高等院校计算机应用系列教材

Java基础教程

(第5版)

吴仁群　编著

清華大學出版社
北　京

Java基础教程(第5版).indd 1　2024-09-27 14:05:29

内 容 简 介

本书是针对 Java 语言初学者编写的基础教程，书中不仅讲解了 Java 程序设计的基础知识，而且提供了大量实用性很强的编程实例。全书共分 10 章，内容包括 Java 概述、Java 基础、类与对象、继承与接口、数组与字符串、Java 的异常处理机制、Java 常见类库、输入输出流及数据库操作、多线程和图形用户界面设计。

本书内容实用，结构清晰，实例丰富，可操作性强，可作为高等院校 Java 程序设计课程的教材，也可作为计算机相关专业的培训教材和自学用书。

本书封面贴有清华大学出版社防伪标签，无标签者不得销售。
版权所有，侵权必究。举报：010-62782989，beiqinquan@tup.tsinghua.edu.cn。

图书在版编目（CIP）数据

Java 基础教程 / 吴仁群编著 . -- 5 版 . -- 北京 : 清华大学出版社 , 2024. 9. -- (高等院校计算机应用系列教材). -- ISBN 978-7-302-67329-3

Ⅰ . TP312.8

中国国家版本馆 CIP 数据核字第 2024E8U494 号

责任编辑：刘金喜
封面设计：高娟妮
版式设计：芃博文化
责任校对：孔祥亮
责任印制：曹婉颖

出版发行：清华大学出版社

网　　址：https://www.tup.com.cn，https://www.wqxuetang.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-83470000　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn

印 装 者：三河市铭诚印务有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**19.5　　**字　　数：**499 千字
版　　次：2009 年 4 月第 1 版　　2024 年 10 月第 5 版　　**印　　次：**2024 年 10 月第 1 次印刷
定　　价：69.00 元

产品编号：106671-01

前言

《Java基础教程》(第2版)已于2013年10月被评为北京市精品教材。为反映Java发展的新特点和新进展，以及为使书的结构更清晰，内容描述更简洁明了，以便更适合初学者学习，本书作者对部分章节进行了修改和完善，增添了新的实例、注解和说明，补充了新的内容，以及对内容进行了重新组织。

随着Flash、HTML5 和 CSS3 等 Web 技术的不断进步和完善，以及受Applet程序的安全性和兼容性等问题的影响，Applet程序已逐渐被开发者所抛弃，因此本版不再对Applet方面进行介绍。本书共分10章。第1章讲述了Java的发展历程、特点及运行平台和Java程序的开发过程；第2章介绍了Java编程的基础语法知识；第3章和第4章讲述了Java的面向对象技术，体现了Java作为一种纯粹的面向对象编程语言的编程特点；第5章介绍了数组与字符串的特点及使用；第6章介绍了Java语言的异常处理机制；第7章介绍了Java类库结构及常用类库；第8章介绍了Java语言中输入输出流和数据库操作方法；第9章介绍了Java多线程的概念、特点及实现；第10章介绍了在Java中如何进行图形用户界面设计及事件处理。

本书附赠教学课件、案例源代码、教学大纲、教案、教学日历，可通过扫描下方二维码下载。

教学资源

本书由吴仁群编写。在编写过程中，编者参考了本书“参考文献”所列举的图书，并得到了清华大学出版社的大力支持，在此对他们表示衷心感谢。

由于时间仓促，书中难免存在一些不足之处，敬请读者批评指正。

服务邮箱：476371891@qq.com。

编　者

2024年5月

Java基础教程(第5版).indd 2 2024-09-27 14:05:30

目录

第1章 Java概述 1

1.1 Java的特点及相关概念 …… 1
1.1.1 Java的发展历程 …… 1
1.1.2 Java的特点 …… 2
1.1.3 Java虚拟机 …… 4
1.1.4 Java的应用领域 …… 6
1.2 Java程序开发 …… 7
1.2.1 运行平台 …… 7
1.2.2 Java程序开发过程 …… 12
1.3 综合应用 …… 14
1.4 本章小结 …… 16
1.5 思考和练习 …… 16

第2章 Java基础 18

2.1 Java程序概况 …… 18
2.1.1 Java程序结构 …… 18
2.1.2 Java注释 …… 19
2.1.3 Java关键字 …… 20
2.1.4 Java标识符 …… 22
2.1.5 变量与常量 …… 22
2.2 基本数据类型 …… 23
2.2.1 基本数据类型概况 …… 23
2.2.2 基本数据类型转换 …… 27
2.3 运算符和表达式 …… 29
2.3.1 算术运算符和算术表达式 …… 29
2.3.2 关系运算符与关系表达式 …… 31
2.3.3 逻辑运算符与逻辑表达式 …… 32
2.3.4 赋值运算符与赋值表达式 …… 32
2.3.5 位运算符 …… 33
2.3.6 条件运算符 …… 34
2.3.7 instanceof运算符 …… 34
2.3.8 一般表达式 …… 34
2.4 Java语句 …… 36
2.4.1 Java语句概述 …… 36
2.4.2 分支语句 …… 37
2.4.3 循环语句 …… 42
2.4.4 跳转语句 …… 44
2.5 综合应用 …… 48
2.6 本章小结 …… 53
2.7 思考和练习 …… 53

第3章 类与对象 56

3.1 面向对象基础 …… 56
3.1.1 编程语言的4个发展阶段 …… 56
3.1.2 面向过程的程序设计 …… 57
3.1.3 面向对象的程序设计 …… 58
3.1.4 两种程序设计语言的简单比较 …… 60
3.2 类 …… 60
3.2.1 类的定义 …… 60
3.2.2 成员变量 …… 62
3.2.3 成员方法 …… 63
3.2.4 修饰符次序 …… 65
3.3 对象 …… 66
3.3.1 对象的创建 …… 67
3.3.2 对象的使用 …… 67
3.3.3 对象的消亡 …… 68
3.4 变量 …… 70
3.4.1 类中变量的分类 …… 70
3.4.2 变量的内存分配 …… 71
3.4.3 实例变量和静态变量的简单比较 …… 72
3.4.4 变量初始化与赋值 …… 74
3.5 方法 …… 76
3.5.1 方法概述 …… 76
3.5.2 方法分类 …… 77
3.5.3 方法调用中的数据传递 …… 79
3.5.4 三个重要方法 …… 82
3.5.5 方法的递归调用 …… 87
3.6 package和import语句 …… 89
3.6.1 package语句 …… 89
3.6.2 import语句 …… 92
3.7 访问权限 …… 92
3.7.1 类的访问控制 …… 92
3.7.2 类成员的访问控制 …… 95
3.8 综合应用 …… 97

3.9 本章小结 …… 102
3.10 思考和练习 …… 102

第4章 继承与接口 105
4.1 继承 …… 105
4.1.1 继承的含义 …… 105
4.1.2 子类继承性的访问控制 …… 107
4.1.3 子类对象的构造过程 …… 109
4.1.4 子类的内存分布 …… 110
4.1.5 子类对象的成员初始化 …… 111
4.1.6 成员变量的隐藏 …… 113
4.1.7 方法的重载与覆盖 …… 114
4.1.8 this关键字 …… 117
4.1.9 super关键字 …… 120
4.1.10 对象的上下转型 …… 121
4.2 接口 …… 121
4.2.1 abstract类 …… 121
4.2.2 接口的含义 …… 123
4.2.3 接口回调 …… 125
4.2.4 接口与抽象类的异同 …… 126
4.3 特殊类 …… 127
4.3.1 final类 …… 127
4.3.2 内部类 …… 128
4.4 综合应用 …… 128
4.5 本章小结 …… 133
4.6 思考和练习 …… 133

第5章 数组与字符串 135
5.1 数组 …… 135
5.1.1 数组的定义及说明 …… 135
5.1.2 数组应用举例 …… 139
5.2 字符串 …… 143
5.2.1 String类 …… 143
5.2.2 StringBuffer类 …… 146
5.2.3 应用举例 …… 147
5.3 综合应用 …… 149
5.4 本章小结 …… 154
5.5 思考和练习 …… 154

第6章 Java的异常处理机制 157
6.1 异常的含义及分类 …… 157
6.2 异常处理 …… 158
6.2.1 异常处理的含义及必要性 …… 158
6.2.2 异常处理的基本结构 …… 159
6.2.3 多个catch块 …… 160
6.2.4 finally语句 …… 161
6.3 两种抛出异常的方式 …… 162
6.3.1 throw——直接抛出 …… 162
6.3.2 throws——间接抛出异常(声明异常) …… 166
6.4 自定义异常 …… 167
6.5 常见异常 …… 168
6.6 综合应用案例 …… 168
6.7 本章小结 …… 170
6.8 思考和练习 …… 170

第7章 Java常见类库 171
7.1 Java类库的结构 …… 171
7.2 常用类 …… 172
7.2.1 System类 …… 172
7.2.2 Math类 …… 177
7.2.3 Random类 …… 178
7.2.4 基本数据类型的包装类 …… 180
7.2.5 Vector类 …… 182
7.2.6 Stack类 …… 186
7.2.7 Queue类 …… 188
7.2.8 Arrays类 …… 191
7.2.9 Hashtable类 …… 194
7.3 本章小结 …… 196
7.4 思考和练习 …… 197

第8章 输入输出流及数据库操作 198
8.1 输入和输出流 …… 198
8.1.1 流的含义 …… 198
8.1.2 流的层次结构 …… 199
8.1.3 标准输入输出流 …… 200
8.1.4 File类 …… 202
8.1.5 FileInputStream类和FileOutputStream类 …… 203
8.1.6 DataInputStream类和DataOutputStream类 …… 206
8.1.7 随机访问文件 …… 209
8.1.8 Reader类和Writer类 …… 211
8.1.9 IOException类的子类 …… 213
8.1.10 综合应用 …… 213
8.2 数据库操作 …… 219
8.2.1 ODBC概述 …… 219

Java基础教程(第5版).indd 4 2024-09-27 14:05:30

8.2.2 JDBC概述 …… 221
8.2.3 使用JDBC-ODBC技术访问数据库 …… 222
8.2.4 综合应用 …… 225
8.2.5 基本SQL语句 …… 228
8.3 建立数据源的操作 …… 230
8.4 本章小结 …… 233
8.5 思考和练习 …… 233

第9章 多线程 234

9.1 多线程的概念 …… 234
9.2 线程类 …… 235
9.2.1 多线程编程中常用的常量和方法 …… 235
9.2.2 线程的生命周期 …… 236
9.2.3 创建多线程的方法 …… 237
9.3 资源的协调与同步 …… 242
9.3.1 线程调度模型 …… 242
9.3.2 资源冲突 …… 243
9.3.3 同步方法 …… 244
9.4 线程间通信 …… 247
9.4.1 共享变量和方法封装在一个类中 …… 247
9.4.2 通过系统方法实现线程通信 …… 248
9.5 本章小结 …… 252
9.6 思考和练习 …… 252

第10章 图形用户界面设计 253

10.1 Java的AWT和Swing基础 …… 253
10.1.1 Java的AWT和Swing概述 …… 253
10.1.2 Java的AWT组件和Swing组件 …… 254
10.1.3 利用AWT组件和Swing组件进行程序设计的基本步骤 …… 256
10.2 常用容器 …… 257
10.2.1 框架 …… 257
10.2.2 面板 …… 259
10.2.3 滚动窗口 …… 260
10.2.4 菜单设计 …… 262
10.2.5 对话框 …… 264
10.3 布局管理器 …… 267
10.3.1 FlowLayout布局 …… 267
10.3.2 BorderLayout布局 …… 268
10.3.3 GridLayout布局 …… 270
10.3.4 CardLayout布局 …… 270
10.3.5 null布局 …… 271
10.4 事件处理 …… 273
10.4.1 委托事件模型 …… 273
10.4.2 键盘事件 …… 276
10.4.3 鼠标事件 …… 277
10.5 常用组件 …… 279
10.5.1 按钮 …… 279
10.5.2 标签 …… 281
10.5.3 文本行 …… 283
10.5.4 文本域 …… 285
10.5.5 复选框 …… 286
10.5.6 单选框 …… 288
10.5.7 选择框 …… 291
10.5.8 列表 …… 292
10.6 综合应用 …… 294
10.7 本章小结 …… 300
10.8 思考和练习 …… 300

参考文献 302

Java基础教程(第5版).indd 5 2024-09-27 14:05:30

Java基础教程(第5版).indd 6 2024-09-27 14:05:30

第 1 章

Java 概述

Java是一种简单、面向对象、分布式、解释、健壮、安全、与平台无关且性能优异的多线程动态语言，也是目前使用较为广泛的编程语言之一。

本章的学习目标：

- 了解Java语言的发展历程
- 了解Java语言的特点
- 了解Java虚拟机JVM
- 掌握Java运行平台的安装与使用
- 掌握Java程序开发的过程
- 学会调试简单的Java程序

1.1 Java的特点及相关概念

1.1.1 Java的发展历程

Java语言的前身是Oak语言。1991年4月，Sun公司(已被Oracle公司收购)的以James Gosling为首的绿色计划项目组计划发展一种分布式系统结构，使其能够在各种消费性电子产品上运行。在项目进行过程中，项目组成员在使用C++编译器时发现了C++的诸多不足，于是他们研发出了Oak语言来替代它，但Oak语言最初仅限于Sun公司内部使用。

1994年下半年，随着Internet的迅速发展和Web的广泛应用，工业界迫切需要一种能够在异构网络环境下使用的语言，在这样的背景下，James Gosling项目组对原有的Oak语言进行了小规模的改造，并在1995年3月推出了Java语言。随后，于1996年1月，他们发布了包含开发支持库的JDK 1.0版本，为Java语言的广泛应用奠定了基础。该版本包括Java运行环境(Java runtime environment，JRE)和Java开发工具包(Java development kit，JDK)，其中JRE包括核心API、集成API、用户界面API、发布技术及JVM(Java虚拟机)5个部分，而JDK包括编译Java程序的编译器

(javac)。在JDK 1.0版本中，除了AWT，其他的库并不完整。

1997年2月，Sun公司发布了JDK 1.1版本，为JVM增加了即时编译器(JIT)。与传统的编译器编译一条指令待其运行完后再将其释放掉不同的是，JIT将常用的指令保存在内存中，这样在下次调用时就没有必要再编译。继JDK 1.1版本后，Sun公司又推出了数个JDK 1.x版本。

在1998年之前，尽管Java已被众多软件企业所采用，但由于当时硬件环境和JVM的技术尚不成熟，其应用范围受到一定限制。在那时，Java主要应用于前端的Applet和一些移动设备中。然而这并不等于Java的应用只限于这些领域。1998年是Java迅猛发展的一年，Sun公司发布了JSP/Servlet、EJB规范，并将Java划分为J2EE、J2SE和J2ME，这一举措标志着Java已经吹响了向企业、桌面和移动3个领域进军的号角。

1998年12月，Sun公司发布了JDK 1.2版本，这是Java语言发展过程中的一个关键阶段，随后Sun公司将Java更名为Java 2。经过随后十年的发展，Java语言持续演进，最终发展到了1.8版本。

JDK 1.2版本将Java划分为J2EE、J2SE和J2ME三大应用平台。JDK 1.2版本的API也相应地分为核心API、可选API和特殊API三大类。其中，核心API是由Sun公司制定的，是Java平台所必须提供的基本API；可选API是Sun为JDK提供的扩充API；特殊API是用于满足特定需求而设计的。此外，Java 2版本增加了Swing图形库，它包含了各式各样的组件，为用户提供了更加灵活和强大的图形用户界面开发能力。

从JDK 1.2版本开始，Sun公司便以平均两年一个版本的速度推出新的JDK。2000年5月，Sun公司发布了JDK 1.3版本；2002年2月，发布了JDK 1.4版本；2004年10月，发布了JDK 1.5版本，同时Sun公司将其改名为JDK 5；2006年4月，发布了JDK 6测试版本，并于2007年初推出JDK 6正式版本。

2009年4月20日，Oracle公司宣布正式以74亿美元的价格收购Sun公司，自此Java商标正式划归Oracle所有。

2011年7月，Oracle公司发布了JDK 7版本；2014年3月18日，Oracle公司发布了JDK 8；2017年9月21日，Oracle公司发布了JDK 9。之后每隔一段时间，Oracle公司都会发布一个新版本的JDK。目前最新的LTS(long-term support，长期支持)版本是JDK17，该版本于2021年9月14日发布。

1.1.2 Java的特点

Java作为一种面向对象且与平台无关的多线程动态语言，具有以下特点。

1. 语法简单

Java语言的简单性主要体现在以下三个方面。

- Java的风格类似于C++，因此C++程序员可以快速掌握Java编程技术。
- Java摒弃了C++中容易引发程序错误的地方，如指针和内存管理。
- Java提供了丰富的类库。

2. 面向对象编程

面向对象编程(object-oriented programming，OOP)是一种先进的编程思想，它使得程序员能够更自然地模拟现实世界，从而更加容易解决复杂的问题。面向对象编程可以说是Java语言的

Java基础教程(第5版).indd 2 2024-09-27 14:05:30

核心特性之一。Java语言的设计完全是基于面向对象的，它不直接支持类似C语言那样的面向过程的程序设计风格，而是支持静态和动态风格的代码继承及重用。从面向对象的特性角度来看，Java与SmallTalk有诸多相似之处，但在其他特性上，尤其是适用于分布式计算环境的特性远远超越了SmallTalk。

3. 分布式

Java自诞生之日起便与网络紧密相联，它强调网络特性，内置了TCP/IP、HTTP和FTP等协议类库，极大地简化了网上应用系统的开发。因此，Java应用程序能够轻松地通过URL打开并访问网络上的对象，其访问方式与访问本地文件系统几乎完全一致。

4. 安全性

Java的安全性可从两个方面得到保证：一方面，在Java语言里，像指针和释放内存等C++中的功能被删除，避免了非法内存操作；另一方面，当Java用来创建浏览器时，其语言功能与浏览器本身提供的功能相结合，使得基于Java的应用程序在浏览器中运行时更为安全。此外，Java语言在机器上执行前，需经过多次测试，其三级安全检验机制可以有效防止非法代码入侵，阻止对内存的越权访问。

5. 健壮性

Java致力于在编译和运行时检查程序的错误。除了运行时异常检查，Java还提供了广泛的编译时异常检查，以便尽早发现可能存在的错误。类型检查功能帮助开发人员识别并纠正许多在早期开发中可能出现的错误。Java自动管理内存，这减少了因手动内存管理不当而导致的错误。Java使用真数组，有效避免了数据覆盖的可能性，从而大大缩短了开发Java应用程序的周期。此外，Java还提供了Null指针检测、数组边界检查、异常出口、字节代码校验等机制，以确保程序的健壮性和安全性。在Java中，对象的创建机制(只能用new操作符)和自动垃圾收集机制大大减少了因内存管理不当引发的错误。

6. 解释运行效率高

Java解释器(运行系统)能直接运行目标代码指令。Java程序经编译器编译后，生成的字节码经过了精心设计，并进行了优化，因此运行速度较快，克服了以往解释性语言运行效率低的缺点。Java使用直接解释器1秒钟内可调用300 000个过程，其翻译目标代码的速度与C/C++几乎一致。

7. 与平台无关

Java编译器将Java程序编译成二进制形式的代码，即字节码。这些字节码有统一的格式，不依赖于具体的硬件环境。

平台无关类型包括源代码级和目标代码级两种类型。C和C++属于源代码级与平台无关，意味着用它编写的应用程序不用修改只需重新编译就可以在不同平台上运行。Java属于目标代码级与平台无关，主要靠Java虚拟机(Java virtual machine，JVM)来实现。

8. 多线程

Java提供的多线程功能使得在一个程序里可同时执行多个小任务。线程有时也称作小进程，是一个大进程里分出来的小的、独立的进程。由于Java实现了多线程技术，所以它在处理并发任务时比C和C++更健壮。多线程带来的主要好处是更好的交互性能和实时控制性能。然

Java基础教程(第5版).indd 3　　2024-09-27 14:05:30

而，实时控制性能也取决于操作系统本身(如UNIX、Windows、Macintosh等)。在开发难易程度和性能上，多线程通常比单线程更具优势。例如，在上网时，等待一幅图片加载完毕可能会让人很烦恼。而在Java中，通过使用一个单线程，可以同时加载图片和访问HTML页面中的其他信息，而无须等待图片加载完成。

9. 动态性

Java的动态特性是其面向对象设计方法的重要发展。它允许程序动态装入运行过程中所需要的类，这是C++语言进行面向对象程序设计时所无法实现的。在C++程序设计过程中，每当在类中增加一个实例变量或成员函数后，引用该类的所有子类都必须重新编译，否则可能会因为类型不匹配而导致程序崩溃。然而，在Java中，编译器并不直接将实例变量和成员函数的引用编译为数值引用，而是将这些符号引用信息在字节码中保存下来并传递给解释器，再由解释器在完成动态连接后，将符号引用信息转换为数值偏移量。这样，在存储器中生成的对象不是在编译过程中确定的，而是延迟到运行时由解释器确定的，因此对类中的变量和方法进行更新时就不会影响现存的代码。当解释器执行字节码时，对于符号信息的查找和转换过程通常只会在遇到新的类或方法名时发生一次，随后代码便可以全速执行。在运行时确定引用的好处是可以使用已被更新的类，而不必担心会影响原有的代码。此外，如果Java程序连接了网络中另一系统中的某一类，那么该类的所有者也可以自由对该类进行更新，而不会使任何引用该类的程序崩溃。Java还简化了使用一个升级的或全新的协议的方法。如果系统运行Java程序时遇到了无法处理的类或方法，Java可通过其类加载机制自动下载所需的类库或功能程序。

1.1.3 Java虚拟机

Java虚拟机(JVM)是一种通过软件实现的计算机物理硬件环境的模拟。它作为一个抽象机器，内部包含一个解释器(interpreter)，该解释器可以将其他高级语言编写的程序编译为虚拟机能够执行的中间代码，我们称为中间语言(intermediate language)。通过这种方法，高级语言程序得以实现可移植性和平台无关性(system independence)。无论是运行在嵌入式设备上，还是运行在包含多个处理器的服务器上，虚拟机都执行相同的指令，其所使用的支持库也具有标准的API，并表现出完全相同或相似的行为。

JVM是一种抽象机器，它附着在具体的操作系统上。JVM拥有一套自己的虚拟机器指令，并且配备了诸如栈、寄存器等运行Java程序所必需的运行时机制。编译后的Java程序并不会直接在硬件系统的CPU上执行，而是被转化为字节码后在JVM上执行。JVM中内置了一个Java解释器，用来解释Java编译器生成的字节码。因此，任何一台机器只要安装了JVM并配备了相应的解释器，就可以运行Java程序，而无须关心这种字节码是在何种平台上生成的。

JVM是编译后的Java程序和硬件系统之间的接口，程序员可以将JVM看作一个虚拟处理器，它负责执行编译后的Java字节码。JVM不仅解释并执行这些字节码指令，还进行了一系列的安全检查。正是JVM的这些特性，确保了Java程序能在多平台间实现无缝移植，同时JVM作为Java程序的安全检查引擎，保证了程序的安全执行。计算机硬件、操作系统、JVM与各种可执行程序之间的关系如图1-1所示。

Java基础教程(第5版).indd 4 2024-09-27 14:05:30

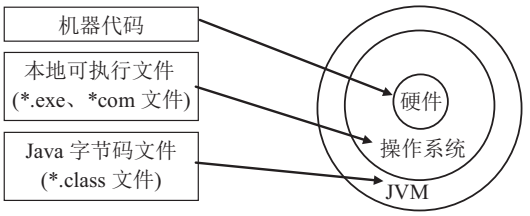

图 1-1　计算机硬件、操作系统、JVM 与各种可执行程序之间的关系

Java虚拟机由多个组件构成，包括类装载器(class loader)、字节码解释器(bytecode interpreter)、安全管理器(security manager)、垃圾收集器(garbage collector)、线程管理(thread management)及图形(graphics)，如图1-2所示。

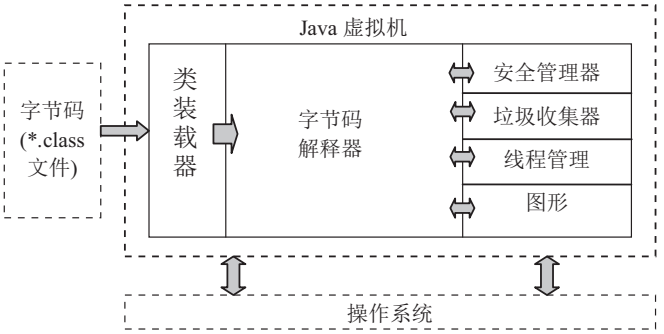

图 1-2　Java 虚拟机体系结构示意图

下面对主要组件进行介绍。

(1) 类装载器：负责加载(load)类的字节码文件，并完成类的链接和初始化工作。首先，它将要加载的类名转换为类的字节码文件名，并在环境变量CLASSPATH指定的每个目录中搜索该文件，然后将字节码文件读入缓冲区；其次，将类转换为JVM内部的数据结构，并使用校验器检查类的合法性。需要注意的是，如果类是第一次被加载，则需对类中的静态数据进行初始化。此外，类装载器加载类中所引用的其他类时，可将其他类中的某些方法编译为本地代码。

(2) 字节码解释器：它是整个JVM的核心组件，负责解释执行由类装载器加载的字节码文件中的指令集合，并通过Java运行环境(JRE)由底层的操作系统实现操作。通常，我们可以通过使用汇编语言编写解释器、重组指令流提高处理器的吞吐量，以及最大限度地使用高速缓存和寄存器等措施来优化字节码解释器。

(3) 安全管理器：它可以根据一定的安全策略对JVM中指令的执行进行控制，主要包括可能影响下层操作系统安全性或完整性的Java服务调用。每个类装载器都与某个安全管理器相关，该安全管理器负责防止由加载器加载的类执行非法操作，从而保护系统安全。默认的类加载器使用的是一个信任型的安全管理器。

(4) 垃圾收集器：垃圾收集器用于检测不再使用的对象，并回收它们所占用的内存。Java语言并非首个采用垃圾收集技术的编程语言。垃圾收集技术已相当成熟，早期的面向对象语言如LISP和SmallTalk就引入了该机制。理想的垃圾收集应该回收所有形式的垃圾，如网络连接、I/O路径等。JVM中垃圾收集的启动方式可分为请求式、要求式和后台式。请求式是调用System.gc( )方法请求JVM进行垃圾收集；要求式是当使用new()方法创建对象时，如果内存资源不足，则请

Java基础教程(第5版).indd 5　2024-09-27 14:05:30

求JVM进行垃圾收集；后台式是如果通过一个独立的线程检测系统的空闲状态发现系统空闲了多个指令周期，则进行垃圾收集。

❖ 提示：

JDK包含Java语言、Java虚拟机、Java类库，是支持Java程序开发的最小环境。

JVM运行于各种操作系统如Linux、Windows、Solaris等上，执行编译好的Java字节码class文件。

JRE包含JavaSE中核心类库API和Java虚拟机，简单理解为JVM+核心类库API。

1.1.4 Java的应用领域

经过三十多年的发展，Java已成为成熟的跨平台高级编程语言，并成为世界上较流行的编程语言之一，广泛应用于各个领域。例如，阿里巴巴、脸书、谷歌、亚马逊和微软等著名公司使用 Java 开发了一系列重要的产品和服务。Java主要应用领域如下。

1. 开发桌面应用程序

Java开发的桌面应用程序可以提供良好的用户体验、稳定性和可靠性。众多大型软件公司都在使用Java开发桌面应用程序，因此，对于希望在大型组织从事软件开发工作的我们来说，Java 无疑是首选。

2. 开发Web应用程序

Java可以用于开发Web应用程序，如电子商务网站、博客、社交媒体网站等。知名网站如淘宝、京东商城、oschina.net等都是使用Java开发的。Java的许多框架和技术使得开发Web应用程序变得更加容易和高效。

3. 开发移动应用程序

当今，有数十亿用户使用Android系统，因此企业对 Android 应用程序的需求日益旺盛。这为 Java 开发人员带来了巨大机遇，因为 Google 已经创建了一个出色的基于 Java 的 Android 开发框架——Android Studio。

4. 开发游戏

Java也可以用于开发游戏，很多知名游戏，如《我的世界》《俄罗斯方块》《贪吃蛇》《模拟人生》及《钻石情迷》等都是基于Java开发的。

5. 开发大数据应用程序

Java可以用于开发大数据应用程序，它提供了很多存储和处理大数据的框架，如Apache Hadoop、Apache Spark和Apache Flink等。这些框架可以帮助企业存储和处理海量数据，从而实现更高效和灵活的大数据处理。

Java基础教程(第5版).indd 6 2024-09-27 14:05:30

1.2　Java程序开发

1.2.1　运行平台

1. 平台简介

Java运行平台主要分为以下3个版本。

- Java SE：Java标准版或Java标准平台，面向桌面应用开发。
- Java EE：Java企业版或Java企业平台，面向企业级开发。
- Java ME：Java微型版或Java小型平台，面向手机等移动终端开发。

❖ 提示：

自JDK 6开始，Java的三大应用平台被重命名为Java SE(标准版)、Java EE(企业版)和Java ME(微型版)，分别对应之前的J2SE、J2EE和J2ME。

本书所有程序均在JDK 17版本下调试通过。

2. 环境变量

环境变量也称系统变量，是操作系统用来与运行中的程序通信的机制，一般可为运行的程序提供配置信息。

常用的Java运行环境变量包括JAVA_HOME、CLASSPATH和PATH。

1) JAVA_HOME环境变量

JAVA_HOME环境变量为需要依赖Java命令和JVM的程序提供了JDK安装目录的路径信息，其值应被设置为JDK的实际安装路径，例如，当Windows平台上JDK的安装目录为“C:\java\jdk17”时，设置如下所示。

```
set  JAVA_HOME= C:\java\jdk17
```

2) CLASSPATH环境变量

CLASSPATH环境变量用于指明Java字节码文件(.class文件)的位置。默认情况下，如果未设置CLASSPATH，Java启动JVM后，会在当前目录下寻找字节码文件。一旦设置了CLASSPATH，JVM会在指定目录下查找字节码文件，至于是否还在当前目录下查找，则包含以下两种情况。

(1) CLASSPATH的路径结尾有“;”，如果在CLASSPATH环境变量值的路径列表的每个路径及其子路径中没有找到指定的字节码文件，则会在当前目录再找一次。

(2) CLASSPATH的路径结尾没有“;”，不会再在当前目录下查找。

另外，如果在所有路径中都找不到该字节码文件，则报告错误。

环境变量CLASSPATH的值一般为一个以分号“;”作为分隔符的路径列表，设置如下所示。

```
set  CLASSPATH=C:\java\jdk17\jre\lib\rt.jar;.;
```


Java基础教程(第5版).indd 7　2024-09-27 14:05:31

3) PATH环境变量

PATH环境变量是操作系统使用的变量，用于搜索在Shell中输入的执行命令。为了便于使用，一般可将JDK中Java命令程序所在目录的路径加入PATH变量的值中，设置如下所示。

```
set PATH= C:\java\jdk17\bin;%PATH%
```

3. JDK17版本的安装

安装步骤如下。

(1) 从www.oracle.com/technetwork/java/javase/downloads网站下载JDK 17(程序名如jdk-17_windows-x64_bin.exe)，然后安装该程序。

(2) 双击，进入安装状态，此时弹出如图1-3所示的对话框。

(3) 单击【下一步】按钮，弹出如图1-4所示的对话框。

图 1-3　安装程序首页界面

图 1-4　目标文件夹 1

(4) 单击【更改】按钮，弹出如图1-5所示的对话框。

(5) 修改安装路径为C:\java\jdk17，单击【确定】按钮，弹出如图1-6所示的对话框。

图 1-5　更改文件夹

图 1-6　目标文件夹 2

(6) 单击【下一步】按钮，弹出如图1-7所示的对话框。

(7) 安装成功后，弹出如图1-8所示的对话框。

(8) 单击【关闭】按钮，安装完毕。

Java基础教程(第5版).indd 8　2024-09-27 14:05:31

图 1-7　安装进度

图 1-8　已完成安装

安装完成后的主要目录有以下几个。

- bin：该路径下存放了JDK的各种工具命令，如javac.exe(Java编译器)、java.exe(Java运行工具)、jar.exe(打包工具)和javadoc.exe(文档生成工具)等。
- conf：该路径下存放了JDK的相关配置文件。
- include：该路径下存放了一些平台特定的头文件。
- jmods：该路径下存放了JDK的各种模块。
- legal：该路径下存放了JDK各模块的授权文档。
- lib：该路径下存放了JDK工具的一些补充JAR包。

❖ 提示：

Java技术官方网站为http:// www.oracle.com/technetwork/java；Eclipse项目网站为http:// www.eclipse.org；各种Java相关开源项目网站为http:// jakarta.apache.org和http:// www.sourceforge.net。

4. 环境变量设置

环境变量设置可采取永久设置和临时设置两种方式。

在Windows系统中永久设置环境变量的步骤如下：右击“此电脑”(或“计算机”)，选择“属性”|“高级系统设置”|“环境变量”选项，在打开的“环境变量设置”窗口中，单击“新建”(或“编辑”“删除”按钮)即可进行环境变量的设置。

一般对“环境变量设置”窗口中没有出现的环境变量采用新建方式来创建一个新的环境变量，而已有的环境变量则采取编辑的方式进行修改，此外，可对一些不用的环境变量进行删除。

❖ 提示：

Windows系统中存在两种环境变量：用户变量和系统变量。其中，用户变量只对当前用户有效，而系统变量对所有用户有效。另外，在这两种环境变量中可以存在重名变量。

Windows系统在执行用户命令时，若用户未给出文件的绝对路径，则应先在当前目录下寻找相应的可执行文件和批处理文件等。若找不到，再依次在系统变量的PATH保存的路径中寻找相应的可执行程序文件(查找顺序是按照路径的录入顺序从左至右，最前面一条的优先级最高，找到命令后不再向后寻找)，如果还找不到，再在用户变量的PATH路径中寻找。如果系统变量和用户变量的PATH中都包含了某个命令，则优先执行系统变量PATH中包含的这个命令。

Windows系统中不区分用户变量和系统变量中名字的大小写，如Path和PATH。

Java基础教程(第5版).indd 9　2024-09-27 14:05:31

下面举例说明JAVA_HOME、CLASSPATH和PATH 3个环境变量的设置。

1) 设置JAVA_HOME环境变量

当“环境变量设置”窗口中没有JAVA_HOME环境变量时，可单击“新建”按钮，然后在弹出的窗口中设置JAVA_HOME环境变量，如设置值为c:\java\jdk17，如图1-9所示。

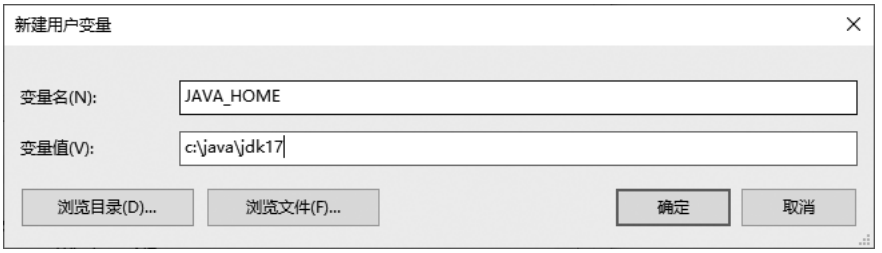

图 1-9　设置 JAVA_HOME 环境变量

2) 设置CLASSPATH环境变量

CLASSPATH环境变量的设置与JAVA_HOME环境变量的设置类似，设置其值为%JAVA_HOME%\lib;.;e:\wu\lib;，如图1-10所示。

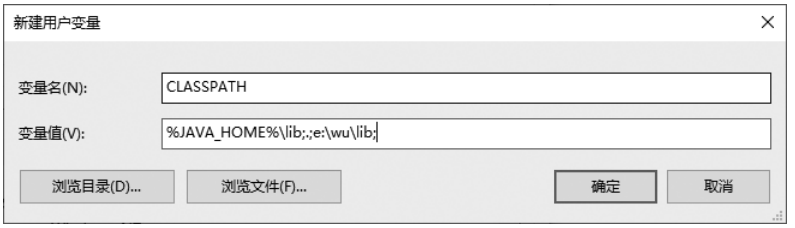

图 1-10　设置 CLASSPATH 环境变量

3) 设置PATH环境变量

PATH环境变量的设置与JAVA_HOME环境变量的设置类似，设置其值为%JAVA_HOME%\bin;%PATH%;，如图1-11所示。

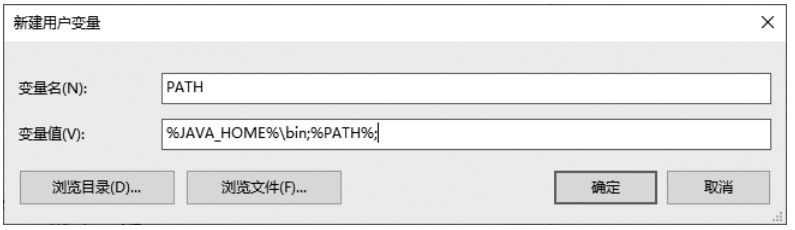

图 1-11　设置 PATH 环境变量

若只是临时使用环境变量，则可在Windows中采取临时设置方式，其具体操作是：运行cmd命令，进入DOS窗口，在DOS提示符下输入设置环境变量的命令，如下所示。

```
set  JAVA_HOME=C:\java\jdk17
set  PATH="%JAVA_HOME%\bin;%PATH%;
……
```


Java基础教程(第5版).indd 10　　2024-09-27 14:05:31

❖ 提示：

为了方便，可以将所有在DOS窗口命令行中需要输入执行的命令放在批命令文件(后缀为.bat)中，本书中将该文件命名为setenv.bat。这样只需在命令行下输入setenv便可执行其中包含的系列命令。

本书中setenv.bat的内容如下。

```
set  JAVA_HOME=C:\java\jdk17
set  CLASSPATH=% JAVA_HOME %\jre\lib\rt.jar;.;e:\wu\lib;
set  PATH=% JAVA_HOME %\bin;%PATH%;
```

读者可根据具体情况来修改批命令文件setenv.bat的内容。

❖ 特别提示：

在安装JDK17时会自动往PATH环境变量中增加一个文件夹(C:\Program Files\Common Files\Oracle\Java\javapath)，称为javapath文件夹，且该文件夹在PATH中置顶。javapath文件夹包含了java.exe、javac.exe、javaw.exe和jshell.exe 4个常用工具。由于javapath提供的工具不完整，所以不建议使用该文件夹中的工具，而使用%JAVA_HOME%\bin文件中的工具。前面在设置环境变量时将PATH设置为%JAVA_HOME%\bin;%PATH%;是为了将文件夹%JAVA_HOME%\bin置顶，以优先使用其下的工具。

5. 如何使用JDK

下面介绍如何在命令行方式下使用JDK。

(1) 按Win+R键，弹出【运行】对话框，如图1-12所示。

(2) 在“打开”文本框中输入cmd命令，单击“确定”按钮，弹出DOS命令窗口，如图1-13所示。

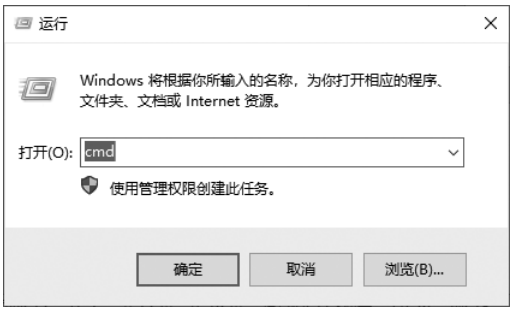

图 1-12　【运行】对话框

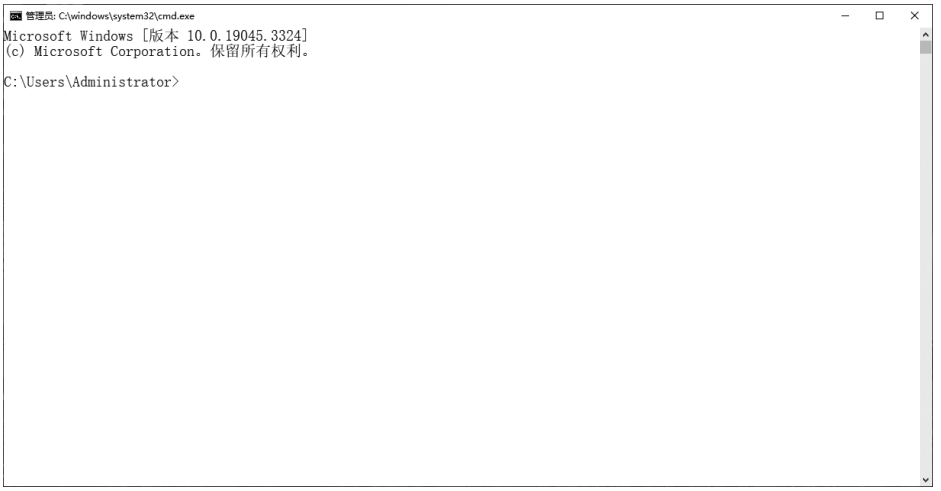

图 1-13　命令窗口

Java基础教程(第5版).indd 11　2024-09-27 14:05:31

(3) 在命令窗口中DOS提示符后面即可输入各种DOS命令，如切换盘符、进入工作目录等。例如，将盘符切换到E盘，并进入工作目录E:\wutask\java后，DOS命令窗口如图1-14所示。

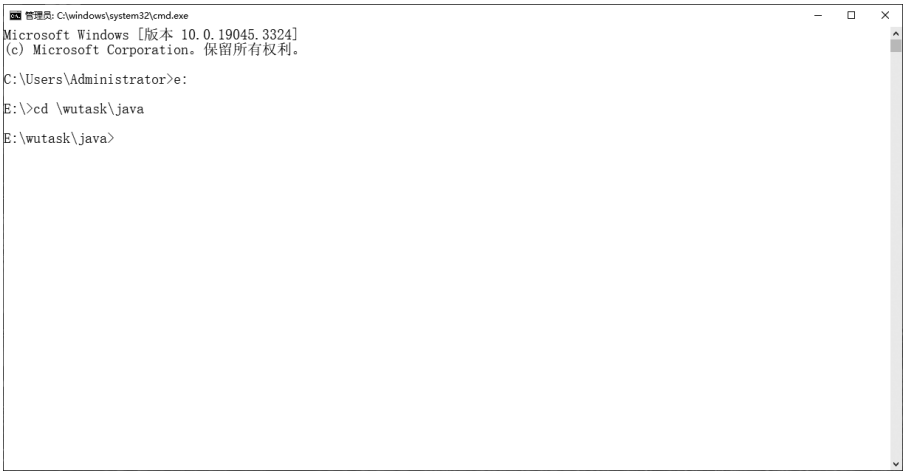

图 1-14　操作后的 DOS 命令窗口

此时，在图1-14中，可在DOS提示符E:\wutask\java>后面输入批命令setenv来设置系统执行文件的位置。

说明：

- 如果已经设置好JAVA_HOME、CLASS和PATH环境变量，则无须执行setenv命令。
- 鉴于许多学生是在公共机房学习，一般不能修改系统信息，因此，建议执行setenv命令来临时设置这些环境变量。

1.2.2　Java程序开发过程

Java程序分为两种：①Application，即“Java应用程序”，是可以独立运行的Java程序，由Java解释器控制执行，也是最常见的类型之一。②Applet，即“Java小程序”，不能独立运行，可以通过 Web 浏览器来加载并运行(嵌入Web页中)。

其中，Applet小程序曾在Web 浏览器中广泛应用，用来实现动态效果、图形呈现、在线游戏等功能，但随着Flash、HTML5 和 CSS3 等 Web 技术的不断发展和完善，以及受Applet程序的安全性和兼容性等问题的影响，Applet小程序已逐渐被开发者抛弃。

本书主要讲授与Application程序相关的知识。在开发Application程序时，需经过编写源文件、编译源文件生成字节码和加载运行字节码3个步骤。

1. 编写源文件

通常，我们可使用任何一个文本编辑器来编写源文件，建议使用Editplus。一个Java程序的源文件由一个或多个互相独立的类组成，并且在这些类中，最多只能有一个类是public类。

对Application程序而言，必须有一个类含有public static void main(String args[ ])方法，其中，args[ ]是main( )方法的一个参数，它是一个字符串类型的数组(注意，String的第一个字母需大写)。

Java基础教程(第5版).indd 12　2024-09-27 14:05:31

假定创建的源文件为JBT11.java，则其内容如下。

```
// 程序名称：JBT11.java
// 功能：演示一个简单Application程序
public class JBT11 {
    public static void main(String args[ ]){
        System.out.println("你好，很高兴学习Java");
    }
}
```

❖ 提示：

Java源文件保存时可选用ANSI、Unicode、UTF-8等编码。

(1)ASNI编码。ASNI是ASCII的扩展，向下包含ASCII。通常，ASCII字符以一个字节来表示，非ASCII字符则以两个字节来表示。在保存Java源文件时并没有固定的ASNI编码，因为使用固定的ASNI编码保存的文件不适合跨地区交流，因此，它与“本地化”(locale)密切相关。例如，在中国内地/大陆地区，ANSI的默认编码是GB2312；在港澳/台地区默认编码是BIG5。再例如，数值0xced2，对于GB2312编码它表示“我”，对于BIG5编码它表示“店”。

(2) Unicode编码。Unicode编码是国际标准字符集，它将世界上的各种语言的每个字符定义一个唯一的编码，以满足跨语言、跨平台的文本信息转换。例如，“我”与“店”的数值分别是0x00006211和0x00006242。

(3) UTF-8编码。UTF-8(universal character set/unicode transformation format，8位元)是针对Unicode的一种可变长度字符编码。它可以用来表示Unicode标准中的任何字符，而且其编码中的第一个字节仍与ASCII相容，使得原来处理ASCII字符的软件无须或只进行少部分修改后，便可继续使用。因此，UTF-8逐渐成为电子邮件、网页及其他存储或传送文字应用中优先采用的编码。

2. 编译Java源程序

Java编译器为执行文件javac.exe。使用编译器的基本格式如下。

```
javac  <options>  <source files>
```

常见的选项options的内容如下。

- -classpath <path>：指定查找用户类文件和注释处理程序的位置。
- -d <directory>：指定放置生成的类文件的位置。
- -encoding <encoding>：指定源文件使用的字符编码。
- -g：生成所有调试信息。
- -version：版本信息。

在Dos提示符下，输入javac并回车，可查看所有选项options。限于篇幅，在此仅列出以上几个常用选项。

下面使用编译器(javac.exe)对源文件JBT11.java进行编译。格式如下。

```
e:\java>javac  JBT11.java
```

编译完后，生成一个名为JBT11.class的字节文件。

Java基础教程(第5版).indd 13　2024-09-27 14:05:31

❖ 提示：

(1) 如果在一个源程序中有多个类定义和接口定义，则在编译时将为每个类生成一个.class文件(每个接口编译后也生成.class文件)。

(2) 保存源文件时，可选择 ANSI、Unicode、UTF-8等编码，在使用javac.exe进行编译时，默认编码为ANSI，如果保存时选择Unicode或UTF-8编码，则按照如下方式编译。

```
e:\java>javac  JBT11.java  -encoding  "unicode"
```

或

```
e:\java>javac  JBT11.java  -encoding  "UTF-8"
```

3. 运行Java源程序

字节码文件必须通过Java虚拟机中的Java解释器(java.exe)来解释执行。Java解释器(java.exe)使用的基本格式如下。

```
java [options] <主类> [args...]
```

其中，常用的选项如下。

- -classpath <目录和 zip/jar 文件的类搜索路径>：指定目录和 zip/jar 文件的类搜索路径。
- -p <模块路径>：指定模块路径。
- -version ：将产品版本输出到错误流并退出。

在Dos提示符下，输入java并回车，可查看所有选项options。限于篇幅，在此仅列出以上几个常用选项。

由于Application程序总是从main( )方法开始执行，因此解释器(java.exe)后面所带的参数应该是包含main( )方法的类对应的class文件名(不含后缀)。格式如下。

```
e:\java>java  JBT11
```

❖ 特别提示：

Java中Application程序命名具有如下特点。

(1) 区分大小写。

(2) 如果程序中有public类，则程序名称必须与public类的名称一致。

(3) 如果程序没有public类，则程序名称可以任取。但建议以包含main( )方法的类的名称作为程序名称，因为无论程序名称如何，使用Java命令运行时，其后的字节码文件一定是main( )方法所在类对应的字节码文件。

1.3 综合应用

【实例1-1】

编写：JBT13.java、JBT14.java、JBT15.java3个源文件，每个源文件只有一个类，分别是JBT13、JBT14和JBT15。JBT13.java是一个应用程序，使用了JBT14.java中的JBT14类和JBT15.java中的JBT15类。将这3个源文件保存到同一目录e:\ java下后，编译JBT13.java。代码如下。

Java基础教程(第5版).indd 14 2024-09-27 14:05:31

```
// 程序名称：JBT13.java
// 功能：演示多类编译运行的情况
public class JBT13{
    public static void main (String args[ ]){
        JBT14 obj1=new JBT14();
        JBT15 obj2=new JBT15();
      int x=1,y=2;
        obj1.add(x,y);
        obj2.minus(x,y);
    }
}

// 程序名称：JBT14.java
class JBT14 {
    void add(int x,int y){
        System.out.println(x+" + "+y+" = "+(x+y));
    }
}
// 程序名称：JBT15.java
public class JBT15 {
    void minus(int x,int y){
        System.out.println(x+" - "+y+" = "+(x-y));
    }
}
```

开发步骤如下。

(1) 打开编辑器，建议使用Editplus。

(2) 按“程序模板”的要求输入源程序。

(3) 保存源文件，分别命名为JBT13.java、JBT14.java和JBT15.java，要求保存在e:\java目录下。

(4) 编译源文件JBT13.java。

```
e:\java>javac  JBT13.java
```

说明：

在编译JBT13.java的过程中，Java系统会自动地编译JBT13.java、JBT14.java和JBT15.java，并相应地生成JBT13.class、JBT14. class和JBT15.class。

(5) 运行程序。

```
e:\java>java  JBT13
```

说明：

运行时，虚拟机仅将JBT13.class、JBT14.class和JBT15.class加载到内存。

【实例1-2】

当程序名称与main( )方法所在类名不一致时，编译命令javac后面的参数是源程序名，而运行java后面的参数是main( )方法所在的类的名称。代码如下。

```
//程序名称：JBT16.java
//功能：演示程序名与main()方法所在类名不一致时的编译和运行
class  A{
    public static void main (Stringargs[ ]){
        JBT16 obj=new JBT16 ();
        obj.show();
    }
```


Java基础教程(第5版).indd 15　　2024-09-29 13:33:45

```
}
public class JBT16{
    void show(){
        System.out.println("Java编译运行测试");
    }
}
```

开发步骤如下。

(1) 打开编辑器，建议使用Editplus。

(2) 按“程序模板”的要求输入源程序。

(3) 保存源文件，命名为JBT16.java。

(4) 编译源文件JBT16.java。

```
e:\java>javac  JBT16.java
```

(5) 运行程序。

```
e:\java>java  A
```

说明：

本程序包含一个public类和一个非public类，因此源程序的前缀名称应该为public类对应的类名称，即程序名称为JBT16.java。由于main()方法所在类为类A，不是类JBT16，因此，运行时java命令后面的字节码文件为类A对应的字节码文件。

1.4 本章小结

本章主要介绍了Java语言的发展历程、Java语言的特点、平台无关性、Java虚拟机JVM的含义、Java运行平台、设置及安装、Java程序开发流程。

1.5 思考和练习

1. 简述Java语言的发展历程。
2. 简述Java语言的特点。
3. 什么是平台无关性？Java语言的平台无关性属于哪种类型？
4. 简述Java虚拟机的工作机制。
5. 简述Java程序的开发过程。
6. 学会安装JDK 17软件，并调试运行本章中所提供的程序。
7. 编写一个简单的Application程序，如下。

```
public  class MyApp{
   public static void main(String args[ ]) {
      MyMath mymath=new MyMath();
      float x=3,y=2;
      System.out.println(x+"+"+y+"="+ mymath.sum(x,y));
   }
}
```


Java基础教程(第5版).indd 16 2024-09-29 13:34:51

```
class  MyMath {
    float sum(float x,float y){
        return x+y;
    }
}
```

要求：

(1) 给出程序的名称。

(2) 写出编译过程。

(3) 写出运行过程。

Java基础教程(第5版).indd 17　2024-09-27 14:05:31