代码检查,并提供安全管理(类SecurityManager)。

Java语言的优良特性使得Java应用具有无比的健壮性和可靠性,这也减少了应用系统的维护费用。Java对对象技术的全面支持和Java平台内嵌的API能缩短应用系统的开发时间并降低成本。Java的编译一次到处可运行的特性使得它能够提供一个随处可用的开放结构和在多平台之间传递信息的低成本方式。特别是Java企业应用编程接口(Java Enterprise APIs),为企业计算及电子商务应用系统提供了有关技术和丰富的类库。

### 1.2.3 Java虚拟机及Java的跨平台原理

1.2.2小节提到了Java语言的跨平台特性,这就不能不讲到“Java虚拟机”。什么是Java虚拟机呢?读者在类似“计算机基础”学科中应该已经具备了对“计算机”组成及相关部件功能的理解。其实,Java虚拟机就是用软件的形式在实际的计算机上仿真模拟各种计算机功能,从而实现了一种抽象化的“计算机”。Java虚拟机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。

Java虚拟机(Java Virtual Machine, JVM),可以用不同的方式(软件或硬件)在不同的



真实计算机及操作系统上加以实现，其编译虚拟机的指令集与编译微处理器的指令集非常类似。Java 虚拟机包括一套字节码指令集、一组寄存器、一个栈、一个垃圾回收堆和一个存储方法域。

用 Java 语言编写的程序或者代码在编译时，其目标操作系统并非是直接运行程序的真实计算机系统，而是 Java 自己定义的“虚拟机”，Java 程序可以直接运行在这种虚拟机之上。因此，理论上只要某个计算机操作系统中装有 Java 的虚拟机程序，那么我们开发的 Java 程序就可以在其上运行，而不需要重新编译。

下面将介绍 Java 程序的编译和运行过程，有助于帮助读者理解跨平台特性，如图 1.2 所示。

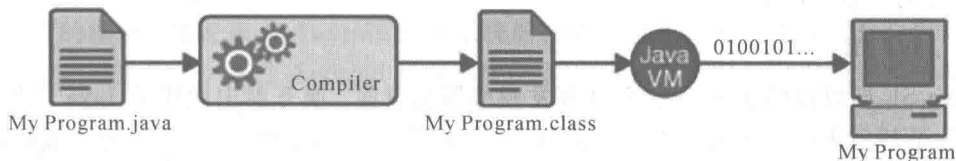


图 1.2 Java 编译与运行过程

图 1.2 中，MyProgram.java (Java 源程序文件，后缀名是 .java) 经过编译命令编译后，将 Java 源文件编译成为 Java 虚拟机目标文件 My Program.class (又称为 Java 字节码文件，byte codes)，字节码文件能被 JVM 解释并在真实计算机上执行。

如果把 Java 源程序想象成我们熟悉的 C 源程序，Java 源程序编译后生成的字节码就相当于 C 源程序编译后的 80x86 的机器码 (二进制程序文件)，JVM 虚拟机相当于 80x86 计算机系统，Java 解释器相当于 80x86 CPU (该“CPU”不是通过硬件实现的，而是用软件实现的)。在 80x86 CPU 上运行的是机器码，在 Java 解释器上运行的是 Java 字节码。

Java 解释器实际上就是特定平台下的一个应用程序。只要实现了特定平台下的解释器程序，Java 字节码就能通过解释器程序在该平台下运行，这是 Java 跨平台的根本原因。当前，并不是在所有的平台下都有相应的 Java 解释器程序，这也是 Java 并不能在所有的平台下都能运行的原因，它只能在已实现了 Java 解释器程序的平台下运行。

如图 1.3 所示为 Java 程序跨平台运行的原理。

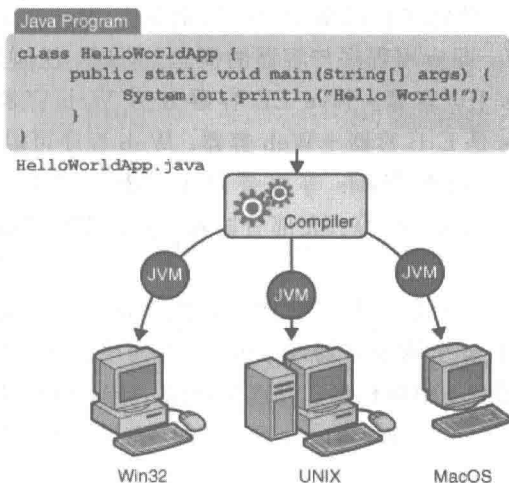


图 1.3 Java 程序跨平台原理示意图

## 1.2.4 Java 的平台版本

Java 发展到今天, 已从最初的单一编程语言发展成为全球第一大通用开发平台。Java 技术已被计算机行业主要公司所采纳。2005 年 6 月, JavaOne 大会的召开将原来称之为 J2SE(Java 2 Platform, Standard Edition), J2EE(Java 2 Platform, Enterprise Edition), J2ME(Java 2 Platform, Micro Edition)的平台修改为 Java SE(Java Platform, Standard Edition)、Java EE(Java Platform, Enterprise Edition)以及 Java ME(Java Platform, Micro Edition)。随着这三大平台的迅速推进, 全球形成了一股巨大的 Java 应用浪潮。目前官网都提供对应的版本可以下载。

### 1. Java SE

Java SE 即以前的 J2SE, 它允许开发和部署在桌面、服务器中使用的 Java 应用程序。Java SE 是基础平台, 通过对 Java SE 的学习, 可帮助读者掌握 Java 语言的基础语法、特性、编程方法及方式。并为 Java EE 提供开发学习的基础。其实, 目前直接使用 Java SE 来开发桌面版程序(Windows 窗体程序)并不流行。

### 2. Java EE

Java EE 以前称之为 J2EE。企业版本帮助程序员开发和部署可移植的、健壮的、可伸缩且安全的服务器端 Java 应用程序, 关于 Java EE 的服务端程序可以从如下几个方面来理解:

对于服务端程序的理解主要包括两种技术标准, Web 技术与 EJB 技术。

简单来说, Web 技术可以认为是一系列开发 B/S(Browser 浏览器/Server 服务器)结构程序的技术, 主要包括诸如 JSP、Servlet、JDBC 这类技术。JSP 与 Servlet 程序都运行在服务器之上, 客户端只是显示程序运行的结果(体现为 HTML 页面)。

EJB(Enterprise JavaBeans, 企业级 JavaBeans), 可以理解为“运行在服务器上的 Java 类”, Java EE 将业务逻辑从客户端软件中抽取出来, 封装在一个组件里, 这个组件运行在一个独立的服务器上, 客户端软件通过网络调用组件提供的服务以实现业务逻辑, 而客户端软件的功能单纯到只负责发送调用请求和显示处理结果。这种设计模式使得开发者可以方便地创建、部署、管理一些跨平台的基于组件的企业应用。

这类应用都有个特点, 即应用程序与容器的概念。所以, 任何开发工具开发和部署 Java EE 程序都需要指定 Java EE 容器。Java EE 容器分为 Web 容器和 EJB 容器, Tomcat/Resin 是 Web 容器, JBoss 是 EJB 容器+Web 容器。Web 程序可以在上面两种容器中运行, 而 Web+EJB 应用则只可以在 JBoss 等服务器上运行, 这类 EJB 容器包括商业产品 WebSphere、WebLogic、GlassFish 等。大部分时候, Java EE 容器与 Java EE 服务器概念是一致的。

总的来说, Java EE 实际上是一种框架和相关技术标准, 能够帮助编程人员开发和部署可移植、健壮、可伸缩且安全的服务器端 Java 应用程序。Java EE 是在 Java SE 的基础上构建的, 它提供 Web 服务、组件模型、管理和通信 API, 可以用来实现企业级的面向服务体系结构(Service-Oriented Architecture, SOA)和 Web 2.0 应用程序。

### 3. Java ME

Java ME 即以前的 J2ME。Java ME 为给移动设备和嵌入式设备提供的 Java 语言平台,