高等学校计算机应用规划教材

Java 基础教程

(第 4 版)

吴仁群　编著

清华大学出版社

北　京

内 容 简 介

本书是针对 Java 语言初学者编写的基础教程，包含 Java 程序设计的基础知识，以及大量实用性很强的编程实例。全书共分 11 章，包括 Java 语言概述、Java 语言基础、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、输入输出及数据库操作、多线程、Applet 程序及应用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，可作为高等学校 Java 程序设计课程的教材，也可作为计算机相关专业的培训和自学教材。

图书在版编目(CIP)数据

Java 基础教程 / 吴仁群编著. —4 版. —北京：清华大学出版社，2021.3
高等学校计算机应用规划教材
ISBN 978-7-302-55060-0

Ⅰ. ①J… Ⅱ. ①吴… Ⅲ. ①JAVA 语言—程序设计—高等学校—教材 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2020)第 039173 号

责任编辑：刘金喜
封面设计：高娟妮
版式设计：孔祥峰
责任校对：牛艳敏
责任印制：丛怀宇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编：**100084
社 总 机：010-62770175　　**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：大厂回族自治县彩虹印刷有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**19.25　　**字　　数：**556 千字
版　　次：2009 年 4 月第 3 版　　2021 年 3 月第 4 版　　**印　　次：**2021 年 3 月第 1 次印刷
定　　价：59.00 元

产品编号：086655-01

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，以及为使教程的结构更清晰，内容描述更简洁明了，以便更适合于初学者学习，本书作者对部分章节进行了修改和完善，主要包括：①对概念进行统一，使得前后章节一致。如将操作元和操作数统一为操作元，将类变量和静态变量统一为静态变量，实例变量和非静态变量统一为非静态变量等。②对部分章节的观点表述进行进一步梳理，使其更简洁、更规范、更容易明白，且不容易出现二义性。③对许多章节的实例进行了更新或补充，以便更能反映所要表达的知识点或知识点的运用，如 swicth(expr)语句应用中，以前举例涉及的 expr 为整型或字符型，事实上 expr 也可以为字符串型，因此增加 expr 为字符串型的实例是必要的；又如增加接口继承的实例，以期对接口应用有更深刻的认识等。④对多线程一章进行大幅度修正，剔除一些淘汰技术，反映最新技术。⑤更正了部分编辑错误。

修改后的第 4 版仍然有 11 章：第 1 章讲述 Java 语言发展历程、Java 语言的特点及开发平台和开发过程；第 2 章介绍 Java 语言编程的基础语法知识；第 3 章和第 4 章讲述 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 章介绍数组和字符串的特点及使用；第 6 章介绍 Java 语言的异常处理机制；第 7 章介绍 Java 类库结构及常用类库；第 8 章介绍 Java 语言中输入输出流和数据库操作方法；第 9 章介绍 Java 语言多线程的含义、特点及实现；第 10 章介绍 Applet 程序的概念及其应用；第 11 章介绍在 Java 语言中如何进行图形用户界面设计及处理功能的实现。

本书 PPT 课件、案例源文件和习题答案等教学资源可通过 http://www.tupwk.com.cn/downpage 下载。

本书由北京印刷学院吴仁群编著。在编写过程中，编者参考了本书“参考文献”所列举的图书，得到了清华大学出版社的大力支持，在此对“参考文献”中所列图书的作者及清华大学出版社表示深深的感谢。

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。

编　者

2020 年 10 月

目　录

∞ 第1章 ∞

Java语言概述

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

本章的学习目标：

- 了解 Java 语言的发展历程。
- 理解 Java 语言的特点。
- 理解 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 语言的特点及相关概念

1.1.1 Java 语言的发展历程

Java 语言的前身是 Oak 语言。1991 年 4 月，Sun 公司(已被 Oracle 公司收购)以 James Gosling 为首的绿色计划项目组(green project)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。项目组成员在使用 C++编译器时发现其有很多不足之处，于是研发出 Oak 语言来替代它，但仅限于 Sun 公司内部使用。

1994 年下半年，由于 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，James Gosling 项目组在对 Oak 语言进行小规模改造的基础上于 1995 年 3 月推出了 Java 语言，并于 1996 年 1 月发布了包含开发支持库的 JDK 1.0 版本。该版本包括 Java 运行环境(JRE)和 Java 开发工具箱(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 外，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。

虽然在 1998 年之前，Java 被众多的软件企业所采用，但由于当时硬件环境和 JVM 的技术尚不成熟，它的应用很有限，那时 Java 主要应用在前端的 Applet 及一些移动设备中，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，在该年 Sun 发布了 JSP/Servlet、EJB

规范，以及将 Java 分成了 J2EE、J2SE 和 J2ME，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，该版本是 Java 语言发展过程中的一个关键阶段，从此 Sun 公司将 Java 更名为 Java 2。经过 10 年的发展，Java 语言已经发展到 1.6 版本。

JDK 1.2 版本可分为 J2EE、J2SE 和 J2ME 三大应用平台。JDK 1.2 版本的 API 分成了核心 API、可选 API 和特殊 API 三大类，其中核心 API 是由 Sun 公司制定的基本的 API，所有的 Java 平台都应该提供，可选 API 是 Sun 为 JDK 提供的扩充 API，特殊 API 是用于满足特殊要求的 API。同时，JDK 1.2 增加了 Swing 图形库，包含各式各样的组件。

从 JDK 1.2 版本开始，Sun 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，Sun 公司发布了 JDK 1.4 版本；2004 年 10 月，Sun 公司发布了 JDK 1.5 版本，同时，Sun 公司将 JDK 1.5 改名为 JDK 5.0。2006 年 4 月，发布了 JDK 6.0 测试版本，并于 2007 年初发布了 JKD 6.0 正式版本;2011 年 7 月发布了 JDK 7.0 版本。2014 年 3 月 18 日，Oracle 公司发表 Java SE 1.8。

在 Java 发展的十几年的时间里，经历了无数的风风雨雨，现在 Java 已经成为一种相当成熟的语言了。在这 10 年的发展中，Java 平台吸引了数百万的开发者，在网络计算遍及全球的今天，Java 已广泛应用于移动电话、桌面计算机、蓝光光碟播放器、机顶盒甚至车载，更是有 30 多亿台设备使用了 Java 技术。

1.1.2 Java 语言的特点

作为一种面向对象且与平台无关的多线程动态语言，Java 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 3 个方面。

- Java 的风格类似于 C++，C++程序员可以很快掌握 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

面向对象编程是一种先进的编程思想，更加容易解决复杂的问题。面向对象可以说是 Java 最重要的特性。Java 语言的设计完全是面向对象的，它不支持类似 C 语言那样的面向过程的程序设计技术。Java 支持静态和动态风格的代码继承及重用。单从面向对象的特性来看，Java 类似于 SmallTalk，但其他特性，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 从诞生起就与网络联系在一起，它强调网络特性，内置 TCP/IP、HTTP 和 FTP 协议类库，便于开发网上应用系统。因此，Java 应用程序可凭借 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完全相同。

4. 安全性

Java 的安全性可从两个方面得到保证：一方面，在 Java 语言中，如指针和释放内存等 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，语言功能和一些浏览器本身提供的功能结合起来，使它更安全。Java 语言在机器上执行前，要经过很多次的测试，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 致力于检查程序在编译和运行时的错误。除了运行时异常检查外，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。类型检查帮助用户检查出许多早期开发中出现的错误。Java 自己操纵内存减少了内存出错的可能性。Java 还实现了真数组，避免了覆盖数据的可能，这项功能大大缩短了开发 Java 应用程序的周期。Java 提供 Null 指针检测数组边界及检测异常出口字节代码校验。同时，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，生成的字节码经过精心设计，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 用直接解释器 1 秒钟内可调用 300 000 个过程。翻译目标代码的速度与 C/C++没什么区别。

7. 与平台无关

Java 编译器将 Java 程序编译成二进制代码，即字节码。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：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(JVM，Java Virtual Machine)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，是从一个大进程中分出来的小的独立的进程。由于 Java 实现了多线程技术，所以比 C 和 C++更健壮。多线程带来的更大好处是更好的交互性能和实时控制性能。当然实时控制性能还取决于系统本身(UNIX、Windows、Macintosh 等)，在开发难易程度和性能上都比单线程要好。例如，上网时都会感觉为查看一幅图片而等待是一件很烦恼的事情，而在 Java 中，可用一个单线程来查看一幅图片，同时可以访问 HTML 中的其他信息而不必等待它。

9. 动态性

Java 的动态特性是其面向对象设计时方法的发展，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一种成员函数后，引用该类的所有子类都必须重新编译，否则将导致程序崩溃。Java 编译器不是将对实例变量和成员函数的引用编译为数值引用，而是将符号引用信息在字节码中保存下来传递给解释器，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，一个在存储器中生成的对象不是在编译过程中确定，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至于影响现存的代码。解释执行字节码时，这种符号信息的查找和转换过程仅在一个新的名字出现时才进行一次，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。如果程序连接了网络中另一系统中的某一类，则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。当系统运行 Java 程序遇到了不知怎样处理的程序时，Java 能自动下载所需要的功能程序。

1.1.3 Java 虚拟机(JVM)

虚拟机是一种对计算机物理硬件计算环境的软件实现。虚拟机是一种抽象机器，内部包含一个解释器(interpreter)，可以将其他高级语言编译为虚拟机的解释器能执行的代码，我们称这种代码为中间语言(intermediate language)，实现高级语言程序的可移植性与平台无关性(system independence)。无论是运行在嵌入式设备还是多个处理器的服务器上，虚拟机都执行相同的指令，所使用的支持库也具有标准的 API 和完全相同或相似的行为。

Java 虚拟机是一种抽象机器，它附着在具体的操作系统上，本身具有一套虚拟机器指令，并有自己的栈、寄存器等运行 Java 程序不可少的机制。编译后的 Java 程序指令并不直接在硬件系统 CPU 上执行，而是在 JVM 上执行。在 JVM 上有一个 Java 解释器用来解释 Java 编译器编译后的程序，任何一台机器只要配备了解释器，就可以运行这个程序，而不管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，程序员可以把 JVM 看作一个虚拟处理器，它不仅解释执行编译后的 Java 指令，而且还进行安全检查，其是 Java 程序能在多平台间进行无缝移植的可靠保证，同时也是 Java 程序的安全检查引擎，如图 1-1 所示。

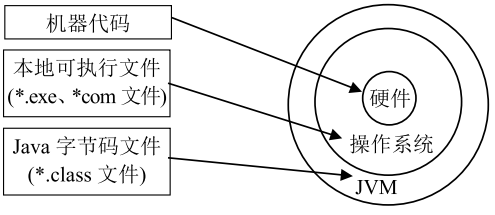

图 1-1　计算机硬件、操作系统、JVM 与各种可执行程序之间的关系

JVM 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。

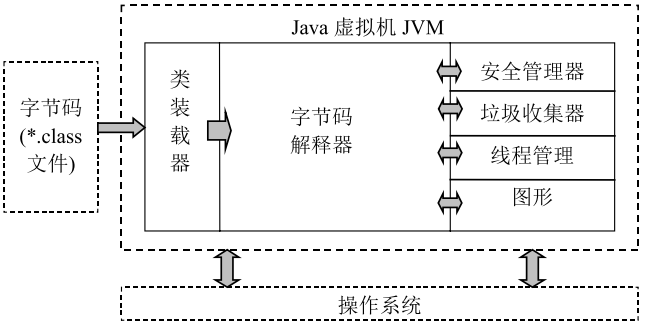

图 1-2　Java 虚拟机体系结构示意图

- 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。类装载器首先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，把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。如果类是第一次被加载，则对类中的静态数据进行初始化。加载类中所引用的其他类，把类中的某些方法编译为本地代码。
- 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字节码指令集合，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，最大限度地使用高速缓存及寄存器等措施来优化字节码解释器。

- 安全管理器：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的安全性或完整性的 Java 服务调用，每个类装载器都与某个安全管理器相关，安全管理器负责保护系统不受由加载器载入系统的类企图执行的违法操作所侵害。默认的类转载器使用信任型安全管理器。
- 垃圾收集器：垃圾收集器用于检测不再使用的对象，并将它们所占用的内存回收。Java 语言并不是第一个使用垃圾收集技术的语言。垃圾收集是一种成熟的技术，早期的面向对象语言 LISP、SmallTalk 等已经提供了垃圾收集机制。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；要求式是使用 new()方法创建对象时，如果内存资源不足，则 JVM 进行垃圾收集；后台式是通过一个独立的线程检测系统的空闲状态，如果发现系统空闲了多个指令周期，则进行垃圾收集。

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。Java SE 提供了标准的 JDK 开发平台。
- Java EE：Java 企业版或 Java 企业平台。
- Java ME：Java 微型版或 Java 小型平台。

提示：

自 JDK 6.0 开始，Java 的 3 个应用平台称为 Java SE、Java EE 与 Java ME(之前的旧名称是 J2SE、J2EE、J2ME)。

本书基于 Java SE 7.0 来介绍 Java 的相关知识，所有程序均在 JDK 7.0 版本下调试通过。

2. 环境变量

环境变量也称为系统变量，是由操作系统提供的一种与操作系统中运行的程序进行通信的机制，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。

环境变量 JAVA_HOME 为需要使用 Java 命令和 JVM 的程序提供了通用的路径信息，其值应设置为 JDK 的安装目录的路径，如在 Windows 平台上 JDK 的安装目录为“C:\java\jdk1.7”时，设置如下所示。

```
set   JAVA_HOME= C: \java \jdk1.7
```

环境变量 CLASSPATH 用于指明字节码文件的位置。没有设置 CLASSPATH 时，Java 启动 JVM 后，会在当前目录下寻找字节码文件(class 文件)。设置后会在指定目录下寻找文件，至于是否还在当前目录下查找，包含以下两种情况。

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

(2) 如果 CLASSPATH 的路径结尾没有“;”，则不会再在当前目录下查找。

如果在所有路径都找不到该字节码文件，就报告错误。

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 \jdk1.7\jre\lib\rt.jar;.;
```

环境变量 PATH 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，一般可把 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 \jdk1.7\bin
```

3. JDK1.7 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 6.0(程序名如 jdk-7u9-windows-i586.exe)，然后安装该程序。

(2) 双击该文件进入安装状态，此时出现一个对话框，如图 1-3 所示。

(3) 在图 1-3 所示对话框中选择【下一步】按钮，此时出现一个对话框，如图 1-4 所示。

图 1-3　协议许可

图 1-4　自定义安装

(4) 在图 1-4 所示对话框中选择【更改】按钮，将安装路径修改为 c:\java\jdk1.7，此时出现一个对话框，如图 1-5 所示。

(5) 在图 1-5 所示对话框中执行【下一步】按钮，此时出现一个对话框，如图 1-6 所示。

图 1-5　修改安装路径

图 1-6　目标文件夹

(6) 在图 1-6 所示对话框中选择【更改】按钮，将此时安装路径变为 c:\java\jre7，出现如图 1-7 所示的对话框。

(7) 在图 1-7 所示的对话框中单击【下一步】按钮，稍等片刻，出现如图 1-8 所示的对话框。

图 1-7　JRE 路径设置

图 1-8　安装完成

(8) 单击【关闭】按钮，至此安装完毕。

安装完毕后的主要目录有以下几个。

- \bin目录：Java开发工具，包括Java编译器和解释器等。
- \demo目录：一些实例程序。
- \lib目录：Java开发类库。
- \jre目录：Java运行环境，包括Java虚拟机和运行类库等。

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

1) 设置环境变量 JAVA_HOME

在 Windows 2000 和 Windows XP 中设置 JAVA_HOME 的步骤如下。

(1) 右击“我的电脑”。

(2) 选择“属性”菜单项。

(3) 在出现的窗口中，选择“高级”选项。

(4) 在出现的窗口中，选择“环境变量”选项。

此时可以设置 JAVA_HOME 变量，结果如图 1-9 所示。

2) 设置环境变量 PATH

为了能在任何目录中使用编译器和解释器，应在系统特性中设置 PATH。

在 Windows 2000 和 Windows XP 中设置 PATH 的步骤同前，结果如图 1-10 所示。

图 1-9　设置环境变量 JAVA_HOME

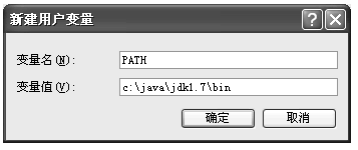

图 1-10　设置环境变量 PATH

3) 设置变量 CLASSPATH

在 Windows 2000 和 Windows XP 中设置 CLASSPATH 的方法同前，结果如图 1-11 所示。

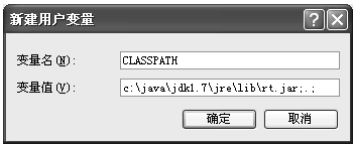

图 1-11　设置变量 CLASSPATH

提示：

在 Windows 7 中设置环境变量的步骤如下。

(1) 右击“计算机”。

(2) 选择“属性”菜单项。

(3) 在出现的窗口中，单击“高级系统设置”。

(4) 在出现的窗口中，单击“环境变量”选项。

(5) 进行环境变量设置。

说明：

Windows 系统中存在两种环境变量：用户变量和系统变量。两种环境变量中是可以存在重名的变量的。用户变量只对当前用户有效，而系统变量对所有用户有效。

Windows 系统在执行用户命令时，若用户未给出文件的绝对路径，则首先在当前目录下寻找相应的可执行文件、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(查找顺序按照路径的录入顺序从左往右寻找，最前面一条的优先级最高，如果找到命令则不再向后寻找)，如果还找不到则在用户变量的 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 设置 Path 和 PATH 并没有区别。

4) 命令行键入命令

若只是临时使用环境变量，则可在 DOS 窗口的命令行输入设置环境变量的命令，如下所示。

```
set   JAVA_HOME=c:\java\jdk1.7
set   PATH="%PATH%";c:\java\jdk1.7\bin;
……
```

提示：

为了方便，可以将所有在 DOS 窗口命令行下需要输入执行的命令放在一个称为批命令文件(后缀为“.bat”)的文件中，本书中将该文件命名为 setpath.bat。这样只需在命令行下输入 setpath 便可以执行其中包含的系列命令。

本书中 setpath.bat 的内容如下。

```
set   PATH=%PATH%;c:\java\jdk1.7\bin;
set   JAVA_HOME=c:\java\jdk1.7
set   CLASSPATH=c:\java\jdk1.7\jre\lib\rt.jar;.;e:\wu\lib;; e:\java;
```

读者可根据具体情况来修改批命令文件 setpath.bat 的内容。

5) 仅安装 JRE

如果只想运行 Java 程序，则可以只安装 Java 运行环境 JRE。JRE 由 Java 虚拟机、Java 的核心类及一些支持文件组成。可以登录 www.oracle.com 网站免费下载 Java 的 JRE。

5. 如何使用 JDK

下面介绍如何在命令行方式下使用 JDK。

(1) 单击【开始】按钮，选择【运行】菜单，此时出现一个对话框，如图 1-12 所示。

图 1-12　【运行】对话框

(2) 在图 1-12 所示的对话框中输入命令“cmd”，单击“确定”按钮，出现一个命令窗口，如图 1-13 所示。

```
C:\Documents and Settings\wu>
```

图 1-13　命令窗口 1

(3) 在图 1-13 所示窗口中 DOS 提示符后面(注：提示符内容视机器而定，这里为 C:\Documents and Settings\wu)输入工作路径所在硬盘的盘符(如 e:)并按 Enter 键，此时出现一个命令窗口，如图 1-14 所示。

```
C:\Documents and Settings\wu>e:
E:\>
```

图 1-14　命令窗口 2

(4) 在图 1-14 所示窗口中 DOS 提示符 E:\>后面输入转换路径的命令“cd 工作路径”，即转换到自己的工作路径，如这里使用的工作路径为 e:\java，则输入的转换工作路径的命令为 cd e:\java，按 Enter 键后出现一个命令窗口，如图 1-15 所示。

```
C:\Documents and Settings\wu>e:
E:\>cd e:\java
E:\java>
```

图 1-15　命令窗口 3

(5) 在图 1-15 所示窗口中 DOS 提示符 e:\java>后面输入批命令 setpath 来设置系统执行文件的位置。

说明：

- 如果已经设置好环境变量 JAVA_HOME、PATH 和 CLASSPATH，则没有必要执行 setpath。
- 鉴于许多学生是在公共机房学习，一般不让修改系统信息，因此，建议执行 setpath 来临时设置这些环境变量。

1.2.2　Java 程序开发过程

利用 Java 可以开发 Application 程序和 Applet 程序。

Application 程序类似于传统的 C 和 C++程序，不需 WWW 浏览器支持就可以直接运行。执行过程：先由 Java compiler 对源代码进行编译，然后由 Java 解释器解释执行。

Applet 程序运行在网页上需要一个驱动的浏览器，如 Sun 的 HotJava、Microsoft 的 Internet Explorer、网景的 Netscape Navigator。执行过程为：编写好 Applet→交给 Java compiler→生成可执

行的字节码→放入 HTML Web 页中→浏览器浏览。

图 1-16 显示了 Application 程序和 Applet 程序的开发过程。

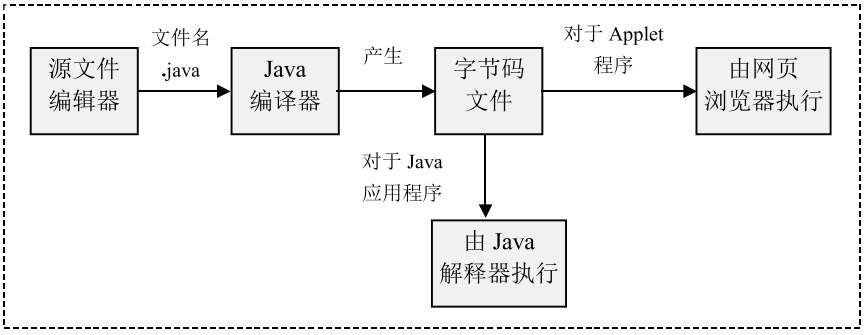

图 1-16　Java 程序开发过程示意图

1. Application 程序的开发

开发一个 Application 程序需经过 3 个步骤：编写源文件、编译源文件生成字节码和加载运行字节码。

1) 编写源文件

我们可使用任何一个文字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书写形式互相独立的类组成。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，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 的第一个字母是大写的)。

假定创建的源文件为 JBT11.java，则其内容如下。

【实例 1-1】

```
//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");
        }
}
```

2) 编译源文件生成字节码

使用编译器(javac.exe)对源文件 JBT11.java 进行编译。

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

编译完后，生成一个名为 JBT11.class 的字节文件。

提示：

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)。

3) 加载运行字节码

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。由于 Application 程序总是从 main()方法开始执行，因此命令 Java 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)。

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

特别提示：

Java 中 Application 程序命名具有如下特点。

- 区分大小写。
- 如果程序中有 public 类，则程序名称必须和 public 类的名称一致。
- 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。

2. Applet 程序的开发

开发一个 Applet 程序需经过 3 个步骤：编写源文件、编译源文件生成字节码，以及通过浏览器加载运行字节码。

1) 编写源文件

一个 Applet 源文件也是由若干个类组成的，一个 Applet 源文件不再需要 main()方法，但必须有且只有一个类扩展了 Applet 类，即它是 Applet 类的子类(Applet 类是系统提供的类)，我们把这个类称作 Applet 源文件的主类。

【实例 1-2】

```
//程序名称：JBT12.java
//功能：演示一个简单 Applet 程序
import java.applet.*;
import java.awt.*;
public class JBT12 extends Applet
{
    public void paint(Graphics g)
    {
        g.setColor(Color.blue);
        g.drawString("Java 是一门很优秀的语言", 12, 30);
        g.setColor(Color.red);
        g.drawString("我一定认真学习 Java", 22, 56);
    }
}
```

2) 编译源文件生成字节码

```
e:\java>java   JBT12.java
```

编译成功后，文件夹 e:\java 下会生成一个 JBT12.class 文件。

3) 通过浏览器运行字节码

Applet 程序由浏览器运行，因此必须编写一个超文本文件(含有 Applet 标记的 Web 页)通知浏览器来运行这个 Applet 程序。

下面是一个最简单的 html 文件(名称由使用者自己确定，这里不妨假定为 JBT12.html)，该文件通知浏览器运行 Applet 程序。使用记事本编辑如下。

```
<applet code=JBT12.class   height=100   width=300>
  </applet>
```

现在可以使用浏览器打开文件 JBT12.html 来运行 Applet 程序，运行结果如图 1-17 所示。

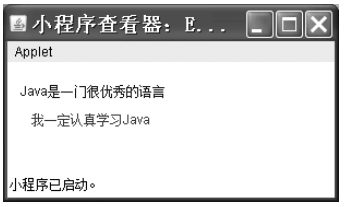

图 1-17　用 IE 打开.html 文件

另外，还可以在 DOS 命令行下使用 appletviewer 来打开网页文件以便执行 Applet 程序如下。

```
e:\java> appletviewer    JBT12.html
```

特别提示：

Java 中 Applet 程序命名具有如下特点。

- 区分大小写。
- 以 Applet 为父类的子类应为 public 类，程序名称与该类的名称一致。

1.3 综合应用

【实例 1-3】

编写 3 个源文件：JBT13.java、JBT14.java、JBT15.java，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 和 JBT15 类。将 3 个源文件提示保存到同一目录 e:\ java 目录下，然后编译 JBT13.java。

```
//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、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 1-4】

当程序名称与 main()方法所在类名不一致时，编译命令 java 后面的参数是源程序名，而运行 java 后面的参数是 main()方法所在类对应的字节码文件。

```
//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辑运行
class   A{
      public static void main (String args[ ]){
          A a=new A();
          a.show();
      }
}
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，因此源程序前缀名称应该为 public 类对应的类名称，即为 JBT16.java。由于 main()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 1.7 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));
   }
}
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。
(4) 写出运行结果。