高等院校课程设计案例精编

赠送光盘中附有完整的案例源代码

Java 课程设计案例精编

张广彬　孟红蕊　张永宝　编著

- 网页浏览器
- 蜘蛛纸牌
- 吃豆子游戏
- 端口扫描器
- 聊天程序
- 连连看游戏
- 中国象棋对弈系统
- 学生管理信息系统

赠送光盘

清华大学出版社

高等院校课程设计案例精编

Java 课程设计案例精编

张广彬　孟红蕊　张永宝　编著

清华大学出版社
北　京

内 容 简 介

Java 语言已成为软件设计开发者应当掌握的一门基础语言。本书为 Java 课程设计指导用书，共分 11 章，具体内容包括： Java 环境的安装与配置、Java 语言编程的基础知识、Java 语言中最重要的类与对象、网页浏览器案例、蜘蛛纸牌案例、吃豆子游戏案例、端口扫描器案例、聊天程序案例、连连看游戏案例、中国象棋对弈系统案例、学生管理信息系统案例。

本书以案例带动知识点的讲解，向读者展示实际项目的设计思想和设计理念，使其可举一反三。每个实例各有侧重点，避免实例罗列和知识点重复，并提供完整的项目实现代码，附于配书光盘中。本书案例典型，选择目前高校课程设计的典型项目，并注重切合实际应用，使读者真正做到学以致用。

本书适合作为高等院校学生 Java 课程设计指导用书，也可作为 Java 语言程序开发人员及爱好者的指导用书。

图书在版编目(CIP)数据

Java 课程设计案例精编/张广彬，孟红蕊，张永宝编著. —北京：清华大学出版社，2007.1
(高等院校课程设计案例精编)
ISBN 978-7-302-14153-2

Ⅰ.J… Ⅱ.①张… ②孟… ③张… Ⅲ.JAVA 语言—程序设计—高等学校—教学参考资料 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2006)第 137584 号

责任编辑：李春明　宋延清
封面设计：山鹰工作室
版式设计：杨玉兰
责任校对：周剑云
责任印制：杜波

出版发行：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
c-service@tup.tsinghua.edu.cn
社 总 机：010-62770175　　**邮购热线**：010-62786544
投稿咨询：010-62772015　　**客户服务**：010-62776969

印 刷 者：北京嘉实印刷有限公司
装 订 者：三河市李旗庄少明装订厂
经　　销：全国新华书店
开　　本：185×260　**印张**：29.5　**字数**：692 千字
版　　次：2007 年 1 月第 1 版　　**印　次**：2007 年 1 月第 1 次印刷
含 1 张光盘
印　　数：1～5000
定　　价：45.00 元

前　言

Java 语言的出现迎合了人们对应用程序跨平台运行的需求，已成为软件设计开发者应当掌握的一门基础语言，很多新的技术领域都涉及到了 Java 语言。目前无论是高校的计算机专业还是 IT 培训学校都将 Java 作为主要的教学内容之一，这对于培养学生的计算机应用能力具有重要的意义，掌握 Java 已经成为人们的共识。

在掌握了 Java 基本知识后，如何快速有效地提高 Java 编程技术成为大家普遍关注的问题。实践证明，案例教学是计算机语言教学最有效的方法之一。好的案例对理解知识和掌握应用十分重要。本书语言通俗，简明实用，书中通过实例来解释相关的概念和方法，并将其运用到实践之中，有助于读者学习。书中各个案例相互独立，给出了详细的设计步骤，包括功能描述、理论基础、总体设计、代码实现、程序运行与发布等；代码都有详细的注释，便于阅读。

本书共分 11 章，具体包括如下内容：

第 1 章　详细讲解 Java 环境的安装与配置。

第 2 章　详细讲解 Java 语言编程的基础知识，包括基本数据类型、运算符与表达式、控制语句、字符串与数组等。

第 3 章　详细讲解 Java 语言中最重要的类与对象的概念。

第 4 章　详细讲解网页浏览器案例。

第 5 章　详细讲解蜘蛛纸牌案例。

第 6 章　详细讲解吃豆子游戏案例。

第 7 章　详细讲解端口扫描器案例。

第 8 章　详细讲解聊天程序案例。

第 9 章　详细讲解连连看游戏案例。

第 10 章　详细讲解中国象棋对弈系统案例。

第 11 章　详细讲解学生管理信息系统案例。

所有案例程序都在 JDK 1.5 运行环境下调试通过。本书代码仅供学习 Java 使用，欢迎读者对不妥之处提出批评和建议。

本书由张广彬、孟红蕊、张永宝编写。由于作者水平有限，书中难免存在疏漏和不足，恳请读者提出宝贵意见，使本书再版时得以改进和完善。

编　者

2006 年 10 月

目　录

第 1 章　Java 概述

学习目标

- 了解 Java 的历史及 Java 语言的特点。
- 熟悉 Java 平台及 Java 虚拟机。
- 掌握 JDK 的安装及环境变量的设置。
- 掌握 JDK 常用工具的使用方法。
- 掌握 Java 程序的编译与运行。

1.1　Java 语言简介

1.1.1　Java 语言的历史

Java 的诞生主要得益于对家用电器的芯片的研制。开始时，开发者想用 C++语言来开发电器的芯片，但是，由于芯片的种类各不相同，因此，程序要进行多次编译。尤其是 C++中的指针操作，稍有不慎，就会引起问题。程序可以出错误，但是家用电器不能出错误。为此，开发者将 C++语言进行简化，去掉指针操作，去掉运算符重载，去掉 C++中的多重继承，得到了 Java 语言，将它变为一种解释执行的语言，并在每个芯片上装上一个 Java 语言虚拟机。刚开始时 Java 语言被称之为 Oak 语言(橡树语言)。

WWW(万维网)的发展则进一步促进了 Java 的应用。刚开始时，WWW 的发展比较缓慢，每个网页上面都是静态的画面，不能与用户进行动态交互操作。即使是后来的 CGI(通用网关接口)也只是在服务器端运行，速度太慢，人们迫切需要能够在浏览器端与用户进行交互，并且使画面能够动起来，但是，WWW 上的计算机各种各样，操作系统也是千差万别，后来人们想到了 Oak 语言，它是解释型执行语言，只要每台计算机的浏览器能够有它的虚拟机，Oak 语言就可以运行，因此 Oak 语言发展起来，后来改名为 Java 语言，成为当前在网络上流行的开发语言。

Java 语言现在逐渐成熟起来，它的类已接近千个，无所不包，而且还可以通过第三方购买中间件，为 Java 语言的发展提供了良好的发展前景。同时它也是跨平台的语言，因此许多软件开发商及硬件开发商也争先恐后地想搭上 Java 语言的快车，都声称支持 Java 语言，它对微软发起了有力的挑战，而且 Sun 公司正努力开发 Java 芯片。

1.1.2　Java 语言的特点

Java 是一种简单的、面向对象的、分布式的、健壮的、安全的、与平台无关的、多线程的、高性能的、动态程序设计语言。

1. 简单易学

Java 语言虽然起源于 C++，但是去掉了 C++语言中难于掌握的指针操作，内存管理非常简单，如果要释放内存资源，你仅需要让其对象的引用等于 null，这样就使操作变得异常简单。

2. 面向对象

Java 是面向对象的编程语言。面向对象技术较好地解决了当今软件开发过程中新出现的种种传统面向过程语言所不能处理的问题，包括软件开发的规模扩大，升级加快，维护量增大以及开发分工日趋细化、专业化和标准化等。面向对象技术的核心是以更接近于人类思维的方式建立计算机模型，它利用类和对象的机制将数据与其上的操作封装在一起，并通过统一的接口与外界交互，使反映现实世界实体的各个类在程序中能够独立、自治和继承。

3. 分布式

Java 包括一个支持 HTTP(超文本传输协议)和 FTP(文件传输协议)等基于 TCP/IP 协议的子库。因此，Java 应用程序可凭借 URL(统一资源定位符)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相同。为分布环境尤其是 Internet 提供动态内容无疑是一项非常宏大的工程，但 Java 的语法特性可很容易地实现这项目标。

4. 健壮性

Java 致力于检查程序在编译和运行时的错误。类型检查可以检查出许多开发早期出现的错误。Java 自行操纵内存，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。这些功能特征大大缩短了开发 Java 应用程序的周期。Java 提供 Null 指针检测数组边界，进行异常出口字节代码校验。

5. 安全稳定

对网络上应用程序的另一个要求是较高的安全可靠性。用户通过网络获取并在本地运行的应用程序必须是可信赖的，不会充当病毒或其他恶意操作的传播者而攻击本地的资源，同时它还应该是稳定的，轻易不会产生死机等错误，使得用户乐于使用。

6. 平台无关性

Java 语言独特的运行机制使得它具有良好的二进制级的可移植性，利用 Java，开发人员可以编写出与具体平台无关，普遍适用的应用程序，大大降低了开发、维护和管理的开销，也就是一次编译，随处运行。

7. 支持多线程

多线程是当今软件开发技术的又一重要成果，已成功应用在操作系统和应用开发等多个领域。多线程技术允许同一个程序有两个或两个以上的执行线索，即同时做两件或多件事情，满足了一些复杂软件的需求。Java 不但内置多线程功能，而且定义了一些用于建立、管理多线程的类和方法，使得开发具有多线程功能的程序变得简单、容易和有效。

8．高性能

如果解释器速度不慢，Java 可以在运行时直接将目标代码翻译成机器指令。Sun 用直接解释器一秒钟内可调用 300,000 个过程。翻译目标代码的速度与 C/C++的性能没什么区别。

9．动态性

Java 的动态特性是其面向对象设计方法的扩展，它允许程序动态地装入运行过程中所需要的类。Java 编译器将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类后，将符号引用信息转换为数值偏移量。这样，一个在存储器生成的对象不在编译过程中决定，而是延迟到运行时由解释器确定，这样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。

1.2　Java 平台简介

1.2.1　Java 平台简介

1998 年 12 月，Sun 发布了 Java 2 平台和 JDK 1.2，这是 Java 发展史上的里程碑。1999 年 6 月，Sun 公司重新组织 Java 平台的集成方法，并将 Java 企业级应用平台作为发展方向。如今，Java 家族已经有三个主要成员：

- J2SE(Java 2 Standard Edition)是用于工作站、PC 机的 Java 标准平台。
- J2EE(Java 2 Enterprise Edition)是可扩展的企业级应用 Java 平台。
- J2ME(Java 2 Micro Edition)是嵌入式电子设备的 Java 应用平台。

本书是基于 J2SE 的教程，利用 Java 可以开发 Java 小程序(Java Applet)、Java 应用程序(Java Application)、服务器端小程序(Servlet)和 JSP 程序(Java Server Page)。Applet 是嵌入在 HTML 文件中的 Java 程序，相当于嵌入在页面之中的脚本。Applet 的大小和复杂性是没有限制的，但由于 Internet 网速的限制，通常 Applet 会很小。对于 Java 开发工具(JDK)而言，应用程序可以理解为从命令行运行的程序。Java 应用程序在最简单的环境中，它的惟一外部输入就是在启动应用程序时所使用的命令行参数。Servlet 和 JSP 都主要工作在服务器端，为 HTTP 服务提供动态的处理。所不同的是 Servlet 是 Java 程序，而 JSP 是 HTML 文件里嵌入了 Java 代码。

1.2.2　Java 虚拟机(JVM)

Java 程序要想运行，必须有 Java 虚拟机(JVM)。Java 虚拟机是编译后的 Java 程序和硬件系统之间的接口，我们可以把 Java 虚拟机看成是一个虚拟的处理器，它可以执行编译后的 Java 指令，还可进行安全检查。Java 虚拟机是在一台真正的计算机上用软件方式实现的一台假想机，其使用的代码存储在.class 文件中。这样一来，利用 Java 虚拟机也便

实现了与平台无关的特点，Java 语言在不同平台上运行时不需要重新编译。Java 虚拟机在执行字节码时，将其解释为具体平台上的机器指令执行。

1.3 Java 运行环境的建立

1.3.1 JDK 简介

JDK(Java Development Kit)即 Java 软件开发工具包，与 Java SDK(Java Software Development Kit)的含义通常是一样的，是 Java 的开发环境。Sun 公司的 Java SDK 是免费的工具，可以到 Sun 公司网站或提供相关下载的网站下载。目前，提供下载的 Java 标准版(J2SE)的版本是称为 Tiger(猛虎)的 1.5.0，不同的版本适合不同的操作系统，读者可以根据自己所用的操作系统下载相应的版本。本书均以 Windows 2000 中文版的操作系统为例进行运行环境的搭建，所使用的 JDK 为最新版本 JDK 1.5。

1.3.2 JDK 的安装

首先要下载 JDK 开发工具，可以从 http://java.sun.com 下载最新的 JDK 开发工具，笔者使用的是 Windows 操作系统，所以下载后的软件名称是 jdk-1_5_0_02-windows-i586-p.exe。下载完成之后运行 jdk-1_5_0_02-windows-i586-p.exe 就可以进行开发工具的安装。安装过程非常简单，接受默认的安装设置就可以了。

需要指明的是，JDK 1.5 与以前的版本有所不同，安装过程分成了开发工具和运行环境两部分的安装，并且默认的安装路径由原来安装在 C:\<jdk-home>改为 C:\Program Files\Java。其中开发工具的默认安装路径是 C:\Program Files\Java\jdk1.5.0_02，运行环境的默认安装路径是 C:\Program Files\Java\jre1.5.0_02。本书选用默认路径进行安装。

提示： 本书用<Java-Home>来代替 JDK 所安装的目录。比如读者将 JDK 安装在 C:\Program Files\Java，则<Java-Home>所代表的路径即为 C:\Program Files\Java。

1.3.3 JDK 运行环境的设置

JDK 运行的环境配置主要有两个方面，即 Path 和 ClassPath 的设置。Path 的设置主要是为了能够在命令行下找到 Java 编译与运行所用的程序；而 ClassPath 的设置主要是为了让 Java 虚拟机能够找到所需的类库。下面均以 Windows 2000 为例，分别讲解 Path 和 ClassPath 的设置方法。

1. Path 的设置

Windows 2000 中 Path 的设置方法如下。

(1) 在 Windows 2000 中用鼠标右击【我的电脑】，在弹出的快捷菜单中选择【属性】命令，打开【系统特性】对话框，单击【高级】标签，打开【高级】选项卡，如图 1.1 所示。

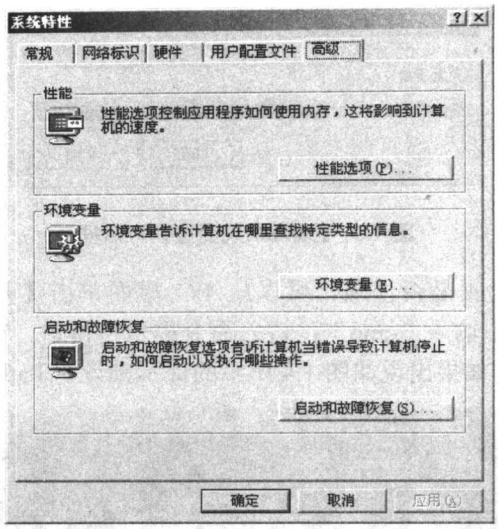

图 1.1　【系统特性】对话框的【高级】选项卡

(2) 在【高级】选项卡内单击【环境变量】按钮，显示出如图 1.2 所示的【环境变量】对话框。

图 1.2　【环境变量】对话框

该对话框中有用户变量和系统变量两项内容，它们的具体区别是使用范围不同，用户变量是针对当前用户所设置的变量，当你用其他用户登录时，该变量将不会影响到其他用户。而系统变量一旦设置任何用户登录都会受到影响。当用户变量与系统变量有相同的变量名而具体的变量值不同时，用户变量优先于系统变量。

(3) 在系统变量中找到 Path 变量，单击【编辑】按钮，如果没有 Path 变量，可单击【新建】按钮添加 Path 变量，在弹出的编辑系统变量窗口中进行编辑，加入<Java-Home>\jdk1.5.0_02\bin，注意目录之间用分号“;”隔开，而且分号的前后不能有空格，单击【确定】按钮，如图 1.3 所示。

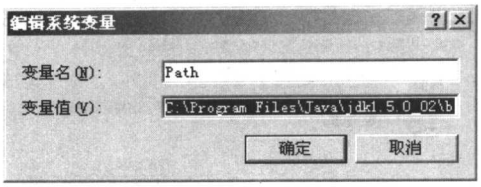

图 1.3 【编辑系统变量】对话框

(4) 为检验 Path 设置是否正确，可以从 PC 桌面单击【开始】按钮，再单击【运行】，在弹出的对话框中输入“cmd”，进入命令提示符窗口。在命令提示符窗口中输入“javac”，按 Enter 键，如果出现如图 1.4 所示的提示则说明 Path 已设置正确。

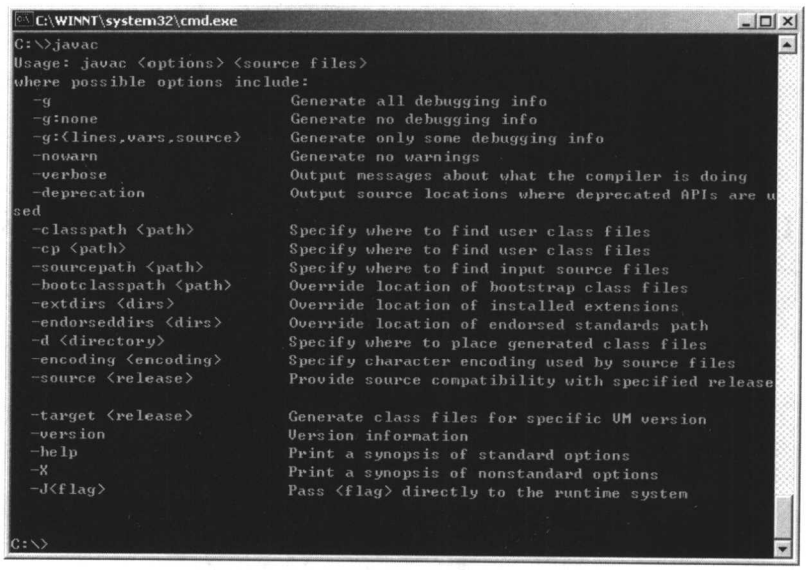

图 1.4 路径设置正确后的命令提示符窗口

2. ClassPath 的设置

如前所述，ClassPath 的设置主要是为了能让 Java 虚拟机能够找到所需的类库，而 Java 虚拟机寻找类库的顺序是：启动类库→扩展类库→用户自定义类库。

启动类库和扩展类库都会在 Java 虚拟机运行时自动加载，而用户自定义类库是不会自动加载的，需要设置路径。所以，我们需要设置的正是用户自定义类库。

设置用户自定义类库的方法比较简单，直接在系统变量中找到 ClassPath 变量(参考设置 Path 变量的方法)，将你所使用的类库的路径加入到 ClassPath 变量的值中即可。例如，你的类库路径为“C:\Javawork\lib”，则 ClassPath 变量的输入值就是该路径，即“C:\Javawork\lib”。也可在命令提示行下输入“set ClassPath = C:\Javawork\lib”，不过该方法只是对当前的命令提示行窗口有效，下次需要连接自定义类库的时候还要重新设置。

1.3.4 JDK 包含的常用工具

JDK 包含的工具均在<Java-Home>\ jdk1.5.0_02\bin 中，其中较常用的工具如下。

- Javac：Java 编译器，用于将 Java 源代码转换成字节码。
- Java：Java 解释器，直接从 Java 的类文件中执行 Java 应用程序字节代码。
- appletviewer：小程序浏览器，一种执行 HTML 文件上的 Java 小程序的 Java 浏览器。
- Javadoc：根据 Java 源码及说明语句生成 HTML 文档。
- Jdb：Java 调试器，可以逐行执行程序，设置断点和检查变量。
- Javah：产生可以调用 Java 过程的 C 过程，或建立能被 Java 程序调用的 C 过程的头文件。
- Javap：Java 反汇编器，显示编译类文件中的可访问功能和数据，同时显示字节代码含义。

提示： 在使用 JDK 所包含的工具中如果遇到困难，可以使用参数“/?”来获得帮助。比如使用“java /?”命令就可以获得 Java 的详细使用帮助。

1.4　JDK 1.5 的新特性

JDK 1.5 的一个重要主题就是通过新增一些特性来简化开发，这些特性包括泛型、for-each 循环、自动装包/拆包、枚举、可变参数和静态导入。使用这些特性有助于编写更加清晰、精悍、安全的代码。下面简单地介绍一下这些新特性。

1．泛型(Generic)

C++通过模板技术可以指定集合的元素类型，而 Java 在 1.5 之前一直没有相对应的功能。一个集合可以放任何类型的对象，相应地从集合里面拿出对象的时候，我们也不得不对它们的类型进行强制转换。JDK 1.5 引入了泛型，它允许指定集合里元素的类型，这样你可以得到强类型在编译时刻进行类型检查的好处。

2．for-each 循环

JDK 1.5 之前是没有 for-each 循环的，JDK 1.5 将 VB 中比较好用的 For-Each 循环引入其中，这样就增强了代码的清晰性与简便性，最突出的优点就是简化了集合的遍历。

3．自动装包/拆包(Autoboxing/unboxing)

自动装包/拆包大大方便了基本类型数据和它们的包装类的使用。

- 自动装包：基本类型自动转为包装类(int >> Integer)。
- 自动拆包：包装类自动转为基本类型(Integer >> int)。

在 JDK 1.5 之前，我们总是对集合不能存放基本类型而耿耿于怀，现在自动转换机制解决了我们的问题。

4．枚举(Enums)

JDK 1.5 加入了一个全新类型的“类”——枚举类型。为此 JDK 1.5 引入了一个新关键字 enum。我们可以定义和使用一个枚举类型，同时还可以使用其两个有用的静态方法

values()和 valueOf()来进行操作。

5. 可变参数(Varargs)

可变参数使程序员可以声明一个接受可变数目参数的方法。但需注意的是，可变参数必须是函数声明中的最后一个参数。

6. 静态导入(Static Imports)

要使用静态成员(方法和变量)，我们必须给出提供这个方法的类。使用静态导入可以使被导入类的所有静态变量和静态方法在当前的类中直接可见，使用这些静态成员无需再给出它们的类名。不过，过度使用这个特性也会在一定程度上降低代码的可读性。

1.5 Java 程序的编写、编译和运行

1.5.1 Java 程序的编译与运行

1. 编译

Java 程序的编译程序是 javac.exe。用 javac 命令将 Java 程序编译成字节码，然后你可用 Java 解释器 java 命令来解释执行这些 Java 字节码。Java 程序源码必须存放在后缀为.java 的文件里。对应 Java 程序里的每一个类，javac 都将生成与类相同名称但后缀为.class 文件。编译器把.class 文件放在.java 文件的同一个目录里，除非你用了-d 选项。当你引用到某些自己定义的类时，必须指明它们的存放目录，这就需要利用环境变量参数 ClassPath。环境变量 ClassPath 是由一些被分号隔开的路径名组成。如果传递给 javac 编译器的源文件里引用到的类定义在本文件和传递的其他文件中找不到，则编译器会按 ClassPath 定义的路径来搜索。例如 ClassPath= .;C:\Javawork\lib 则编译器先搜索当前目录，如果没搜索到则继续搜索 C:\Javawork\lib 目录。系统总是将系统类的目录默认地加在 ClassPath 后面，除非你用-classpath 选项来编译。

提示： 在 ClassPath 的值中，“.”表示当前目录。路径之间的间隔用“;”，表示环境变量 ClassPath 具有多个取值，即一次寻找中的搜索路径。

javac 的具体用法如下：

```
javac [-g][-O][-debug][-depend][-nowarn][-verbose][-classpath path]
[-nowrite][-d directory] file.java
```

以下是对主要选项的解释。

- -classpath path：定义 javac 搜索类的路径。它将覆盖默认的 ClassPath 环境变量的设置。
- -d directory：指明类层次的根目录，格式为：javac -d <my_dir> MyProgram.java，这样就可以将 MyProgram.java 程序里产生的.class 文件存放在 my_dir 目录里。
- -g：带调试信息编译，调试信息包括行号与使用 Java 调试工具时用到的局部变

量信息。如果编译没有加上-O 优化选项，则只包含行号信息。

- -nowarn：关闭警告信息，编译器将不显示任何警告信息。
- -O：优化编译 static、final、private 函数，注意你的类文件可能更大。
- -verbose：让编译器与解释器显示被编译的源文件名和被加载的类名。

2. 运行

当 Java 程序已经编译好，生成.class 文件后，便可以用 java 命令运行了。.class 文件就是 Java 的字节码文件，而 java 命令就是解释运行字节码的解释器。

用 java 命令解释 Java 字节码的语法如下：

```
java [options] classname <args>
java_g [options] classname <args>
```

其中 classname 参数是要执行类的类名、args 是传递给要执行类中 main 函数的参数、options 为可选参数，主要包括如下参数。

- -cs, -checksource：当一个编译过的类被调入时，这个选项将比较字节码更改时间与源文件更改时间，如果源文件更改时间靠后，则重新编译此类并调入此新类。
- -classpath path：定义 javac 搜索类的路径。
- -mx x：设置最大内存分配池，大小为 x，x 必须大于 1000B。默认为 16MB。
- -ms x：设置垃圾回收堆的大小为 x，x 必须大于 1000B。默认为 1MB。
- -noasyncgc：关闭异步垃圾回收功能。此选项打开后，除非显式调用或程序内存溢出，垃圾内存都不回收。本选项不打开时，垃圾回收线程与其他线程异步地同时执行。
- -ss x：每个 Java 线程有两个堆栈，一个是 Java 代码堆栈，一个是 C 代码堆栈。-ss 选项将线程里 C 代码用的堆栈设置成最大为 x。
- -oss x：每个 Java 线程有两个堆栈，一个是 Java 代码堆栈，一个是 C 代码堆栈。-oss 选项将线程里 Java 代码用的堆栈设置成最大为 x。
- -v, -verbose：让 Java 解释器在每一个类被调入时，在标准输出中打印相应信息。

1.5.2　编写简单的 Java 程序

1. 编写 Java 应用程序

【实例 1.1】第一个程序 HelloWorld。

```
/**
 * @HelloWorld.java
 * @author zgb
 */
public class HelloWorld{
    public static void main(String args[]) {
        System.out.println("Hello, World!");
    }
}
```

下面对这个例子进行分析，其中会提到一些概念，而这些概念将在后续章节中讲述。这里你需要记住的，只是写一个 Java 应用程序所需要的基本结构。

首先需要说明的一点是，Java 程序是大小写敏感的。也就是说，Java 程序中需要区分字母的大小写。比如“public”、“Public”和“PUBLIC”就是 3 个不同的标识符。上例中如果把“public”改为“Public”或者“PUBLIC”，编译就不能通过。

程序一开始就是一个以“/** ... */”对包含的注释。关于注释的使用会在下一节详细说明，这里使用注释主要为了告诉你这段程序是写在 HelloWorld.java 文件中的，作者是 zgb。

上例中两处用到 public，这是一个应用范围的限定符。public 限定的内容几乎可以不受限制地应用，相应地还有 protected 和 private，第 3 章将会讲述这些内容。

Java 是完全的面向对象的程序设计语言，即使一个最简单的 Java 程序，也不能没有类。class 用于定义一个类，如 class HelloWorld 定义了类 HelloWorld，其后的一对大括号中定义该类的成员，所以定义一个类的基本结构就是：class 类名{}。有关类和对象的内容，也将在第 3 章讲述。

注意到这里的类名和主文件名是完全一致的，包括大小写。请记住，一个源文件中可以定义一个或多个类，但源文件的主文件名必须和其中一个类的名称完全相同；如果一个源文件中定义了多个类，那么其中只能有一个类可以加上 public 限定符，而且源文件的主文件名必须和该类的名称完全相同。Java 源文件通常以 .java 作为扩展名。

提示： 虽然一个源文件中可以定义多个类，但是强烈推荐一个源文件中只定义一个类，这样会带来很多方便。

本例中为 HelloWorld 类定义了一个 main 方法(函数)，并通过一条输出语句，将“Hello, World!”输出到标准输出设备(控制台显示器)。对于一个可运行的 Java 类来说，这个 main 方法是必需的，而且需要注意以下几点：

- 该 main 方法必须是公有的，即使用 public 限定。
- 该 main 方法必须是类方法，也即静态方法，使用 static 限定。
- 该 main 方法必须且只能带一个 String[] 型(字符串数组类型)的参数。
- 该 main 方法必须没有返回值，即定义返回值的类型为 void。

Java 中向命令提示行输出的函数是 System.out.println()，另外还有一个与之相近的输出语句是 System.out.print()，两者的不同之处是前者在该行输出完之后会自动换行，而后者不会自动换行。

例子已经分析完了，下面做个总结：

- Java 是大小写敏感的。
- 在一个 Java 源文件中只写一个类(可以写多个，建议只写一个)，且该类的类名与源文件的主文件名称完全一致。
- 一个可执行的 Java 类需要像下面这样定义一个 main 方法：

```
public static void main(String[] args) {}
```