新世纪计算机基础教育丛书 | 丛书主编 谭浩强

# Java程序设计题解与上机指导（第二版）

辛运帏 温小艳 蒋慧科 编著

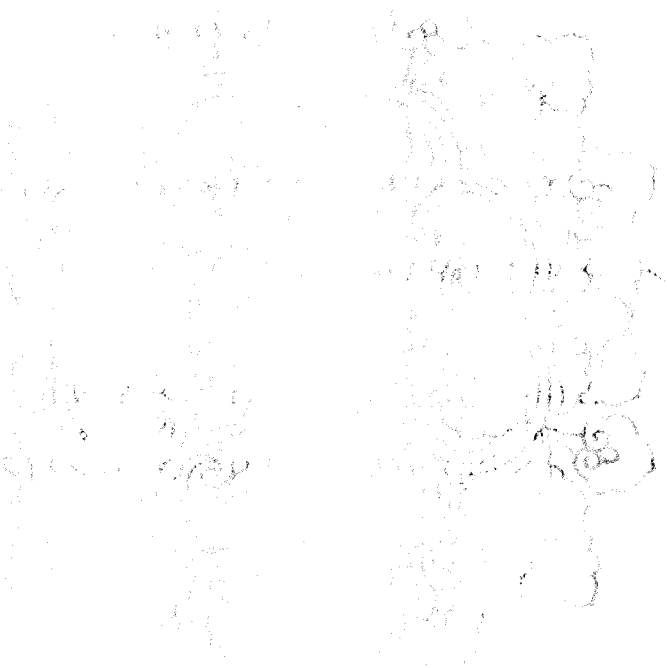
清华大学出版社

新世纪计算机基础教育丛书　　丛书主编 谭浩强

# Java程序设计题解与上机指导

## （第二版）

辛运帏 温小艳 蒋慧科 编著



清华大学出版社
北京

## 内 容 简 介

本书是和《Java 程序设计(第二版)》(辛运帏、饶一梅、马素霞编著,北京,清华大学出版社)一书配套使用的参考书。本书对《Java 程序设计(第二版)》一书中 12 章的全部习题做了完整解答。书中对所有的论述题都给出简单的答案;对所有的编程题都给出了题目的简单分析,论述了设计思路,并给出了完整的程序代码。这些代码均在 Java 5.0 环境下调试通过,程序运行结果用截图的方式示出,供读者参考。

通过学习《Java 程序设计(第二版)》并配合本书的使用,能使读者更深入地了解 Java 语言,熟练掌握它,并能使用该语言编程完成特定的任务。

本书概念清晰、实用性强,可供高等院校学生和学习 Java 语言的读者参考使用。



**图书在版编目(CIP)数据**

Java 程序设计题解与上机指导/辛运帏,温小艳,蒋慧科编著. —2 版. —北京:清华大学出版社,2006.11

(新世纪计算机基础教育丛书)

ISBN 7-302-13852-4

Ⅰ. J… Ⅱ. ①辛…②温…③蒋… 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 113966 号

**出 版 者**:清华大学出版社 **地 址**:北京清华大学学研大厦

http://www.tup.com.cn **邮 编**:100084

**社 总 机**:010-62770175 **客户服务**:010-62776969

**组稿编辑**:焦 虹

**文稿编辑**:孙建春

**印 刷 者**:北京市清华园胶印厂

**装 订 者**:三河市化甲屯小学装订二厂

**发 行 者**:新华书店总店北京发行所

**开 本**:185×260 **印张**:16 **字数**:375 千字

**版 次**:2006 年 11 月第 2 版 2006 年 11 月第 1 次印刷

**书 号**:ISBN 7-302-13852-4/TP·8334

**印 数**:1～4000

**定 价**:21.00 元

# 丛书序言

Preface Preface Preface Preface

现代科学技术的飞速发展，改变了世界，也改变了人类的生活。作为新世纪的大学生，应当站在时代发展的前列，掌握现代科学技术知识，调整自己的知识结构和能力结构，以适应社会发展的要求。新世纪需要具有丰富的现代科学知识，能够独立解决面临的任务，充满活力，有创新意识的新型人才。

掌握计算机知识和应用，无疑是培养新型人才的一个重要环节。计算机技术已深入到人类生活的各个角落，与其他学科紧密结合，成为推动各学科飞速发展的有力的催化剂。无论学什么专业的学生，都必须具备计算机的基础知识和应用能力。计算机既是现代科学技术的结晶，又是大众化的工具。学习计算机知识，不仅能够掌握有关的知识，而且能培养人们的信息素养。它是高等学校全面素质教育中极为重要的一部分。

高校计算机基础教育应当遵循的理念是：面向应用需要；采用多种模式；启发自主学习；重视实践训练；加强创新意识；树立团队精神；培养信息素养。

计算机应用人才队伍由两部分人组成：一部分是计算机专业出身的计算机专业人才，他们是计算机应用人才队伍中的骨干力量；另一部分是各行各业中应用计算机的人员。这后一部分人一般并非计算机专业毕业，他们人数众多，既熟悉自己所从事的专业，又掌握计算机的应用知识，善于用计算机作为工具解决本领域中的任务。他们是计算机应用人才队伍中的基本力量。事实上，大部分应用软件都是由非计算机专业出身的计算机应用人员研制的。他们具有的这个优势是其他人难以代替的。从这个事实可以看到在非计算机专业中深入进行计算机教育的必要性。

非计算机专业中的计算机教育，无论目的、内容、教学体系、教材、教学方法等各方面都与计算机专业有很大的不同，绝不能照搬计算机专业的模式和做法。全国高等院校计算机基础教育研究会自 1984 年成立以来，始终不渝地探索高校计算机基础教育的特点和规律。2004 年，全国高等院校计算机基础教育研究会与清华大学出版社共同推出了《中国高等院校计算机基础教育课程体系 2004》(简称 CFC2004)，2006 年，又推出了《中国高等院校计算机基础教育课程体系 2006》(简称 CFC2006)，由清华大学出版社正式出版发行。

1988 年起，我们根据教学实际的需要，组织编写了《计算机基础教育丛书》，邀请有丰富教学经验的专家、学者先后编写了多种教材，由清华大学出版社出版。丛书出版后，迅速受到广大高校师生的欢迎，对高等学校

的计算机基础教育起了积极的推动作用。广大读者反映这套教材定位准确，内容丰富，通俗易懂，符合大学生的特点。

1999年，根据新世纪的需要，在原有基础上组织出版了《新世纪计算机基础教育丛书》。由于内容符合需要，质量较高，被许多高校选为教材。丛书总发行量突破1000多万册，这在国内是罕见的。

最近，我们又对丛书做进一步的修订，根据发展的需要，增加了新的书目和内容。本丛书有以下特点：

(1) 内容新颖。根据21世纪的需要，重新确定丛书的内容，以符合计算机科学技术的发展和教学改革的要求。本丛书除保留了原丛书中经过实践考验且深受群众欢迎的优秀教材外，还编写了许多新的教材。在这些教材中反映了近年来迅速得到推广应用的一些计算机新技术，以后还将根据发展不断补充新的内容。

(2) 适合不同学校组织教学的需要。本丛书采用模块形式，提供了各种课程的教材，内容覆盖高校计算机基础教育的各个方面。既有供理工类专业用的，也有供文科和经济类专业用的；既有必修课的教材，也包括一些选修课的教材。各类学校都可以从中选择到合适的教材。

(3) 符合初学者的特点。本丛书针对初学者的特点，以应用为目的，以应用为出发点，强调实用性。本丛书的作者都是长期在第一线从事高校计算机基础教育的教师，对学生的基础、特点和认识规律有深入的研究，在教学实践中积累了丰富的经验。可以说，每一本教材都是他们长期教学经验的总结。在教材的写法上，既注意概念的严谨和清晰，又特别注意采用读者容易理解的方法阐明看似深奥难懂的问题，做到例题丰富，通俗易懂，便于自学。这一点是本丛书一个十分重要的特点。

(4) 采用多样化的形式。除了教材这一基本形式外，有些教材还配有习题解答和上机指导，并提供电子教案。

总之，本丛书的指导思想是内容新颖、概念清晰、实用性强、通俗易懂、教材配套。简单概括为："新颖、清晰、实用、通俗、配套"。我们经过多年实践形成的这一套行之有效的创作风格，相信会受到广大读者的欢迎。

本丛书多年来得到各方面人士的指导、支持和帮助，尤其是得到全国高等院校计算机基础教育研究会的各位专家和各高校的老师们的支持和帮助，我们在此表示由衷的感谢。

本丛书肯定有不足之处，竭诚希望得到广大读者的批评指正。

欢迎访问谭浩强网站：http://www.tanhaoqiang.com

丛书主编

全国高等院校计算机基础教育研究会会长

谭浩强

2005年1月1日

# 前言

Java语言自问世以来，经历了多次的版本升级，从安全机制、语法成分到API函数都有较大的修改。基于JDK 1.1写就的《Java程序设计》(2001年9月由清华大学出版社出版)也应该与时俱进，以便向读者介绍Java最新版本的内容。为此我们全面修订了《Java程序设计》，将新版本的内容与Java基础知识一起集成到《Java程序设计(第二版)》(已由清华大学出版社于2006年8月出版)中。为配合使用该书，我们编写了这本《Java程序设计题解与上机指导(第二版)》，旨在帮助读者尽快掌握Java语言。

本书对《Java程序设计(第二版)》一书中12章的全部习题做了完整解答。所有的论述题都给出简单的答案，内容主要摘自《Java程序设计(第二版)》一书。所有的编程题都给出了题目的简单分析，论述了设计思路，并给出了完整的程序代码。这些题目均在Java 5.0环境下调试通过，程序运行结果以截图的方式提供给读者，以供参考。

众所周知，大部分的习题解答不都具有唯一性，特别是程序设计题目，给读者发挥潜能的余地非常大。本书中给出的这些解答仅供参考，希望能起到抛砖引玉的作用。因为编者水平的局限性，书中的答案难免存在这样那样的问题，实现的代码也并不一定是最优的，读者可以参考本书中的内容和其他参考书中的内容，得出自己的更全面的答案。至于程序代码，其实现的方式就更加多种多样，相信读者能在本书的代码基础之上，编写出功能更全面、效率更高的程序。

计算机技术是不断发展、不断完善的技术，Java语言也是如此。从诞生之日起，它的版本一直在更新中。就在本书编写过程及读者使用本书期间，相信Java又有了新的发展。读者应及时把握这些新动向，了解最新版本的相关信息，特别是及时更新自己机器上的JDK，以保持自己设计的代码与新版本的同步。

本书是一本教学参考书，希望读者在使用、调试本书中代码的同时，既能加深对语言的理解，又能提高程序设计的能力，并在此过程中不断发现问题、思考问题、解决问题，把本书作为掌握知识的一个工具和桥梁。

本书由辛运帏、温小艳、蒋慧科编写，并运行通过了所有程序代码。

由于作者水平有限，对 Java 语言的掌握不够全面，书中难免有错误和不妥之处，恳请广大读者特别是同行专家批评指正，在此我们表示深深的谢意。

编者

2006 年 8 月

于南开园

# 目 录

Catalog Catalog Catalog Catalog

# 第1章 概 述

**1.1** 简述 Java 语言的特点。

**解**：Java 是简单的、面向对象的语言，并具有分布性、安全性和健壮性等特点。在 1995 年由美国 Sun 公司向公众推出。具体地说，Java 语言有如下显著的特点。

1. 语法简单，功能强大

Java 语言的语法非常像 C ++，同时去掉了 C ++ 中不常用且容易出错的地方。例如，Java 中没有指针、结构等概念，没有预处理器，程序员不必自己释放占用的内存空间，因此在一定程度上减少了因内存混乱而导致的系统崩溃。另外，Java 强调其面向对象的特性，用它可以编制出非常复杂的系统。Java 具备强大功能的同时，其解释器只占用很少的内存，适合在各种类型的机器上运行。

2. 分布式与安全性

Java 强调网络特性，内置了 TCP/IP、HTTP、FTP 协议类库，具备强大且易于使用的联网能力，便于开发网上应用系统。

Java 程序的三级代码安全检查机制可以有效地防止非法代码的侵入，阻止对内存的越权访问，能够避免病毒的侵害，成为 Internet 上最安全的技术之一。其新的安全机制在具备足够的安全特性的基础之上，又给了程序设计人员一定的自由度，允许他们设计功能更加强大的程序。

3. 与平台无关

提到 Java，有一句著名的口号：一次编写，到处运行。Java 语言规定了统一的数据类型，它们在任何机器上占用的内存大小都是不变的。Java 编译器将 Java 程序编译成二进制代码，即字节码(bytecode)。运行时环境针对不同的处理器指令系统，把字节码转换为不同的具体指令，因此 Java 可以跨平台使用，特别适合于网络应用，同时也为 Java 程序跨平台的无缝移植提供了很大的便利。

4. 解释编译两种运行方式

Java 程序可以经解释器得到字节码，所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，突破了以往解释性语言运行效率低的瓶颈。在现在的 Java 版本中又加入了编译功能(即 just-in-time 编译器，简称 JIT 编译器)，生成器将字节码转换成本机的机器代码，然后可以以较高速度执行，使得执行效率大幅度提高，达到了编译语言的水平。

5. 多线程

Java 内置了语言级多线程功能，可使用户程序并行执行。Java 提供的同步机制可保证各线程对共享数据的正确操作，完成各自的特定任务。在硬件条件允许的情况下，这些线程可以直接对应到各个 CPU 上，充分发挥硬件性能，减少用户等待时间。

6. 动态执行

Java 执行代码是在运行时动态载入的，程序可以自动进行版本升级，在网络环境下，可用于瘦客户机架构，减少维护工作。另外，类库中增加的新方法和其他实例，不会影响到原有程序的执行。

7. 丰富的 API 文档和类库

Java 为用户提供了详尽的 API 文档说明。Java 开发工具包中的类库包罗万象，应有尽有，程序员的开发工作可以在一个较高的层次上展开。这也正是 Java 受欢迎的重要原因之一。

**1.2** 什么是 Java 虚拟机？它包括哪几部分？

**解**：Java 虚拟机(Java virtual machine，JVM)规范中给出了它的定义：JVM 是在一台真正的机器上用软件方式实现的一台假想机。Java 虚拟机是运行 Java 程序必不可少的机制，它是编译后的 Java 程序和硬件系统之间的接口，程序员可以把 JVM 看作一个虚拟的处理器。编译后的 Java 程序指令并不直接在硬件系统的 CPU 上执行，而是由 JVM 执行。它不仅解释执行编译后的 Java 指令，而且还进行安全检查。它是 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 程序的安全检验引擎。

JVM 的具体实现包括：指令集(等价于 CPU 的指令集)、寄存器组、类文件格式、栈、垃圾收集堆、内存区。

**1.3** 简述 JVM 的工作机制。

**解**：JVM(Java 虚拟机)是运行 Java 程序必不可少的机制。编译后的 Java 程序指令并不直接在硬件系统的 CPU 上执行，而是由 JVM 执行。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 看作一个虚拟的处理器。它不仅解释执行编译后的 Java 指令，而且还进行安全检查。它是 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 程序的安全检验引擎。

JVM 是在一台真正的机器上用软件方式实现的一台假想机。JVM 使用的代码存储在.class 文件中。JVM 的部分指令很像真正的 CPU 指令，包括算术运算、流控制和数组元素访问等。

Java 虚拟机规范提供了编译所有 Java 代码的硬件平台。因为编译是针对假想机的，所以该规范能让 Java 程序独立于平台。它适用于每个具体的硬件平台，以保证为 JVM 编译的代码的运行。JVM 不但可以用软件实现，而且可以用硬件实现。

JVM 的代码格式为压缩的字节码，效率较高。由 JVM 字节码表示的程序必须保持原来的类型规定。Java 主要的类型检查是在编译时由字节码校验器完成的。Java 的任何解释器必须能执行符合 JVM 定义的类文件格式的任何类文件。

Java 虚拟机规范对运行时数据区域的划分及字节码的优化并不做严格的限制，它们的实现依平台的不同而有所不同。

**1.4** Java 语言的安全机制有哪些？

**解**：在 Java 的不同版本中，都有不同的安全机制。在最初的 JDK 1.0 版本中，安全模型是所谓的“沙箱”模型，从网络上下载的代码只能在一个受限的环境中运行，这个环境像个箱子一样限制了代码能访问的资源。

在随后发布的 JDK 1.1 版本中，提出了“签名 Applet”的概念。有正确签名的 Applet 视同本地代码一样，可以使用本地的资源。没有签名的 Applet 还与前一版本一样，只在沙箱中运行。

在 Java 2 平台下，安全机制又有较大改善。它允许用户自己设定相关的安全级别。另外，对于应用程序，也采取了和 Applet 一样的安全策略，程序员可以根据需要对本地代码或是远程代码进行设定，以保证程序更安全高效地运行。

在 Java 程序环境中，重要的几个组成部分包括 Java 解释器、类下载器及字节码校验器。

1. Java 解释器

Java 解释器只能执行为 JVM 编译的代码。Java 解释器有三项主要工作：

(1) 下载代码——由类下载器完成。

(2) 校验代码——由字节码校验器完成。

(3) 运行代码——由运行时解释器完成。

2. 类下载器

Java 运行时系统区别对待来自不同源的类文件。它可能从本地文件系统中下载类文件，也可能从 Internet 上使用类下载器下载类文件。运行时系统动态决定程序运行时所需的类文件，并把它们下载到内存中，将类、接口与运行时系统相连接。类下载器把本地文件系统的类名空间和网络源输入的类名空间区分开来，以增加安全性。因为内置的类总是先被检查，所以可以防止起破坏作用的应用程序的侵袭。

所有的类下载完毕后，开始确定可执行文件的内存分配。此时，指定具体的内存地址，并创建查询表。因为内存分配是在运行时进行的，并且 Java 解释器阻止访问可能给操作系统带来破坏的非法代码地址，从而增加了保护性。

3. 字节码校验器

Java 代码在机器上真正执行前要经过几次测试。程序通过字节码校验器检查代码的安全性，字节码校验器检测代码段的格式，并使用规则来检查非法代码段——伪造的指针、对目标的访问权限违例或是试图改变目标类型或类的代码。通过网络传送的所有类文件都要经过字节码校验器的检验。

字节码校验器要对程序中的代码进行四趟扫描，这可以保证代码将依从 JVM 规范，并且不破坏系统的完整性。校验器主要检查以下几项内容：

(1) 类遵从 JVM 的类文件格式。

(2) 不出现访问违例情况。

(3) 代码不会引起运算栈溢出。

(4) 所有运算代码的参数类型总是正确的。

(5) 不会发生非法数据转换，如把整数转换为指针。

(6) 对象域访问是合法的。

如果完成所有的扫描之后不返回任何错误信息，就可以保证 Java 程序的安全性了。

**1.5** Java 的垃圾收集机制与其他语言相比有什么特点？

**解**：许多程序设计语言允许在程序运行时动态分配内存，并将所分配的内存块的开

始地址以指针的形式返回给程序，供程序员使用。

一旦不再需要所分配的内存，程序或运行时环境最好将内存释放，避免内存越界时得到意外结果。

在 C 和 C ++（及其他许多语言）中，由程序开发人员负责内存的释放，但程序开发人员并不总是清楚内存应该在何时释放。而如果不及时释放不再需要的内存，或释放了仍在使用的内存，可能导致程序崩溃或系统混乱。这些不能正确使用内存的程序被称为有“内存漏洞”。

在 Java 中，程序员不必亲自释放内存，它提供了后台系统级线程，记录每次内存分配的情况，并统计每个内存指针的引用次数，引用次数为 0 则表示不再使用。在 Java 虚拟机运行时环境闲置时，垃圾收集线程将检查是否存在引用次数为 0 的内存指针；如果有的话，则垃圾收集线程把该内存“标记”为“清除”（释放）。

在 Java 程序生存期内，垃圾收集将自动进行，无须用户释放内存，从而消除了内存漏洞。

**1.6** 上机调试 1.2 节中的程序 1-1，直到得到正确结果。

**解**：程序中，第 6 行代码将在屏幕上显示一条信息：

Hello World!

使用编辑器（例如记事本）编辑程序 1-1，并保存之。文件名为 HelloWorldApp.java，保存文件时注意文件名的大小写。然后使用命令行 javac HelloWorldApp.java 编译程序，得到类文件 HelloWorldApp.class；使用命令行 java HelloWorldApp 执行该程序。执行结果如图 1-1 所示。

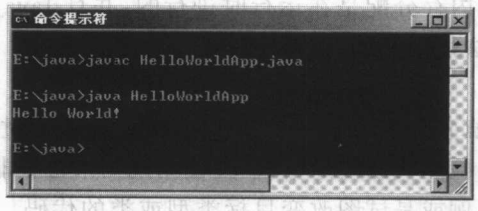



**图 1-1　程序 1-1 的执行结果**

**1.7** 练习使用浏览器查看 Java API 文档。

**解**：目前大多数读者使用的个人工作环境可能是 Windows 系列操作系统，包括 Windows 98、Windows NT、Windows 2000、Windows Me、Windows XP 等，所以本习题集中的解答以 Windows 环境为平台，并使用最新的 JDK 5.0 Documentation 版本。

假设读者目前的系统中已安装有 JDK 5.0 Documentation 文档，双击图标并启动它。此时出现如图 1-2 所示的窗口。

单击 API & Language 链接，进入 API & Language Documentation 窗口，如图 1-3 所示。

单击 Java 2 Platform API Specification 链接，进入如图 1-4 所示的窗口。

这个窗口分为三部分：左上部分显示 Java JDK 5.0 中提供的所有包的信息，选中某个包后，将在左下部分显示这个包中所有接口及类的信息。例如，我们选择查看 java.

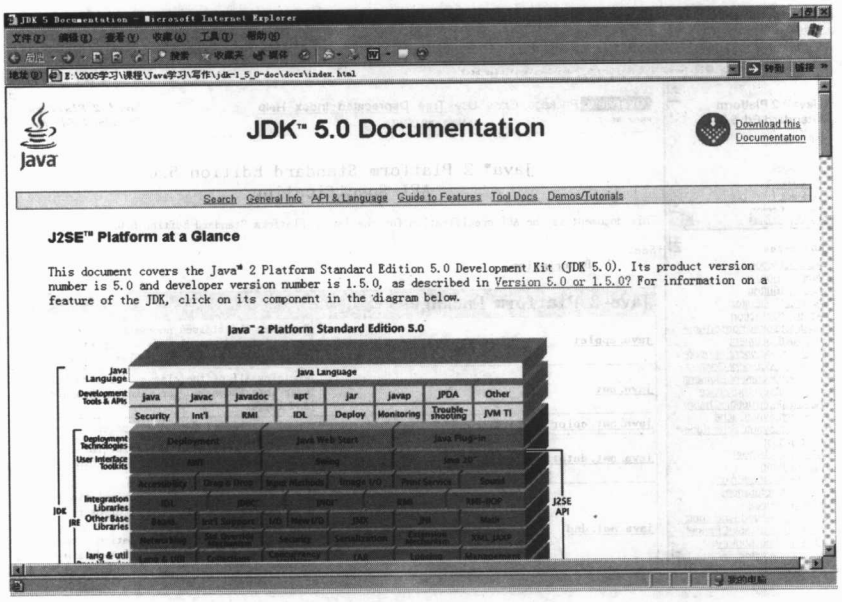



**图 1-2　JDK 的初始界面**

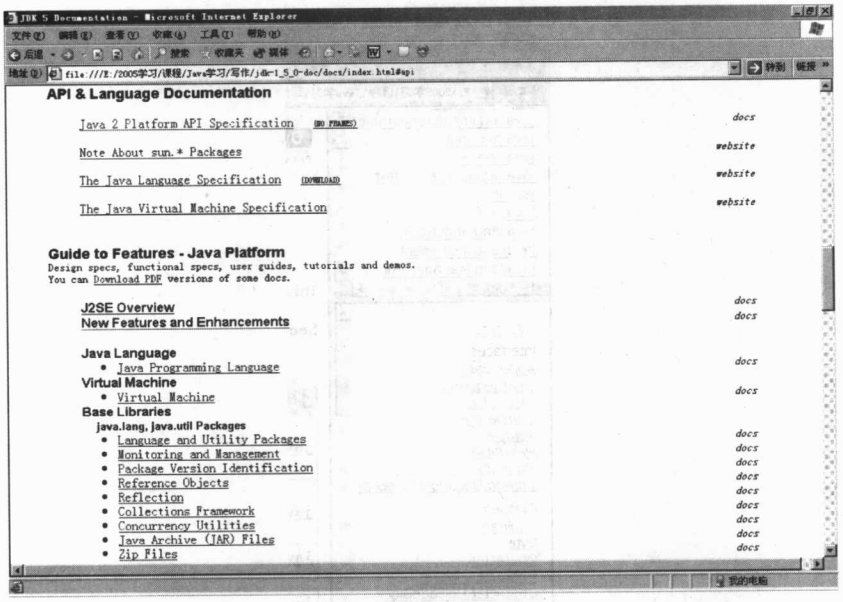



**图 1-3　API 文档**

lang 包，窗口显示的内容如图 1-5 所示。

如果想进一步查看包中 Integer 类的信息，单击 Integer 链接，右侧窗口部分将显示 java. lang 中 Integer 类的所有接口及类的内容，向下拉动滚卷条，定位到所需的位置就可以了。

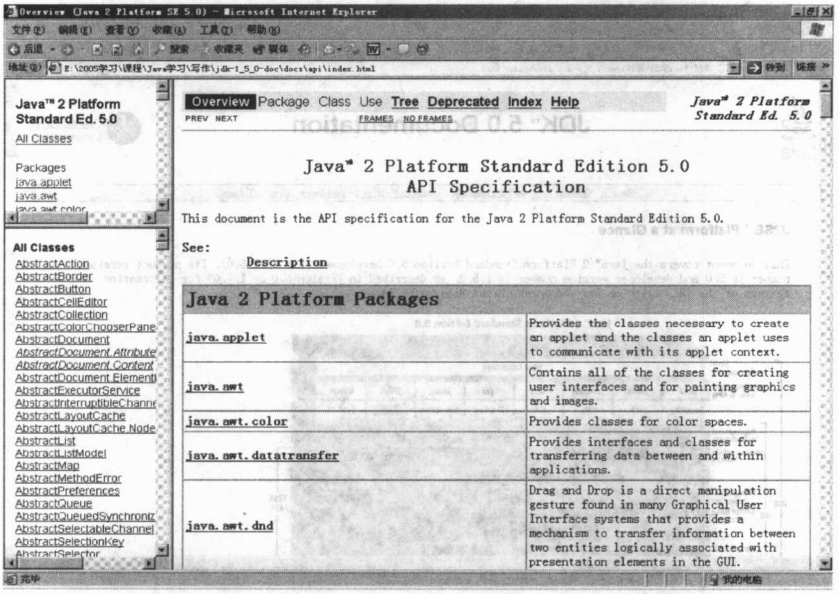



图 1-4　API 文档(续)


图 1-5　java.lang 包内信息列表

一般地,一个类中的信息包括以下几部分:

- Field Summary;

- Constructor Summary;
- Method Summary;
- Field Detail;
- Constructor Detail;
- Method Detail。

Field Summary 中列出类中成员变量的信息，包括名字、类型及含义。Field Detail 中将详细介绍这些成员变量。

Constructor Summary 中列出类构造方法的信息，包括参数列表并解释所创建的实例。构造方法的详细信息显示在 Constructor Detail 部分中。

在 Method Summary 中可以查找到要使用的方法名，在 Method Detail 中将详细介绍该方法的使用方法，包括调用参数表及返回值情况。例如，图 1-6 所示的窗口中显示的是 toString 方法的详细情况，该方法将返回整数所对应的字符串。

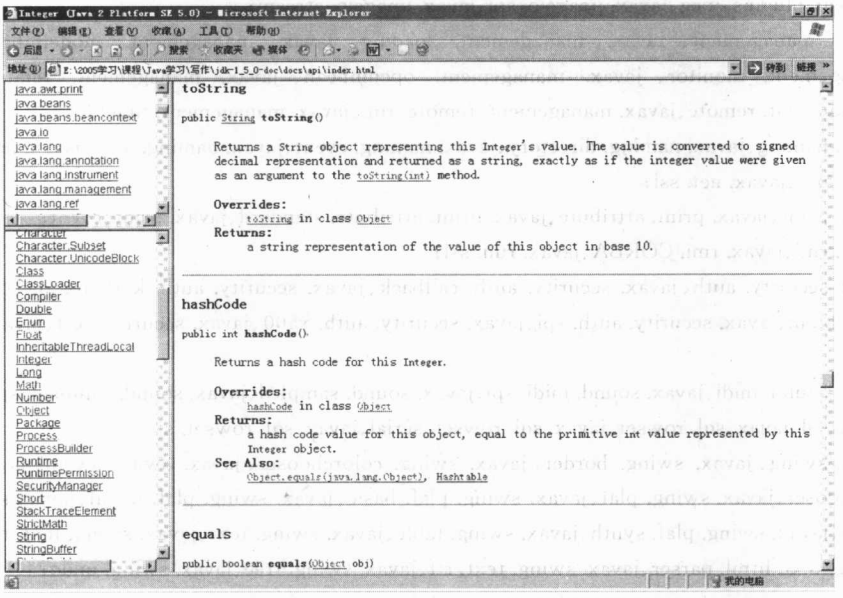



**图 1-6　toString 方法的介绍**

**1.8**　列出 Java API 文档中所有的包名。

**解**：进入 API 文档，可以查看预提供的所有的包的信息。Java API 文档中共包含 32 个大包，有些大包中又包含子包。所有包名如下：

java. applet;

java. awt、java. awt. color、java. awt. datatransfer、java. awt. dnd、java. awt. event、java. awt. font、java. awt. geom、java. awt. im、java. awt. im. spi、java. awt. image、java. awt. image. renderable、java. awt. print;

java. beans、java. beans. beancontext;

java. io;

java. lang、java. lang. annotation、java. lang. instrument、java. lang. management、java. lang. ref、java.

lang. reflect;
java. math;
java. net;
java. nio、java. nio. channels、java. nio. channels. spi、java. nio. charset、java. nio. charset. spi;
java. rmi、java. rmi. activation、java. rmi. dgc、java. rmi. registry、java. rmi. server;
java. security、java. security. acl、java. security. cert、java. security. interfaces、java. security. spec;
java. sql;
java. text;
java. util、java. util. concurrent、java. util. concurrent. atomic、java. util. concurrent. locks、java. util. jar、java. util. logging、java. util. prefs、java. util. regex、java. util. zip;
javax. accessibility;
javax. activity;
javax. crypto、javax. crypto. interfaces、javax. crypto. spec;
javax. imageio、javax. imageio. event、javax. imageio. metadata、javax. imageio. plugins. bmp、javax. imageio. plugins. jpeg、javax. imageio. spi、javax. imageio. stream;
javax. management、javax. management. loading、javax. management. modelmbean、javax. management. monitor、javax. management. openmbean、javax. management. relation、javax. management. remote、javax. management. remote. rmi、javax. management. timer;
javax. naming、javax. naming. directory、javax. naming. event、javax. naming. ldap、javax. naming. spi;
javax. net、javax. net. ssl;
javax. print、javax. print. attribute、javax. print. attribute. standard、javax. print. event;
javax. rmi、javax. rmi. CORBA、javax. rmi. ssl;
javax. security. auth、javax. security. auth. callback、javax. security. auth. kerberos、javax. security. auth. login、javax. security. auth. spi、javax. security. auth. x500、javax. security. cert、javax. security. sasl;
javax. sound. midi、javax. sound. midi. spi、javax. sound. sampled、javax. sound. sampled. spi;
javax. sql、javax. sql. rowset、javax. sql. rowset. serial、javax. sql. rowset. spi;
javax. swing、javax. swing. border、javax. swing. colorchooser、javax. swing. event、javax. swing. filechooser、javax. swing. plaf、javax. swing. plaf. basic、javax. swing. plaf. metal、javax. swing. plaf. multi、javax. swing. plaf. synth、javax. swing. table、javax. swing. text、javax. swing. text. html、javax. swing. text. html. parser、javax. swing. text. rtf、javax. swing. tree、javax. swing. undo;
javax. transaction、javax. transaction. xa;
javax. xml、javax. xml. datatype、javax. xml. namespace、javax. xml. parsers、javax. xml. transform、javax. xml. transform. dom、javax. xml. transform. sax、javax. xml. transform. stream、javax. xml. validation、javax. xml. xpath;
org. ietf. jgss;
org. omg. CORBA、org. omg. CORBA_2_3、org. omg. CORBA_2_3. portable、org. omg. CORBA. DynAnyPackage、org. omg. CORBA. ORBPackage、org. omg. CORBA. portable、org. omg. CORBA. TypeCodePackage;org. omg. CosNaming、org. omg. CosNaming. NamingContextExtPackage、org. omg. CosNaming. NamingContextPackage; org. omg. Dynamic、org. omg. DynamicAny、org. omg. DynamicAny. DynAnyFactoryPackage、org. omg. DynamicAny. DynAnyPackage;org. omg. IOP、org. omg. IOP. CodecFactoryPackage、org. omg. IOP. CodecPackage; org. omg. Messaging; org. omg. PortableInterceptor、org. omg. PortableInterceptor. ORBInitInfoPackage、org. omg. PortableServer、org. omg. PortableServer. CurrentPackage、org. omg. PortableServer. POAManagerPackage、org.