

软件开发视频大讲堂

# Java 从入门到精通

(第4版)

明日科技 编著

清华大学出版社

北 京

## 内 容 简 介

《Java 从入门到精通（第4版）》从初学者角度出发，通过通俗易懂的语言、丰富多彩的实例，详细介绍了使用Java语言进行程序开发需要掌握的知识。全书分为28章，包括初识Java，熟悉Eclipse开发工具，Java语言基础，流程控制，字符串，数组，类和对象，包装类，数字处理类，接口、继承与多态，类的高级特性，异常处理，Swing程序设计，集合类，I/O（输入/输出），反射，枚举类型与泛型，多线程，网络通信，数据库操作，Swing表格组件，Swing树组件，Swing其他高级组件，高级布局管理器，高级事件处理，AWT绘图与音频播放，打印技术和企业进销存管理系统等。书中所有知识都结合具体实例进行介绍，涉及的程序代码给出了详细的注释，可以使读者轻松领会Java程序开发的精髓，快速提高开发技能。另外，本书除了纸质内容之外，配书光盘中还给出了海量开发资源库，主要内容如下：

- |                                                           |                                                             |
|-----------------------------------------------------------|-------------------------------------------------------------|
| <input checked="" type="checkbox"/> 音视频讲解：总时长32小时，共319段   | <input checked="" type="checkbox"/> 实例资源库：732个实例及源码详细分析     |
| <input checked="" type="checkbox"/> 模块资源库：15个经典模块开发过程完整展现 | <input checked="" type="checkbox"/> 项目案例资源库：15个企业项目开发过程完整展现 |
| <input checked="" type="checkbox"/> 测试题库系统：616道能力测试题目     | <input checked="" type="checkbox"/> 面试资源库：369个企业面试真题        |
| <input checked="" type="checkbox"/> PPT电子教案               |                                                             |

本书适合作为软件开发入门者的自学用书，也适合作为高等院校相关专业的教学参考书，还可供开发人员查阅、参考。

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。侵权举报电话：010-62782989 13701121933

### 图书在版编目（CIP）数据

Java从入门到精通/明日科技编著. —4版. —北京：清华大学出版社，2016（2016.10重印）

（软件开发视频大讲堂）

ISBN 978-7-302-44454-1

I. ①J… II. ①明… III. ①JAVA语言—程序设计 IV. ①TP312

中国版本图书馆CIP数据核字（2016）第169630号

责任编辑：赵洛育

封面设计：刘洪利

版式设计：刘艳庆

责任校对：赵丽杰

责任印制：宋 林

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](mailto:c-service@tup.tsinghua.edu.cn)

质 量 反 馈：010-62772015, [zhiliang@tup.tsinghua.edu.cn](mailto:zhiliang@tup.tsinghua.edu.cn)

印 刷 者：清华大学印刷厂

装 订 者：北京市密云县京文制本装订厂

经 销：全国新华书店

开 本：203mm×260mm

印 张：38

字 数：1036千字

（附海量开发资源库DVD1张）

版 次：2008年9月第1版 2016年10月第4版

印 次：2016年10月第2次印刷

印 数：6001~16000

定 价：69.60元

---

产品编号：058847-02



# 如何使用 Java 开发资源库

在学习《Java 从入门到精通（第 4 版）》一书时，随书附配光盘提供了“Java 开发资源库”系统，可以帮助读者快速提升编程水平和解决实际问题的能力。《Java 从入门到精通（第 4 版）》和 Java 开发资源库配合学习流程如图 1 所示。

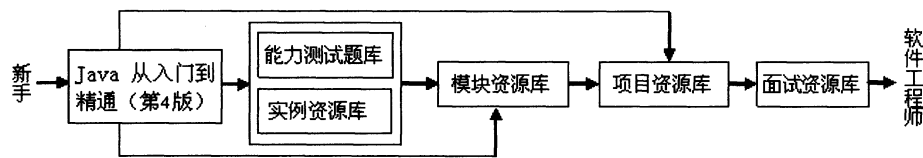


图 1 图书与开发资源库配合学习流程图

打开光盘中的“Java 开发资源库”文件夹，运行 Java 开发资源库.exe 程序，即可进入“Java 开发资源库”系统，主界面如图 2 所示。

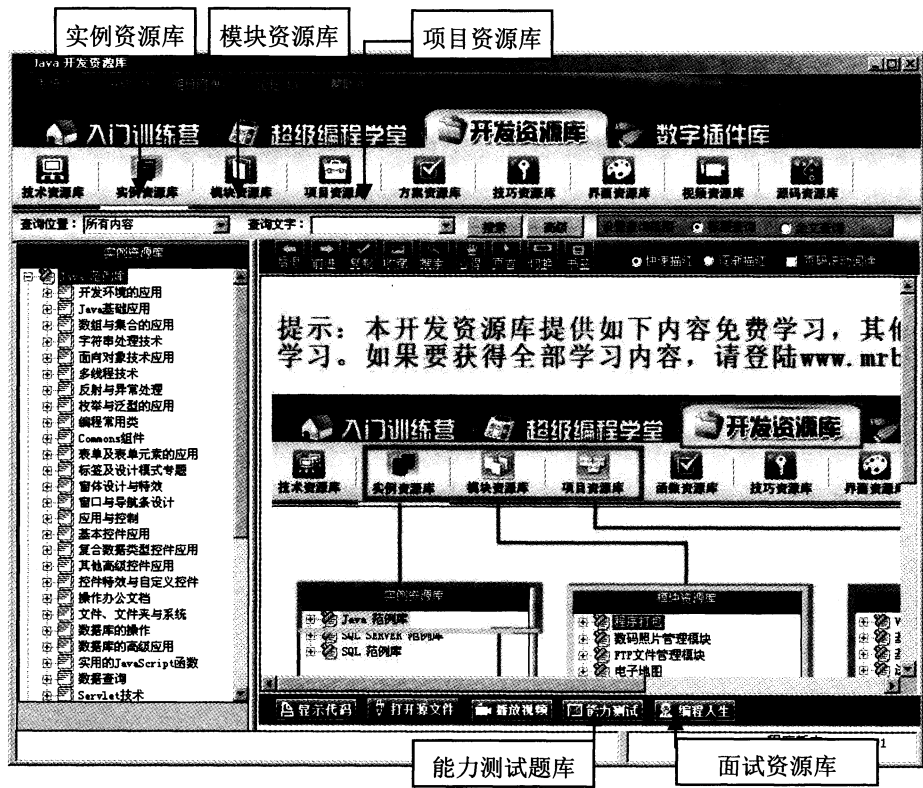


图 2 Java 开发资源库主界面

在学习某一章节时，可以配合实例资源库的相应章节，利用实例资源库提供的大量热点实例和关键实例巩固所学编程技能，提高编程兴趣和自信心；也可以配合能力测试题库的对应章节进行测试，检验学习成果。具体流程如图3所示。

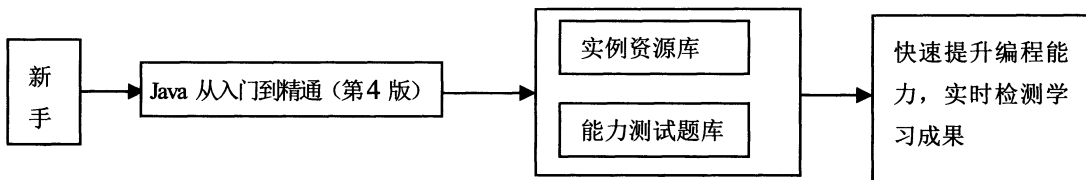


图3 使用实例资源库和能力测试题库

对于数学逻辑能力和英语基础较为薄弱的读者，或者想了解个人数学逻辑思维能力和编程英语基础的用户，本书提供了数学及逻辑思维能力和编程英语能力测试供练习和测试，如图4所示。

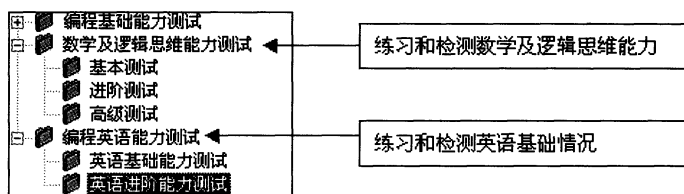


图4 数学及逻辑思维能力和编程英语能力测试目录

当本书学习完成时，可以配合模块资源库和项目资源库的30个模块和项目，全面提升个人综合编程技能和解决实际开发问题的能力，为成为Java软件开发工程师打下坚实基础。具体模块和项目目录如图5所示。

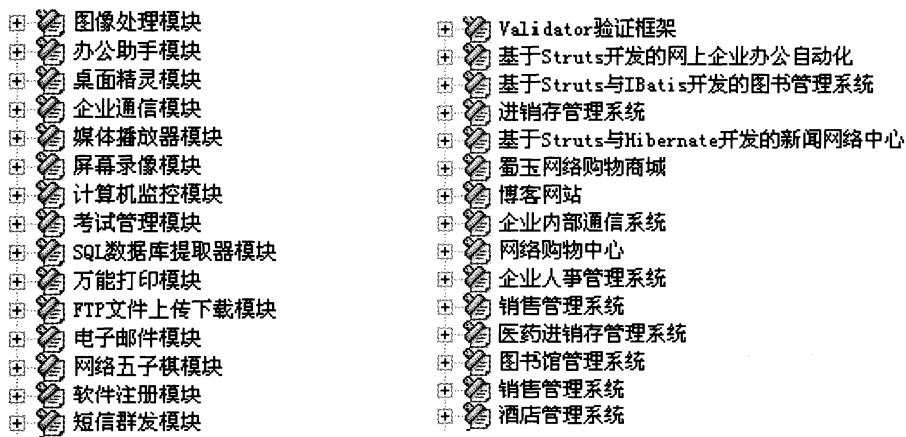


图5 模块资源库和项目资源库目录

万事俱备，该到软件开发的主战场上接受洗礼了。面试资源库提供了大量国内外软件企业的常见面试真题，同时还提供了程序员职业规划、程序员面试技巧、企业面试真题汇编和虚拟面试系统等精彩内容，是程序员求职面试的绝佳指南。面试资源库的具体内容如图6所示。

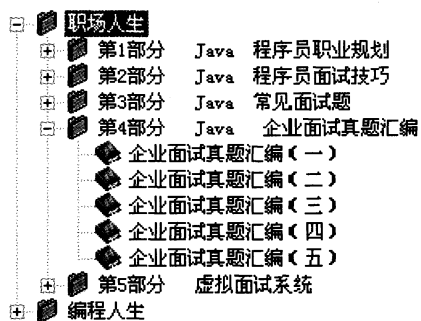


图 6 面试资源库具体内容

如果您在使用本书开发资源库时遇到问题，可加我们的 QQ: 4006751066（可容纳 10 万人），我们将竭诚为您服务。

# 前言

## Preface

**丛书说明：**“软件开发视频大讲堂”丛书（第1版）于2008年8月出版，因其编写细腻，易学实用，配备全程视频等，在软件开发类图书市场上产生了很大反响，绝大部分品种在全国软件开发零售图书排行榜中名列前茅，2009年多个品种被评为“全国优秀畅销书”。

“软件开发视频大讲堂”丛书（第2版）于2010年8月出版，出版后，绝大部分品种在全国软件开发类零售图书排行榜中依然名列前茅。丛书中多个品种被百余所高校计算机相关专业、软件学院选为教学参考书，在众多的软件开发类图书中成为最耀眼的品牌之一。丛书累计销售40多万册。

“软件开发视频大讲堂”丛书（第3版）于2012年8月出版，根据读者需要，增删了品种，重新录制了视频，提供了从“入门学习→实例应用→模块开发→项目开发→能力测试→面试”等各个阶段的海量开发资源库。因丛书编写结构合理、实例选择经典实用，丛书迄今累计销售90多万册。

“软件开发视频大讲堂”丛书（第4版）在继承前3版所有优点的基础上，修正了前3版图书中发现的疏漏之处，并结合目前市场需要，进一步对丛书品种进行了完善，对相关内容进行了更新优化，使之更适合读者学习，为了方便教学，还提供了教学课件PPT。

Java是Sun公司推出的能够跨越多平台的、可移植性最高的一种面向对象的编程语言。自面世以来，Java凭借其易学易用、功能强大的特点得到了广泛的应用。其强大的跨平台特性使Java程序可以运行在大部分系统平台上，甚至可在手持电话、商务助理等移动电子产品上运行，真正做到“一次编写，到处运行”。Java可用于编写桌面应用程序、Web应用程序、分布式系统和嵌入式系统应用程序等，这使得它成为应用范围最广泛的开发语言。随着Java技术的不断更新，在全球云计算和移动互联网的产业环境下，Java的显著优势和广阔前景将进一步呈现出来。

## 本书内容

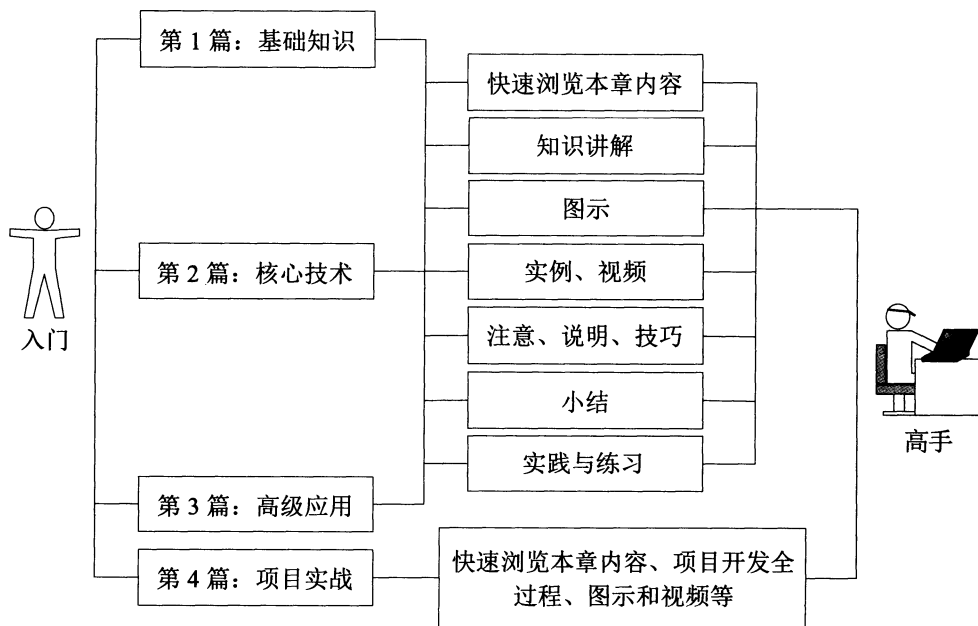
本书提供了从入门到编程高手所必备的各类知识，共分4篇，大体结构如下图所示。

**第1篇：基础知识。**本篇通过初识Java、熟悉Eclipse开发工具、Java语言基础、流程控制、字符串、数组、类和对象、包装类、数字处理类等内容的介绍，并结合大量的图示、实例、视频等，使读者快速掌握Java语言，为以后编程奠定坚实的基础。

**第2篇：核心技术。**本篇介绍了接口、继承与多态，类的高级特性，异常处理，Swing程序设计，集合类，I/O（输入/输出），反射，枚举类型与泛型，多线程，网络通信和数据库操作等内容。学习完本篇，能够开发一些小型应用程序。

**第3篇：高级应用。**本篇介绍了Swing表格组件、Swing树组件、Swing其他高级组件、高级布局管理器、高级事件处理、AWT绘图与音频播放、打印技术等内容。学习完本篇，能够开发高级的桌面应用程序、多媒体程序和打印程序等。

**第 4 篇：项目实战。**本篇通过一个大型、完整的企业进销存管理系统，运用软件工程的设计思想，让读者学习如何进行软件项目的实际开发。书中按照“编写项目计划书→系统设计→数据库设计→创建项目→实现项目→运行项目→项目打包部署→解决开发常见问题”的流程进行介绍，带领读者一步步亲身体验开发项目的全过程。



## 本书特点

- ☑ **由浅入深，循序渐进。**本书以初、中级程序员为对象，先从 Java 语言基础学起，再学习 Java 的核心技术，然后学习 Swing 的高级应用，最后学习开发一个完整项目。讲解过程中步骤详尽，版式新颖，在操作的内容图片上以 ❶❷❸……的编号+内容的方式进行标注，使读者在阅读时一目了然，从而快速掌握书中内容。
- ☑ **语音视频，讲解详尽。**书中每一章节均提供声图并茂的教学视频，读者可以根据书中提供的视频位置在光盘中找到。这些视频能够引导初学者快速入门，感受编程的快乐和成就感，增强进一步学习的信心，从而快速成为编程高手。
- ☑ **实例典型，轻松易学。**通过例子学习是最好的学习方式，本书通过“一个知识点、一个例子、一个结果、一段评析、一个综合应用”的模式，透彻详尽地讲述了实际开发中所需的各类知识。另外，为了便于读者阅读程序代码，快速学习编程技能，书中几乎每行代码都提供了注释。
- ☑ **精彩栏目，贴心提醒。**本书根据需要在各章安排了很多“注意”“说明”“技巧”等小栏目，让读者可以在学习过程中更轻松的理解相关知识点及概念，更快地掌握个别技术的应用技巧。

- ☑ 应用实践，随时练习。书中几乎每章都提供了“实践与练习”，使读者能够通过对问题的解答重新回顾、熟悉所学知识，举一反三，为进一步学习做好充分的准备。

## 读者对象

- |                   |                |
|-------------------|----------------|
| ☑ 初学编程的自学者        | ☑ 编程爱好者        |
| ☑ 大中专院校的老师 and 学生 | ☑ 相关培训机构的老师和学员 |
| ☑ 毕业设计的学生         | ☑ 初、中级程序开发人员   |
| ☑ 程序测试及维护人员       | ☑ 参加实习的“菜鸟”程序员 |

## 读者服务

为了方便解决本书疑难问题，读者朋友可加我们的 QQ: 4006751066（可容纳 10 万人），也可以登录 [www.mingribook.com](http://www.mingribook.com) 留言，我们将竭诚为您服务。

## 致谢

本书在出版过程中，得到了清华大学出版社策划编辑及本书项目负责人刘利民先生的大力支持，在此表示衷心感谢。另外，本书所有的编审、发行人员为本书的出版和发行付出了辛勤劳动，在此一并致谢。

## 致读者

本书由明日科技有限责任公司 Java 程序开发小组编写。明日科技是一家专业从事软件开发、教育培训以及软件开发教育资源整合的高科技公司，其编写的教材既注重选取软件开发中的必需、常用内容，又注重内容的易学、方便以及相关知识的拓展，深受读者喜爱。其编写的教材多次荣获“全行业优秀畅销品种”“中国大学出版社优秀畅销书”等奖项，多个品种长期位居同类图书销售排行榜的前列。

本书主要参与编写的程序员有申小琦、王小科、王国辉、董刚、赛奎春、房德山、杨丽、高春艳、辛洪郁、周佳星、张鑫、张宝华、葛忠月、刘杰、白宏健、张雳霆、马新新、冯春龙、宋万勇、李文欣、王东东、柳琳、王盛鑫、徐明明、杨柳、赵宁、王佳雪、于国良、李磊、李彦峻、王泽奇、贾景波、谭慧、李丹、吕玉翠、孙巧辰、赵颖、江玉贞、周艳梅、房雪坤、裴莹、郭铁、张金辉、王敬杰、高茹、李贺、陈威、高飞、刘志铭、高润岭、于国槐、郭锐、郭鑫、邹淑芳、李根福、杨贵发、王喜平等。在编写过程中，我们以科学、严谨的态度，力求精益求精，但错误、疏漏之处在所难免，敬请广大读者批评指正。

感谢您购买本书，希望本书能成为您编程路上的领航者。

“零门槛”编程，一切皆有可能。

祝读书快乐！

编者

# 目 录

## Contents

## 第 1 篇 基础知识

|                                                                                                |    |                                                                                                     |    |
|------------------------------------------------------------------------------------------------|----|-----------------------------------------------------------------------------------------------------|----|
| 第 1 章 初识 Java .....                                                                            | 2  | 2.1.1 Eclipse 简介 .....                                                                              | 18 |
|  视频讲解：25 分钟   |    | 2.1.2 下载 Eclipse .....                                                                              | 18 |
| 1.1 Java 简介 .....                                                                              | 3  | 2.1.3 安装 Eclipse 的中文语言包 .....                                                                       | 20 |
| 1.1.1 什么是 Java 语言 .....                                                                        | 3  | 2.1.4 Eclipse 的配置与启动 .....                                                                          | 23 |
| 1.1.2 Java 的应用领域 .....                                                                         | 3  | 2.1.5 Eclipse 工作台 .....                                                                             | 25 |
| 1.1.3 Java 的版本 .....                                                                           | 4  | 2.1.6 透视图与视图 .....                                                                                  | 25 |
| 1.1.4 怎样学好 Java .....                                                                          | 5  | 2.1.7 菜单栏 .....                                                                                     | 26 |
| 1.1.5 Java API 文档 .....                                                                        | 6  | 2.1.8 工具栏 .....                                                                                     | 29 |
| 1.2 Java 语言的特性 .....                                                                           | 7  | 2.1.9 “包资源管理器”视图 .....                                                                              | 30 |
| 1.2.1 简单 .....                                                                                 | 7  | 2.1.10 “控制台”视图 .....                                                                                | 30 |
| 1.2.2 面向对象 .....                                                                               | 7  | 2.2 使用 Eclipse .....                                                                                | 31 |
| 1.2.3 分布性 .....                                                                                | 7  | 2.2.1 创建 Java 项目 .....                                                                              | 31 |
| 1.2.4 可移植性 .....                                                                               | 7  | 2.2.2 创建 Java 类文件 .....                                                                             | 31 |
| 1.2.5 解释型 .....                                                                                | 8  | 2.2.3 使用编辑器编写程序代码 .....                                                                             | 32 |
| 1.2.6 安全性 .....                                                                                | 8  | 2.2.4 运行 Java 程序 .....                                                                              | 35 |
| 1.2.7 健壮性 .....                                                                                | 8  | 2.3 程序调试 .....                                                                                      | 35 |
| 1.2.8 多线程 .....                                                                                | 8  | 2.4 小结 .....                                                                                        | 37 |
| 1.2.9 高性能 .....                                                                                | 8  | 2.5 实践与练习 .....                                                                                     | 37 |
| 1.2.10 动态 .....                                                                                | 8  |                                                                                                     |    |
| 1.3 搭建 Java 环境 .....                                                                           | 9  | 第 3 章 Java 语言基础 .....                                                                               | 38 |
| 1.3.1 JDK 下载 .....                                                                             | 9  |  视频讲解：1 小时 45 分钟 |    |
| 1.3.2 Windows 系统的 JDK 环境 .....                                                                 | 11 | 3.1 Java 主类结构 .....                                                                                 | 39 |
| 1.4 第一个 Java 程序 .....                                                                          | 14 | 3.1.1 包声明 .....                                                                                     | 39 |
| 1.5 小结 .....                                                                                   | 15 | 3.1.2 声明成员变量和局部变量 .....                                                                             | 40 |
| 1.6 实践与练习 .....                                                                                | 16 | 3.1.3 编写主方法 .....                                                                                   | 40 |
|                                                                                                |    | 3.1.4 导入 API 类库 .....                                                                               | 40 |
| 第 2 章 熟悉 Eclipse 开发工具 .....                                                                    | 17 | 3.2 基本数据类型 .....                                                                                    | 40 |
|  视频讲解：13 分钟 |    | 3.2.1 整数类型 .....                                                                                    | 41 |
| 2.1 熟悉 Eclipse .....                                                                           | 18 | 3.2.2 浮点类型 .....                                                                                    | 42 |

|                                                                                                             |           |                                                                                                             |            |
|-------------------------------------------------------------------------------------------------------------|-----------|-------------------------------------------------------------------------------------------------------------|------------|
| 3.2.3 字符类型.....                                                                                             | 43        | 4.5 小结.....                                                                                                 | 79         |
| 3.2.4 布尔类型.....                                                                                             | 44        | 4.6 实践与练习.....                                                                                              | 79         |
| 3.3 变量与常量.....                                                                                              | 44        | <b>第5章 字符串.....</b>                                                                                         | <b>80</b>  |
| 3.3.1 标识符和关键字.....                                                                                          | 45        |  <b>视频讲解: 1 小时 53 分钟</b>   |            |
| 3.3.2 声明变量.....                                                                                             | 45        | 5.1 String 类.....                                                                                           | 81         |
| 3.3.3 声明常量.....                                                                                             | 46        | 5.1.1 声明字符串.....                                                                                            | 81         |
| 3.3.4 变量的有效范围.....                                                                                          | 47        | 5.1.2 创建字符串.....                                                                                            | 81         |
| 3.4 运算符.....                                                                                                | 48        | 5.2 连接字符串.....                                                                                              | 82         |
| 3.4.1 赋值运算符.....                                                                                            | 49        | 5.2.1 连接多个字符串.....                                                                                          | 83         |
| 3.4.2 算术运算符.....                                                                                            | 50        | 5.2.2 连接其他数据类型.....                                                                                         | 83         |
| 3.4.3 自增和自减运算符.....                                                                                         | 51        | 5.3 获取字符串信息.....                                                                                            | 84         |
| 3.4.4 比较运算符.....                                                                                            | 51        | 5.3.1 获取字符串长度.....                                                                                          | 85         |
| 3.4.5 逻辑运算符.....                                                                                            | 52        | 5.3.2 字符串查找.....                                                                                            | 85         |
| 3.4.6 位运算符.....                                                                                             | 53        | 5.3.3 获取指定索引位置的字符.....                                                                                      | 86         |
| 3.4.7 三元运算符.....                                                                                            | 55        | 5.4 字符串操作.....                                                                                              | 87         |
| 3.4.8 运算符优先级.....                                                                                           | 56        | 5.4.1 获取子字符串.....                                                                                           | 87         |
| 3.5 数据类型转换.....                                                                                             | 56        | 5.4.2 去除空格.....                                                                                             | 88         |
| 3.5.1 隐式类型转换.....                                                                                           | 57        | 5.4.3 字符串替换.....                                                                                            | 89         |
| 3.5.2 显式类型转换.....                                                                                           | 58        | 5.4.4 判断字符串的开始与结尾.....                                                                                      | 90         |
| 3.6 代码注释与编码规范.....                                                                                          | 59        | 5.4.5 判断字符串是否相等.....                                                                                        | 91         |
| 3.6.1 代码注释.....                                                                                             | 59        | 5.4.6 按字典顺序比较两个字符串.....                                                                                     | 93         |
| 3.6.2 编码规范.....                                                                                             | 60        | 5.4.7 字母大小写转换.....                                                                                          | 94         |
| 3.7 小结.....                                                                                                 | 61        | 5.4.8 字符串分割.....                                                                                            | 95         |
| 3.8 实践与练习.....                                                                                              | 61        | 5.5 格式化字符串.....                                                                                             | 96         |
| <b>第4章 流程控制.....</b>                                                                                        | <b>62</b> | 5.5.1 日期和时间字符串格式化.....                                                                                      | 97         |
|  <b>视频讲解: 1 小时 21 分钟</b> |           | 5.5.2 常规类型格式化.....                                                                                          | 100        |
| 4.1 复合语句.....                                                                                               | 63        | 5.6 使用正则表达式.....                                                                                            | 101        |
| 4.2 条件语句.....                                                                                               | 64        | 5.7 字符串生成器.....                                                                                             | 103        |
| 4.2.1 if 条件语句.....                                                                                          | 64        | 5.8 小结.....                                                                                                 | 105        |
| 4.2.2 switch 多分支语句.....                                                                                     | 68        | 5.9 实践与练习.....                                                                                              | 105        |
| 4.3 循环语句.....                                                                                               | 71        | <b>第6章 数组.....</b>                                                                                          | <b>106</b> |
| 4.3.1 while 循环语句.....                                                                                       | 71        |  <b>视频讲解: 1 小时 21 分钟</b> |            |
| 4.3.2 do...while 循环语句.....                                                                                  | 72        | 6.1 数组概述.....                                                                                               | 107        |
| 4.3.3 for 循环语句.....                                                                                         | 73        | 6.2 一维数组的创建及使用.....                                                                                         | 107        |
| 4.4 循环控制.....                                                                                               | 75        | 6.2.1 创建一维数组.....                                                                                           | 107        |
| 4.4.1 break 语句.....                                                                                         | 75        | 6.2.2 初始化一维数组.....                                                                                          | 108        |
| 4.4.2 continue 语句.....                                                                                      | 78        |                                                                                                             |            |





|                                                                                                    |     |                                                                                                 |     |
|----------------------------------------------------------------------------------------------------|-----|-------------------------------------------------------------------------------------------------|-----|
| 6.2.3 使用一维数组.....                                                                                  | 109 | 7.5 类的主方法.....                                                                                  | 140 |
| 6.3 二维数组的创建及使用.....                                                                                | 109 | 7.6 对象.....                                                                                     | 141 |
| 6.3.1 二维数组的创建.....                                                                                 | 110 | 7.6.1 对象的创建.....                                                                                | 142 |
| 6.3.2 二维数组初始化.....                                                                                 | 111 | 7.6.2 访问对象的属性和行为.....                                                                           | 143 |
| 6.3.3 使用二维数组.....                                                                                  | 111 | 7.6.3 对象的引用.....                                                                                | 145 |
| 6.4 数组的基本操作.....                                                                                   | 112 | 7.6.4 对象的比较.....                                                                                | 146 |
| 6.4.1 遍历数组.....                                                                                    | 112 | 7.6.5 对象的销毁.....                                                                                | 147 |
| 6.4.2 填充替换数组元素.....                                                                                | 113 | 7.7 小结.....                                                                                     | 148 |
| 6.4.3 对数组进行排序.....                                                                                 | 115 | 7.8 实践与练习.....                                                                                  | 148 |
| 6.4.4 复制数组.....                                                                                    | 116 |                                                                                                 |     |
| 6.4.5 数组查询.....                                                                                    | 117 | 第8章 包装类.....                                                                                    | 149 |
| 6.5 数组排序算法.....                                                                                    | 119 |  视频讲解: 11 分钟   |     |
| 6.5.1 冒泡排序.....                                                                                    | 119 | 8.1 Integer.....                                                                                | 150 |
| 6.5.2 直接选择排序.....                                                                                  | 122 | 8.2 Boolean.....                                                                                | 152 |
| 6.5.3 反转排序.....                                                                                    | 124 | 8.3 Byte.....                                                                                   | 154 |
| 6.6 小结.....                                                                                        | 125 | 8.4 Character.....                                                                              | 155 |
| 6.7 实践与练习.....                                                                                     | 126 | 8.5 Double.....                                                                                 | 156 |
| 第7章 类和对象.....                                                                                      | 127 | 8.6 Number.....                                                                                 | 158 |
|  视频讲解: 1 小时 27 分钟 |     | 8.7 小结.....                                                                                     | 158 |
| 7.1 面向对象概述.....                                                                                    | 128 | 8.8 实践与练习.....                                                                                  | 158 |
| 7.1.1 对象.....                                                                                      | 128 | 第9章 数字处理类.....                                                                                  | 159 |
| 7.1.2 类.....                                                                                       | 129 |  视频讲解: 16 分钟 |     |
| 7.1.3 封装.....                                                                                      | 130 | 9.1 数字格式化.....                                                                                  | 160 |
| 7.1.4 继承.....                                                                                      | 130 | 9.2 数学运算.....                                                                                   | 162 |
| 7.1.5 多态.....                                                                                      | 131 | 9.2.1 Math 类.....                                                                               | 163 |
| 7.2 类.....                                                                                         | 132 | 9.2.2 常用数学运算方法.....                                                                             | 163 |
| 7.2.1 成员变量.....                                                                                    | 132 | 9.3 随机数.....                                                                                    | 167 |
| 7.2.2 成员方法.....                                                                                    | 133 | 9.3.1 Math.random()方法.....                                                                      | 167 |
| 7.2.3 权限修饰符.....                                                                                   | 133 | 9.3.2 Random 类.....                                                                             | 169 |
| 7.2.4 局部变量.....                                                                                    | 135 | 9.4 大数字运算.....                                                                                  | 170 |
| 7.2.5 局部变量的有效范围.....                                                                               | 135 | 9.4.1 BigInteger.....                                                                           | 171 |
| 7.2.6 this 关键字.....                                                                                | 136 | 9.4.2 BigDecimal.....                                                                           | 173 |
| 7.3 类的构造方法.....                                                                                    | 137 | 9.5 小结.....                                                                                     | 176 |
| 7.4 静态变量、常量和方法.....                                                                                | 138 | 9.6 实践与练习.....                                                                                  | 176 |

## 第2篇 核心技术

|                          |     |                                |     |
|--------------------------|-----|--------------------------------|-----|
| 第10章 接口、继承与多态 .....      | 178 | 第12章 异常处理 .....                | 216 |
| 📺 视频讲解: 23 分钟            |     | 📺 视频讲解: 17 分钟                  |     |
| 10.1 类的继承 .....          | 179 | 12.1 异常概述 .....                | 217 |
| 10.2 Object 类 .....      | 182 | 12.2 处理程序异常错误 .....            | 217 |
| 10.3 对象类型的转换 .....       | 184 | 12.2.1 错误 .....                | 218 |
| 10.3.1 向上转型 .....        | 184 | 12.2.2 捕捉异常 .....              | 218 |
| 10.3.2 向下转型 .....        | 185 | 12.3 Java 常见异常 .....           | 220 |
| 10.4 使用 instanceof 操作符判断 |     | 12.4 自定义异常 .....               | 221 |
| 对象类型 .....               | 186 | 12.5 在方法中抛出异常 .....            | 222 |
| 10.5 方法的重载 .....         | 188 | 12.5.1 使用 throws 关键字抛出异常 ..... | 222 |
| 10.6 多态 .....            | 191 | 12.5.2 使用 throw 关键字抛出异常 .....  | 223 |
| 10.7 抽象类与接口 .....        | 193 | 12.6 运行时异常 .....               | 225 |
| 10.7.1 抽象类 .....         | 193 | 12.7 异常的使用原则 .....             | 226 |
| 10.7.2 接口 .....          | 194 | 12.8 小结 .....                  | 226 |
| 10.8 小结 .....            | 197 | 12.9 实践与练习 .....               | 226 |
| 10.9 实践与练习 .....         | 197 |                                |     |
| 第11章 类的高级特性 .....        | 198 | 第13章 Swing 程序设计 .....          | 227 |
| 📺 视频讲解: 23 分钟            |     | 📺 视频讲解: 1 小时 3 分钟              |     |
| 11.1 Java 类包 .....       | 199 | 13.1 Swing 概述 .....            | 228 |
| 11.1.1 类名冲突 .....        | 199 | 13.1.1 Swing 特点 .....          | 228 |
| 11.1.2 完整的类路径 .....      | 199 | 13.1.2 Swing 包 .....           | 228 |
| 11.1.3 创建包 .....         | 200 | 13.1.3 常用 Swing 组件概述 .....     | 229 |
| 11.1.4 导入包 .....         | 201 | 13.2 常用窗体 .....                | 229 |
| 11.2 final 变量 .....      | 203 | 13.2.1 JFrame 窗体 .....         | 229 |
| 11.3 final 方法 .....      | 206 | 13.2.2 JDialog 窗体 .....        | 231 |
| 11.4 final 类 .....       | 207 | 13.3 标签组件与图标 .....             | 233 |
| 11.5 内部类 .....           | 208 | 13.3.1 标签的使用 .....             | 233 |
| 11.5.1 成员内部类 .....       | 208 | 13.3.2 图标的使用 .....             | 234 |
| 11.5.2 局部内部类 .....       | 212 | 13.4 常用布局管理器 .....             | 236 |
| 11.5.3 匿名内部类 .....       | 213 | 13.4.1 绝对布局 .....              | 237 |
| 11.5.4 静态内部类 .....       | 214 | 13.4.2 流布局管理器 .....            | 238 |
| 11.5.5 内部类的继承 .....      | 215 | 13.4.3 边界布局管理器 .....           | 239 |
| 11.6 小结 .....            | 215 | 13.4.4 网格布局管理器 .....           | 241 |
| 11.7 实践与练习 .....         | 215 | 13.5 常用面板 .....                | 242 |
|                          |     | 13.5.1 JPanel 面板 .....         | 242 |



|                                                                                                 |     |                                                                                                 |     |
|-------------------------------------------------------------------------------------------------|-----|-------------------------------------------------------------------------------------------------|-----|
| 13.5.2 JScrollPane 面板 .....                                                                     | 243 | 15.2.2 输出流 .....                                                                                | 274 |
| 13.6 按钮组件 .....                                                                                 | 244 | 15.3 File 类 .....                                                                               | 275 |
| 13.6.1 提交按钮组件 .....                                                                             | 244 | 15.3.1 文件的创建与删除 .....                                                                           | 275 |
| 13.6.2 单选按钮组件 .....                                                                             | 246 | 15.3.2 获取文件信息 .....                                                                             | 276 |
| 13.6.3 复选框组件 .....                                                                              | 247 | 15.4 文件输入/输出流 .....                                                                             | 278 |
| 13.7 列表组件 .....                                                                                 | 248 | 15.4.1 FileInputStream 与 FileOutputStream 类 .....                                               | 278 |
| 13.7.1 下拉列表框组件 .....                                                                            | 248 | 15.4.2 FileReader 和 FileWriter 类 .....                                                          | 279 |
| 13.7.2 列表框组件 .....                                                                              | 250 | 15.5 带缓存的输入/输出流 .....                                                                           | 282 |
| 13.8 文本组件 .....                                                                                 | 252 | 15.5.1 BufferedInputStream 与<br>BufferedOutputStream 类 .....                                    | 282 |
| 13.8.1 文本框组件 .....                                                                              | 252 | 15.5.2 BufferedReader 与 BufferedWriter 类 .....                                                  | 282 |
| 13.8.2 密码框组件 .....                                                                              | 253 | 15.6 数据输入/输出流 .....                                                                             | 284 |
| 13.8.3 文本域组件 .....                                                                              | 253 | 15.7 ZIP 压缩输入/输出流 .....                                                                         | 285 |
| 13.9 常用事件监听器 .....                                                                              | 254 | 15.7.1 压缩文件 .....                                                                               | 286 |
| 13.9.1 监听事件简介 .....                                                                             | 255 | 15.7.2 解压缩 ZIP 文件 .....                                                                         | 287 |
| 13.9.2 动作事件监听器 .....                                                                            | 255 | 15.8 小结 .....                                                                                   | 289 |
| 13.9.3 焦点事件监听器 .....                                                                            | 257 | 15.9 实践与练习 .....                                                                                | 289 |
| 13.10 小结 .....                                                                                  | 258 |                                                                                                 |     |
| 13.11 实践与练习 .....                                                                               | 258 |                                                                                                 |     |
| 第 14 章 集合类 .....                                                                                | 259 | 第 16 章 反射 .....                                                                                 | 290 |
|  视频讲解: 13 分钟 |     |  视频讲解: 22 分钟  |     |
| 14.1 集合类概述 .....                                                                                | 260 | 16.1 Class 类与 Java 反射 .....                                                                     | 291 |
| 14.2 Collection 接口 .....                                                                        | 260 | 16.1.1 访问构造方法 .....                                                                             | 292 |
| 14.3 List 集合 .....                                                                              | 261 | 16.1.2 访问成员变量 .....                                                                             | 295 |
| 14.3.1 List 接口 .....                                                                            | 261 | 16.1.3 访问方法 .....                                                                               | 298 |
| 14.3.2 List 接口的实现类 .....                                                                        | 262 | 16.2 使用 Annotation 功能 .....                                                                     | 301 |
| 14.4 Set 集合 .....                                                                               | 263 | 16.2.1 定义 Annotation 类型 .....                                                                   | 301 |
| 14.5 Map 集合 .....                                                                               | 266 | 16.2.2 访问 Annotation 信息 .....                                                                   | 304 |
| 14.5.1 Map 接口 .....                                                                             | 266 | 16.3 小结 .....                                                                                   | 307 |
| 14.5.2 Map 接口的实现类 .....                                                                         | 267 | 16.4 实践与练习 .....                                                                                | 307 |
| 14.6 小结 .....                                                                                   | 269 |                                                                                                 |     |
| 14.7 实践与练习 .....                                                                                | 270 | 第 17 章 枚举类型与泛型 .....                                                                            | 308 |
| 第 15 章 I/O (输入/输出) .....                                                                        | 271 |  视频讲解: 20 分钟 |     |
|  视频讲解: 22 分钟 |     | 17.1 枚举类型 .....                                                                                 | 309 |
| 15.1 流概述 .....                                                                                  | 272 | 17.1.1 使用枚举类型设置常量 .....                                                                         | 309 |
| 15.2 输入/输出流 .....                                                                               | 272 | 17.1.2 深入了解枚举类型 .....                                                                           | 311 |
| 15.2.1 输入流 .....                                                                                | 272 | 17.1.3 使用枚举类型的优势 .....                                                                          | 316 |
|                                                                                                 |     | 17.2 泛型 .....                                                                                   | 316 |
|                                                                                                 |     | 17.2.1 回顾向上转型与向下转型 .....                                                                        | 317 |



|                            |     |                                  |     |
|----------------------------|-----|----------------------------------|-----|
| 17.2.2 定义泛型类.....          | 318 | 19.2.3 TCP 网络程序.....             | 351 |
| 17.2.3 泛型的常规用法.....        | 319 | 19.3 UDP 程序设计基础.....             | 355 |
| 17.2.4 泛型的高级用法.....        | 322 | 19.3.1 DatagramPacket 类.....     | 355 |
| 17.2.5 泛型总结.....           | 325 | 19.3.2 DatagramSocket 类.....     | 356 |
| 17.3 小结.....               | 325 | 19.3.3 UDP 网络程序.....             | 356 |
| 17.4 实践与练习.....            | 325 | 19.4 小结.....                     | 359 |
| 第 18 章 多线程.....            | 326 | 19.5 实践与练习.....                  | 360 |
| 📺 视频讲解: 21 分钟              |     | 第 20 章 数据库操作.....                | 361 |
| 18.1 线程简介.....             | 327 | 📺 视频讲解: 48 分钟                    |     |
| 18.2 实现线程的两种方式.....        | 327 | 20.1 数据库基础知识.....                | 362 |
| 18.2.1 继承 Thread 类.....    | 328 | 20.1.1 什么是数据库.....               | 362 |
| 18.2.2 实现 Runnable 接口..... | 329 | 20.1.2 数据库的种类及功能.....            | 363 |
| 18.3 线程的生命周期.....          | 331 | 20.1.3 SQL 语言.....               | 363 |
| 18.4 操作线程的方法.....          | 333 | 20.2 JDBC 概述.....                | 365 |
| 18.4.1 线程的休眠.....          | 333 | 20.2.1 JDBC-ODBC 桥.....          | 365 |
| 18.4.2 线程的加入.....          | 335 | 20.2.2 JDBC 技术.....              | 366 |
| 18.4.3 线程的中断.....          | 336 | 20.2.3 JDBC 驱动程序的类型.....         | 366 |
| 18.4.4 线程的礼让.....          | 338 | 20.3 JDBC 中常用的类和接口.....          | 366 |
| 18.5 线程的优先级.....           | 338 | 20.3.1 Connection 接口.....        | 367 |
| 18.6 线程同步.....             | 340 | 20.3.2 Statement 接口.....         | 367 |
| 18.6.1 线程安全.....           | 341 | 20.3.3 PreparedStatement 接口..... | 368 |
| 18.6.2 线程同步机制.....         | 342 | 20.3.4 DriverManager 类.....      | 368 |
| 18.7 小结.....               | 344 | 20.3.5 ResultSet 接口.....         | 368 |
| 18.8 实践与练习.....            | 345 | 20.4 数据库操作.....                  | 369 |
| 第 19 章 网络通信.....           | 346 | 20.4.1 连接数据库.....                | 370 |
| 📺 视频讲解: 24 分钟              |     | 20.4.2 向数据库发送 SQL 语句.....        | 371 |
| 19.1 网络程序设计基础.....         | 347 | 20.4.3 处理查询结果集.....              | 371 |
| 19.1.1 局域网与因特网.....        | 347 | 20.4.4 顺序查询.....                 | 372 |
| 19.1.2 网络协议.....           | 347 | 20.4.5 模糊查询.....                 | 373 |
| 19.1.3 端口和套接字.....         | 348 | 20.4.6 预处理语句.....                | 374 |
| 19.2 TCP 程序设计基础.....       | 349 | 20.4.7 添加、修改、删除记录.....           | 376 |
| 19.2.1 InetAddress 类.....  | 349 | 20.5 小结.....                     | 378 |
| 19.2.2 ServerSocket 类..... | 350 | 20.6 实践与练习.....                  | 378 |



## 第 3 篇 高级应用

|                                                                                                 |     |                                                                                                 |     |
|-------------------------------------------------------------------------------------------------|-----|-------------------------------------------------------------------------------------------------|-----|
| 第 21 章 Swing 表格组件 .....                                                                         | 380 | 23.3 工具栏 .....                                                                                  | 430 |
|  视频讲解: 20 分钟   |     | 23.4 文件选择器 .....                                                                                | 432 |
| 21.1 利用 JTable 类直接创建表格 .....                                                                    | 381 | 23.4.1 文件选择对话框 .....                                                                            | 432 |
| 21.1.1 创建表格 .....                                                                               | 381 | 23.4.2 使用文件过滤器 .....                                                                            | 433 |
| 21.1.2 定制表格 .....                                                                               | 383 | 23.5 进度条 .....                                                                                  | 435 |
| 21.1.3 操纵表格 .....                                                                               | 387 | 23.6 系统托盘 .....                                                                                 | 436 |
| 21.2 表格模型与表格 .....                                                                              | 389 | 23.7 桌面集成控件 .....                                                                               | 437 |
| 21.2.1 利用表格模型创建表格 .....                                                                         | 389 | 23.8 小结 .....                                                                                   | 438 |
| 21.2.2 维护表格模型 .....                                                                             | 391 | 23.9 实践与练习 .....                                                                                | 438 |
| 21.3 提供行标题栏的表格 .....                                                                            | 393 | 第 24 章 高级布局管理器 .....                                                                            | 439 |
| 21.4 小结 .....                                                                                   | 398 |  视频讲解: 32 分钟   |     |
| 21.5 实践与练习 .....                                                                                | 398 | 24.1 箱式布局管理器 .....                                                                              | 440 |
| 第 22 章 Swing 树组件 .....                                                                          | 399 | 24.2 卡片布局管理器 .....                                                                              | 442 |
|  视频讲解: 20 分钟   |     | 24.3 网格组布局管理器 .....                                                                             | 445 |
| 22.1 简单的树 .....                                                                                 | 400 | 24.4 弹簧布局管理器 .....                                                                              | 451 |
| 22.2 处理选中节点事件 .....                                                                             | 401 | 24.4.1 使用弹簧布局管理器 .....                                                                          | 451 |
| 22.3 遍历树节点 .....                                                                                | 404 | 24.4.2 使用弹簧和支柱 .....                                                                            | 454 |
| 22.4 定制树 .....                                                                                  | 406 | 24.4.3 利用弹簧控制组件大小 .....                                                                         | 455 |
| 22.5 维护树模型 .....                                                                                | 408 | 24.5 小结 .....                                                                                   | 456 |
| 22.6 处理展开节点事件 .....                                                                             | 410 | 24.6 实践与练习 .....                                                                                | 456 |
| 22.7 小结 .....                                                                                   | 412 | 第 25 章 高级事件处理 .....                                                                             | 457 |
| 22.8 实践与练习 .....                                                                                | 412 |  视频讲解: 23 分钟 |     |
| 第 23 章 Swing 其他高级组件 .....                                                                       | 413 | 25.1 键盘事件 .....                                                                                 | 458 |
|  视频讲解: 41 分钟 |     | 25.2 鼠标事件 .....                                                                                 | 460 |
| 23.1 高级组件面板 .....                                                                               | 414 | 25.3 窗体事件 .....                                                                                 | 462 |
| 23.1.1 分割面板 .....                                                                               | 414 | 25.3.1 捕获窗体焦点变化事件 .....                                                                         | 462 |
| 23.1.2 选项卡面板 .....                                                                              | 417 | 25.3.2 捕获窗体状态变化事件 .....                                                                         | 463 |
| 23.1.3 桌面面板和内部窗体 .....                                                                          | 419 | 25.3.3 捕获其他窗体事件 .....                                                                           | 465 |
| 23.2 菜单 .....                                                                                   | 424 | 25.4 选项事件 .....                                                                                 | 467 |
| 23.2.1 创建菜单栏 .....                                                                              | 424 | 25.5 表格模型事件 .....                                                                               | 468 |
| 23.2.2 创建弹出式菜单 .....                                                                            | 426 | 25.6 小结 .....                                                                                   | 471 |
| 23.2.3 定制个性化菜单 .....                                                                            | 426 | 25.7 实践与练习 .....                                                                                | 471 |

|                                                                                               |     |                                                                                               |     |
|-----------------------------------------------------------------------------------------------|-----|-----------------------------------------------------------------------------------------------|-----|
| 第 26 章 AWT 绘图与音频播放 .....                                                                      | 472 | 26.6.2 图像翻转 .....                                                                             | 485 |
|  视频讲解: 36 分钟 |     | 26.6.3 图像旋转 .....                                                                             | 487 |
| 26.1 Java 绘图 .....                                                                            | 473 | 26.6.4 图像倾斜 .....                                                                             | 488 |
| 26.1.1 Graphics .....                                                                         | 473 | 26.7 播放音频文件 .....                                                                             | 490 |
| 26.1.2 Graphics2D .....                                                                       | 473 | 26.8 小结 .....                                                                                 | 492 |
| 26.2 绘制图形 .....                                                                               | 473 | 26.9 实践与练习 .....                                                                              | 492 |
| 26.3 绘图颜色与笔画属性 .....                                                                          | 477 | 第 27 章 打印技术 .....                                                                             | 493 |
| 26.3.1 设置颜色 .....                                                                             | 477 |  视频讲解: 13 分钟 |     |
| 26.3.2 笔画属性 .....                                                                             | 478 | 27.1 打印控制类 .....                                                                              | 494 |
| 26.4 绘制文本 .....                                                                               | 479 | 27.2 “打印”对话框 .....                                                                            | 495 |
| 26.4.1 设置字体 .....                                                                             | 480 | 27.3 打印页面 .....                                                                               | 497 |
| 26.4.2 显示文字 .....                                                                             | 480 | 27.4 多页打印 .....                                                                               | 499 |
| 26.5 绘制图片 .....                                                                               | 481 | 27.5 打印预览 .....                                                                               | 501 |
| 26.6 图像处理 .....                                                                               | 483 | 27.6 小结 .....                                                                                 | 503 |
| 26.6.1 放大与缩小 .....                                                                            | 483 | 27.7 实践与练习 .....                                                                              | 503 |

## 第 4 篇 项目实战

|                                                                                                      |     |                           |     |
|------------------------------------------------------------------------------------------------------|-----|---------------------------|-----|
| 第 28 章 企业进销存管理系统 .....                                                                               | 506 | 28.7.1 Item 公共类 .....     | 520 |
|  视频讲解: 1 小时 18 分钟 |     | 28.7.2 数据模型公共类 .....      | 521 |
| 28.1 系统分析 .....                                                                                      | 507 | 28.7.3 Dao 公共类 .....      | 523 |
| 28.1.1 需求分析 .....                                                                                    | 507 | 28.8 系统登录模块设计 .....       | 533 |
| 28.1.2 可行性分析 .....                                                                                   | 507 | 28.8.1 设计登录窗体 .....       | 534 |
| 28.1.3 编写项目计划书 .....                                                                                 | 508 | 28.8.2 “密码”文本框的回车事件 ..... | 535 |
| 28.2 系统设计 .....                                                                                      | 510 | 28.8.3 “登录”按钮的事件处理 .....  | 535 |
| 28.2.1 系统目标 .....                                                                                    | 510 | 28.9 系统主窗体设计 .....        | 536 |
| 28.2.2 系统功能结构 .....                                                                                  | 510 | 28.9.1 设计菜单栏 .....        | 537 |
| 28.2.3 系统业务流程图 .....                                                                                 | 511 | 28.9.2 设计工具栏 .....        | 539 |
| 28.2.4 系统编码规范 .....                                                                                  | 512 | 28.9.3 设计状态栏 .....        | 540 |
| 28.3 开发环境 .....                                                                                      | 513 | 28.10 进货单模块设计 .....       | 541 |
| 28.4 数据库与数据表设计 .....                                                                                 | 514 | 28.10.1 设计进货单窗体 .....     | 542 |
| 28.4.1 数据库分析 .....                                                                                   | 514 | 28.10.2 添加进货商品 .....      | 542 |
| 28.4.2 创建数据库 .....                                                                                   | 514 | 28.10.3 进货统计 .....        | 544 |
| 28.4.3 创建数据表 .....                                                                                   | 515 | 28.10.4 商品入库 .....        | 545 |
| 28.5 创建项目 .....                                                                                      | 519 | 28.11 销售单模块设计 .....       | 547 |
| 28.6 系统文件夹组织结构 .....                                                                                 | 520 | 28.11.1 设计销售单窗体 .....     | 548 |
| 28.7 公共类设计 .....                                                                                     | 520 | 28.11.2 添加销售商品 .....      | 548 |



|                          |     |                               |     |
|--------------------------|-----|-------------------------------|-----|
| 28.11.3 销售统计 .....       | 549 | 28.13.4 恢复数据库 .....           | 557 |
| 28.11.4 商品销售 .....       | 550 | 28.14 运行项目 .....              | 558 |
| 28.12 库存盘点模块设计 .....     | 551 | 28.15 系统打包发行 .....            | 559 |
| 28.12.1 设计库存盘点窗体 .....   | 552 | 28.16 开发常见问题与解决 .....         | 562 |
| 28.12.2 读取库存商品 .....     | 552 | 28.16.1 打包的 JAR 文件无法登录 .....  | 562 |
| 28.12.3 统计损益数量 .....     | 554 | 28.16.2 无法打开内部窗体 .....        | 562 |
| 28.13 数据库备份与恢复模块设计 ..... | 555 | 28.16.3 “关于”界面被其他窗体覆盖 .....   | 563 |
| 28.13.1 设计窗体 .....       | 555 | 28.16.4 打包 JAR 文件之后无法运行 ..... | 563 |
| 28.13.2 文件浏览 .....       | 555 | 28.16.5 程序运行后没有出现闪屏界面 .....   | 563 |
| 28.13.3 备份数据库 .....      | 556 | 28.17 小结 .....                | 564 |

# 光盘“开发资源库”目录

## 第 1 大部分 实例资源库

(732 个完整实例分析, 光盘路径: 开发资源库/实例资源库)

### 开发环境的应用

- 下载 JDK 开发工具包
- 把 JDK 工具包安装到指定磁盘
- 设置 JDK 的环境变量
- 验证 Java 开发环境
- 下载并安装 JRE 执行环境
- 编程输出星号组成的等腰三角形
- 下载最新的 Eclipse
- 为最新的 Eclipse 安装中文语言包
- 活用 Eclipse 的工作空间
- 在 Eclipse 项目中编程输出字符表情
- 为 Eclipse 添加新的 JDK 环境
- 设置 Eclipse 中文 API 提示信息
- 为项目添加类库
- 使当前项目依赖另一个项目
- 安装界面设计器
- 设计 Windows 系统的运行对话框界面
- 设计计算器程序界面
- 设计进销存管理系统的关于界面
- JDK 1.5 的安装与配置
- Tomcat 5.5 的安装
- 配置 Windows 2000+SQL Server 2000+Tomcat 运行
- 配置 Windows 2000+Oracle +Tomcat 运行环境
- 配置 Windows 2000+ Access +Tomcat 运行环境
- 配置 Windows 2000+MySQL+Tomcat 运行环境
- 配置 Windows XP 2003+SQL Server 2000+Tomcat 运行环境
- 在 Linux 下安装 JDK 1.5
- 在 Linux 下配置 Tomcat 服务器

- 配置 Linux+MySQL+ Tomcat 运行环境
- 配置 Windows+Resin 运行环境
- 配置 Linux+Resin 运行
- 安装与配置 WebLogic 服务器
- WebLogic 中 SQL Server 2000 的 JDBC 连接池配置
- 应用 Dreamweaver 开发 JSP 程序
- 配置 Windows 2000+MySQL+Tomcat 运行环境
- 应用 MyEclipse 开发 JSP 程序
- 应用 NetBeans 开发 JSP 程序

### Java 基础应用

- 基本语法
- 运算符
- 条件语句
- 循环控制

### 数组与集合的应用

- 数组演练
- 数组操作
- 数组排序与查询
- 常用集合的使用

### 字符串处理技术

- 格式化字符串
- 辨别字符串
- 操作字符串

### 面向对象技术应用

- Java 中类的定义
- 修饰符的使用
- 包装类的使用
- 面向对象的特征
- Object 类的应用



-  克隆与序列化
-  接口和内部类
-  多线程技术
  -  线程的基础
  -  线程的同步
  -  线程的进阶
-  反射与异常处理
  -  反射的基础
  -  反射的进阶
  -  常见的未检查型异常
  -  常见的已检查型异常
  -  处理异常
-  枚举与泛型的应用
  -  枚举使用的简介
  -  泛型使用的简介
-  编程常用类
  -  Calendar 类的使用
  -  SimpleDateFormat 与 TimeZone 类的使用
  -  System 类的使用
  -  Math 类的使用
  -  其他常用类的使用
-  Commons 组件
  -  Commons Lang 组件简介
  -  Commons Math 组件简介
  -  Commons IO 组件简介
  -  Commons BeanUtils 组件简介
  -  其他 Commons 组件简介
-  表单及表单元素的应用
  -  获取文本框 编辑框 隐藏域的值
  -  获取下拉列表 菜单的值
  -  获取复选框的值
  -  获取单选按钮的值
  -  把数据库中的记录显示到下拉列表中
  -  将数组中的数据添加到下拉列表中
  -  级联菜单
  -  修改数据时下拉列表的默认值为数据库中原数据信息
  -  可以输入文字的下拉列表
  -  根据下拉列表的值显示不同控件
-  根据数据表结构自动生成数据录入页面
-  投票信息一次性设置
-  自动计算金额
-  设置文本框的只读属性
-  让您的密码域更安全
-  限制多行文本域输入的字符个数
-  不提交表单获取单选按钮的值
-  选中单选按钮后显示其他表单元素
-  防止表单在网站外部提交
-  同一个页中的多表单提交
-  标签及设计模式专题
  -  利用 <c:forEach> 循环标签实现数据显示
  -  利用 EL 表达式语言实现页面逻辑处理简单化
  -  自定义文件下载标签
  -  自定义图片浏览标签
  -  自定义数据查询标签
  -  应用 HQL 检索方式查询数据
  -  应用 QBC 检索方式查询数据
  -  应用一对关联实现级联添加数据
  -  应用一对多关联实现级联操作
  -  使用本地 SQL 检索
  -  DispatchAction 类实现用户查询
  -  LookupDispatchAction 类实现用户管理
  -  利用 Token 令牌机制处理用户重复提交
  -  利用 Validator 验证框架处理用户登录
  -  解决用户提交的中文乱码
  -  利用动态 FormBean 实现对用户的操作
  -  Struts 与 Hibernate 结合实现数据添加和查询
  -  在 Spring 中的表单控制器中实现验证处理
  -  利用表单控制器实现数据添加操作
  -  利用 Spring 中的多方法控制器实现数据查询和删除操作
  -  通过 Spring+Hibernate 框架实现大批量数据添加
-  窗体设计与特效
  -  设置窗体位置
  -  设置窗体大小
  -  设置窗体的标题栏
  -  设置窗体的背景
  -  窗体形状及应用

-  对话框
-  MDI 窗体的使用
-  让窗体更有活力
-  窗体与控件外观
-   窗口与导航条设计
  -  打开新窗口显示广告信息
  -  自动关闭的广告窗口
  -  弹出窗口居中显示
  -  打开新窗口显示详细信息
  -  弹出窗口的 Cookie 控制
  -  为弹出的窗口加入关闭按钮
  -  关闭弹出窗口时刷新父窗口
  -  关闭 IE 主窗口时, 不弹出询问对话框
  -  弹出网页模式对话框
  -  弹出全屏显示的网页(模式)对话框
  -  网页拾色器
  -  日期选择器
  -  全屏显示无边框有滚动条窗口
  -  应用 JavaScript 实现指定尺寸的无边框窗口
  -  应用 CSS+DIV 实现无边框窗口
  -  带图标的文字导航条
  -  Flash 导航条
  -  图片按钮导航条
  -  导航条的动画效果
  -  不用图片实现质感导航条
  -  二级导航菜单
  -  半透明背景的下拉菜单
  -  弹出式下拉菜单
  -  展开式导航条
  -  收缩式导航菜单
  -  树状导航菜单
-   应用与控制
  -  将表单数据输出到 Word
  -  将查询结果输出到 Word
  -  通过 ODBC 访问 Excel
  -  利用 Java Excel 访问 Excel
  -  在 JSP 页面通过按钮打开新的 Excel 文件
  -  将查询结果导出到 Excel
  -  导出到 Access 数据库中
-  导出到 Excel 数据库中
-  在 JSP 中压缩 ZIP 文件
-  在 JSP 中解压缩 ZIP 文件
-  利用 Spring 生成 Excel 工作表
-  利用 Spring 生成 PDF 文件
-   基本控件应用
  -  顶层容器的应用
  -  布局管理器应用
  -  输入控件的应用
  -  选择控件的应用
  -  菜单控件的应用
  -  其他技术的应用
-   复合数据类型控件应用
  -  列表的简单应用
  -  列表的高级应用
  -  表格的简单应用
  -  表格的高级应用
  -  树控件简单应用
  -  树控件高级应用
-   其他高级控件应用
  -  JTextPane 控件的应用
  -  JEditorPane 控件的应用
  -  其他文本控件的应用
  -  进度指示器的应用
  -  控件组织器的应用
-   控件特效与自定义控件
  -  控件边框效果
  -  控件渲染让界面 UI 更灵活
  -  让控件活起来
  -  自定义控件
-   操作办公文档
  -  操作 Word
  -  操作 Excel
  -  操作 PDF
-   文件、文件夹与系统
  -  单表单元素上传文件到数据库
  -  多表单元素上传文件到数据库
  -  上传文件到服务器
  -  限制文件大小的文件上传

-  遍历指定目录下的所有文件
-  获取驱动器信息
-  遍历指定驱动器
-  访问类路径上的资源文件
-  获取文件信息
-  查看文件是否存在
-  重命名文件
-  对文件夹创建、删除的操作
-  使用 Java 的输入输出流从文本文件中读取注册服务条款
-  使用 Java 的输入输出流实现永久计数器
-  通过文本文件向数据库传递数据
-  读取属性文件
-  复制文件夹
-  文件下载
-  使用 JSP 生成 XML 文档
-  使用 DOM 读取 XML 文件
-  使用 SAX 读取 XML 文件
-  修改文件属性
-  显示指定类型的文件
-  以树结构显示文件路径
-  查找替换文本文件内容
-  支持图片预览的文件选择对话框
-  设置 Windows 的文件属性
-  文件批量重命名
-  快速批量移动文件
-  删除磁盘所有 .tmp 临时文件
-  提取数据库内容到文件
-  提取文本文件的内容到 MySQL 数据库
-  将图片文件保存到 SQL Server 数据库
-  显示数据库中的图片信息
-  提取技术网站数据到文件夹
-  读取文件路径到数据库
-  在数据库中建立磁盘文件索引
-  窗体动态加载磁盘文件
-  删除文件夹中所有文件
-  创建磁盘索引文件
-  快速全盘查找文件
-  获取磁盘所有文本文件
-  网络文件夹备份
-  键盘录入内容保存到文本文件
-  将数组写入到文件中并逆序输出
-  利用 StringBuffer 避免文件的多次写入
-  合并多个 txt 文件
-  实现文件简单加密与解密
-  对大文件实现分割处理
-  将分割后的文件重新合并
-  读取属性文件的单个属性值
-  向属性文件中添加信息
-  在复制文件时使用进度条
-  从 XML 文件中读取数据
-  读取 Jar 文件属性
-  电子通讯录
-  批量复制指定扩展名的文件
-  计数器小程序
-  将某文件夹中的文件进行分类存储
-  利用 StreamTokenizer 统计文件的字符数
-  在指定目录下搜索文件
-  序列化与反序列化对象
-  文件锁定
-  投票统计
-  压缩所有文本文件
-  压缩包解压到指定文件夹
-  压缩所有子文件夹
-  深层文件夹压缩包的释放
-  解决压缩包中文乱码
-  Apache 实现文件解压缩
-  把窗体压缩成 ZIP 文件
-  解压缩 Java 对象
-  文件压缩为 RAR 文档
-  解压缩 RAR 压缩包
-  文件分卷压缩
-  为 RAR 压缩包添加注释
-  获取压缩包详细文件列表
-  从 RAR 压缩包中删除文件
-  在压缩文件中查找字符串
-  重命名 RAR 压缩包中的文件
-  创建自解压 RAR 压缩包

-  设置 RAR 压缩包密码
-  以压缩格式传输网络数据
-  压缩远程文件夹
-  压缩存储网页
-  JSP 批量文件上传
-  Struts 的文件上传
-  Spring 的文件上传
-  从 FTP 下载文件
-  文件列表维护
-  文件在线压缩与解压缩
-  判断远程文件是否存在
-  通过文本文件实现网站计数器
-  JSP 生成 XML 文件
-  数据库的操作
  -  通过 JDBC-ODBC 桥连接 SQL Server 数据库
  -  通过 JDBC 连接 SQL Server 数据库
  -  通过 Tomcat 连接池连接 SQL Server 数据库
  -  通过 WebLogic 连接池连接 SQL Server 数据库
  -  应用 Hibernate 连接 SQL Server 数据库
  -  通过 JDBC-ODBC 桥连接 Access 数据库
  -  应用 Hibernate 连接 Access 数据库
  -  通过 JDBC 连接 MySQL 数据库
  -  通过 Tomcat 连接池连接 MySQL 数据库
  -  应用 Hibernate 连接 MySQL 数据库
  -  通过 JDBC 连接 Oracle 数据库
  -  应用 Hibernate 连接
  -  利用 SQL 语句实现分页
  -  利用结果集进行分页
  -  转到指定页的分页
  -  具有页码跳转功能的分页
  -  分栏显示
  -  分类、分栏显示
  -  对超长文本数据进行分页显示
  -  单条数据录入
  -  批量数据插入
  -  插入用户登录日志信息
  -  更新指定记录
  -  批量更新
  -  商品价格调整
-  修改密码
-  找回密码
-  动态创建 SQL Server 数据库
-  动态创建 SQL Server 数据表和字段
-  动态创建 MySQL 数据库
-  列举 SQL Server 数据库中的数据表
-  列举 MySQL 数据库中的数据表
-  查看数据表结构
-  在线维护投票数据库
-  通过 JDBC 获取插入记录的自动编号
-  通过 Hibernate 获取插入记录的自动编号
-  在线删除指定的一个数据表
-  在线删除多个指定的数据表
-  在线删除指定数据表中的指定索引
-  清空指定数据表中的所有数据
-  快速清空指定数据表中的所有记录
-  批量清空数据表中的数据
-  生成 SQL 数据库脚本
-  恢复 SQL 数据库脚本
-  删除指定记录
-  批量删除数据
-  删除数据前给予提示
-  获取从数据库里删除的记录数
-  生成有规律的编号
-  生成无规律的编号
-  SQL Server 数据备份
-  SQL Server 数据恢复
-  动态附加数据库
-  应用 JDBC 事务
-  Hibernate 中应用事务
-  对数据进行降序查询
-  对数据进行多条件排序查询
-  对统计结果进行排序
-  查询 SQLServer 数据表中前 3 条数据
-  查询 SQLServer 数据库后 3 条数据
-  查询 MySQL 数据库中的前 3 条数据
-  查询 MySQL 数据库后 3 条数据
-  按照字母顺序对留学生表进行排序
-  按姓氏笔画排序

-  将汉字按音序排序
-  按列的编号排序
-  从表中随机返回记录
-  使用 GROUP BY 函数实现对数据的分组统计
-  使用 GROUP BY 函数实现多表分组统计
-  利用 SUM 函数实现数据汇总
-  利用 AVG 函数实现计算平均值
-  利用 MIN 函数求数据表中的最小数据
-  利用 MAX 函数求数据表中的最大值
-  利用 COUNT 函数求销售额大于某值的图书种类
-  查询编程词典 6 月的销售量
-  查询与张静同一天入司的员工信息
-  使用 IN 谓词查询某几个时间的数据
-  日期查询中避免千年虫问题
-  在查询结果中不显示重复记录
-  使用 NOT 查询不满足条件的记录
-  使用 between 进行区间查询
-  列出销量表中重复记录和记录条数
-  使用关系运算符查询某一时间段数据
-  计算两个日期之间的月份数
-  格式化金额
-  在查询语句中过滤掉字符串中的空格
-  通过 JDBC-ODBC 桥连接 SQL Server 2000
-  通过 JDBC 连接 SQL Server 2000 数据库
-  JDBC 连接 SQL Server 2005 数据库
-  JDBC 技术连接 Oracle 数据库
-  JDBC 连接 Java DB 数据库
-  列举 SQL Server 数据库下的数据表
-  列举 MySQL 数据库下的数据表
-  动态维护投票数据库
-  MySQL 数据备份
-  MySQL 数据恢复
-  获取 SQL Server 数据表字段的描述信息
-  将员工信息添加到数据表
-  添加数据时使用数据验证
-  在插入数据时过滤掉危险字符
-  将用户选择爱好以字符串形式保存到数据库
-  将数据从一张表复制到另张表
-  使用 UNION ALL 语句批量插入数据
-  在删除数据时给出提示信息
-  将数据表清空
-  字符串大小写转换
-  使用 JDBC 连接常用数据库
-  JDBC 实现简单的搜索引擎
-  使用 JDBC 操作数据库
-  使用 JDBC 查看数据表结构
-  用户密码管理
-  使用 JDBC 批量处理数据
-  在 JDBC 中使用存储过程
-  使用 Tomcat 连接池
-  使用 dbcp 连接池
-  使用 c3p0 连接池
-  MySQL 数据库的备份和恢复
-  MySQL 数据库分页
-  SQL Server 数据库备份与恢复
-  SQL Server 数据库附加与分离
-  SQL Server 数据库分页
-  Hibernate 实现分页
-  将数据表导出到 XML 文件中
-  从 XML 文件导入数据表
-  在 MySQL 数据库增加新用户权限
-  使用事务提高数据库操作的安全性
-  在数据库中添加或读取文件数据
-  利用触发器记录系统登录日志
-  查询数值型数据
-  查询字符串
-  查询日期型数据
-  查询逻辑型数据
-  查询非空数据
-  查询文本框中指定的字符串
-  查询下拉列表框中指定的数值数据
-  查询下拉列表框中的日期数据
-  将表单元素中的内容作为字段、运算符和内容进行查询
-  利用变量查询字符串
-  利用变量查询数值型数据
-  查询前 5 名数据
-  查询后 5 名数据

-  取出数据统计结果前 3 名数据
-  查询指定 SQL Server 数据库中的日期型数据
-  查询指定 Access 数据库中的日期型数据
-  查询指定时间段的数据
-  按月查询数据
-  查询大于指定条件的数据
-  查询时不显示重复记录
-  NOT 与谓词进行组合条件的查询
-  列出数据中的重复记录和记录条数
-  单列数据分组统计
-  多列数据分组统计
-  多表分组统计
-  利用聚集函数 SUM 对学生成绩进行汇总
-  利用聚集函数 AVG 求某班学生的平均成绩
-  利用聚集函数 MIN 求销售额最少的商品
-  利用聚集函数 MAX 求月销售额完成最多的员工
-  利用聚集函数 COUNT 求日销售额大于某值的图书种类数
-  利用 FROM 子句进行多表查询
-  使用表的别名
-  合并多个结果集
-  简单的嵌套查询
-  复杂的嵌套查询
-  用子查询作派生的表
-  用子查询作表达式
-  用子查询关联数据
-  多表联合查询
-  对联合查询后的结果进行排序
-  条件联合查询
-  简单内连接查询
-  复杂内连接查询
-  自连接
-  LEFT OUTER JOIN 查询
-  RIGHT OUTER JOIN 查询
-  使用外连接进行多表联合查询
-  利用 IN 谓词限定查询范围
-  用 IN 查询表中的记录信息
-  由 IN 引入的关联子查询
-  静态交叉表

-  动态交叉表
-  对查询结果进行格式化（四舍五入）
-  在查询中使用字符串函数
-  在查询中使用日期函数
-  利用 HAVING 语句过滤分组数据
-  复杂条件查询

## 数据库的高级应用

-  创建视图
-  使用视图过滤不想要的数据库
-  使用视图与计算数据
-  使用视图重新格式化检索出来的数据
-  获取数据库中的全部用户视图
-  修改视图
-  删除视图
-  视图的应用
-  创建存储过程
-  调用存储过程实现用户身份验证
-  应用存储过程添加数据
-  调用加密存储过程
-  获取数据库中所有存储过程
-  修改存储过程
-  删除存储过程
-  应用存储过程实现数据分页
-  创建触发器
-  应用触发器自动插入回复记录
-  应用触发器添加日志信息
-  在删除成绩表时将学生表中数据删除
-  在程序中调用 UPDATE 触发器
-  获取数据库中的触发器名称
-  创建带有触发条件的触发器
-  使用批处理删除数据
-  使用批处理提升部门员工工资
-  将教师表中数据全部添加到选课表
-  在批处理中使用事务

## 实用的 JavaScript 函数

-  小写金额转换为大写金额
-  处理字符串中的空格
-  验证输入的日期格式是否正确
-  检查表单元素是否为空



 验证 E-mail 是否正确 通过正则表达式验证电话号码 验证输入的字符串是否为汉字 验证身份证号码 客户端验证用户名和密码 验证网址是否合法 验证数量和金额 限制输入字符串的长度 显示长日期格式的系统日期 倒计时 实时显示系统时间 特殊日期提示 **数据查询** 使用子查询 嵌套查询 连接查询 函数查询 **Servlet 技术** 将 HTML 元素嵌入到 Servlet 在 Servlet 中实现页面转发的操作 在 Servlet 中获取当前页的绝对路径 在 Servlet 中对 Cookie 的操作 利用 JavaBean 由 Servlet 向 JSP 传递数据 在 Servlet 中使用 JDBC- ODBC 桥访问数据库 在 Servlet 中使用 JDBC 访问数据库 使用 Servlet 访问数据库连接池 使用过滤器验证用户身份 使用过滤器进行网站流量统计 使用过滤器对响应页面中的敏感字符进行过滤 使用监听器查看在线用户 利用监听器使服务器端免登录 过滤非法文字 编码过滤器解决中文问题 过滤器验证用户 过滤器分析流量 使用过滤器禁止浏览器缓存页面 监听在线用户 监听器实现免登录 **JavaBean 技术** 连接数据库的方法 数据查询的方法 带参数的数据查询 数据增加的方法 数据修改的方法 数据删除的方法 数据库分页的方法 对结果集进行分页的方法 关闭数据库的方法 数据库事务处理的方法 调用数据库存储过程的方法 附加数据库的方法 备份数据库的方法 还原数据库的方法 自动获得汉字的拼音简码 转换输入文本中的回车和空格 小写金额转换为大写金额 判断字符串是否以指定字符开头 计算字符串的实际长度 字符串截取 字符串转换成数组 检查字符是否有英文字母 小写字母转换为大写字母 大写字母转换为小写字母 把数组转换成字符串 将整型数据格式化为指定长度的字符串 把一个长数字分位显示 过滤输入字符串中的危险符号 判断是否为当前时间的方法 判断用户输入的是否是数字的方法 对输入数据中的 HTML 字符进行转换的方法 过滤字符串中的空格与 null 值的方法 对 SQL 语句中输入的空值进行处理的方法 将整型值转换为字符型的方法 判断用户输入的是否为有效 id 值的方法 获取年份的方法 获取月份的方法 获取日的方法

-  显示指定格式的日期的方法
-  显示指定格式的时间的方法
-  显示完整日期时间的方法
-  对字符串进行 GBK 编码
-  对字符串进行 ISO-8859-1 编码
-  随机产生指定位数的验证码
-  生成指定位数的随机字符串
-  用户登录模块
-  带验证码的用户登录模块
-  带识别状态的用户登录模块
-  输出提示页面的方法
-  输出分页导航的方法
-  版权信息生成的方法
-  生成柱形图
-  生成折线图
-  生成饼状图
-  实现进度条
-  弹出提示对话框并重定向网页
-  打开指定大小的新窗口并居中显示
-  **Ajax 技术**
  -  Ajax 无刷新分页
  -  Ajax 实现聊天室
  -  Ajax 验证用户名是否被注册
  -  Ajax 刷新 DIV 内容
  -  Ajax 级联选择框
  -  Ajax 实现带进度条的文件上传
-  **在线统计**
  -  通过 Application 对象实现网站计数器
  -  网站图形计数器
  -  记录用户 IP 地址的计数器
  -  只对新用户计数的计数器
  -  统计用户在某一页停留的时间
  -  统计用户在站点停留的时间
  -  判断用户是否在线
  -  实时统计在线人数
  -  统计日访问量
  -  利用柱形图统计分析网站访问量
-  **网络通信**
  -  发送电子邮件
  -  发送 HTML 格式邮件
-  带附件的邮件发送程序
-  邮件群发
-  Spring 利用 Web Service 发送手机短信
-  利用短信猫发送手机短信
-  实现邮件发送
-  实现邮件接收
-  发送带附件的邮件
-  接收带附件的邮件
-  显示 POP3 未读邮件和已读邮件
-  IP 地址转换成整数
-  获取本地天气预报
-  **信息提取与图表分析**
  -  远程获取其他网页信息
  -  网站访问量显示图表
  -  投票结果显示图表
  -  利用折线图分析多种商品的价格走势
  -  利用折线图分析某种商品的价格走势
  -  年销售额及利润图表分析
-  **行业应用**
  -  禁止重复投票的在线投票系统
  -  每个 IP 一个月只能投票一次的投票系统
  -  一般用户注册
  -  带检测用户名的用户注册
  -  分步用户注册
  -  论坛
  -  购物车
  -  Application 形式的聊天室
  -  聊天室（私聊）
  -  数据库形式的聊天室（私聊）
  -  简易万年历
  -  带有备忘录的万年历
-  **网站策略与安全**
  -  通过邮箱激活注册用户
  -  越过表单限制漏洞
  -  文件上传漏洞
  -  防止 SQL 注入式攻击
  -  获取客户端信息
  -  防止网站文件盗链下载
  -  禁止网页刷新



禁止复制和另存网页内容

防止页面重复提交

获取指定网页源代码并盗取数据

隐藏 JSP 网址扩展名

数据加密

MD5 加密

## 安全技术

确定对方的 IP 地址

获取客户端 TCP / IP 端口的方法

替换输入字符串中的危险字符

禁止用户输入危险字符

用户安全登录

带验证码的用户登录模块

防止用户直接输入地址访问 JSP 文件

禁止复制网页内容

禁止网页被另存为

屏蔽 IE 主菜单

屏蔽键盘相关事件

屏蔽鼠标右键

对登录密码进行加密

字符串加密

## 实用工具

在线查询 IP 地理位置

手机号码归属地查询

工行在线支付

支付宝的在线支付

快钱在线支付

在线文本编辑器

网页拾色器

在线验证 18 位身份证

在线汉字转拼音

在线万年历

进制转换工具

## 高级应用开发

自动选择语言跳转

JSP 防刷计数器

用 JSP 操作 XML 实现留言板

网站支持 RSS 订阅

JSP 系统流量分析

用 JSP 生成 WEB 静态网页

## 图形图像与多媒体

别致的图形计数器

预览并上传图片

分页浏览图像

生成中文验证码

生成关键字验证码

生成隐藏的验证码

生成带背景和雪花的验证码

生成带有干扰线的验证码

为图像添加文字水印和图片水印

制作相册幻灯片

在线连续播放音乐

在线播放 FLV 视频

同步显示 LRC 歌词

JSP 生成条形码

通过下拉列表框选择头像

从网页对话框中选择头像

通过滑动鼠标放大或缩小图片

随机显示图片

幻灯片式图片播放

浮动广告

插入 Flash 动画

插入背景透明的 Flash

在线播放 MP3 歌曲列表

MP3 文件下载

自制视频播放器

在线影片欣赏

## 报表打印

利用 JavaScript 调用 IE 自身的打印功能实现打印

利用 WebBrowser 打印

将页面中的客户列表导出到 Word 并打印

利用 Word 自动打印指定格式的会议记录

利用 Excel 打印工资报表

将 Web 页面中的数据导出到 Excel 并自动打印

打印库存明细表

打印库存盘点报表

打印库存汇总报表

打印指定条件的库存报表

- 打印汇款单
- 打印信封
- 利用柱形图分析报表
- 利用饼形图分析报表
- 利用折线图分析报表
- 利用区域图分析报表
- 导出报表到 Excel 表格
- 导出报表为 PDF 文档

- 实现打印报表功能
- 实现打印预览功能
- 用 JSP 实现 Word 打印
- JSP+CSS 打印简单的数据报表
- JSP 套打印快递单
- JSP 生成便于打印的网页
- 将数据库中的数据写入到 Excel
- 将数据库中的数据写入到 Word

## 第 2 大部分 模块资源库

(15 个经典模块, 光盘路径: 开发资源库/模块资源库)

### 模块 1 程序打包

#### 概述

- jar 文件
- MANIFEST 文件

#### 使用命令实现程序打包

- 完成单个文件打包
- 完成打包多个文件

#### 在 Eclipse 中实现程序打包

#### 常见问题与解决

### 模块 2 数码照片管理模块

#### 模块概述

- 设计思路
- 模块架构

#### 关键技术

- 捕获树的选中节点事件
- 捕获树的展开节点事件
- 浏览方式切换技术
- 随意选取照片技术
- 图片缩放与内存溢出
- 工具提示回行显示技术

#### 效果预览

#### 实现对相册树的维护

#### 实现添加照片的功能

#### 实现修改照片信息的功能

#### 实现删除照片的功能

#### 实现全屏查看照片功能

#### 实现浏览方式的切换

#### 实现查找照片功能

#### 实现图片播放器

#### 保存选中图片到指定路径

### 模块 3 FTP 文件管理模块

#### FTP 文件管理模块概述

- 模块简介
- 功能结构
- 业务流程
- 程序预览

#### 关键技术

- 架设 FTP 服务器
- 登录 FTP 服务器
- 浏览本地资源
- 浏览 FTP 服务器资源
- FTP 文件上传与下载
- 向 FTP 服务器发送命令
- 获取文件在本系统的显示图标
- 任务队列

#### 实现 FTP 站点管理功能

- 装载属性文件
- 装载 FTP 站点信息
- 编写站点维护对话框
- 维护 FTP 站点

#### 实现登录面板



- [-] 实现本地资源管理
  - [-] 呈现本地资源
  - [-] 本地资源的控制面板
- [-] 实现 FTP 资源管理
  - [-] 呈现本地资源
  - [-] FTP 服务器资源的控制面板
- [-] 实现队列管理
  - [-] 任务队列
  - [-] 本地队列文件上传
  - [-] FTP 队列文件下载

## 模块 4 电子地图

- [-] 模块概述
  - [-] 设计思路
  - [-] 模块架构
  - [-] 效果预览
- [-] 关键技术
  - [-] Java DB 数据库技术
  - [-] 万年历选择框技术
  - [-] 滑块组件使用技术
  - [-] 列表组件使用技术
  - [-] 维护树模型技术
- [-] 实现地图处理器类
- [-] 实现用来绘制地图的标签组件
  - [-] 绘制地图显示区的大地图
  - [-] 绘制鹰眼漫游区的小地图
- [-] 实现操作地图功能
  - [-] 实现缩放地图功能
  - [-] 实现移动地图功能
- [-] 实现维护标记功能
  - [-] 实现弹出菜单功能
  - [-] 实现对标记的维护
  - [-] 实现查看标记信息功能
- [-] 实现搜索标记功能
  - [-] 实现常用搜索功能
  - [-] 实现高级搜索功能
  - [-] 描红并居中显示选中标记

## 模块 5 网络五子棋游戏

- [-] 五子棋模块概述
  - [-] 模块简介
  - [-] 程序预览
- [-] 关键技术
  - [-] 实现透明的登录界面
  - [-] 监控网络连接状态
  - [-] 绑定属性的 JavaBean
  - [-] 在棋盘上绘制棋子
  - [-] 实现动态调整棋盘大小
  - [-] 游戏悔棋
  - [-] 游戏回放
- [-] 实现登录界面
- [-] 编写游戏主窗体
- [-] 编写下棋面板
- [-] 编写棋盘面板
- [-] 实现游戏规则算法
- [-] 编写棋盘模型
- [-] 编写联机通讯类

## 模块 6 远程协助模块

- [-] 远程协助模块介绍
  - [-] 模块简介
  - [-] 功能结构
  - [-] 程序预览
- [-] 关键技术
  - [-] 截取屏幕图像
  - [-] 控制计算机的输入
  - [-] 在网络中发送和接收图片
  - [-] RMI 实现远程控制
  - [-] 自定义组件显示远程屏幕
- [-] 联系人管理
  - [-] 添加联系人
  - [-] 修改联系人
  - [-] 删除联系人
  - [-] 显示联系人列表
  - [-] 处理联系人选择事件
- [-] 创建网络服务器
- [-] 编写远程连接面板



□ 启动 RMI 远程方法服务

□ 实现远程监控界面

□ 实现系统托盘

## 模块 7 软件注册模块

□ 软件注册模块概述

□ 模块概述

□ 功能结构

□ 程序预览

□ 关键技术

□ 读取客户端 MAC 地址

□ Java 操作注册表

□ 避免用户修改系统时间

□ 右键单击弹出菜单

□ 一次性粘贴注册码

□ 获取两个时间的相隔天数

□ ini 文件的读写

□ RSA 加密解密算法

□ 软件注册导航窗体的实现

□ 窗体概述

□ 窗体界面设计

□ 软件试用部分的实现

□ 软件注册窗体的实现

□ 窗体概述

□ 窗体界面设计

□ 验证注册码

□ 限制注册用户使用时间

□ 根据注册机器的硬件信息保证软件使用唯一性

□ 注册机的实现

□ 窗体概述

□ 窗体界面设计

□ 生成注册码

## 模块 8 多媒体播放器模块

□ 模块概述

□ 模块概述

□ 功能结构

□ 程序预览

□ 关键技术

□ 安装 jmf-2\_1\_1e-windows-i586.exe

□ JMF 播放视频文件

□ 设置窗体外观感觉

□ 窗体全屏显示

□ 友情链接

□ 实现播放媒体文件

□ 选择本地媒体文件

□ 实现媒体播放

□ 实现播放控制

□ 播放列表维护

□ 添加列表数据

□ 实现列表“上移”“下移”

□ 实现列表元素重命名

□ 实现删除列表内容

□ 实现播放列表中元素全部删除

□ 通过双击列表选择播放文件

□ 播放控制

□ 创建最近播放列表

□ 编写数据库操作方法

□ 动态添加菜单项

□ 实现自动检索系统中媒体文件

□ 创建选择文件夹对话框

□ 获取媒体文件集合

□ 将媒体文件添加到播放列表

## 模块 9 决策分析模块

□ 模块概述

□ 设计思路

□ 模块架构

□ 效果预览

□ 数据接口

□ 接口设计

□ 测试数据

□ 关键技术

□ 支持固定列表格技术

□ 使用 JFreeChart 绘制统计图技术

□ 使用 JavaExcel 生成 Excel 文件

□ 使用 IText 生成 PDF 格式的文件

□ 多线程与进度条的使用



## ▣ 实现过程

- ▣ 实现动态控制表格的固定列数量
- ▣ 实现组件间的可用性控制
- ▣ 生成统计图与使用进度条
- ▣ 保存统计图到指定路径
- ▣ 导出报表到 Excel 表格
- ▣ 利用报表和统计图生成 PDF 文件

## 模块 10 网站留言簿

### ▣ 概述与开发环境

- ▣ 概述
- ▣ 开发环境

### ▣ 实例运行结果

### ▣ 设计与分析

- ▣ 系统分析
- ▣ 系统流程
- ▣ 文件夹及文件架构
- ▣ Hibernate 配置文件及类的分布

### ▣ 技术要点

- ▣ 获取留言及回复信息
- ▣ 获取系统日期和时间
- ▣ 保存留言信息时自动插入回复记录

### ▣ 数据表结构

### ▣ 创建 Hibernate 配置文件

### ▣ 创建实体类及映射文件——创建实体类

### ▣ 创建实体类及映射文件——创建映射文件

### ▣ 业务处理逻辑类——对留言及回复信息操作的 Topic 类 1

### ▣ 业务处理逻辑类——对留言及回复信息操作的 Topic 类 2

### ▣ 业务处理逻辑类——对管理员操作的 Manager 类

### ▣ 创建公共类——ChStr 类和 GetTime 类

### ▣ 创建公共类——style 样式文件和 check 类

### ▣ 添加留言信息

### ▣ 显示留言信息

### ▣ 回复留言

### ▣ 删除留言

### ▣ 用户登录页面

### ▣ 公共页

## ▣ 调试、发布与运行

- ▣ 调试
- ▣ 发布与运行

## 模块 11 桌面精灵

### ▣ 模块概述

### ▣ 关键技术

### ▣ 实现滚动字幕

### ▣ 实现支持农历的万年历

### ▣ 实现维护记录功能

### ▣ 实现搜索记录功能

## 模块 12 短信发送模块

### ▣ 短信发送模块概述

### ▣ 关键技术

### ▣ 数据库设计

### ▣ 设置并连接短信猫

### ▣ 读取短信

### ▣ 发送短信

### ▣ 发信箱的实现

### ▣ 联系人管理

## 模块 13 数据分页

### ▣ 概述与开发环境

### ▣ 实例运行结果

### ▣ 设计与分析

### ▣ 技术要点

### ▣ 开发过程

### ▣ 发布与运行

## 模块 14 电子阅读器模块

### ▣ 电子阅读模块概述

### ▣ 关键技术

### ▣ 实现主窗体

### ▣ PDF 文档读取的实现

### ▣ 缩位图的实现

### ▣ 书签的实现

### ▣ 全屏显示 PDF 文档

## 模块 15 复杂条件查询

.....



## 第3大部分 项目资源库

(15 个企业开发项目, 光盘路径: 开发资源库/项目资源库)

### 项目 1 Validator 验证框架

.....

### 项目 2 网上办公自动化系统

#### □ 概述与开发环境

□ 概述

□ 开发环境

#### □ 需求分析

□ 项目规划

□ 系统功能结构图

#### □ 数据库设计

□ 数据表概要说明

□ 数据表的结构

#### □ 前期准备

□ 配置应用 Struts 结构文件及数据库连接文件

□ 配置 web.xml 文件

□ 配置 Struts 标签库文件

□ 编写 Struts 框架配置文件

#### □ 网站总体设计

□ 系统架构设计

□ 类的分布架构设计

□ 文件架构设计

□ 网站首页运行效果图

□ 删除发文子模块

#### □ 公共类的编写

□ 数据库的连接及操作方法类: DB

□ 数据表信息类: Content

□ 分页类: Page

□ 类型转换类: Change

□ 检查用户权限类: CheckUserAble

□ 解决 Struts 中的中文乱码的类: FormToChinese

□ 检查用户是否已经在线的公共类

#### □ 登录模块

□ 创建登录的页面: index.jsp

□ 配置 struts-config.xml 文件

□ 创建 LogonForm 类

□ 创建 LogonAction 类

#### □ 自定义标签的开发

□ 如何开发及使用自定义标签

□ 本例中 office 自定义标签的开发过程

#### □ 收/发文管理模块

□ 建立发文子模块

□ 浏览发文子模块

#### □ 会议管理模块

□ 查看会议记录子模块

□ 添加会议记录子模块

□ 浏览会议的详细内容

□ 删除会议子模块

#### □ 公告管理模块

□ 修改公告子模块

□ 浏览公告信息

□ 查看公告详细信息

□ 添加公告信息

#### □ 人力资源管理模块

□ 浏览员工信息模块中的查询功能

□ 个人信息子模块

□ 查看员工详细信息

□ 修改员工信息

□ 修改个人信息

□ 添加员工信息

#### □ 文档管理

□ 浏览文件详细内容

□ 删除文件子模块

□ 文件上传子模块

□ 文件下载子模块

□ 浏览文件信息

#### □ 资产管理模块

□ 资产管理模块

- 浏览办公用品信息
- 更新办公用品信息
- 添加办公用品信息
- 浏览车辆信息
- 浏览车辆详细信息
- 添加车辆信息
- 修改车辆信息
- 资产管理模块公共类
- 内部邮件管理模块
  - 查看收件箱
  - 查看发件箱
  - 查看邮件详细信息
  - 添加邮件信息
  - 内部邮件管理模块公共类
- 意见箱模块
  - 查看意见箱
  - 查看意见详细信息
  - 发送新的建议
  - 意见箱模块公共类
- 退出模块
- 疑难解答
- 公共模块

### 项目 3 图书管理系统

- 概述与开发环境
  - 概述
  - 开发环境
- 需求分析
- 系统设计
  - 项目规划
  - 系统功能结构图
- 数据库设计
  - 数据表概要说明
  - 数据表关系概要说明
  - 主要数据表的结构
- 网站总体设计
  - 系统架构设计
  - 类的分布架构设计
- 公共类的编写
  - 自动转码类

- 对系统时间操作类
- 取得自动编号操作类
- 自动转码类
- IBatis 设计模式的介绍
  - IBatis 设计模式的组成
  - 构建 IBatis 设计模式的 SQL Map 配置代码
- 配置 Struts
- ActionForm 类的编写及配置
  - ActionForm 类的编写及配置
  - 管理员信息 ActionForm 类的代码
  - 图书信息操作的 ActionForm 类
  - 图书借阅信息操作的 ActionForm 类
- 对数据表操作持久类的编写
  - IBatis 设计模式的 SQL Map 映射文件
  - IBatis 基本组件
  - 管理员信息表的操作
  - 图书信息表的操作
  - 图书借阅信息表的操作
- 图书管理系统总体架构
  - 图书管理系统文件架构设计
  - 图书管理系统首页设计
- 管理员功能模块
  - 模块功能介绍
  - 创建管理员的 Action 实现类
  - 系统登录设计
  - PaginatedList 类实现分页显示管理员列表
  - 添加管理员信息
  - 设置管理员权限
  - 删除管理员信息
  - 管理员查询
  - 修改密码
- 图书管理功能模块
  - 图书管理功能模块
  - 创建图书的 Action 实现类
  - 查看图书信息列表
  - 添加图书信息
  - 删除图书信息
  - 查询图书详细信息
  - 修改图书信息

- 库存进书管理
  - 查询库存量不足的图书
  - 库存查询
- 图书借还管理功能模块
    - 图书借还功能模块
      - 创建图书借阅的 Action 实现类
      - 图书借阅
      - 图书归还
      - 图书续借
      - 借阅查询
  - 图书设置模块
    - 图书存放位置管理
      - 图书类别管理
      - 读者类别管理
      - 读者信息管理
      - 图书设置模块公共类
  - 图书销售模块
    - 图书销售管理
      - 图书销售查询
      - 图书销售模块公共类
  - 图表分析模块
    - 图书销售图表分析
      - 图书借阅图表分析
      - 图表分析模块公共类
  - 公共模块
  - 疑难问题分析与解决
      - 映射对数据表操作的 XML 文件的路径错误
        - 映射属性的错误操作

#### 项目 4 进销存管理系统

- 开发背景和系统分析
    - 开发背景
      - 需求分析
      - 可行性分析
      - 编写项目计划书
  - 系统设计
    - 系统目标
      - 系统功能结构
      - 业务逻辑编码规则
      - 系统流程图

- 构建开发环境
  - 系统预览
  - 文件夹组织结构
- 数据库设计
    - 数据库分析
      - 进销存管理系统的 E-R 图
      - 使用 PowerDesigner 建模
      - 创建数据库
  - 主窗体设计
    - 创建主窗体
      - 创建导航面板
  - 公共模块设计
    - 编写 Dao 公共类
      - 编写 Item 类
  - 基础信息模块设计
    - 基础信息模块概述
      - 基础信息模块技术分析
      - 供应商添加实现过程
      - 供应商修改与删除实现过程
      - 单元测试
  - 进货管理模块设计
    - 进货管理模块概述
      - 进货管理模块技术分析
      - 进货单实现过程
  - 查询统计模块设计
    - 查询统计模块概述
      - 查询统计模块技术分析
      - 销售查询实现过程
  - 库存管理模块设计
    - 库存管理模块概述
      - 库存管理模块技术分析
      - 价格调整实现过程
      - 单元测试
  - 系统打包发布
  - 开发技巧与难点分析
  - 使用 PowerDesigner 逆向生成数据库 E-R 图

#### 项目 5 基于 Struts 与 Hibernate 开发的新闻网络中心

.....



## 项目 6 网上购物商城

- 概述与系统分析
- 总体设计
  - ▢ 项目规划
  - ▢ 系统功能结构图
- 系统设计
  - ▢ 设计目标
  - ▢ 开发及运行环境
  - ▢ 逻辑结构设计
- 技术准备
  - ▢ MVC 概述
  - ▢ Struts 概述
  - ▢ 在 MyEclipse 中配置应用 Struts 结构文件
- 系统架构设计
  - ▢ 系统文件夹架构图
  - ▢ 文件架构设计
- JavaBean 的设计
  - ▢ JavaBean 的设计
  - ▢ 数据库连接的 JavaBean 的编写
  - ▢ 设置系统中使用的 SQL 语句的 JavaBean
  - ▢ 解决 Struts 中文乱码问题
  - ▢ 检查用户是否已经在线的公共类
- 会员管理模块
  - ▢ 会员登录
  - ▢ 用户注册
  - ▢ 找回密码
- 网站主页设计
  - ▢ 网站主页设计
  - ▢ 网站首页面导航信息版块
  - ▢ 网站首页面左部信息版块
  - ▢ 网站首页面右部信息版块
  - ▢ 网站首页面版权信息版块
- 会员资料修改模块
- 购物车模块
  - ▢ 添加购物车
  - ▢ 查看购物车
  - ▢ 生成订单
  - ▢ 清空购物车

- 商品销售排行模块
  - ▢ 商品销售排行榜
  - ▢ 分页显示特价商品
- 网站后台主要功能模块设计
  - ▢ 网站后台首页设计
  - ▢ 后台管理员身份验证模块
  - ▢ 商品设置模块
  - ▢ 订单设置模块
- 退出模块
- 疑难问题分析
  - ▢ 中文乱码问题的处理
  - ▢ 关闭网站后 session 没有被注销

## 项目 7 博客系统

- 概述与系统分析
- 总体设计
  - ▢ 项目规划
  - ▢ 系统功能结构图
- 系统设计
  - ▢ 设计目标
  - ▢ 开发及运行环境
  - ▢ 逻辑结构设计
- 技术准备
  - ▢ Hibernate 框架概述
  - ▢ Hibernate 配置文件
  - ▢ 创建持久化类
  - ▢ Hibernate 映射文件
- 系统构架设计
  - ▢ 系统文件夹架构图
  - ▢ 文件夹架构设计
- 公共类设计
  - ▢ 获得当前系统时间类
  - ▢ 字符处理类的编写
  - ▢ 将字符串转化成字符数组类
  - ▢ Hibernate 的初始化与 Session 管理类的编写
- 网站前台主要功能设计
  - ▢ 网站首页页面设计
  - ▢ 网站计数功能实现
  - ▢ 网络日历功能
  - ▢ 博主信息显示模块



- 浏览博主发表文章模块
- 添加评论模块
- 网站后台主要功能模块设计
  - 网站后台主要功能模块设计
  - 后台首页设计
  - 个人相片设置模块
  - 博主设置模块
  - 博客文章管理模块
- 疑难问题分析
  - Hibernate 的映射类型
  - 如何使用 Hibernate 声明事务边界
- 程序调试与错误处理

## 项目 8 企业内部通信系统

- 开发背景和系统分析
  - 开发背景
  - 需求分析
  - 可行性分析
  - 编写项目计划书
- 系统设计
  - 系统目标
  - 系统功能结构
  - 数据库设计
  - 系统预览
  - 文件夹组织结构
- 主窗体设计
  - 创建主窗体
  - 记录窗体位置
- 公共模块设计
  - 数据库操作类
  - 系统工具类
- 系统托盘模块设计
  - 系统托盘模块概述
  - 系统托盘模块技术分析
  - 系统托盘模块实现过程
- 系统工具模块设计
  - 系统工具模块概述
  - 系统工具模块技术分析
  - 系统工具模块实现过程

- 用户管理模块设计
  - 用户管理模块概述
  - 用户管理模块技术分析
  - 用户管理模块实现过程
  - 单元测试
- 通信模块设计
  - 通信模块概述
  - 通信模块技术分析
  - 通信模块实现过程
- 开发技巧和 JDK 6 新增的系统托盘
  - 开发技巧与难点分析
  - 使用 JDK 6 新增的系统托盘

## 项目 9 网络购物中心

.....

## 项目 10 企业人事管理系统

- 开发背景和系统分析
  - 开发背景
  - 系统分析
- 系统设计
  - 系统目标
  - 系统功能结构
  - 系统预览
  - 业务流程图
  - 文件夹结构设计
- 数据库设计
  - 数据库分析
  - 数据库概念设计
  - 数据库逻辑结构设计
- 主窗体设计
  - 导航栏的设计
  - 工具栏的设计
- 公共模块设计
  - 编写 Hibernate 配置文件
  - 编写 Hibernate 持久化类和映射文件
  - 编写通过 Hibernate 操作持久化对象的常用方法
  - 创建用于特殊效果的部门树对话框
  - 创建通过部门树选取员工的面板和对话框

## 人事管理模块设计

-  人事管理模块功能概述
-  人事管理模块技术分析
-  人事管理模块实现过程
-  单元测试

## 待遇管理模块设计

-  待遇管理模块功能概述
-  待遇管理模块技术分析
-  待遇管理模块实现过程

## 系统维护模块设计

-  系统维护模块功能概述
-  系统维护模块技术分析
-  系统维护模块实现过程
-  单元测试

## 开发技巧与难点分析

## Hibernate 关联关系的建立方法

-  建立一对一关联
-  建立一对多关联

## 项目 11 销售管理系统

.....

## 项目 12 医药进销存管理系统

-  概述与系统分析
-  总体设计
-  系统设计
-  技术准备
-  系统总体架构设计
-  系统公共类设计
-  Java 实体类及 Hibernate 映射文件的设计
-  系统功能模块设计
-  系统主界面设计
-  设计药品基本情况模块
-  设计客户基本情况模块

## 设计供应商基本情况模块

-  设计药品采购模块
-  设计药品销售模块
-  设计库存盘点模块
-  设计销售退货模块
-  设计客户回款模块
-  设计基本信息查询模块
-  设计入库明细查询模块
-  设计销售明细查询模块
-  设计回款信息查询模块
-  设计增加用户功能模块
-  设计用户维护功能模块
-  系统提示模块设计
-  疑难问题解析
-  系统常见错误处理
- .....

## 项目 15 酒店管理系统

-  概述与系统分析
-  总体设计
-  系统设计
-  技术准备
-  系统架构设计
-  数据持久层设计
-  主窗体的格局设计
-  开台签单功能的设计与实现
-  自动结账功能的设计与实现
-  销售统计功能的设计与实现
-  人员管理功能的设计与实现
-  系统维护功能的设计与实现
-  系统安全功能的设计与实现
-  疑难问题分析

## 第4大部分 能力测试题库

(616 道能力测试题目, 光盘路径: 开发资源库/能力测试)

### 第1部分 Java 编程基础能力测试

.....

### 第2部分 数学及逻辑思维能力测试

 基本测试

 进阶测试

 高级测试

### 第3部分 编程英语能力测试

 英语基础能力测试

 英语进阶能力测试

## 第5大部分 面试资源库

(369 项面试真题, 光盘路径: 开发资源库/编程人生)

### 第1部分 Java 程序员职业规划

 你了解程序员吗

 程序员自我定位

### 第2部分 Java 程序员面试技巧

 面试的三种方式

 如何应对企业面试

 英语面试

 电话面试

 智力测试

### 第3部分 Java 常见面试题

 Java 语法面试题

 字符串与数组面试题

 面向对象试题

 Java 异常面试题

 多线程试题

 集合类试题

 数据库相关试题

 网络与数据流的试题

 数据结构与算法试题

 软件工程与设计模式

### 第4部分 Java 企业面试真题汇编

 企业面试真题汇编(一)

 企业面试真题汇编(二)

 企业面试真题汇编(三)

 企业面试真题汇编(四)

### 第5部分 Java 虚拟面试系统

# 第 1 篇

## 基础知识

- » 第 1 章 初识 Java
- » 第 2 章 熟悉 Eclipse 开发工具
- » 第 3 章 Java 语言基础
- » 第 4 章 流程控制
- » 第 5 章 字符串
- » 第 6 章 数组
- » 第 7 章 类和对象
- » 第 8 章 包装类
- » 第 9 章 数字处理类

本篇通过初识 Java、熟悉 Eclipse 开发工具、Java 语言基础、流程控制、字符串、数组、类和对象、包装类、数字处理类等内容的介绍，结合大量的图示、举例、视频等，使读者快速掌握 Java 语言，为以后编程奠定坚实的基础。

# 第 1 章

---

## 初识 Java

(  视频讲解：25 分钟 )

Java 是一种跨平台的、面向对象的程序设计语言。本章将简单介绍 Java 语言的不同版本及其相关特性以及学好 Java 语言的方法等，主要目的是让读者对 Java 语言有一个整体的了解，然后再慢慢地学习具体内容，最后达到完全掌握 Java 语言的目的。

通过阅读本章，您可以：

- » 了解 Java 语言
- » 了解 Java 的版本
- » 了解 Java 的应用领域
- » 了解如何学好 Java
- » 了解 Java 语言特性
- » 掌握不同平台的 JDK 环境搭建
- » 掌握 Java 程序的编写方法

## 1.1 Java 简介

 视频讲解：光盘\TM\lx\1\Java 简介.mp4

Java 是一种高级的面向对象的程序设计语言。使用 Java 语言编写的程序是跨平台的，从 PC 机到手持电话都有 Java 开发的程序和游戏，Java 程序可以在任何计算机、操作系统和支持 Java 的硬件设备上运行。

### 1.1.1 什么是 Java 语言

Java 是于 1995 年由 Sun 公司推出的一种极富创造力的面向对象的程序设计语言，它是由有 Java 之父之称的 Sun 研究院院士詹姆斯·戈士林博士亲手设计而成的，并完成了 Java 技术的原始编译器和虚拟机。Java 最初的名字是 OAK，在 1995 年被重命名为 Java，正式发布。

Java 是一种通过解释方式来执行的语言，其语法规则和 C++ 类似。同时，Java 也是一种跨平台的程序设计语言。用 Java 语言编写的程序，可以运行在任何平台和设备上，如跨越 IBM 个人电脑、MAC 苹果计算机、各种微处理器硬件平台，以及 Windows、UNIX、OS/2、MAC OS 等系统平台，真正实现“一次编写，到处运行”。Java 非常适于企业网络和 Internet 环境，并且已成为 Internet 中最具有影响力、最受欢迎的编程语言之一。

与目前常用的 C++ 相比，Java 语言简洁得多，而且提高了可靠性，除去了最大的程序错误根源，此外它还有较高的安全性，可以说它是有史以来最为卓越的编程语言。

Java 语言编写的程序既是编译型的，又是解释型的。程序代码经过编译之后转换为一种称为 Java 字节码的中间语言，Java 虚拟机（JVM）将对字节码进行解释和运行。编译只进行一次，而解释在每次运行程序时都会进行。编译后的字节码采用一种针对 JVM 优化过的机器码形式保存，虚拟机将字节码解释为机器码，然后在计算机上运行。Java 语言程序代码的编译和运行过程如图 1.1 所示。

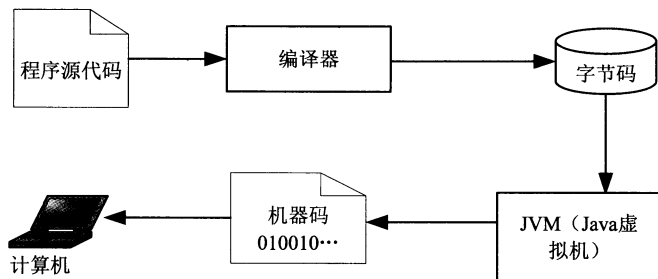


图 1.1 Java 程序的编译和运行过程

### 1.1.2 Java 的应用领域

借助 Java，程序开发人员可以自由地使用现有的硬件和软件系统平台。这是因为 Java 是独立于平台的，它还可以应用于计算机之外的领域，Java 程序可以在便携式计算机、电视、电话、手机和其他

的大量电子设备上运行。Java 的用途不胜枚举，它拥有无可比拟的能力，其节省的时间和费用也十分可观。Java 的应用领域主要有以下方面：

- ☒ 桌面应用系统开发。
- ☒ 嵌入式系统开发。
- ☒ 电子商务应用。
- ☒ 企业级应用开发。
- ☒ 交互式系统开发。
- ☒ 多媒体系统开发。
- ☒ 分布式系统开发。
- ☒ Web 应用系统开发。

Java 无处不在，它已经拥有几百万个用户，其发展速度要快于在它之前的任何一种计算机语言。Java 能够给企业和最终用户带来数不尽的好处。Oracle 公司董事长和首席执行官 Larru Ellison 说过：“Java 正在进入企业、家庭和学校。它正在像 Internet 本身一样成为普遍存在的技术。”

如果仔细观察，就会发现 Java 就在我们身边，如我们经常使用的 Java 开发工具 Eclipse、NetBeans、JBuilder 等，另外还有 RSSOwl、Limewire、Azureus、CyberDuck、OpenOffice 等优秀软件都是使用 Java 编写的。此外，各手机厂商都为自己的产品提供了 Java 技术的支持，各种手机上的 Java 程序和游戏已经数不胜数。

### 1.1.3 Java 的版本

自从 Sun 公司推出 Java 以来，就力图使之无所不能。Java 发展至今，按应用范围分为 3 个版本，即 Java SE、Java EE 和 Java ME，也就是 Sun ONE（Open Net Environment）体系。本节将对 Java 的这 3 个版本分别进行介绍。

#### 1. Java SE

Java SE 是 Java 的标准版，主要用于桌面应用程序的开发，同时也是 Java 的基础，它包含 Java 语言基础、JDBC（Java 数据库连接性）操作、I/O（输入/输出）、网络通信、多线程等技术。Java SE 的结构如图 1.2 所示。

#### 2. Java EE

Java EE 是 Java 的企业版，主要用于开发企业级分布式的网络程序，如电子商务网站和 ERP（企业资源规划）系统，其核心为 EJB（企业 Java 组件模型）。Java EE 的结构如图 1.3 所示。

#### 3. Java ME

Java ME 主要应用于嵌入式系统开发，如掌上电脑、手机等移动通信电子设备，现在大部分手机厂商所生产的手机都支持 Java 技术。Java ME 的结构如图 1.4 所示。





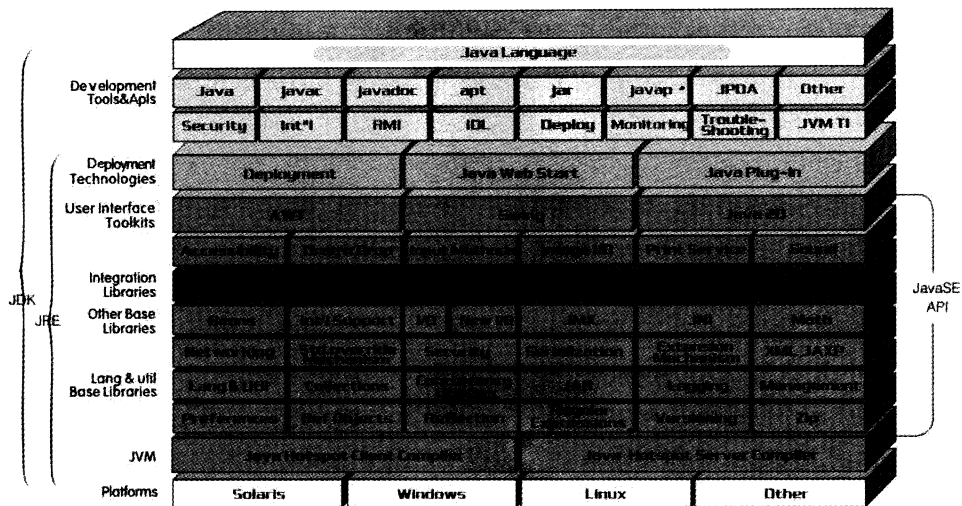


图 1.2 Java SE 的结构

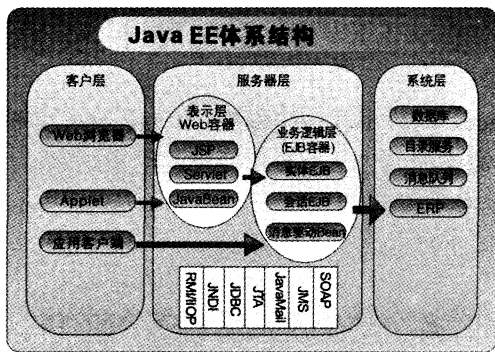


图 1.3 Java EE 的结构

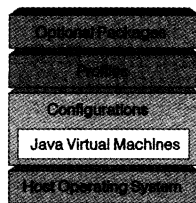


图 1.4 Java ME 的结构

### 1.1.4 怎样学好 Java

如何学好 Java 语言，是所有初学者需要共同面对的问题，其实每种语言的学习方法都大同小异，初学者需要注意的主要有以下几点：

- ☑ 明确自己的学习目标和大的方向，选择并锁定一门语言，按照自己的学习方向努力学习，认真研究。
- ☑ 初学者不用看太多的书，先找本相对基础的书系统地学习。很多程序开发人员工作了很久也只是熟悉部分基础而已，并没有系统地学习 Java 语言。
- ☑ 了解设计模式。开发程序必须编写程序代码，这些代码必须具有高度的可读性，这样编写的程序才有调试、维护和升级的价值。学习一些设计模式，能够更好地把握项目的整体结构。
- ☑ 不要死记语法。在刚接触一门语言，特别是 Java 语言时，掌握好基本语法，并大概了解一些功能即可。借助开发工具（如 Eclipse 或 NetBeans）的代码辅助功能，完成代码的录入，这样可以快速进入学习状态。

- ☑ 多实践，多思考，多请教。仅读懂书本中的内容和技术是不行的，必须动手编写程序代码，并运行程序、分析运行结构，从而对学习内容有整体的认识和肯定。用自己的方式思考问题、编写代码来提高编程思想。平时多请教老师或经理，和其他人多沟通技术问题，提高自己的技术和见识。
- ☑ 不要急躁。遇到技术问题，必须冷静对待，不要让自己思维混乱，保持清醒的头脑才能分析和解决各种问题。可以尝试用听歌、散步等方式来放松自己。
- ☑ 遇到问题，首先尝试自己解决，这样可以提高自己的程序调试能力，并对常见问题有一定的了解，明白出错的原因，甚至举一反三，解决其他关联的错误问题。
- ☑ 多查阅资料。可以经常到 Internet 上搜索相关资料或解决问题的方法，网络上已经摘录了很多人的问题和不同的解决方法，分析这些解决问题的方法，找出最适合自己的方法。
- ☑ 多阅读别人的源代码。不但要看懂别人的程序代码，还要分析编程者的编程思想和设计模式，并融为己用。

### 1.1.5 Java API 文档

API 的全称是 Application Programming Interface，即应用程序编程接口。Java API 文档是 Java 程序开发不可缺少的编程词典，它记录了 Java 语言中海量的 API，主要包括类的继承结构、成员变量和成员方法、构造方法、静态成员的详细说明和描述信息。可以在网站 <http://docs.oracle.com/javase/8/docs/api/index.html> 中找到 JDK 8 的 API 文档，页面效果如图 1.5 所示。

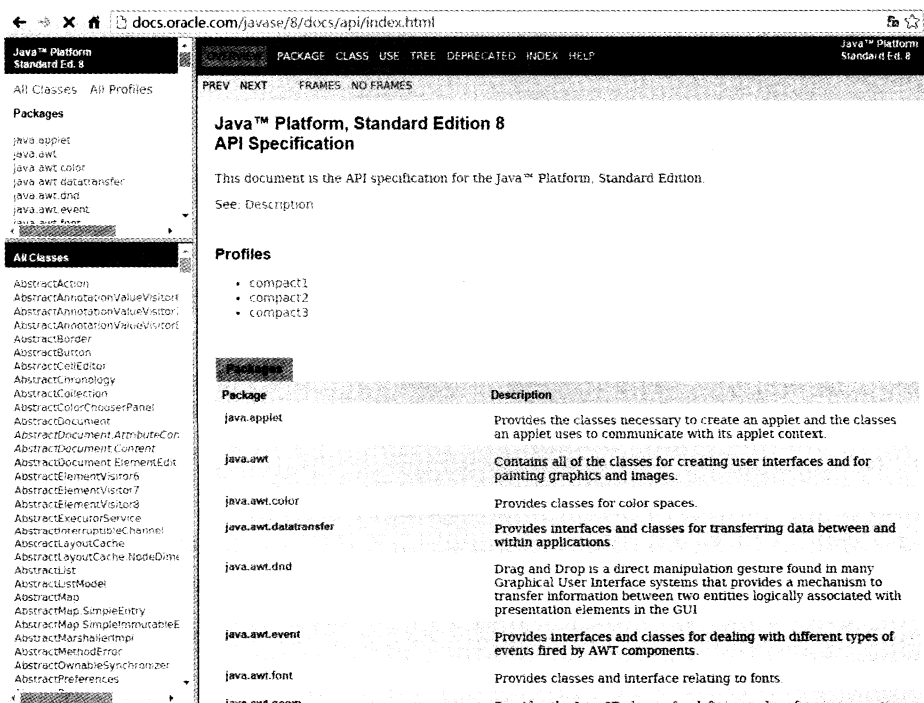


图 1.5 Java API 文档页面

## 1.2 Java 语言的特性

Java 语言的作者们编写了具有广泛影响的 Java 白皮书，里面详细地介绍了他们的设计目标以及实现成果，还用简短的篇幅介绍了 Java 语言的特性。下面将对这些特性进行扼要的介绍。

### 1.2.1 简单

Java 语言的语法简单明了，容易掌握，而且是纯面向对象的语言。Java 语言的简单性主要体现在以下几个方面：

- ☑ 语法规则和 C++ 类似。从某种意义上讲，Java 语言是由 C 和 C++ 语言转变而来的，所以 C 程序设计人员可以很容易地掌握 Java 语言的语法。
- ☑ Java 语言对 C++ 进行了简化和提高。例如，Java 使用接口取代了多重继承，并取消了指针，因为指针和多重继承通常使程序变得复杂。Java 语言还通过实现垃圾自动收集，大大简化了程序设计人员的资源释放管理工作。
- ☑ Java 提供了丰富的类库和 API 文档以及第三方开发包，另外还有大量的基于 Java 的开源项目，JDK（Java 开发者工具箱）已经开放源代码，读者可以通过分析项目的源代码，提高自己的编程水平。

### 1.2.2 面向对象

面向对象是 Java 语言的基础，也是 Java 语言的重要特性，它本身就是一种纯面向对象的程序设计语言。Java 提倡万物皆对象，语法中不能在类外面定义单独的数据和函数，也就是说，Java 语言最外部的数据类型是对象，所有的元素都要通过类和对象来访问。

### 1.2.3 分布性

Java 的分布性包括操作分布和数据分布，其中操作分布是指在多个不同的主机上布置相关操作，而数据分布是将数据分别存放在多个不同的主机上，这些主机是网络中的不同成员。Java 可以凭借 URL（统一资源定位符）对象访问网络对象，访问方式与访问本地系统相同。

### 1.2.4 可移植性

Java 程序具有与体系结构无关的特性，可以方便地移植到网络上的不同计算机中。同时，Java 的类库中也实现了针对不同平台的接口，使这些类库可以移植。



### 1.2.5 解释型

运行 Java 程序需要解释器。任何移植了 Java 解释器的计算机或其他设备都可以用 Java 字节码进行解释执行。字节码独立于平台，它本身携带了许多编译时的信息，使得连接过程更加简单，开发过程更加迅速，更具探索性。

### 1.2.6 安全性

Java 语言删除了类似 C 语言中的指针和内存释放等语法，有效地避免了非法操作内存。Java 程序代码要经过代码校验、指针校验等很多测试步骤才能够运行，所以未经允许的 Java 程序不可能出现损害系统平台的行为，而且使用 Java 可以编写防病毒和防修改的系统。

### 1.2.7 健壮性

Java 程序的设计目标之一，是编写多方面的、可靠的应用程序，Java 将检查程序在编译和运行时的错误，并消除错误。类型检查能帮助用户检查出许多在开发早期出现的错误。集成开发工具（如 Eclipse、NetBeans）的出现也使编译和运行 Java 程序更加容易。

### 1.2.8 多线程

多线程机制能够使应用程序在同一时间并行执行多项任务，而且相应的同步机制可以保证不同线程能够正确地共享数据。使用多线程，可以带来更好的交互能力和实时行为。

### 1.2.9 高性能

Java 编译后的字节码是在解释器中运行的，所以它的速度较多数交互式应用程序提高了很多。另外，字节码可以在程序运行时被翻译成特定平台的机器指令，从而进一步提高运行速度。

### 1.2.10 动态

Java 在很多方面比 C 和 C++ 更能够适应发展的环境，可以动态调整库中方法和增加变量，而客户端却不需要任何更改。在 Java 中进行动态调整是非常简单和直接的。



## 1.3 搭建 Java 环境

 视频讲解：光盘\TM\lx\1\搭建 Java 环境.mp4

“工欲善其事，必先利其器。”在学习 Java 语言之前，必须了解并搭建好它所需要的开发环境。要编译和执行 Java 程序，JDK（Java Developers Kits）是必备的。下面将具体介绍下载并安装 JDK 和配置环境变量的方法。

### 1.3.1 JDK 下载

Java 的 JDK 又称 Java SE（以前称 J2SE），是 Sun 公司的产品，由于 Sun 公司已经被 Oracle 公司收购，因此 JDK 可以在 Oracle 公司的官方网站 <http://www.oracle.com/index.html> 下载。



在 Java 6 出版之后，J2SE、J2EE 和 J2ME 正式更名，将名称中的 2 去掉，更名后分别为 Java SE、Java EE 和 Java ME。

下面以目前最新版本的 JDK 8 为例介绍下载 JDK 的方法，具体步骤如下：

（1）打开 IE 浏览器，输入网址 <http://www.oracle.com/index.html>，浏览 Oracle 官方主页。将光标移动到工具栏上的 Downloads 菜单项上，将显示下载列表下拉菜单，单击 Java SE 超链接，如图 1.6 所示。

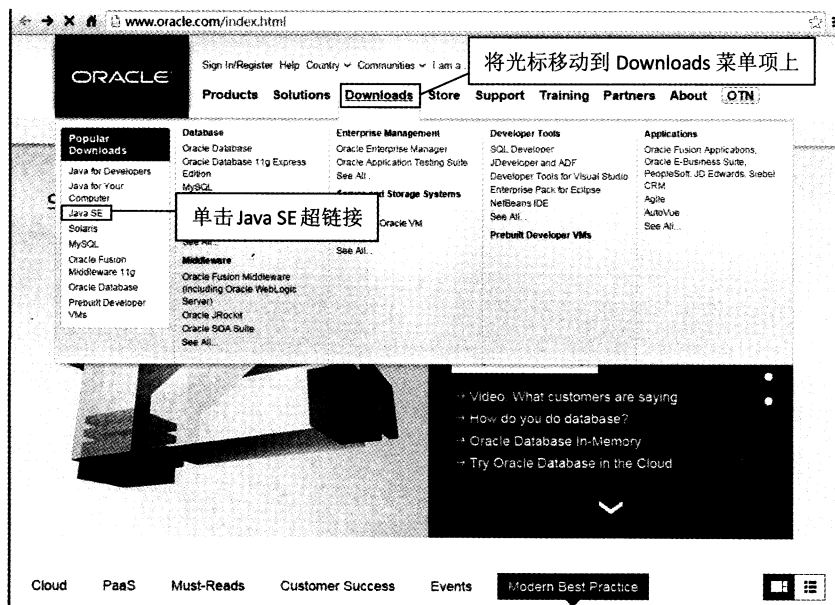


图 1.6 Oracle 主页

(2) 单击 Java SE 超链接后，将跳转到 JDK 的下载页面，在该页面中单击最新版本 JDK 的超链接，即如图 1.7 所示的 DOWNLOAD 按钮。在撰写本书时，最新的 JDK 版本为 JDK 8u65。

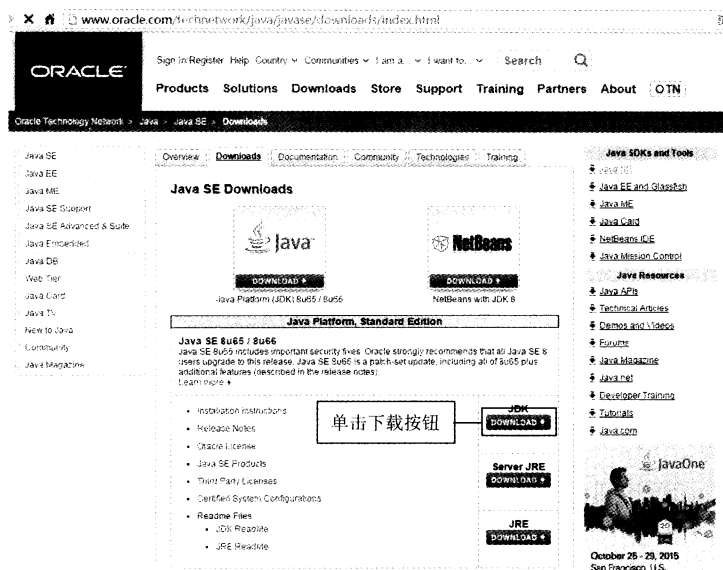


图 1.7 JDK 的下载页

(3) 在进入的新页面中，需要先选中同意协议的单选按钮，这时将显示如图 1.8 所示的页面，否则单击要下载的超链接时将不能进行下载。

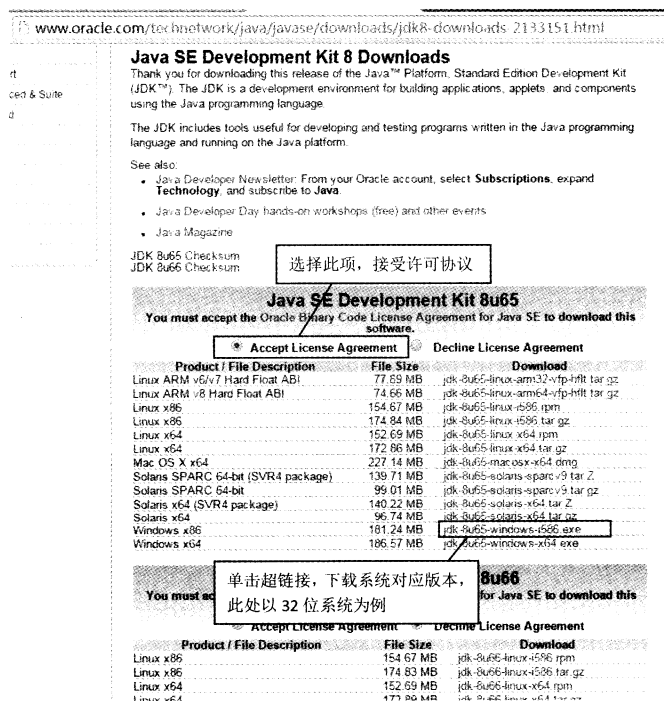


图 1.8 JDK 8u65 的下载列表

**注意**

下载时要选择适合自己操作系统平台的安装文件,如 Windows 系统平台是无法运行 Linux 系统平台的安装文件的。

(4)在下载列表中,可以根据计算机硬件和系统选择适当的版本进行下载。如果是 32 位的 Windows 操作系统,那么需要下载 jdk-8u65-windows-i586.exe 文件,直接在页面单击该文件的超链接即可。

## 1.3.2 Windows 系统的 JDK 环境

### 1. JDK 安装

下载 Windows 平台的 JDK 安装文件 jdk-8u65-windows-i586.exe 后即可安装,步骤如下:

- (1) 双击刚刚下载的安装文件,将弹出欢迎对话框,单击“下一步”按钮,如图 1.9 所示。
- (2) 在弹出的对话框中,可以选择安装的功能组件,这里选择默认设置,如图 1.10 所示。

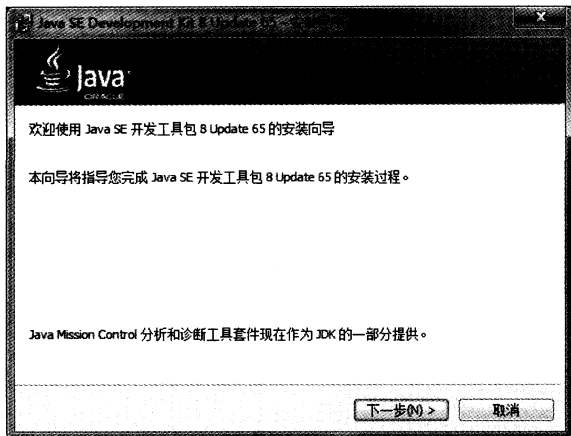


图 1.9 欢迎对话框

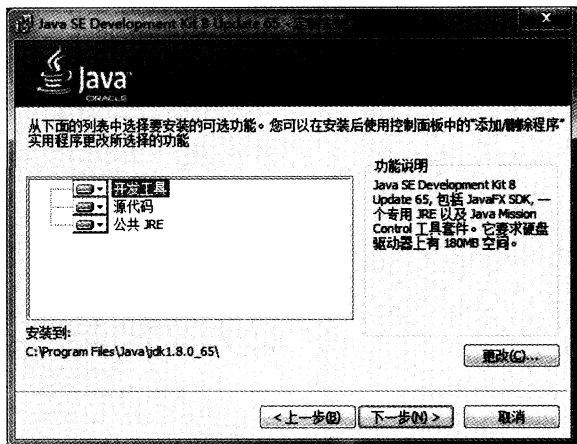


图 1.10 “定制安装”对话框

(3) 单击“更改”按钮,将弹出“更改文件夹”对话框,在该对话框中将 JDK 的安装路径更改为 C:\Java\jdk1.8.0\_65\,如图 1.11 所示,单击“确定”按钮,将返回到“定制安装”对话框中。

(4) 单击“下一步”按钮,开始安装 JDK。在安装过程中会弹出 JRE 的“目标文件夹”对话框,这里更改 JRE 的安装路径为 C:\Java\jre8\ (若此路径没有,可手动添加),然后单击“下一步”按钮,安装向导会继续完成安装进程。

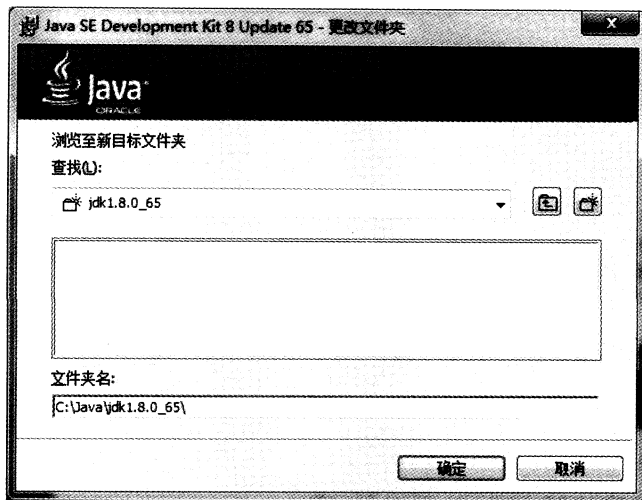


图 1.11 更改 JDK 的安装路径对话框

**说明**

JRE 全称为 Java Runtime Environment, 它是 Java 运行环境, 主要负责 Java 程序的运行, 而 JDK 包含了 Java 程序开发所需要的编译、调试等工具, 另外还包含了 JDK 的源代码。

(5) 安装完成后, 将弹出如图 1.12 所示的对话框, 单击“关闭”按钮即可。

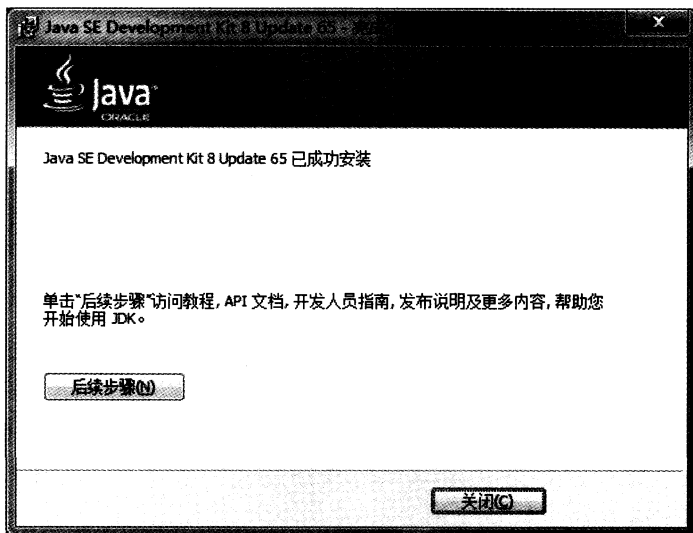


图 1.12 “完成”对话框

**说明**

JavaFX 2.0 由 Oracle 公司宣布推出, 是一款为企业业务应用提供的先进 Java 用户界面 (UI) 平台, 它能帮助开发人员无缝地实现与本地 Java 功能及 Web 技术动态能力的混合与匹配。



## 2. 在 Windows 7 系统中配置环境变量

在 Windows 7 系统中配置环境变量的步骤如下：

(1) 在“计算机”图标上右击，在弹出的快捷菜单中选择“属性”命令，将弹出“属性”对话框，在其左侧单击“高级系统设置”超链接，将打开如图 1.13 所示的“系统属性”对话框。

(2) 单击“环境变量”按钮，将弹出“环境变量”对话框，如图 1.14 所示，单击“系统变量”栏下的“新建”按钮，创建新的系统变量。

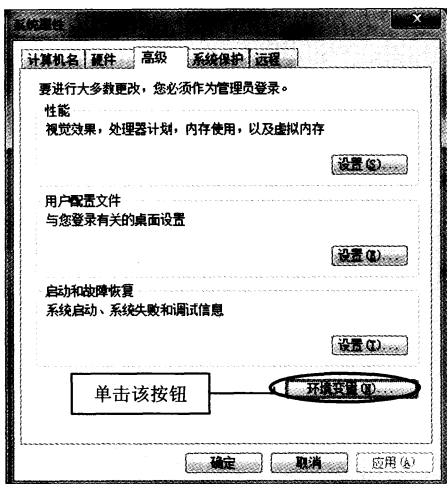


图 1.13 “系统属性”对话框

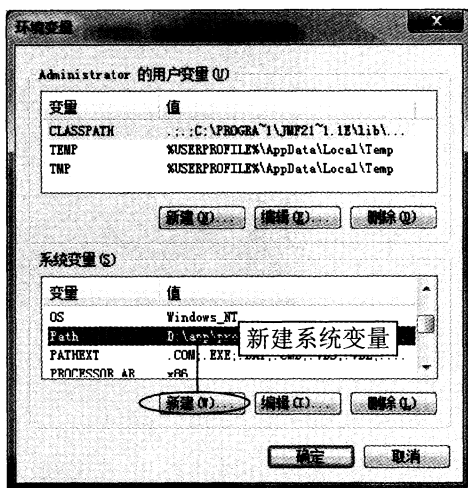


图 1.14 “环境变量”对话框

(3) 弹出“新建系统变量”对话框，分别输入变量名“JAVA\_HOME”和变量值（即 JDK 的安装路径），其中变量值是笔者的 JDK 安装路径，读者需要根据自己的计算机环境进行修改，如图 1.15 所示。单击“确定”按钮，关闭“新建系统变量”对话框。

(4) 在图 1.14 所示的“环境变量”对话框中双击 Path 变量对其进行修改，在原变量值最前端添加“.;%JAVA\_HOME%\bin;%JAVA\_HOME%\jre\bin;”变量值（注意，最后的“;”不要丢掉，它用于分隔不同的变量值），如图 1.16 所示。单击“确定”按钮完成环境变量的设置。

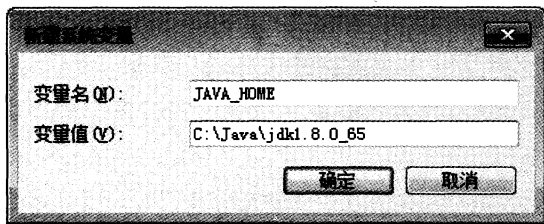


图 1.15 “新建系统变量”对话框

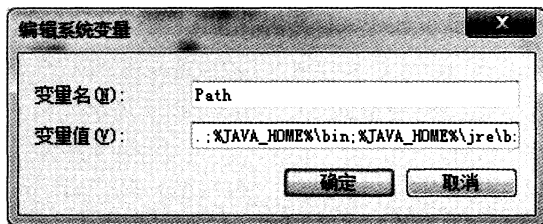


图 1.16 设置 Path 环境变量值

(5) JDK 安装成功之后必须确认环境配置是否正确。在 Windows 系统中测试 JDK 环境需要选择“开始”/“运行”命令（没有“运行”命令可以按 Windows+R 快捷键），然后在“运行”对话框中输入“cmd”并单击“确定”按钮启动控制台。在控制台中输入 javac 命令，按 Enter 键，将输出如图 1.17 所示的 JDK 的编译器信息，其中包括修改命令的语法和参数选项等信息。这说明 JDK 环境搭建成功。

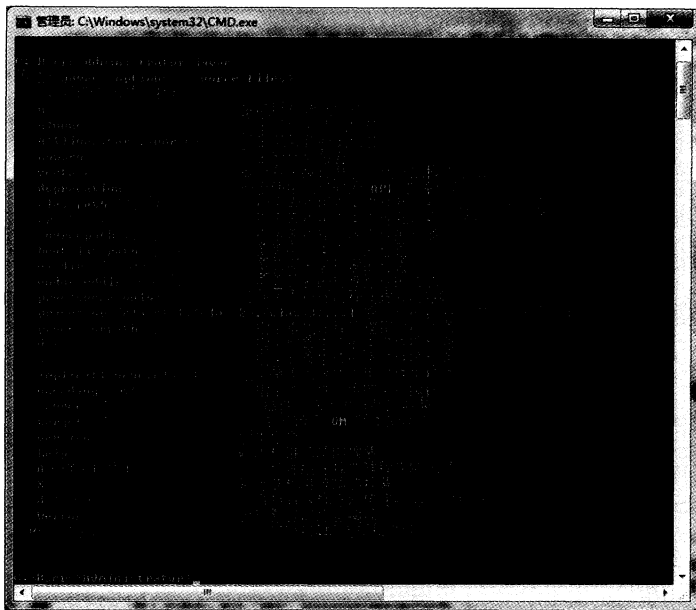


图 1.17 JDK 的编译器信息

## 1.4 第一个 Java 程序

 视频讲解：光盘\TM\1\1\第一个 Java 程序.mp4

编写 Java 应用程序，可以使用任何一个文本编辑器来编写程序的源代码，然后使用 JDK 搭配的工具进行编译和运行。现在流行的开发工具可以自动完成 Java 程序的编译和运行，还带有代码辅助功能，可以提示完整的语法代码，但是大型的 IDE 开发工具需要的系统资源较大，在开发一个简单的程序时，还是原始的文本编辑器速度较快。本节将介绍使用文本编辑器开发一个简单 Java 程序的步骤。

**【例 1.1】** 下面编写本书的第一个 Java 程序，它在屏幕（也称控制台）上输出“Hello Java”信息。程序编写步骤如下：（实例位置：光盘\TM\1\1.01）

（1）使用文本编辑器编写 Java 程序代码的过程和平时编写文本文件是一样的，只需注意 Java 语法格式和编码规则即可。选择“开始”/“所有程序”/“附件”/“记事本”命令，在记事本程序中输入以下代码：

```
public class HelloJava {  
    public static void main(String[] args) {  
        System.out.println("Hello Java");  
    }  
}
```

（2）选择“文件”/“保存”命令，选择存储位置为 C 盘根目录，在输入文件名称时，使用英文双引号（"）把文件名称包含起来，如 "HelloJava.java"。这样可以防止记事本程序为文件自动添加.txt 扩展名。

(3) Java 源程序需要编译成字节码才能够被 JVM 识别, 需要使用 JDK 的 `javac.exe` 命令。假设 `HelloJava.java` 文件保存在 C 盘, 选择“开始”/“运行”命令, 在“运行”对话框中输入“`cmd`”, 单击“确定”按钮, 启动控制台, 在控制台中输入 `cd\` 命令将当前位置切换到 C 盘根目录, 输入 `javac HelloJava.java` 命令编译源程序。源程序被编译后, 会在相同的位置生成相应的 `.class` 文件, 这是编译后的 Java 字节码文件。

### 注意

输入 `javac HelloJava.java` 命令时, 要注意 `javac` 和 `HelloJava.java` 之间有一个空格符。如果没有输入这个空格符, 将导致命令出错, 无法执行。

(4) 在控制台中输入 `java HelloJava` 命令将执行编译后的 `HelloJava.class` 字节码文件。编译与运行 Java 程序的步骤以及运行结果如图 1.18 所示。

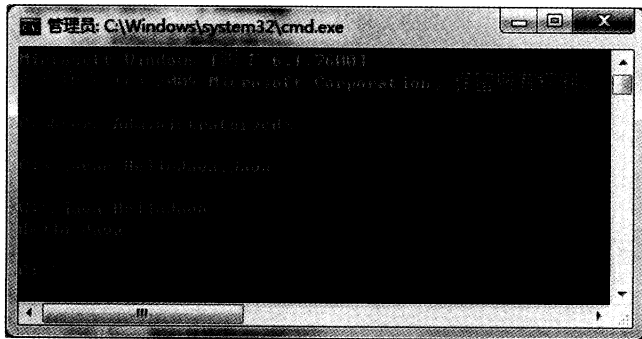


图 1.18 编译与运行 Java 程序的步骤以及运行结果

### 注意

使用 `Javac × × java` 命令编译 Java 源程序时, 如果没有弹出错误提示信息, 说明编译成功, 也许会提示一些警告信息, 但编译也能够通过; 但是如果出现 `Exception` 类的异常错误信息, 则说明源程序的代码有问题, 无法完成编译过程, 这时可以根据相应的 `Exception` 异常判断错误原因和代码位置, 解决代码错误。有关异常的知识将在本书的第 12 章中介绍。

## 1.5 小 结

本章简单介绍了 Java 语言及其相关特性, 另外还介绍了在 Windows 系统平台中搭建 Java 环境的方法, 以及编写 Java 程序的简单步骤。通过本章的学习, 读者应该能够了解什么是 Java 和 Java 的不同版本以及如何学习 Java 语言。搭建 Java 环境是本章的重点, 读者应该熟练掌握。

## 1.6 实践与练习

1. 尝试修改例 1.1, 使程序输出“这是我的第一个 Java 程序”。(答案位置: 光盘\TM\s\1.02)
2. 尝试编写 Java 程序, 使程序输出如图 1.19 所示的运行结果。(答案位置: 光盘\TM\s\1.03)

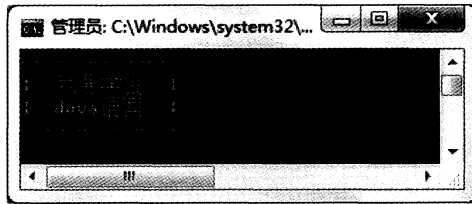


图 1.19 Java 程序运行结果

# 第 2 章

---

## 熟悉 Eclipse 开发工具

(  视频讲解：13 分钟 )

学习 Java 语言程序设计必须选择一个功能强大、使用简单、能够辅助程序设计的 IDE (集成开发工具)。Eclipse 是目前最流行的 Java 语言开发工具之一，它具有强大的代码辅助功能，可以帮助程序开发人员自动完成语法修正、补全文字、代码修正、API 提示等编码工作，大量节省程序开发所需的时间和精力。本章将简要介绍 Eclipse 开发工具，使读者能够初步了解 Eclipse 并使用它完成程序设计工作。

通过阅读本章，您可以：

- » 掌握安装 Eclipse 的国际化语言包
- » 学习 Eclipse 中经常使用的菜单和工具栏
- » 学习使用 Eclipse 编写程序代码
- » 掌握 Eclipse 中调试器的使用

## 2.1 熟悉 Eclipse

 视频讲解：光盘\TM\lx\2\熟悉 Eclipse.mp4

虽然使用记事本和 JDK 编译工具已经可以编写 Java 程序，但是在项目开发过程中必须使用大型的集成开发工具（IDE）来编写 Java 程序，这样可以避免编码错误，方便管理项目结构，而且使用 IDE 工具的代码辅助功能可以快速输入程序代码。本节将介绍 Eclipse 开发工具，包括它的安装、配置与启动、菜单栏、工具栏以及各种视图的作用等。

### 2.1.1 Eclipse 简介

Eclipse 是由 IBM 公司投资 4000 万美元开发的集成开发工具。它基于 Java 语言编写，并且是开放源代码的、可扩展的，也是目前最流行的 Java 集成开发工具之一。另外，IBM 公司捐出 Eclipse 源代码，组建了 Eclipse 联盟，由该联盟负责这种工具的后续开发。Eclipse 为编程人员提供了一流的 Java 程序开发环境，它的平台体系结构是在插件概念的基础上构建的，插件是 Eclipse 平台最具特色的特征之一，也是其区别于其他开发工具的特征之一。学习了本章之后，读者将对 Eclipse 有一个初步的了解，为后面深入学习做准备。



#### 说明

本书所使用的 Eclipse 开发工具的版本是 4.5，读者可以从 <http://www.eclipse.org> 网站下载。同时，该网站的 Babel 项目组提供了 Eclipse 4.5 的多国语言包，它能够在任何国家或地区以本地语言显示程序界面，其中包括菜单栏、工具栏、窗体和所有程序模块（不包括其他插件）。

### 2.1.2 下载 Eclipse

本节将介绍如何到 Eclipse 的官方网站下载本书所使用的 Eclipse 开发环境。掌握 Eclipse 的下载与使用，并不只是为了学习，以后工作中 Eclipse 也是程序开发的好帮手。事实上，Eclipse 已经成为使用最广泛、应用最多的 Java 开发工具，并且是由 Java 语言编写的。其下载步骤如下：

(1) 打开浏览器，在地址栏中输入“<http://www.eclipse.org>”后，按 Enter 键开始访问 Eclipse 的官方网站，该网站的首页面包含了下载超链接，如图 2.1 所示。单击页面上的 DOWNLOAD 菜单项进入到下载页面。

(2) Eclipse 下载页面中包含各种版本的 Eclipse 下载区域，其中第 3 个栏目是 Java 开发版的 Eclipse（它包含 Java IDE、CVS 客户端、XML 编辑器和 WindowBuilder 等），在每个栏目的右侧是各种平台的下载超链接，本书使用的是 32 位的 Windows 平台，所以单击 Windows 32 bit 超链接，如图 2.2 所示。

(3) 最后的 Eclipse 下载页面会根据客户端所在的地理位置，分配合理的下载镜像站点，用户只需在 Eclipse 下载页面中单击 DOWNLOAD 按钮即可下载 Eclipse 文件，如图 2.3 所示。

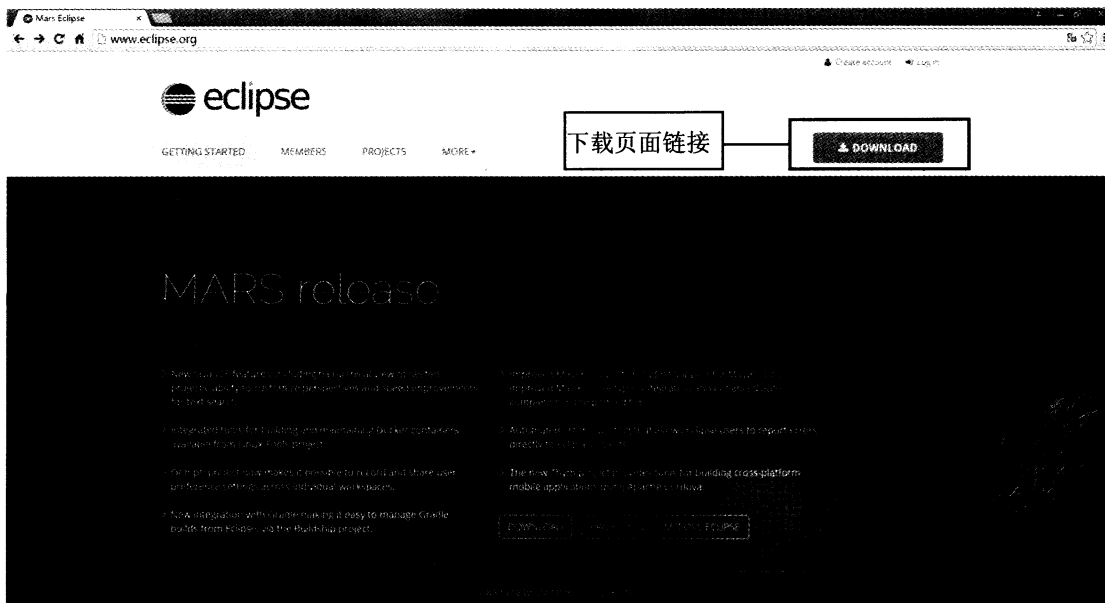


图 2.1 Eclipse 网站首页

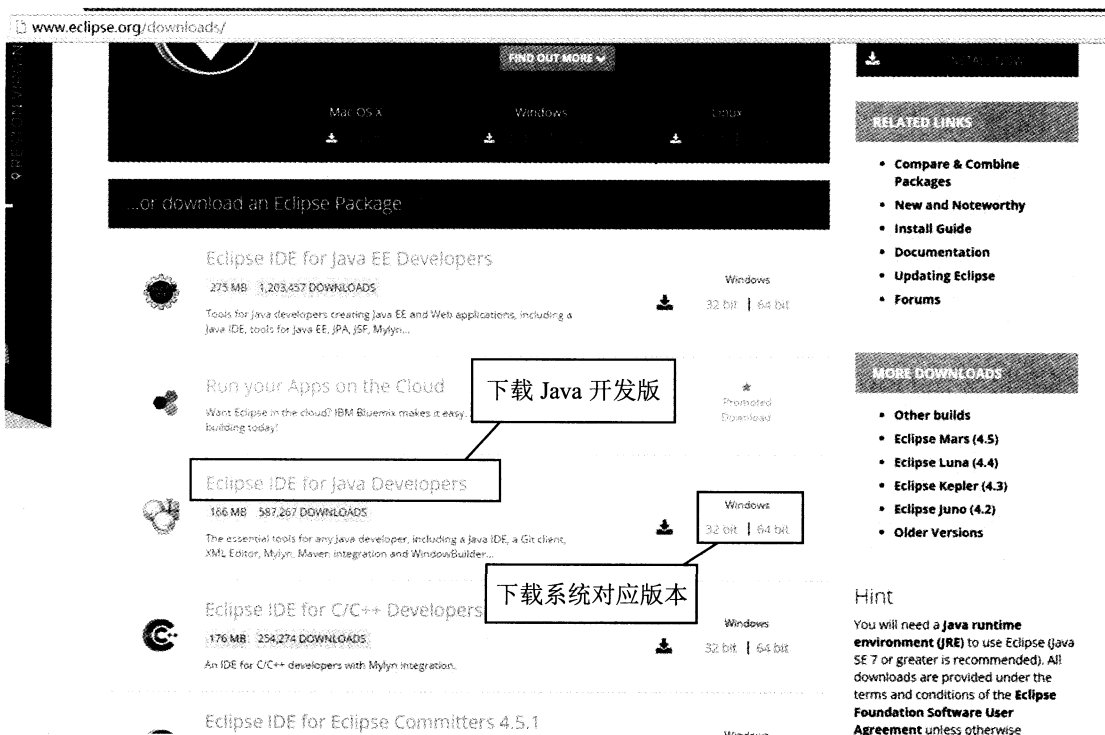


图 2.2 Eclipse 下载页面

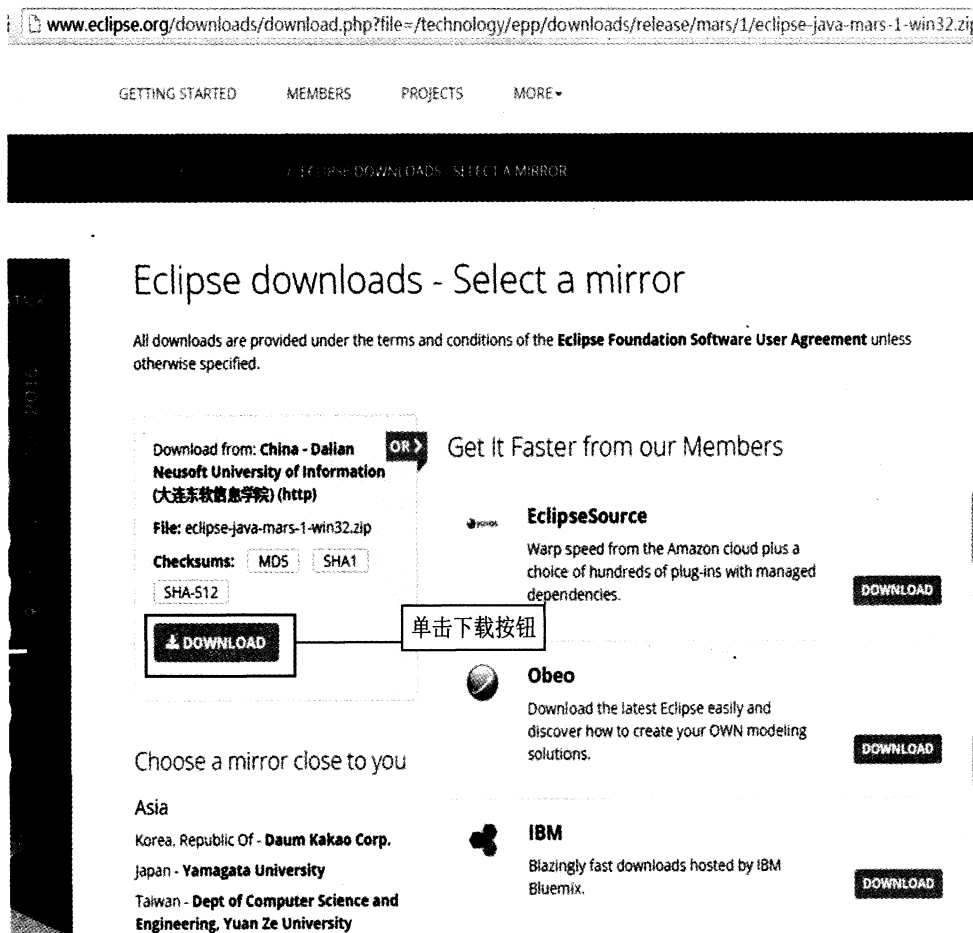


图 2.3 最后的 Eclipse 下载页面

### 2.1.3 安装 Eclipse 的中文语言包

从网站中下载的 Eclipse 安装文件是一个压缩包, 将其解压缩到指定的文件夹, 然后运行文件夹中的 Eclipse.exe 文件, 即可启动 Eclipse 开发工具。但是在启动 Eclipse 之前需要安装中文语言包, 以降低读者的学习难度。

Eclipse 的国际语言包可以到 <http://www.eclipse.org/babel> 下载, 具体的下载和使用步骤如下:

(1) 在浏览器的地址栏中输入 <http://www.eclipse.org/babel>, 并按 Enter 键, 进入如图 2.4 所示的 Babel 项目组首页。



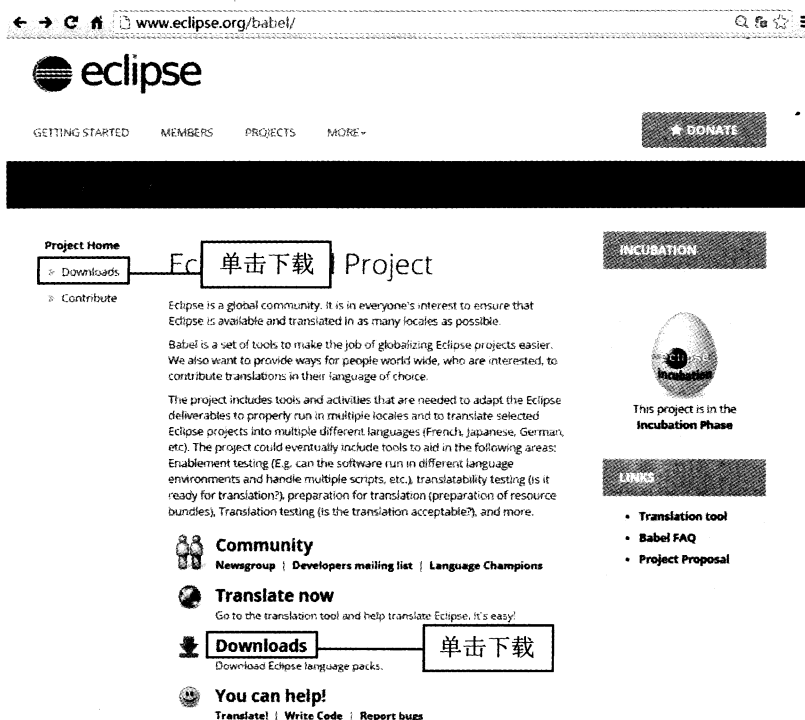


图 2.4 Babel 项目组首页

(2) 单击页面左侧导航中的 Downloads 超链接或者单击页面下方的绿色箭头都可以进入到语言包的下载页面。



### 注意

下载 Eclipse 多国语言包时，一定要注意语言包所匹配的 Eclipse 版本，否则可能无法实现 Eclipse 的国际化。

(3) 在下载页面的 Babel Language Packs 标题下选择对应 Eclipse 版本的超链接下载语言包，Eclipse 4.5 的名称是 Mars，所以单击该超链接，如图 2.5 所示，进入 Eclipse 4.5 的 Babel 语言包下载页面，该页面中包含了对应各国语言的资源包，而每个语言的资源包又按插件与功能模块分为多个 zip 压缩包。



- Open the install wizard with 'Help' > 'Install new software...'
- Add the Babel p2 repository: <http://download.eclipse.org/technology/babel/update-site/R0.13.0/mars>
- Select/install your language pack of choice
- Restart Eclipse and you should get a translated Eclipse

## Using the language packs

If the system locale for your operating system is already in the language you want to launch Eclipse in, all you have to do is to launch Eclipse normally and the language packs will be loaded.

If the system locale is in English, and you want to launch Eclipse in another language, you have to launch Eclipse with the `-nl` command line argument:

```
eclipse.exe -nl fr
```

Replace `fr` with the locale you wish to use. See "[Running Eclipse](#)" for more details. Also see the "[Teach Your Eclipse to Speak the Local Lingo](#)" Eclipse Corner Article for more examples on how to launch Eclipse in different languages (including bi-directional language like Arabic) and different operating systems (like Linux and Mac OS X).

## Babel Language Pack Zips and Update Sites - R0.13.0 (2015/08/04)

Babel Language Pack Zips
 

Mars

Luna

Kepler

选择合适的版本

Babel Language Pack Update Site for Mars  
<http://download.eclipse.org/technology/babel/update-site/R0.13.0/mars>  
 Zipped p2 repository for Mars (114 MB)

Babel Language Pack Update Site for Luna  
<http://download.eclipse.org/technology/babel/update-site/R0.13.0/luna>  
 Zipped p2 repository for Luna (108 MB)

Babel Language Pack Update Site for Kepler  
<http://download.eclipse.org/technology/babel/update-site/R0.13.0/kepler>  
 Zipped p2 repository for Kepler (135 MB)

## Important notes for Babel Language Packs R0.13.0

- Babel Language Packs for older release trains can be found at [Eclipse Babel Project Archived Downloads](#).
- Not all of the existing Eclipse Foundation Projects are included in Babel yet. If you encounter a project that you would like to help translate, **tell that project's leaders**.

图 2.5 Babel 语言包下载页面

(4) 在页面中找到简体中文的语言包分类, 如图 2.6 所示, 可以单独选择 Eclipse 的语言包下载, 也可以下载全部语言包。例如, 单独下载 Eclipse 的语言包, 可以单击 BabelLanguagePack-eclipse-zh\_4.5.0.v20151128060001.zip (87.61%)超链接, 下载后的文件名为 BabelLanguagePack-eclipse-zh\_4.5.0.v20151128060001.zip。

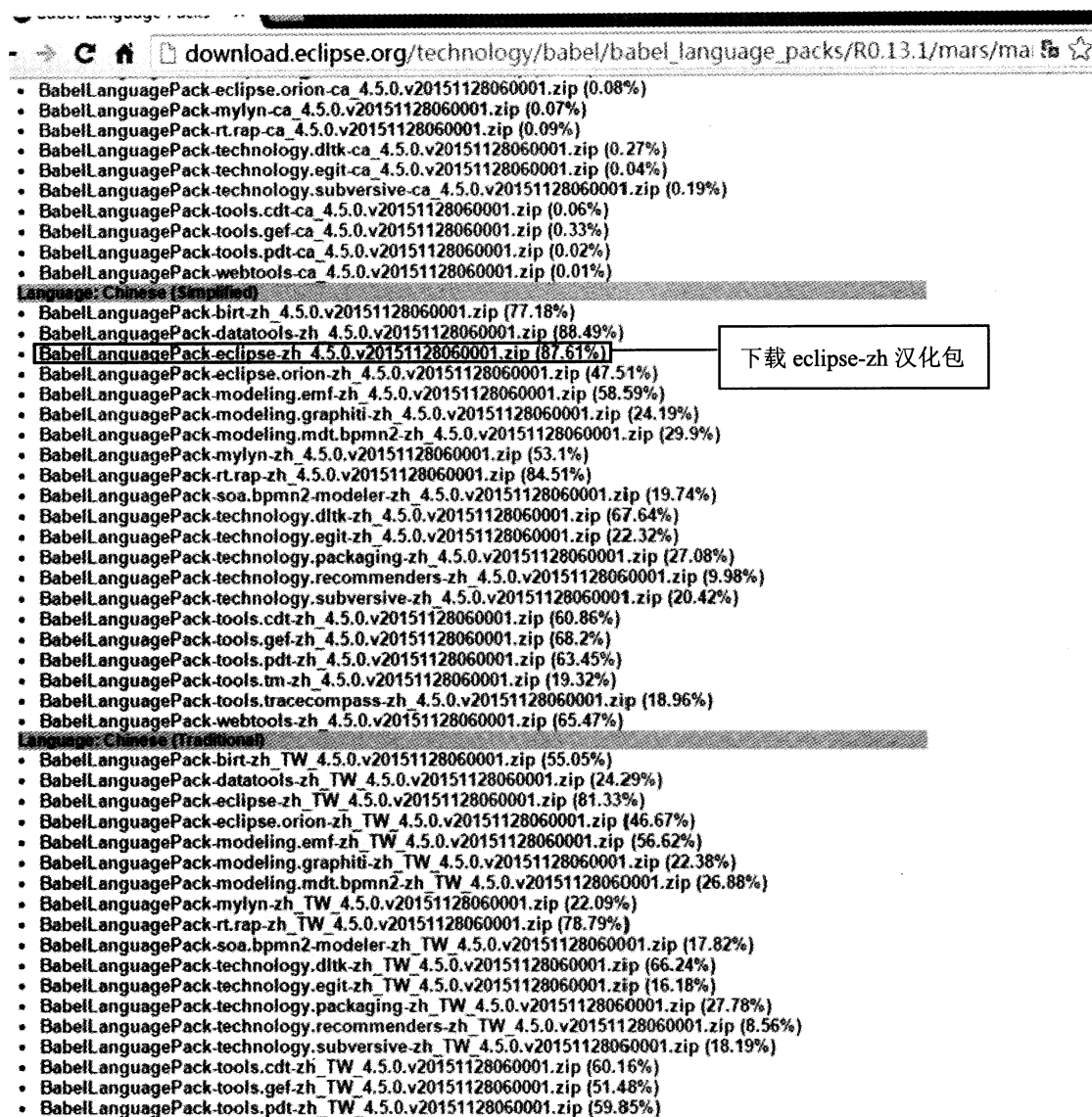


图 2.6 中文语言包下载分类

(5) 将下载的所有语言包解压缩并覆盖 Eclipse 文件夹中同名的两个文件夹 features 和 plugins, 这样在启动 Eclipse 时便会自动加载语言包。

## 2.1.4 Eclipse 的配置与启动

现在已经配置好 Eclipse 的多国语言包, 可以启动 Eclipse 了。在 Eclipse 的安装文件夹中运行 eclipse.exe 文件, 即开始启动 Eclipse, 将弹出“工作空间启动程序”对话框, 该对话框用于设置 Eclipse 的工作空间 (用于保存 Eclipse 建立的程序项目和相关设置)。本书的开发环境统一设置工作空间为

Eclipse 安装位置的 workspace 文件夹, 在“工作空间启动程序”对话框的“工作空间”文本框中输入“\workspace”, 单击“确定”按钮, 即可启动 Eclipse, 如图 2.7 所示。

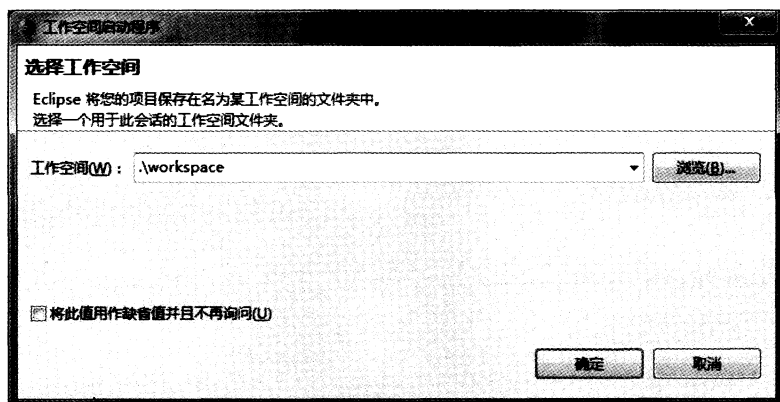


图 2.7 设置工作空间



#### 注意

每次启动 Eclipse 时都会出现设置工作空间的对话框, 通过选中“将此值用作缺省值并且不再询问”复选框可以设置默认工作空间, 这样 Eclipse 启动时就不会再询问工作空间的设置了。



#### 技巧

如果在启动 Eclipse 时, 选中“将此值用作缺省值并且不再询问”复选框, 设置不再询问工作空间设置后, 可以通过以下方法恢复提示。首先选择“窗口”/“首选项”命令, 打开“首选项”对话框, 然后在左侧选择“常规”/“启动和关闭”/“工作空间”节点, 并且选中右侧的“启动时提示工作空间”复选框, 单击“应用”按钮后, 再单击“确定”按钮即可。

Eclipse 首次启动时, 会显示 Eclipse 欢迎界面, 如图 2.8 所示。

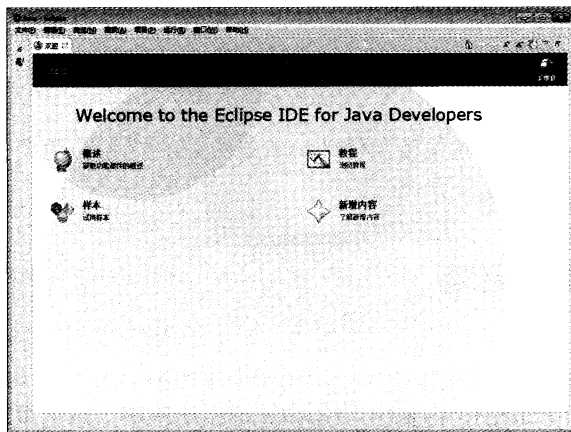


图 2.8 Eclipse 的欢迎界面

## 2.1.5 Eclipse 工作台

在 Eclipse 的欢迎界面中, 单击“工作台”(Workbench)按钮或关闭欢迎界面, 将显示 Eclipse 的工作台, 它是程序开发人员开发程序的主要场所。Eclipse 还可以将各种插件无缝地集成到工作台中, 也可以在工作台中开发各种插件。Eclipse 工作台主要包括标题栏、菜单栏、工具栏、编辑器、透视图和相关的视图等, 如图 2.9 所示。在接下来的章节中将介绍 Eclipse 的菜单栏与工具栏, 以及什么是透视图、视图, 并介绍常用视图。

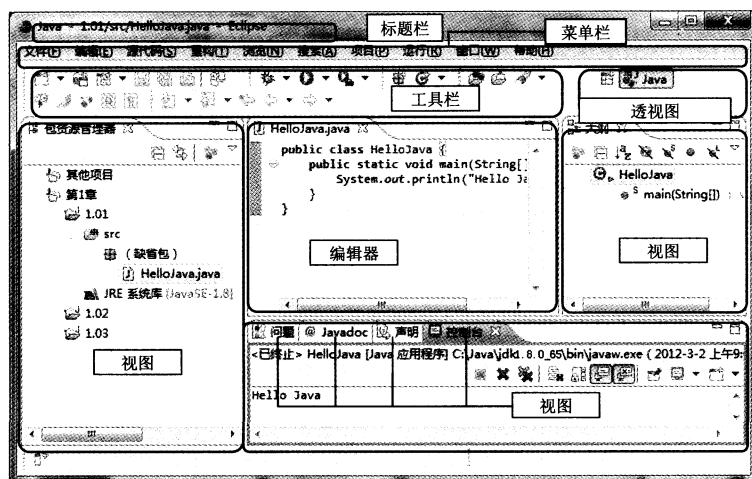


图 2.9 Eclipse 工作台

## 2.1.6 透视图与视图

透视图和视图是 Eclipse 中的概念, 本节将分别介绍透视图、视图及其在 Eclipse 中的作用。

### 1. 透视图

透视图是 Eclipse 工作台提供的附加组织层, 它实现多个视图的布局 and 可用操作的集合, 并为这个集合定义一个名称, 起到一个组织的作用。例如, Eclipse 提供的 Java 透视图组织了与 Java 程序设计有关的视图和操作的集合, 而“调试”透视图负责组织与程序调试有关的视图和操作集。在 Eclipse 的 Java 开发环境中提供了几种常用的透视图, 如 Java 透视图、“资源”透视图、“调试”透视图、“小组同步”透视图等。不同的透视图之间可以进行切换, 但是同一时刻只能使用一个透视图。

### 2. 视图

视图多用于浏览信息的层次结构和显示活动编辑器的属性, 例如, “控制台”视图用于显示程序运行时的输出信息和异常错误, 而“包资源管理器”视图可以浏览项目的文件组织结构。视图可以单独出现, 也可以与其他视图以选项卡样式叠加在一起, 它们可以有自己独立的菜单和工具栏, 并且可以通过拖动随意改变布局位置。



## 技巧

在视图标题上右击，在弹出的快捷菜单中选择“已拆离”命令可以将视图从 Eclipse 工作台界面中分离，以单独窗口存在，同样的操作也可以将视图再次合并到工作台界面中。本书大部分视图都是从 Eclipse 中分离后截取的界面效果。

## 2.1.7 菜单栏

Eclipse 的菜单栏包含了 Eclipse 的基本命令，在使用不同的编辑器时，还会动态地添加有关该编辑器的菜单。基本的菜单栏中除了常用的“文件”“编辑”“窗口”“帮助”等菜单以外，还提供了一些功能菜单，如“源代码”和“重构”等，如图 2.10 所示。

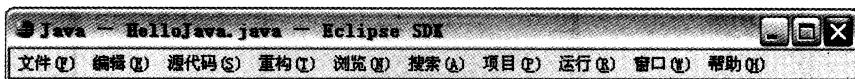


图 2.10 Eclipse 的菜单栏

每个菜单中都包含不同的命令，这些命令用于完成最终的操作，如文件的打开与保存、代码格式化、程序的运行与分步调试等。每个菜单所包含的命令如图 2.11 所示。



图 2.11 Eclipse 菜单命令

菜单中的命令虽多，但不是所有命令都经常使用。本节将介绍几个最常用的菜单及其命令，其他不常用的菜单，读者可以在日后程序开发过程中慢慢掌握。

### 1. “文件”菜单

“文件”菜单中包含“新建”“保存”“关闭”“打印”“切换工作空间”“属性”等命令。菜单中包含的内容虽多，但也有常用的和不常用的，例如，如果不常使用打印功能，那么“文件”菜单中的“打印”命令会很少用到。“文件”菜单中的常用命令列于表 2.1 中，这样更方便阅读和查询。

表 2.1 “文件”菜单中常用命令

| 图 标                                                                               | 命 令    | 说 明                         | 快 捷 键        |
|-----------------------------------------------------------------------------------|--------|-----------------------------|--------------|
|                                                                                   | 新建     | 创建新项目、元素或资源                 | Alt +Shift+N |
|                                                                                   | 打开文件   | 打开已经存在的文件                   |              |
|                                                                                   | 关闭     | 关闭当前编辑器                     | Ctrl+W       |
|                                                                                   | 全部关闭   | 关闭所有编辑器                     | Ctrl+Shift+W |
|  | 保存     | 保存当前编辑器的内容                  | Ctrl+S       |
|                                                                                   | 刷新     | 刷新所选元素的内容                   | F5           |
|                                                                                   | 切换工作空间 | 切换工作空间到其他位置，这将导致 Eclipse 重启 |              |
|  | 导入     | 打开导入向导对话框                   |              |
|  | 导出     | 打开导出向导对话框                   |              |
|                                                                                   | 属性     | 打开所选元素的属性对话框                | Alt+Enter    |

### 2. “编辑”菜单

“编辑”菜单用于辅助程序代码设计工作，除常用的“剪切”“复制”“粘贴”命令之外，还提供了“快速修正”“将选择范围扩展到”“内容辅助”等高级命令。“编辑”菜单中的常用命令如表 2.2 所示。

表 2.2 “编辑”菜单中常用命令

| 命 令      | 说 明                                       | 快 捷 键             |
|----------|-------------------------------------------|-------------------|
| 将选择范围扩展到 | 将选择编辑内容的范围扩大到外层元素、下一个元素、上一个元素或者恢复上一次选择的元素 |                   |
| 查找/替换    | 搜索编辑器中的内容片段，并根据需要替换为新的内容                  | Ctrl+F            |
| 查找下一个    | 搜索当前所选内容下一次出现的地方                          | Ctrl+K            |
| 查找上一个    | 搜索当前所选内容上一次出现的地方                          | Ctrl+Shift+K      |
| 添加书签     | 在当前光标所在行添加书签                              |                   |
| 添加任务     | 在当前光标所在行添加任务                              |                   |
| 灵活插入方式   | 切换插入方式。当禁用灵活插入方式时，将禁用自动缩进、添加右方括号等辅助功能     | Ctrl+Shift+Insert |
| 内容辅助     | 在当前光标位置打开内容辅助对话框                          |                   |
| 文字补全     | 补全当前编辑器中正在输入的文字                           | Ctrl+Alt+/        |
| 快速修正     | 如果光标位于问题代码附近，则打开一个解决方案对话框                 | Ctrl+I            |

### 3. “源代码”菜单

“源代码”菜单中包含的命令都是和代码编写相关的命令，主要用于辅助编程。“源代码”菜单的常用命令如表 2.3 所示。

表 2.3 “源代码”菜单中常用命令

| 命 令                     | 说 明                                         | 快 捷 键          |
|-------------------------|---------------------------------------------|----------------|
| 切换注释                    | 注释或取消注释当前选择的所有行                             | Ctrl+/或 Ctrl+7 |
| 添加块注释                   | 在当前选择的多行代码周围添加块注释                           | Ctrl+Shift+/   |
| 除去块注释                   | 从当前选择的多行代码中除去块注释                            | Ctrl+Shift+\   |
| 更正缩进                    | 更正当前选择的代码行的缩进                               | Ctrl+I         |
| 格式                      | 使用代码格式化程序来格式化当前 Java 代码                     | Ctrl+Shift+F   |
| 组织导入                    | 导入当前类所使用的类包                                 | Ctrl+Shift+O   |
| 覆盖/实现方法                 | 使用向导覆盖父类或实现接口中的方法                           |                |
| 生成 Getter 和 Setter      | 使用向导创建成员变量的 getXXX()/setXXX()方法             |                |
| 生成 hashCode()和 equals() | 打开“生成 hashCode()和 equals()”对话框              |                |
| 使用字段生成构造函数              | 添加构造函数，这些构造函数初始化当前所选类型的字段。可用于类型、字段或类型中的文本选择 |                |
| 从超类中生成构造函数              | 对于当前所选类型，按照超类中的定义来添加构造函数                    |                |
| 包围方式                    | 使用代码模板包围所选语句                                | Alt+Shift+Z    |
| 外部化字符串                  | 打开“将字符串外部化”向导，此向导允许通过使用语句访问属性文件来替换代码中的所有字符串 |                |

### 4. “重构”菜单

“重构”菜单是 Eclipse 最关键的菜单，主要包括项目重构的相关命令，应该重点掌握。“重构”菜单中的常用命令如表 2.4 所示。

表 2.4 “重构”菜单中常用命令

| 命 令        | 说 明                                        | 快 捷 键       |
|------------|--------------------------------------------|-------------|
| 重命名        | 重命名所选择的 Java 元素                            | Alt+Shift+R |
| 移动         | 移动所选择的 Java 元素                             | Alt+Shift+V |
| 抽取方法       | 创建一个包含当前所选语句或表达式的新方法，并相关地引用                | Alt+Shift+M |
| 抽取局部变量     | 创建为当前所选表达式指定的新变量，并将选择替换为对新变量的引用            | Alt+Shift+L |
| 抽取常量       | 从所选表达式创建静态终态字段并替换字段引用，并且可以选择重写同一表达式的其他出现位置 |             |
| 内联         | 直接插入局部变量、方法或常量                             | Alt+Shift+I |
| 将匿名类转换为嵌套类 | 将匿名内部类转换为成员类                               |             |
| 将成员类型转换为顶级 | 为所选成员类型创建新的 Java 编译单元，并根据需要更新所有引用          |             |
| 将局部变量转换为字段 | 如果该变量是在创建时初始化的，则此操作将把初始化移至新字段的声明或类的构造函数    |             |
| 抽取超类       | 从一组同类型中抽取公共超类                              |             |
| 抽取接口       | 根据当前类的方法创建接口，并使该类实现这个接口                    |             |



续表

| 命 令  | 说 明                               | 快 捷 键 |
|------|-----------------------------------|-------|
| 包括字段 | 将对变量的所有引用替换为 getXXX()/setXXX()方法  |       |
| 历史记录 | 浏览工作空间重构历史记录，并提供用于从重构历史记录中删除重构的选项 |       |

2.1.8 工具栏

和大多数软件的布局格式相同，Eclipse 的工具栏位于菜单栏的下方。工具栏中的按钮都是菜单命令对应的快捷图标，在打开不同的编辑器时，还会动态地添加与编辑器相关的新工具栏按钮。另外，除了菜单栏下面的主工具栏，Eclipse 中还有视图工具栏、透视图工具栏和快速视图工具栏等多种工具栏。

(1) 主工具栏

主工具栏就是位于 Eclipse 菜单栏下方的工具栏，其内容将根据不同的透视图和不同类型的编辑器显示相关工具按钮，如图 2.12 所示。



图 2.12 Eclipse 主工具栏

(2) 视图工具栏

Eclipse 界面中包含多种视图，这些视图有不同的用途（有关视图的概念已在 2.1.6 节中讲述），可以根据视图的功能需求在视图的标题栏位置添加相应的视图工具栏。例如，“控制台”视图用于输出程序运行中的输出结果和运行时异常信息，其工具栏如图 2.13 所示。

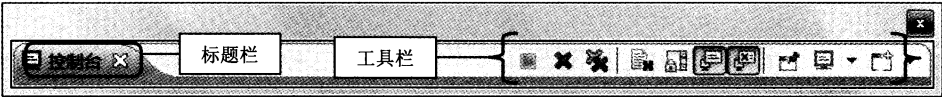


图 2.13 “控制台”视图的标题栏和工具栏

(3) 透视图工具栏

透视图工具栏主要包括切换已经打开的不同透视图的缩略按钮以及打开其他视图的按钮。在相应的工具按钮上右击会弹出透视图的管理菜单，实现透视图的定制、关闭、复位、布局位置、是否显示文本等操作，如图 2.14 所示。

(4) 快速视图工具栏

快速视图工具栏包含了一组快速视图的图标按钮，该工具栏中的视图会以图标按钮形式隐藏，单击指定工具按钮时会显示指定的视图，该视图失去焦点时会自动隐藏到快速工具栏。默认情况下，快速视图工具栏出现在工作台的左下角，如图 2.15 所示。



图 2.14 透视图工具栏



图 2.15 快速视图工具栏



### 2.1.9 “包资源管理器” 视图

“包资源管理器”视图用于浏览项目结构中的 Java 元素，包括包、类、类库的引用等，但最主要的用途还是操作项目中的源代码文件。“包资源管理器”视图的界面如图 2.16 所示。

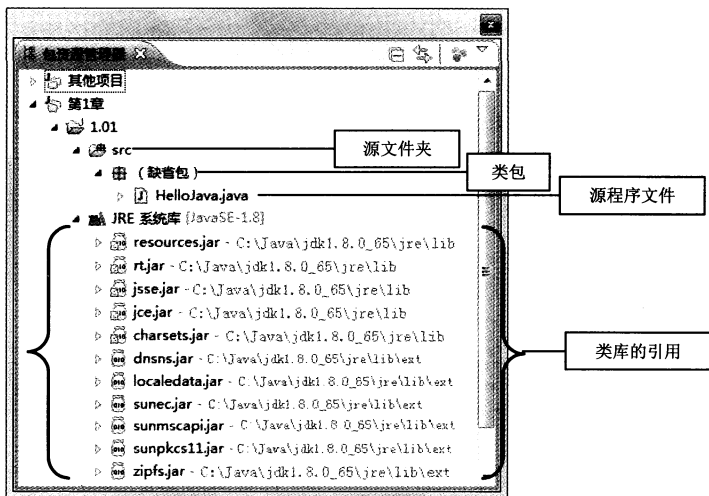


图 2.16 “包资源管理器” 视图

### 2.1.10 “控制台” 视图

“控制台”视图用于显示程序运行的输出结果和异常信息（Runtime Exception）。在学习 Swing 程序设计之前，必须使用控制台实现与程序的交互，例如，为方便调试某个方法，该视图在方法执行前后会分别输出“方法开始”和“方法结束”信息。“控制台”视图的界面如图 2.17 所示。

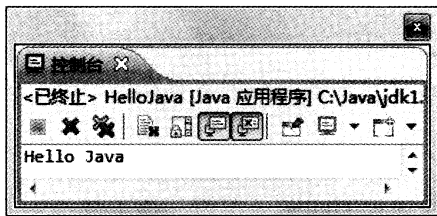


图 2.17 “控制台” 视图



#### 说明

英文原版的 Eclipse 开发环境中，控制台视图的标题叫做 Console。由于英文版的开发环境更加稳定，所以更推荐大家使用英文原版的 Eclipse 来学习 Java。

## 2.2 使用 Eclipse

 视频讲解：光盘\TM\lx\2\使用 Eclipse.mp4

现在读者对 Eclipse 工具已经有大体的认识了，本节将介绍如何使用 Eclipse 完成 HelloJava 程序的编写和运行。（实例位置：光盘\TM\sl\Hello.java）

### 2.2.1 创建 Java 项目

在 Eclipse 中编写程序，必须先创建项目。Eclipse 中有很多项目，其中 Java 项目用于管理和编写 Java 程序。创建该项目的步骤如下：

（1）选择“文件”/“新建”/“项目”命令，打开“新建项目”对话框，该对话框包含创建项目的向导，在向导中选择“Java 项目”节点，单击“下一步”按钮。

（2）弹出“新建 Java 项目”对话框，在“项目名”文本框中输入“HelloJava”，在“项目布局”栏中选中“为源文件和类文件创建单独的文件夹”单选按钮，如图 2.18 所示，然后单击“完成”按钮，完成项目的创建。

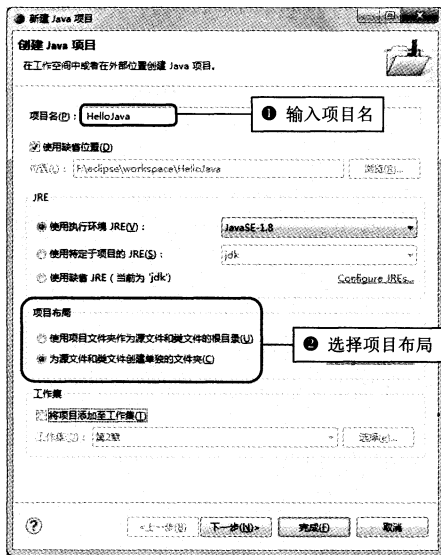


图 2.18 “新建 Java 项目”对话框

### 2.2.2 创建 Java 类文件

创建 Java 类文件时，会自动打开 Java 编辑器。创建 Java 类文件可以通过“新建 Java 类”向导来完成。在 Eclipse 菜单栏中选择“文件”/“新建”/“类”命令，将打开“新建 Java 类”向导对话框，

如图 2.19 所示。

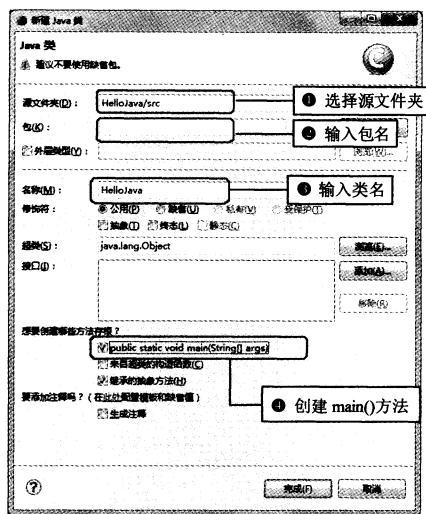


图 2.19 “新建 Java 类”向导对话框

使用该向导对话框创建 Java 类的步骤如下：

(1) 在“源文件夹”文本框中输入项目源程序文件夹的位置。通常向导会自动填写该文本框，没有特殊情况，不需要修改。

(2) 在“包”文本框中输入类文件的包名，这里暂时默认为空，不输入任何信息，这样就会使用 Java 工程的默认包。

(3) 在“名称”文本框中输入新建类的名称，如 HelloJava。



#### 注意

虽然 HelloJava 类名与 Java 项目同名，但是它们分别代表类文件和 Java 项目，读者需要注意区分它们的含义。

(4) 选中 `public static void main(String[] args)` 复选框，向导在创建类文件时，会自动为该添加 `main()` 方法，使该类成为可以运行的主类。

## 2.2.3 使用编辑器编写程序代码

编辑器总是位于 Eclipse 工作台的中间区域，该区域可以重叠放置多个编辑器。编辑器的类型可以不同，但它们的主要功能都是完成 Java 程序、XML 配置等代码编写或可视化设计工作。本节将介绍如何使用 Java 编辑器和其他代码辅助功能快速编写 Java 程序。

### 1. 打开 Java 编辑器

在使用向导创建 Java 类文件之后，会自动打开 Java 编辑器编辑新创建的 Java 类文件。除此之外，打开 Java 编辑器最常用的方法是在“包资源管理器”视图中双击 Java 源文件或在 Java 源文件处右击

并在弹出的快捷菜单中选择“打开方式”/“Java 编辑器”命令。Java 编辑器的界面如图 2.20 所示。

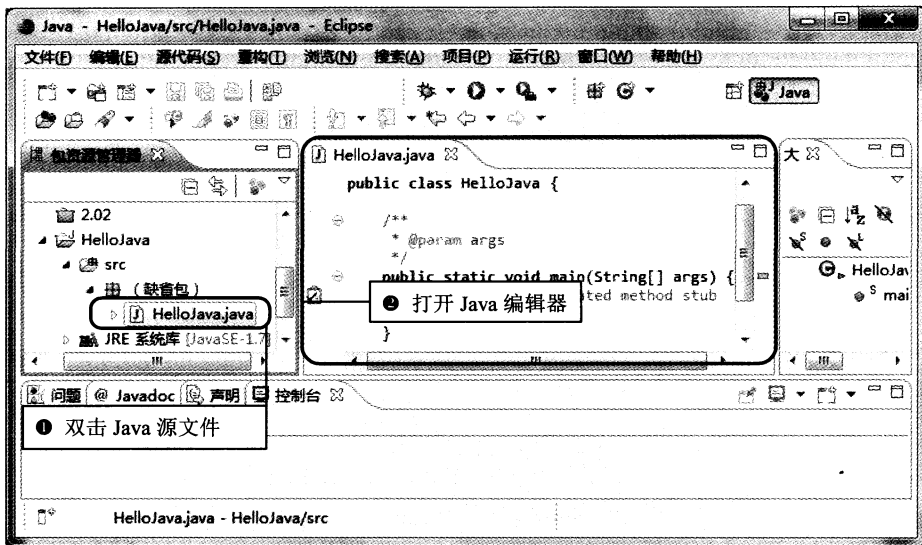


图 2.20 Java 编辑器界面

从图 2.20 中可以看到，Java 编辑器以不同的样式和颜色突出显示 Java 语法。这些突出显示的语法包括以下几个方面：

- ☒ 程序代码注释。
- ☒ Javadoc 注释。
- ☒ Java 关键字。



#### 技巧

在 Java 编辑器左侧右击，在弹出的快捷菜单中选择“显示行号”命令，可以开启 Java 编辑器显示行号的功能。

## 2. 编写 Java 代码

Eclipse 的强大之处并不在于编辑器能突出显示 Java 语法，而在于它强大的代码辅助功能。在编写 Java 程序代码时，可以使用 Ctrl+Alt+/快捷键自动补全 Java 关键字，也可以使用 Alt+/快捷键启动 Eclipse 代码辅助菜单。

在使用向导创建 HelloJava 类之后，向导会自动构建 HelloJava 类结构的部分代码，并建立 main() 方法，程序开发人员需要做的就是将代码补全，为程序添加相应的业务逻辑。本程序的完整代码如图 2.21 所示。

## 技巧

在 Eclipse 安装后, Java 编辑器文本字体为 Consolas 10。采用这个字体时, 中文显示比较小, 不方便查看。这时, 可以选择主菜单上的“窗口”/“首选项”命令, 打开“首选项”对话框, 在左侧的列表中选择“常规”/“外观”/“颜色和字体”节点, 在右侧选择 Java/“Java 编辑器文本字体”节点, 并单击“编辑”按钮, 在弹出的对话框中选择 Courier New 字体, 单击“确定”按钮, 返回到“首选项”对话框中, 单击“应用”按钮即可。另外, “调试”/“控制台字体”节点也需要进行以上修改。

在 HelloJava 程序代码中, 第 2、4、5、6 行是由向导创建的, 完成这个程序只要编写第 3 行和第 8 行代码即可。

首先来看一下第 3 行代码。它包括 private、static、String 3 个关键字, 这 3 个关键字如果在记事本程序中手动输入可能不会花多长时间, 但是无法避免出现输入错误的情况, 如将 private 关键字输入为“privat”, 缺少了字母“e”, 这个错误可能在程序编译时才会被发现。如果是名称更长、更复杂的关键字, 就更容易出现错误。而在 Eclipse 的 Java 编辑器中可以输入关键字的部分字母, 然后使用 Ctrl+Alt+/ 快捷键自动补全 Java 关键字, 如图 2.22 所示。

```

1
2 public class HelloJava {
3     private static String say = "我要学会你。";
4     /**
5      * @param args
6      */
7     public static void main(String[] args) {
8         System.out.println("你好 Java " + say);
9     }
10 }
11 }

```

图 2.21 HelloJava 程序代码

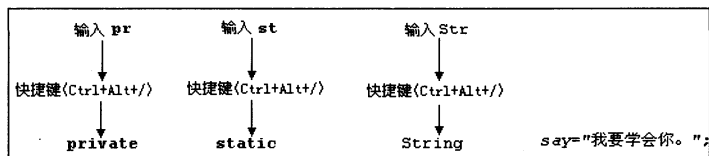


图 2.22 使用快捷键补全关键字

其次是第 8 行的程序代码。它使用 System.out.println() 方法输出文字信息到控制台, 这是程序开发时最常使用的方法之一。当输入“.”操作符时, 编辑器会自动弹出代码辅助菜单, 也可以在输入部分文字之后使用 Alt+/ 快捷键调出代码辅助菜单, 完成关键语法的输入, 如图 2.23 所示。

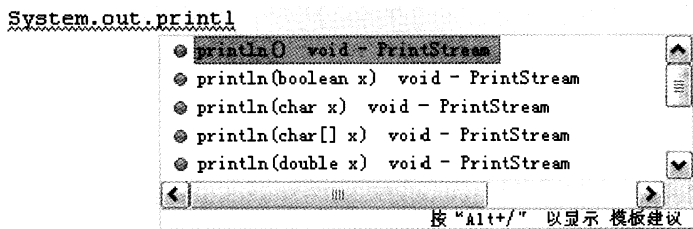


图 2.23 代码辅助菜单

## 注意

System.out.println() 方法在 Java 编辑器中可以通过输入“syso”和按 Alt+/ 快捷键完成快速输入。

**技巧**

将光标移动到 Java 编辑器的错误代码位置，按 Ctrl+1 快捷键可以激活“代码修正”菜单，进而选择菜单中合适的修正方法。

## 2.2.4 运行 Java 程序

HelloJava 类包含 main() 方法，它是一个可以运行的主类。例如，在 Eclipse 中运行 HelloJava 程序，可以在“包资源管理器”视图中 HelloJava 文件处右击，在弹出的菜单中选择“运行方式”/“ Java 应用程序”命令。程序运行结果如图 2.24 所示。

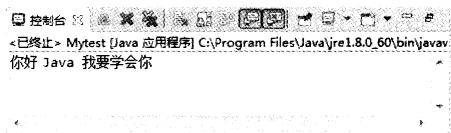


图 2.24 HelloJava 程序在控制台的输出结果

## 2.3 程序调试

### 视频讲解：光盘\TM\lx\2\程序调试.mp4

读者在程序开发过程中会不断体会到程序调试的重要性。为验证 Java 单元的运行状况，以往会在某个方法调用的开始和结束位置分别使用 System.out.println() 方法输出状态信息，并根据这些信息判断程序执行状况，但这种方法比较原始，而且经常导致程序代码混乱（导出的都是 System.out.println() 方法）。

本节将简单介绍 Eclipse 内置的 Java 调试器的使用方法，使用该调试器可以进行设置程序的断点，实现程序单步执行，在调试过程中查看变量和表达式的值等调试操作，这样可以避免在程序中编写大量的 System.out.println() 方法输出调试信息。

使用 Eclipse 的 Java 调试器需要设置程序断点，然后使用单步调试分别执行程序代码的每一行。

### 1. 断点

设置断点是程序调试中必不可少的手段，Java 调试器每次遇到程序断点时都会将当前线程挂起，即暂停当前程序的运行。

可以在 Java 编辑器中显示代码行号的位置双击添加或删除当前行的断点，或者在当前行号的位置右击，在弹出的快捷菜单中选择“切换断点”命令实现断点的添加与删除，如图 2.25 所示。

### 2. 以调试方式运行 Java 程序

要在 Eclipse 中调试 HelloJava 程序，可以在“包资源管理器”视图中 HelloJava 文件处右击，在弹出的快捷菜单中选择“调试方式”/“ Java 应用程序”命令。在图 2.25 中第 8 行代码设置了断点，调试器将在该断点处挂起当前线程，使程序暂停，如图 2.26 所示。

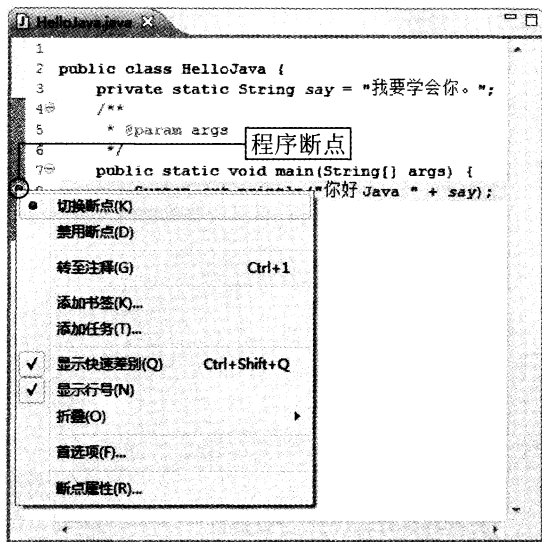


图 2.25 Java 编辑器中的断点

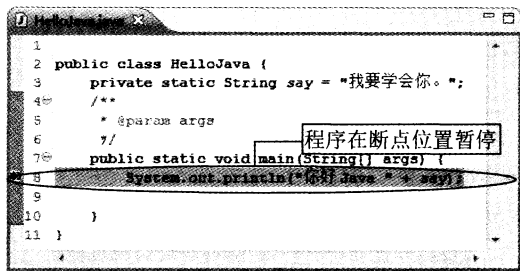


图 2.26 程序执行到断点后暂停

### 3. 程序调试

程序执行到断点被暂停后，可以通过“调试”视图工具栏上的按钮执行相应的调试操作，如运行、停止等。“调试”视图如图 2.27 所示。

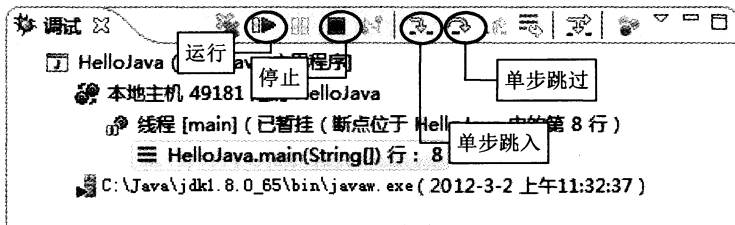


图 2.27 “调试”视图

#### (1) 单步跳过

在“调试”视图的工具栏中单击  按钮或按 F6 键，将执行单步跳过操作，即运行单独的一行程序代码，但是不进入调用方法的内部，然后跳到下一个可执行点并暂停线程。



#### 说明

不停地执行单步跳过操作，会每次执行一行程序代码，直到程序结束或等待用户操作。

#### (2) 单步跳入

在“调试”视图的工具栏中单击  按钮或按 F5 键，执行该操作将跳入调用方法或对象的内部单步执行程序并暂停线程。



## 2.4 小 结

本章主要介绍了 Eclipse 开发工具，包括它的配置与启动、菜单栏、工具栏，以及常用的“包资源管理器”和“控制台”视图；另外还介绍了 Eclipse 的使用方法，包括创建 Java 项目、创建 Java 类、编写程序代码和运行程序；最后还讲解了 Java 程序调试器的简单使用方法。

通过对本章的学习，读者应该能初步认识并掌握 Eclipse 开发工具的使用。在学习本书内容时，应灵活运用 Eclipse 的代码辅助功能（使用 Ctrl+Alt+/, Alt+/和 Ctrl+1 快捷键）快速编写程序代码。

## 2.5 实践与练习

1. 尝试在 Eclipse 中创建项目并编写程序，实现如图 2.28 所示的程序输出结果。（答案位置：光盘\TM\sl\2.01）

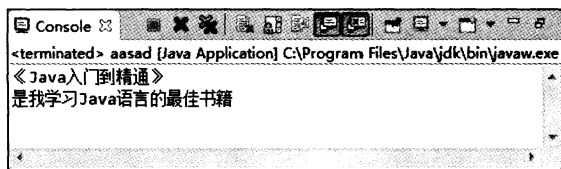


图 2.28 程序输出结果

2. 尝试编写 Java 程序，使程序分别输出两个整数的加、减、乘、除运算结果。（答案位置：光盘\TM\sl\2.02）

# 第 3 章

---

## Java 语言基础

(  视频讲解：1 小时 45 分钟 )

很多人认为学习 Java 语言之前必须要学习 C++ 语言，其实并非如此，产生这种错误的认识是因为很多人在学习 Java 语言之前都学过 C++ 语言。事实上，Java 语言比 C++ 语言更容易掌握。要掌握并熟练应用 Java 语言，就需要对 Java 语言的基础进行充分的了解。本章对 Java 语言基础进行了比较详细的介绍，初学者应该对本章的各个小节进行仔细的阅读、思考，这样才能达到事半功倍的效果。

通过阅读本章，您可以：

- » 了解 Java 主类结构
- » 了解 Java 语言中的基本数据类型
- » 理解 Java 语言中的常量与变量
- » 掌握 Java 语言运算符的使用
- » 理解 Java 语言数据类型的转换
- » 了解 Java 语言中的代码注释与编码规范

## 3.1 Java 主类结构

Java 语言是面向对象的程序设计语言，Java 程序的基本组成单元是类，类体中又包括属性与方法两部分（本书将在以下章节中逐一介绍）。每一个应用程序都必须包含一个 `main()` 方法，含有 `main()` 方法的类称为主类。下面通过程序来介绍 Java 主类结构。

**【例 3.1】** 在 Eclipse 下依次创建项目 `item`、包 `Number` 和类 `Frist`。在类体中输入以下代码，实现在控制台上输出“你好 Java”。（实例位置：光盘\TM\sl\3.01）

```
package Number;
public class Frist {
    static String s1 = "你好";
    public static void main(String[] args) {
        String s2 = "Java";
        System.out.println(s1);
        System.out.println(s2);
    }
}
```

运行结果如图 3.1 所示。

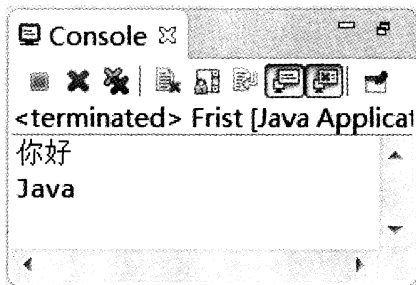


图 3.1 在控制台上输出字符串

### 注意

代码中的所有标点符号都是英文字符，不要在中文输入法状态下输入标点符号，如双引号和分号，否则会导致编译错误。

文件名必须和类名 `Frist` 同名，即 `Frist.java`。还要注意大小写，Java 是区分大小写的。

### 3.1.1 包声明

一个 Java 应用程序是由若干个类组成的。在例 3.1 中就是一个类名为 `Frist` 的类，语句 `package Number` 为声明该类所在的包，`package` 为包的关键字（关于包的详细介绍可参见第 11 章）。

### 3.1.2 声明成员变量和局部变量

通常将类的属性称为类的全局变量（成员变量），将方法中的属性称为局部变量。全局变量声明在类体中，局部变量声明在方法体中。全局变量和局部变量都有各自的应用范围。在例 3.1 中 s1 是成员变量，s2 是局部变量。

### 3.1.3 编写主方法

main()方法是类体中的主方法。该方法从“{”开始，至“}”结束。public、static 和 void 分别是 main()方法的权限修饰符、静态修饰符和返回值修饰符，Java 程序中的 main()方法必须声明为 public static void。String[] args 是一个字符串类型的数组，它是 main()方法的参数（以后章节中将作详细的介绍）。main()方法是程序开始执行的位置。

### 3.1.4 导入 API 类库

在 Java 语言中可以通过 import 关键字导入相关的类。在 JDK 的 API 中（应用程序接口）提供了 130 多个包，如 java.awt、java.io 等。可以通过 JDK 的 API 文档来查看这些类，其中主要包括类的继承结构、类的应用、成员变量表、构造方法表等，并对每个变量的使用目的作了详细的描述，API 文档是程序开发人员不可或缺的工具。

**注意**

Java 语言是严格区分大小写的。例如，不能将关键字 class 等同于 Class。

## 3.2 基本数据类型

视频讲解：光盘\TM\lx\3\基本数据类型.mp4

在 Java 中有 8 种基本数据类型来存储数值、字符和布尔值，如图 3.2 所示。

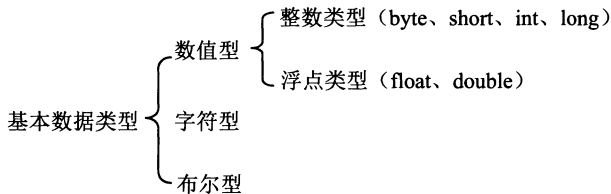


图 3.2 Java 基本数据类型

## 3.2.1 整数类型

整数类型用来存储整数数值，即没有小数部分的数值。可以是正数，也可以是负数。整型数据在 Java 程序中有 3 种表示形式，分别为十进制、八进制和十六进制。

☑ 十进制：十进制的表现形式大家都很熟悉，如 120、0、-127。



不能以 0 作为十进制数的开头（0 除外）。

☑ 八进制：如 0123（转换成十进制数为 83）、-0123（转换成十进制数为-83）。



八进制必须以 0 开头。

☑ 十六进制：如 0x25（转换成十进制数为 37）、0Xb01e（转换成十进制数为 45086）。



十六进制必须以 0X 或 0x 开头。

整型数据根据它所占内存大小的不同，可分为 byte、short、int 和 long 4 种类型。它们具有不同的取值范围，如表 3.1 所示。

表 3.1 整型数据类型

| 数据类型  | 内存空间（8 位等于 1 字节） | 取值范围                                     |
|-------|------------------|------------------------------------------|
| byte  | 8 位              | -128~127                                 |
| short | 16 位             | -32768~32767                             |
| int   | 32 位             | -2147483648~2147483647                   |
| long  | 64 位             | -9223372036854775808~9223372036854775807 |

下面以 int 型变量为例讲解整型变量的定义。

【例 3.2】定义 int 型变量，实例代码如下：

```
int x;           //定义 int 型变量 x
int x,y;         //定义 int 型变量 x、y
int x = 450,y = -462; //定义 int 型变量 x、y 并赋给初值
```

在定义以上 4 种类型变量时，要注意变量的取值范围，超出相应范围就会出错。对于 long 型值，若赋给的值大于 int 型的最大值或小于 int 型的最小值，则需要在数字后加 L 或 l，表示该数值为长整数，如 long num = 2147483650L。

【例 3.3】 在项目中创建类 Number，在主方法中创建不同数值型变量，并将这些变量相加，将和输出。（实例位置：光盘\TM\sl\3.02）

```
public class Number {                                //创建类
    public static void main(String[] args) {          //主方法
        byte mybyte = 124;                            //声明 byte 型变量并赋值
        short myshort = 32564;                        //声明 short 型变量并赋值
        int myint = 45784612;                         //声明 int 型变量并赋值
        long mylong = 46789451;                      //声明 long 型变量并赋值
        long result = mybyte + myshort + myint + mylong; //获得各数相加后的结果
        System.out.println("结果为: " + result);      //将以上变量相加的结果输出
    }
}
```

程序运行结果如图 3.3 所示。

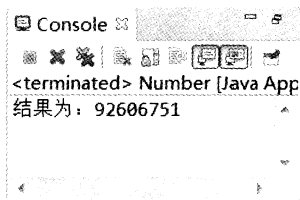


图 3.3 不同数值型变量相加的和

## 3.2.2 浮点类型

浮点类型表示有小数部分的数字。Java 语言中浮点类型分为单精度浮点类型（float）和双精度浮点类型（double），它们具有不同的取值范围，如表 3.2 所示。

表 3.2 浮点型数据类型

| 数 据 类 型 | 内存空间（8 位等于 1 字节） | 取 值 范 围                         |
|---------|------------------|---------------------------------|
| float   | 32 位             | 1.4E-45~3.4028235E38            |
| double  | 64 位             | 4.9E-324~1.7976931348623157E308 |

在默认情况下，小数都被看作 double 型，若使用 float 型小数，则需要在小数后面添加 F 或 f。可以使用后缀 d 或 D 来明确表明这是一个 double 类型数据，不加 d 不会出错，但声明 float 型变量时如果不加 f，系统会认为变量是 double 类型而出错。下面举例介绍声明浮点类型变量的方法。

【例 3.4】 定义浮点类型变量，实例代码如下：

```
float f1 = 13.23f;
double d1 = 4562.12d;
double d2 = 45678.1564;
```

### 3.2.3 字符类型

#### 1. char 型

字符类型 (char) 用于存储单个字符, 占用 16 位 (两个字节) 的内存空间。在定义字符型变量时, 要以单引号表示, 如 's' 表示一个字符, 而 "s" 则表示一个字符串, 虽然只有一个字符, 但由于使用双引号, 它仍然表示字符串, 而不是字符。

使用 char 关键字可定义字符变量, 下面举例说明。

【例 3.5】声明字符型变量, 实例代码如下:

```
char x = 'a';
```

由于字符 a 在 unicode 表中的排序位置是 97, 因此允许将上面的语句写成:

```
char x = 97;
```

同 C 和 C++ 语言一样, Java 语言也可以把字符作为整数对待。由于 unicode 编码采用无符号编码, 可以存储 65536 个字符 (0x0000~0xffff), 所以 Java 中的字符几乎可以处理所有国家的语言文字。若想得到一个 0~65536 之间的数所代表的 unicode 表中相应位置上的字符, 也必须使用 char 型显式转换。

【例 3.6】在项目中创建类 Gess, 编写如下代码, 实现将 unicode 表中某些位置上的字符以及一些字符在 unicode 表中的位置在控制台上输出。(实例位置: 光盘\TM\sl\3.03)

```
public class Gess { //定义类
    public static void main(String[] args) { //主方法
        char word = 'd', word2 = '@'; //定义 char 型变量
        int p = 23045, p2 = 45213; //定义 int 型变量
        System.out.println("d 在 unicode 表中的顺序位置是: " + (int) word);
        System.out.println("@ 在 unicode 表中的顺序位置是: " + (int) word2);
        System.out.println("unicode 表中的第 23045 位是: " + (char) p);
        System.out.println("unicode 表中的第 45213 位是: " + (char) p2);
    }
}
```

运行结果如图 3.4 所示。

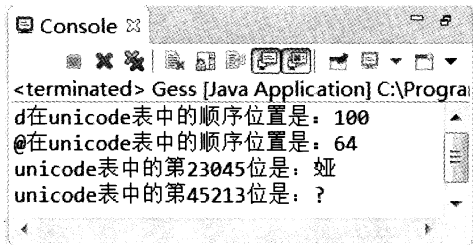


图 3.4 例 3.6 的运行结果

## 2. 转义字符

转义字符是一种特殊的字符变量，它以反斜杠“\”开头，后跟一个或多个字符。转义字符具有特定的含义，不同于字符原有的意义，故称“转义”。例如，`printf`函数的格式串中用到的“`\n`”就是一个转义字符，意思是“回车换行”。Java 中的转义字符如表 3.3 所示。

表 3.3 转义字符

| 转 义 字 符             | 含 义                                   |
|---------------------|---------------------------------------|
| <code>\ddd</code>   | 1~3 位八进制数据所表示的字符，如 <code>\123</code>  |
| <code>\uxxxx</code> | 4 位十六进制数据所表示的字符，如 <code>\u0052</code> |
| <code>'\'</code>    | 单引号字符                                 |
| <code>\\</code>     | 反斜杠字符                                 |
| <code>\t</code>     | 垂直制表符，将光标移到下一个制表符的位置                  |
| <code>\r</code>     | 回车                                    |
| <code>\n</code>     | 换行                                    |
| <code>\b</code>     | 退格                                    |
| <code>\f</code>     | 换页                                    |

将转义字符赋值给字符变量时，与字符常量值一样需要使用单引号。

【例 3.7】 使用转义字符，实例代码如下：

```
char c1 = '\\';           //将转义字符 '\\' 赋值给变量 c1
char char1 = '\u2605';    //将转义字符 '\u2605' 赋值给变量 char1
System.out.println(c1);   //输出结果\
System.out.println(char1); //输出结果★
```

## 3.2.4 布尔类型

布尔类型又称逻辑类型，通过关键字 `boolean` 来定义布尔类型变量，只有 `true` 和 `false` 两个值，分别代表布尔逻辑中的“真”和“假”。布尔值不能与整数类型进行转换。布尔类型通常被用在流程控制中作为判断条件。

【例 3.8】 声明 `boolean` 型变量，实例代码如下：

```
boolean b;               //定义布尔型变量 b
boolean b1,b2;           //定义布尔型变量 b1、b2
boolean b = true;        //定义布尔型变量 b，并赋给初值 true
```

## 3.3 变量与常量

 视频讲解：光盘\TM\lx\3\变量与常量.mp4

在程序执行过程中，其值不能被改变的量称为常量，其值能被改变的量称为变量。变量与常量的命名都必须使用合法的标识符。本节将向读者介绍标识符与关键字、变量与常量的命名。



### 3.3.1 标识符和关键字

#### 1. 标识符

标识符可以简单地理解为一个名字，用来标识类名、变量名、方法名、数组名、文件名的有效字符序列。

Java 语言规定标识符由任意顺序的字母、下划线（\_）、美元符号（\$）和数字组成，并且第一个字符不能是数字。标识符不能是 Java 中的保留关键字。

下面是合法标识符：

```
name
user_age
$page
```

下面是非法标识符：

```
4word
String
User name
```

在 Java 语言中标识符中的字母是严格区分大小写的，如 good 和 Good 是不同的两个标识符。Java 语言使用 unicode 标准字符集，最多可以标识 65535 个字符，因此，Java 语言中的字母不仅包括通常的拉丁文字 a、b、c 等，还包括汉字、日文以及其他许多语言中的文字。

#### 2. 关键字

关键字是 Java 语言中已经被赋予特定意义的一些单词，不可以把这些字作为标识符来使用。3.2 节介绍的数据类型中提到的 int、boolean 等都是关键字。Java 语言中的关键字如表 3.4 所示。

表 3.4 Java 关键字

|           |          |            |              |            |           |
|-----------|----------|------------|--------------|------------|-----------|
| int       | public   | this       | finally      | boolean    | abstract  |
| continue  | float    | long       | short        | throw      | throws    |
| return    | break    | for        | static       | new        | interface |
| if        | goto     | default    | byte         | do         | case      |
| strictfp  | package  | super      | void         | try        | switch    |
| else      | catch    | implements | private      | final      | class     |
| extends   | volatile | while      | synchronized | instanceof | char      |
| protected | import   | transient  | default      | double     |           |

### 3.3.2 声明变量

变量的使用是程序设计中一个十分重要的环节。定义变量就是要告诉编译器（compiler）这个变量的数据类型，这样编译器才知道需要配置多少空间给它，以及它能存放什么样的数据。在程序运行过程中，空间内的值是变化的，这个内存空间就称为变量。为了便于操作，给这个空间取个名字，称为

变量名。变量的命名必须是合法的标识符。内存空间内的值就是变量值。在声明变量时可以是没有赋值，也可以是直接赋给初值。

【例 3.9】 声明变量，实例代码如下：

```
int age;           //声明 int 型变量
char char1 = 'r'; //声明 char 型变量并赋值
```

编写以上程序代码，究竟会产生什么样的效果呢？要了解这个问题，就需要对变量的内存配置有一定的认识。用图解的方式将上例程序代码在内存中的状况表现出来，如图 3.5 所示。

由图 3.5 可知，系统的内存可大略分为 3 个区域，即系统(OS)区、程序(Program)区和数据(Data)区。当程序执行时，程序代码会加载到内存中的程序区，数据暂时存储在数据区中。假设上述两个变量定义在方法体中，则程序加载到程序区中。当执行此程序代码时，会在数据区配置空间给出这两个变量。

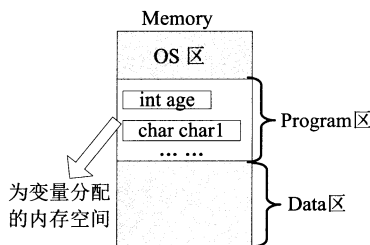


图 3.5 变量占用的内存空间

对于变量的命名并不是任意的，应遵循以下几条规则：

- ☑ 变量名必须是一个有效的标识符。
- ☑ 变量名不可以使用 Java 中的关键字。
- ☑ 变量名不能重复。
- ☑ 应选择有意义的单词作为变量名。

#### 说明

在 Java 语言中允许使用汉字或其他语言文字作为变量名，如“int 年龄 = 21”，在程序运行时不会出现错误，但建议读者尽量不要使用这些语言文字作为变量名。

### 3.3.3 声明常量

在程序运行过程中一直不会改变的量称为常量 (constant)，通常也被称为“final 变量”。常量在整个程序中只能被赋值一次。在为所有的对象共享值时，常量是非常有用的。

在 Java 语言中声明一个常量，除了要指定数据类型外，还需要通过 final 关键字进行限定。声明常量的标准语法如下：

```
final 数据类型 常量名称[=值]
```

常量名通常使用大写字母，但这并不是必需的。很多 Java 程序员使用大写字母表示常量，是为了清楚地表明正在使用常量。

【例 3.10】 声明常量，实例代码如下：

```
final double PI = 3.1415926D; //声明 double 型常量 PI 并赋值
final boolean BOOL = true;    //声明 boolean 型常量 BOOL 并赋值
```

当定义的 `final` 变量属于“成员变量”时，必须在定义时就设定它的初值，否则将会产生编译错误。从下面的实例中可看出变量与常量的区别。

【例 3.11】 在项目中创建类 `Part`，在类体中创建变量 `age` 与常量 `PI`。在主方法中分别将变量与常量赋值，通过输出信息可测试变量与常量的有效范围。（实例位置：光盘\TM\sl\3.04）

```
public class Part {                                //新建类 Part
    //声明常量 PI，此时如不对 PI 进行赋值，则会出现错误提示
    static final double PI = 3.14;
    static int age = 23;                            //声明 int 型变量 age 并进行赋值

    public static void main(String[] args) {        //主方法
        final int number;                          //声明 int 型常量 number
        number = 1235;                             //对常量进行赋值
        age = 22;                                   //再次对变量进行赋值
        //number = 1236;                          //错误的代码，因为 number 为常量，只能进行一次赋值
        System.out.println("常量 PI 的值为: " + PI); //将 PI 的值输出
        System.out.println("赋值后 number 的值为:" + number); //将 number 的值输出
        System.out.println("int 型变量 age 的值为: " + age); //将 age 的值输出
    }
}
```

运行结果如图 3.6 所示。

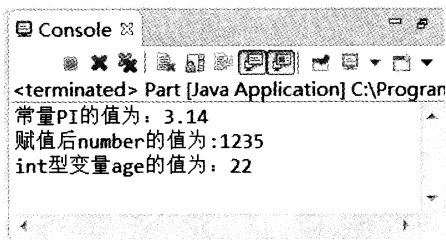


图 3.6 例 3.11 的运行结果

### 3.3.4 变量的有效范围

由于变量被定义出来后只是暂存在内存中，等到程序执行到某一个点，该变量会被释放掉，也就是说变量有它的使用寿命。因此，变量的有效范围是指程序代码能够访问该变量的区域，若超出该区域，则在编译时会出现错误。在程序中，一般会根据变量的“有效范围”将变量分为“成员变量”和“局部变量”。

#### 1. 成员变量

在类体中所定义的变量被称为成员变量，成员变量在整个类中都有效。类的成员变量又可分为两种，即静态变量和实例变量。

【例 3.12】 声明静态变量和实例变量，实例代码如下：

```
class var{
    int x = 45;
```

```
static int y = 90
}
```

其中， $x$  为实例变量， $y$  为静态变量（也称类变量）。如果在成员变量的类型前面加上关键字 `static`，这样的成员变量称为静态变量。静态变量的有效范围可以跨类，甚至可达到整个应用程序之内。对于静态变量，除了能在定义它的类内存取，还能直接以“类名.静态变量”的方式在其他类内使用。

## 2. 局部变量

在类的方法体中定义的变量（方法内部定义，“{”与“}”之间的代码中声明的变量）称为局部变量。局部变量只在当前代码块中有效。

在类的方法中声明的变量，包括方法的参数，都属于局部变量。局部变量只在当前定义的方法内有效，不能用于类的其他方法中。局部变量的生命周期取决于方法，当方法被调用时，Java 虚拟机为方法中的局部变量分配内存空间，当该方法的调用结束后，则会释放方法中局部变量占用的内存空间，局部变量也将会销毁。

局部变量可与成员变量的名字相同，此时成员变量将被隐藏，即这个成员变量在此方法中暂时失效。变量的有效范围如图 3.7 所示。

【例 3.13】在项目中创建类 `Val`，分别定义名称相同的局部变量与成员变量，当名称相同时成员变量将被隐藏。（实例位置：光盘\TM\sl\3.05）

```
public class Val {
    static int times = 3;
    public static void main(String[] args) {
        int times = 4;
        System.out.println("times 的值为: " + times);
    }
}
```

//新建类  
//定义成员变量 times  
//主方法  
//定义局部变量 times  
//将 times 的值输出

运行结果如图 3.8 所示。

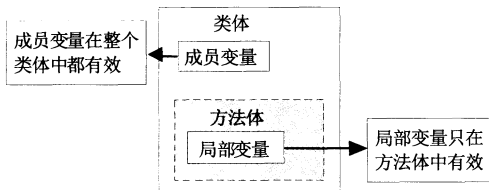


图 3.7 变量的有效范围

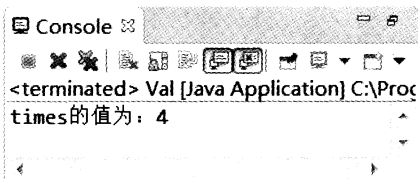


图 3.8 例 3.13 的运行结果

## 3.4 运算符

运算符是一些特殊的符号，主要用于数学函数、一些类型的赋值语句和逻辑比较方面。Java 中提供了丰富的运算符，如赋值运算符、算术运算符、比较运算符等。本节将向读者介绍这些运算符。

### 3.4.1 赋值运算符

 视频讲解：光盘\TM\lx\3\赋值运算符.mp4

赋值运算符以符号“=”表示，它是一个二元运算符（对两个操作数作处理），其功能是将右方操作数所含的值赋给左方的操作数。例如：

```
int a = 100;
```

该表达式是将 100 赋值给变量 a。左方的操作数必须是一个变量，而右边的操作数则可以是任何表达式，包括变量（如 a、number）、常量（如 123、'book'）、有效的表达式（如 45\*12）。

【例 3.14】 使用赋值运算符为变量赋值，实例代码如下：

```
int a = 10;           //声明 int 型变量 a
int b = 5;           //声明 int 型变量 b
int c = a+b;         //将变量 a 与 b 运算后的结果赋值给 c
```

遵循赋值运算符的运算规则，可知系统将先计算 a+b 的值，结果为 15，然后将 15 赋值给变量 c，因此“c=15”。

由于赋值运算符“=”处理时会先取得右方表达式处理后的结果，因此一个表达式中若含有两个以上的“=”运算符，会从最右方的“=”开始处理。

【例 3.15】 在项目中创建类 Eval，在主方法中定义变量，使用赋值运算符为变量赋值。（实例位置：光盘\TM\sl\3.06）

```
public class Eval {           //创建类
    public static void main(String[] args) {           //主方法
        int a, b, c;           //声明 int 型变量 a、b、c
        a = 15;               //将 15 赋值给变量 a
        c = b = a + 4;         //将 a 与 4 的和赋值给变量 b，然后再赋值给变量 c
        System.out.println("c 值为: " + c);           //将变量 c 的值输出
        System.out.println("b 值为: " + b);           //将变量 b 的值输出
    }
}
```

运行结果如图 3.9 所示。

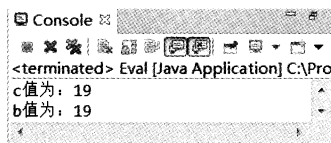


图 3.9 例 3.15 的运行结果

#### 说明

在 Java 中可以把赋值运算符连在一起使用。如：

```
x = y = z = 5;
```

在这个语句中，变量 x、y、z 都得到同样的值 5。但在程序开发中不建议使用这种赋值语句。

### 3.4.2 算术运算符

 视频讲解：光盘\TM\lx\3\算术运算符.mp4

Java 中的算术运算符主要有+（加）、-（减）、\*（乘）、/（除）、%（求余），它们都是二元运算符。Java 中算术运算符的功能及使用方式如表 3.5 所示。

表 3.5 Java 算术运算符

| 运 算 符 | 说 明 | 实 例       | 结 果   |
|-------|-----|-----------|-------|
| +     | 加   | 12.45f+15 | 27.45 |
| -     | 减   | 4.56-0.16 | 4.4   |
| *     | 乘   | 5L*12.45f | 62.25 |
| /     | 除   | 7/2       | 3     |
| %     | 取余数 | 12%10     | 2     |

其中，“+”和“-”运算符还可以作为数据的正负符号，如+5、-7。



#### 注意

在进行除法运算时，0 不能做除数。例如，对于语句“int a = 5/0;”，系统会抛出 ArithmeticException 异常。

下面通过一个小程序来介绍算术运算符的使用方法。

**【例 3.16】** 在项目中创建类 Arith，在主方法中定义变量，使用算术运算符将变量的计算结果输出。（实例位置：光盘\TM\sl\3.07）

```

public class Arith {
    public static void main(String[] args) {
        float number1 = 45.56f;
        int number2 = 152;
        System.out.println("和为：" + (number1 + number2));
        System.out.println("差为：" + (number2 - number1));
        System.out.println("积为：" + number1 * number2);
        System.out.println("商为：" + number1 / number2);
    }
}

```

//创建类  
 //主方法  
 //声明 float 型变量并赋值  
 //声明 int 型变量并赋值  
 //将变量相加之和输出  
 //将变量相减之差输出  
 //将变量相乘的积输出  
 //将变量相除的商输出

运行结果如图 3.10 所示。

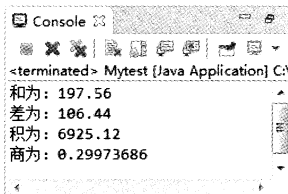


图 3.10 使用算术运算符将变量的计算结果输出

### 3.4.3 自增和自减运算符

 视频讲解：光盘\TM\lx\3\自增和自减运算符.mp4

自增、自减运算符是单目运算符，可以放在操作元之前，也可以放在操作元之后。操作元必须是一个整型或浮点型变量。自增、自减运算符的作用是使变量的值增 1 或减 1。放在操作元前面的自增、自减运算符，会先将变量的值加 1（减 1），然后再使该变量参与表达式的运算。放在操作元后面的自增、自减运算符，会先使变量参与表达式的运算，然后再将该变量加 1（减 1）。例如：

```
++a(--a)           //表示在使用变量 a 之前，先使 a 的值加（减）1
a++(a--)           //表示在使用变量 a 之后，使 a 的值加（减）1
```

粗略地分析，++a 与 a++的作用都相当于 a = a+1。假设 a = 4，则：

```
b = ++a;           //先将 a 的值加 1，然后赋给 b，此时 a 值为 5，b 值为 5
```

再看另一个语法，同样假设 a = 4，则：

```
b = a++;           //先将 a 的值赋给 b，再将 a 的值变为 5，此时 a 值为 5，b 值为 4
```

### 3.4.4 比较运算符

 视频讲解：光盘\TM\lx\3\比较（关系）运算符.mp4

比较运算符属于二元运算符，用于程序中的变量之间、变量和自变量之间以及其他类型的信息之间的比较。比较运算符的运算结果是 boolean 型。当运算符对应的关系成立时，运算结果为 true，否则为 false。所有比较运算符通常作为判断的依据用在条件语句中。比较运算符共有 6 个，如表 3.6 所示。

表 3.6 比较运算符

| 运 算 符 | 作 用          | 举 例         | 操 作 数 据    | 结 果   |
|-------|--------------|-------------|------------|-------|
| >     | 比较左方是否大于右方   | 'a'>'b'     | 整型、浮点型、字符型 | false |
| <     | 比较左方是否小于右方   | 156 < 456   | 整型、浮点型、字符型 | true  |
| ==    | 比较左方是否等于右方   | 'c'=='c'    | 基本数据类型、引用型 | true  |
| >=    | 比较左方是否大于等于右方 | 479>=426    | 整型、浮点型、字符型 | true  |
| <=    | 比较左方是否小于等于右方 | 12.45<=45.5 | 整型、浮点型、字符型 | true  |
| !=    | 比较左方是否不等于右方  | 'y'!='t'    | 基本数据类型、引用型 | true  |

【例 3.17】 在项目中创建类 Compare，在主方法中创建整型变量，使用比较运算符对变量进行比较运算，并将运算后的结果输出。（实例位置：光盘\TM\sl\3.08）

```
public class Compare {           //创建类
    public static void main(String[] args) {
        int number1 = 4;         //声明 int 型变量 number1
        int number2 = 5;         //声明 int 型变量 number2
    }
}
```

```

/* 依次将变量 number1 与变量 number2 的比较结果输出 */
System.out.println("number1>number 的返回值为: " + (number1 > number2));
System.out.println("number1< number2 返回值为: " + (number1 < number2));
System.out.println("number1==number2 返回值为: " + (number1 == number2));
System.out.println("number1!=number2 返回值为: " + (number1 != number2));
System.out.println("number1>= number2 返回值为: " + (number1 >= number2));
System.out.println("number1<=number2 返回值为: " + (number1 <= number2));
}
}

```

运行结果如图 3.11 所示。

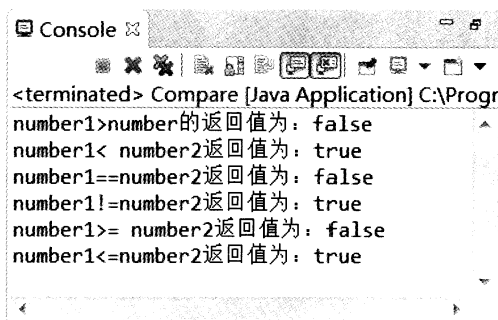


图 3.11 例 3.17 的运行结果

### 3.4.5 逻辑运算符

 视频讲解：光盘\TM\lx\3\逻辑运算符.mp4

返回类型为布尔值的表达式，如比较运算符，可以被组合在一起构成一个更复杂的表达式。这是通过逻辑运算符来实现的。逻辑运算符包括&（&&）（逻辑与）、||（逻辑或）、！（逻辑非）。逻辑运算符的操作元必须是 boolean 型数据。在逻辑运算符中，除了“！”是一元运算符之外，其他都是二元运算符。表 3.7 给出了逻辑运算符的用法和含义。

表 3.7 逻辑运算符

| 运 算 符 | 含 义 | 用 法      | 结 合 方 向 |
|-------|-----|----------|---------|
| &&、&  | 逻辑与 | op1&&op2 | 从左到右    |
|       | 逻辑或 | op1  op2 | 从左到右    |
| !     | 逻辑非 | !op      | 从右到左    |

结果为 boolean 型的变量或表达式可以通过逻辑运算符组合为逻辑表达式。

用逻辑运算符进行逻辑运算时，结果如表 3.8 所示。

表 3.8 使用逻辑运算符进行逻辑运算

| 表达式 1 | 表达式 2 | 表达式 1&&表达式 2 | 表达式 1  表达式 2 | !表达式 1 |
|-------|-------|--------------|--------------|--------|
| true  | true  | true         | true         | false  |
| true  | false | false        | true         | false  |



续表

| 表达式 1 | 表达式 2 | 表达式 1&&表达式 2 | 表达式 1  表达式 2 | !表达式 1 |
|-------|-------|--------------|--------------|--------|
| false | false | false        | false        | true   |
| false | true  | false        | true         | true   |

逻辑运算符“&&”与“&”都表示“逻辑与”，那么它们之间的区别在哪里呢？从表 3.8 可以看出，当两个表达式都为 true 时，“逻辑与”的结果才会是 true。使用逻辑运算符“&”会判断两个表达式；而逻辑运算符“&&”则是针对 boolean 类型的类进行判断，当第一个表达式为 false 时则不去判断第二个表达式，直接输出结果从而节省计算机判断的次数。通常将这种在逻辑表达式中从左端的表达式可推断出整个表达式的值称为“短路”，而那些始终执行逻辑运算符两边的表达式称为“非短路”。“&&”属于“短路”运算符，而“&”则属于“非短路”运算符。

**【例 3.18】** 在项目中创建类 Calculation，在主方法中创建整型变量，使用逻辑运算符对变量进行运算，并将运算结果输出。（实例位置：光盘\TM\sl\3.09）

```
public class Calculation {           //创建类
    public static void main(String[] args) {
        int a = 2;                   //声明 int 型变量 a
        int b = 5;                   //声明 int 型变量 b
        //声明 boolean 型变量，用于保存应用逻辑运算符“&&”后的返回值
        boolean result = ((a > b) && (a != b));
        //声明 boolean 型变量，用于保存应用逻辑运算符“||”后的返回值
        boolean result2 = ((a > b) || (a != b));
        System.out.println(result);   //将变量 result 输出
        System.out.println(result2);  //将变量 result2 输出
    }
}
```

运行结果如图 3.12 所示。

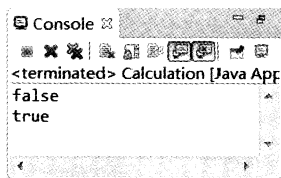


图 3.12 例 3.18 的运行结果

### 3.4.6 位运算符



视频讲解：光盘\TM\lx\3\位运算符.mp4

位运算符除“按位与”和“按位或”运算符外，其他只能用于处理整数的操作数。位运算是完全针对位方面的操作。整型数据在内存中以二进制的形式表示，如 int 型变量 7 的二进制表示是 00000000 00000000 00000000 00000111。

左边最高位是符号位，最高位是 0 表示正数，若为 1 则表示负数。负数采用补码表示，如 -8 的二进制表示为 11111111 11111111 11111111 11111000。这样就可以对整型数据进行按位运算。

### 1. “按位与” 运算

“按位与”运算的运算符为“&”，为双目运算符。“按位与”运算的运算法则是：如果两个整型数据 a、b 对应位都是 1，则结果位才是 1，否则为 0。如果两个操作数的精度不同，则结果的精度与精度高的操作数相同，如图 3.13 所示。

### 2. “按位或” 运算

“按位或”运算的运算符为“|”，为双目运算符。“按位或”运算的运算法则是：如果两个操作数对应位都是 0，则结果位才是 0，否则为 1。如果两个操作数的精度不同，则结果的精度与精度高的操作数相同，如图 3.14 所示。

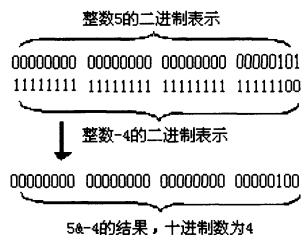


图 3.13 5&amp;-4 的运算过程

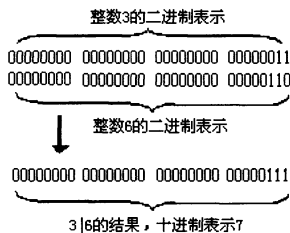


图 3.14 3|6 的运算过程

### 3. “按位取反” 运算

“按位取反”运算也称“按位非”运算，运算符为“~”，为单目运算符。“按位取反”就是将操作数二进制中的 1 修改为 0，0 修改为 1，如图 3.15 所示。

### 4. “按位异或” 运算

“按位异或”运算的运算符是“^”，为双目运算符。“按位异或”运算的运算法则是：当两个操作数的二进制表示相同（同时为 0 或同时为 1）时，结果为 0，否则为 1。若两个操作数的精度不同，则结果数的精度与精度高的操作数相同，如图 3.16 所示。

### 5. 移位操作

除了上述运算符之外，还可以对数据按二进制位进行移位操作。Java 中的移位运算符有以下 3 种。

- ☑ <<: 左移。
- ☑ >>: 右移。
- ☑ >>>: 无符号右移。

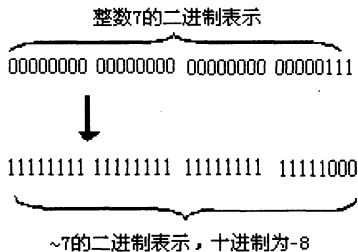


图 3.15 ~7 的运算过程

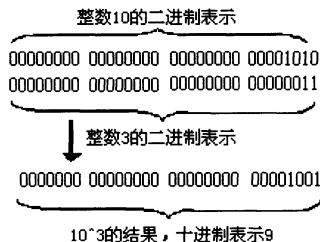


图 3.16 10^3 的运算过程

左移就是将运算符左边的操作数的二进制数据按照运算符右边操作数指定的位数向左移动，右边移空的部分补 0。右移则复杂一些。当使用“>>”符号时，如果最高位是 0，右移空的位就填入 0；如果最高位是 1，右移空的位就填入 1，如图 3.17 所示。

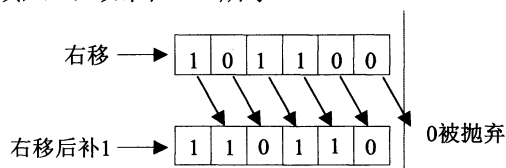


图 3.17 右移

Java 还提供了无符号右移“>>>”，无论最高位是 0 还是 1，左侧被移空的高位都填入 0。

**注意**

移位运算符适用的数据类型有 byte、short、char、int 和 long。

**技巧**

移位可以实现整数除以或乘以  $2^n$  的效果。例如， $y << 2$  与  $y * 4$  的结果相同； $y >> 1$  的结果与  $y / 2$  的结果相同。总之，一个数左移  $n$  位，就是将这个数乘以  $2^n$ ；一个数右移  $n$  位，就是将这个数除以  $2^n$ 。

### 3.4.7 三元运算符

视频讲解：光盘\TM\lx\3\三元运算符.mp4

三元运算符的使用格式为：

条件式?值 1:值 2

三元运算符的运算规则为：若条件式的值为 true，则整个表达式取值 1，否则取值 2。例如：

```
boolean b = 20 < 45 ? true : false;
```

如上例所示，表达式“20<45”的运算结果返回真，那么 boolean 型变量 b 取值为 true；相反，表达式“45<20”返回为假，则 boolean 型变量 b 取值 false。

三元运算符等价于 if...else 语句。

【例 3.19】 等价于三元运算符的 if...else 语句，实例代码如下：（实例位置：光盘\TM\sl\3.10）

```
boolean a;           //声明 boolean 型变量
if(20<45)             //将 20<45 作为判断条件
    a = true;         //条件成立将 true 赋值给 a
else
    a = false;        //条件不成立将 false 赋值给 a
```

### 3.4.8 运算符优先级

 视频讲解：光盘\TM\lx\3\运算符优先级.mp4

Java 中的表达式就是使用运算符连接起来的符合 Java 规则的式子。运算符的优先级决定了表达式中运算执行的先后顺序。通常优先级由高到低的顺序依次是：

- ☒ 增量和减量运算。
- ☒ 算术运算。
- ☒ 比较运算。
- ☒ 逻辑运算。
- ☒ 赋值运算。

如果两个运算有相同的优先级，那么左边的表达式要比右边的表达式先被处理。表 3.9 显示了在 Java 中众多运算符特定的优先级。

表 3.9 运算符的优先级

| 优 先 级 | 描 述    | 运 算 符        |
|-------|--------|--------------|
| 1     | 括号     | ()           |
| 2     | 正负号    | +, -         |
| 3     | 一元运算符  | ++, --, !    |
| 4     | 乘除     | *, /, %      |
| 5     | 加减     | +, -         |
| 6     | 移位运算   | >>, >>>, <<  |
| 7     | 比较大小   | <, >, >=, <= |
| 8     | 比较是否相等 | ==, !=       |
| 9     | 按位与运算  | &            |
| 10    | 按位异或运算 | ^            |
| 11    | 按位或运算  |              |
| 12    | 逻辑与运算  | &&           |
| 13    | 逻辑或运算  |              |
| 14    | 三元运算符  | ?:           |
| 15    | 赋值运算符  | =            |

#### 技巧

在编写程序时尽量使用括号运算符来限定运算次序，以免产生错误的运算顺序。

## 3.5 数据类型转换

 视频讲解：光盘\TM\lx\3\数据类型转换.mp4

类型转换是将一个值从一种类型更改为另一种类型的过程。例如，可以将 String 类型数据“457”

转换为一个数值型，而且可以将任意类型的数据转换为 String 类型。

如果从低精度数据类型向高精度数据类型转换，则永远不会溢出，并且总是成功的；而把高精度数据类型向低精度数据类型转换则必然会有信息丢失，有可能失败。

数据类型转换有两种方式，即隐式转换与显式转换。

### 3.5.1 隐式类型转换

从低级类型向高级类型的转换，系统将自动执行，程序员无须进行任何操作。这种类型的转换称为隐式转换。下列基本数据类型会涉及数据转换，不包括逻辑类型和字符类型。这些类型按精度从低到高排列的顺序为 `byte < short < int < long < float < double`。

【例 3.20】使用 `int` 型变量为 `float` 型变量赋值，此时 `int` 型变量将隐式转换成 `float` 型变量。实例代码如下：

```
int x = 50;           //声明 int 型变量 x
float y = x;          //将 x 赋值给 y
```

此时执行输出语句，`y` 的结果将是 50.0。

隐式类型的转换也要遵循一定的规则，来解决在什么情况下将哪种类型的数据转换成另一种类型的数据。表 3.10 列出了各种数据类型转换的一般规则。

表 3.10 隐式类型转换规则

| 操作数 1 的数据类型                    | 操作数 2 的数据类型 | 转换后的数据类型 |
|--------------------------------|-------------|----------|
| byte、short、char                | int         | int      |
| byte、short、char、int、           | long        | long     |
| byte、short、char、int、long       | float       | float    |
| byte、short、char、int、long、float | double      | double   |

下面通过一个简单实例介绍数据类型隐式转换。

【例 3.21】在项目中创建类 `Conver`，在主方法中创建不同数值型的变量，实现将各变量隐式转换。（实例位置：光盘\TM\sl\3.11）

```
public class Conver {           //创建类
    public static void main(String[] args) {
        //定义 byte 型变量 mybyte，并把 byte 型变量允许的最大值赋给 mybyte
        byte mybyte = 127;
        int myint = 150;         //定义 int 型变量 myint，并赋值 150
        float myfloat = 452.12f; //定义 float 型变量 myfloat，并赋值
        char mychar = 10;       //定义 char 型变量 mychar，并赋值
        double mydouble = 45.46546; //定义 double 型变量，并赋值
        System.out.println("byte 型与 float 型数据进行运算结果为："
            + (mybyte + myfloat));
        /* 将运算结果输出 */
        System.out.println("byte 型与 int 型数据进行运算结果为："
            + mybyte * myint);
    }
}
```

```

        System.out.println("byte 型与 char 型数据进行运算结果为: "
            + mybyte / mychar);
        System.out.println("double 型与 char 型数据进行运算结果为: "
            + (mydouble + mychar));
    }
}

```

运行结果如图 3.18 所示。

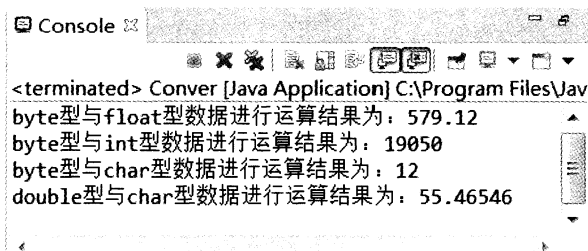


图 3.18 例 3.21 的运行结果



#### 技巧

要理解类型转换，读者可以这么想象，大脑前面是一片内存，源和目标分别是两个大小不同的内存块（由变量及数据的类型来决定），将源数据赋值给目标内存的过程，就是用目标内存块尽可能地套取源内存中的数据。

## 3.5.2 显式类型转换

当把高精度的变量的值赋给低精度的变量时，必须使用显式类型转换运算（又称强制类型转换）。语法如下：

**(类型名)要转换的值**

下面通过几种常见的显式数据类型转换实例来说明。

**【例 3.22】** 将不同的数据类型进行显式类型转换，实例代码如下：

```

int a = (int)45.23;           //此时输出 a 的值为 45
long y = (long)456.6F;       //此时输出 y 的值为 456
int b = (int)d';              //此时输出 b 的值为 100

```

当执行显式类型转换时可能会导致精度损失。只要是 boolean 类型以外其他基本类型之间的转换，全部都能以显式类型转换的方法达到。

**说明**

当把整数赋值给一个 `byte`、`short`、`int`、`long` 型变量时，不可以超出这些变量的取值范围，否则必须进行强制类型转换。例如：

```
byte b = (byte)129;
```

## 3.6 代码注释与编码规范

在程序代码中适当地添加注释可以提高程序的可读性和可维护性。好的编码规范可以使程序更易阅读和理解。本节将介绍 Java 中的几种代码注释以及应该注意的编码规范。

### 3.6.1 代码注释



视频讲解：光盘\TM\lx\3\代码注释.mp4

通过在程序代码中添加注释可提高程序的可读性。注释中包含了程序的信息，可以帮助程序员更好地阅读和理解程序。在 Java 源程序文件的任意位置都可添加注释语句。注释中的文字 Java 编译器不进行编译，所有代码中的注释文字对程序不产生任何影响。Java 语言提供了 3 种添加注释的方法，分别为单行注释、多行注释和文档注释。

#### 1. 单行注释

“//”为单行注释标记，从符号“//”开始直到换行为止的所有内容均作为注释而被编译器忽略。语法如下：

```
//注释内容
```

例如，以下代码为声明的 `int` 型变量添加注释：

```
int age;           //定义 int 型变量用于保存年龄信息
```

#### 2. 多行注释

“/\* \*/”为多行注释标记，符号“/\*”与“\*/”之间的所有内容均为注释内容。注释中的内容可以换行。

语法如下：

```
/*  
  注释内容 1  
  注释内容 2  
  ...  
*/
```

 **注意**

在多行注释中可嵌套单行注释。例如:

```
/*
  程序名称: Hello word  //开发时间: 2008-03-05
*/
```

 **注意**

但在多行注释中不可以嵌套多行注释, 以下代码为非法:

```
/*
  程序名称: Hello word
  /*开发时间: 2008-03-05
    作者: 张先生
*/
*/
```

### 3. 文档注释

“/\*\* \*/”为文档注释标记。符号“/\*\*”与“\*/”之间的内容均为文档注释内容。当文档注释出现在声明(如类的声明、类的成员变量的声明、类的成员方法声明等)之前时, 会被 Javadoc 文档工具读取作为 Javadoc 文档内容。文档注释的格式与多行注释的格式相同。对于初学者而言, 文档注释并不是很重要, 了解即可。

 **说明**

一定要养成良好的编程风格。软件编码规范中提到“可读性第一, 效率第二”, 所以程序员必须要在程序中添加适量的注释来提高程序的可读性和可维护性。程序中注释要占程序代码总量的 20%~50%。

### 3.6.2 编码规范

 **视频讲解:** 光盘\TM\lx\3\编码规范.mp4

在学习开发的过程中要养成良好的编码习惯, 因为规整的代码格式会给程序的开发与日后的维护提供很大方便。在此对编码规则作了以下总结, 供读者学习。

- ☒ 每条语句要单独占一行, 一条命令要以分号结束。

 **注意**

程序代码中的分号必须为英文状态下输入的, 初学者经常会将“;”写成中文状态下的“;”, 此时编译器会报出 illegal character (非法字符) 这样的错误信息。



- ☑ 在声明变量时，尽量使每个变量的声明单独占一行，即使是相同的数据类型也要将其放置在单独的一行上，这样有助于添加注释。对于局部变量应在声明的同时对其进行初始化。
- ☑ 在 Java 代码中，关键字与关键字间如果有多个空格，这些空格均被视作一个。例如：

等价于

```
public    static void main(String args[])  ⇨  public static void main(String args[])
```

多行空格没有任何意义，为了便于理解、阅读，应控制好空格的数量。

- ☑ 为了方便日后的维护，不要使用技术性很高、难懂、易混淆判断的语句。由于程序的开发与维护不能是同一个人，所以应尽量使用简单的技术完成程序需要的功能。
- ☑ 对于关键的方法要多加注释，这样有助于阅读者了解代码结构。

## 3.7 小 结

本章向读者介绍的是 Java 语言基础，其中需要读者重点掌握的是 Java 语言的基本数据类型、变量与常量以及运算符三大知识点。初学者经常会将 String 类型认为是 Java 语言的基本数据类型，在此提醒读者 Java 语言的基本数据类型中并没有 String 类型。另外，要对数据类型之间的转换有一定的了解。在使用变量时，需要读者注意的是变量的有效范围，否则在使用时会出现编译错误或浪费内存资源。此外，Java 中的各种运算符也是 Java 基础中的重点，正确使用这些运算符，才能得到预期的结果。

## 3.8 实践与练习

1. 编写 Java 程序将两个数相加的结果输出。（答案位置：光盘\TM\sl\3.12）
2. 编写 Java 程序，声明成员变量 age 与局部变量 name。比较一下两个变量的区别，并添加相应的注释。（答案位置：光盘\TM\sl\3.13）
3. 编写 Java 程序，将所有整型数值全部转换成 int 型，并将转换后的值输出。（答案位置：光盘\TM\sl\3.14）

# 第 4 章

---

## 流程控制

(  视频讲解：1 小时 21 分钟 )

做任何事情都要遵循一定的原则。例如，到图书馆去借书，就必须要有借书证，并且借书证不能过期，这两个条件缺一不可。程序设计也是如此，需要有流程控制语言实现与用户的交流，并根据用户的输入决定程序要“做什么”“怎么做”等。

流程控制对于任何一门编程语言来说都是至关重要的，它提供了控制程序步骤的基本手段。如果没有流程控制语句，整个程序将按照线性的顺序来执行，不能根据用户的输入决定执行的序列。本章将向读者介绍 Java 语言中的流程控制语句。

通过阅读本章，您可以：

- » 理解 Java 语言中复合语句的使用方法
- » 掌握 if 条件语句的使用方法
- » 了解 if 语句与 switch 语句的区别
- » 掌握 while 循环语句的使用方法
- » 掌握 do...while 循环语句的使用方法
- » 了解 while 语句与 do...while 语句的区别
- » 掌握 for 语句的使用方法
- » 了解 foreach 语句的使用方法

## 4.1 复合语句

与 C 语言及其他语言相同, Java 语言的复合语句是以整个块区为单位的语句, 所以又称块语句。复合语句由开括号“{”开始, 闭括号“}”结束。

在前面的学习中已经接触到了这种复合语句, 例如, 在定义一个类或方法时, 类体就是以“{ }”作为开始与结束的标记, 方法体同样也是以“{ }”作为标记。复合语句中的每个语句都是从上到下被执行。复合语句以整个块为单位, 能够用在任何一个单独语句可以用到的地方, 并且在复合语句中还可以嵌套复合语句。

【例 4.1】在项目中创建类 Compound, 在主方法中定义复合语句块, 其中包含另一复合语句块。代码如下: (实例位置: 光盘\TM\sl\4.01)

```
public class Compound {  
    public static void main(String args[]) {  
        int x = 20;  
        {  
            int y = 40;  
            System.out.println(y);  
            int z = 245;  
            boolean b;  
            {  
                b = y > z;  
                System.out.println(b);  
            }  
        }  
        String word = "hello java";  
        System.out.println(word);  
    }  
}
```

复合语句

复合语句

运行结果如图 4.1 所示。

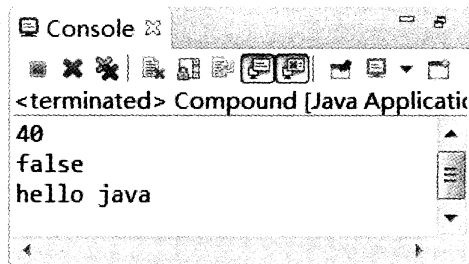


图 4.1 例 4.1 的运行结果

在使用复合语句时要注意, 复合语句为局部变量创建了一个作用域, 该作用域为程序的一部分, 在该作用域中某个变量被创建并能够被使用。如果在某个变量的作用域外使用该变量, 则会发生错误, 例如, 在本实例中, 如果在复合语句外使用变量 y、z、b 将会出现错误, 而变量 x 可在整个方法体中使用。

## 4.2 条件语句

条件语句可根据不同的条件执行不同的语句。条件语句包括 if 条件语句与 switch 多分支语句。本节将向读者介绍条件语句的用法。

### 4.2.1 if 条件语句

 视频讲解：光盘\TM\lx\4\if 条件语句.mp4

if 条件语句是一个重要的编程语句，用于告诉程序在某个条件成立的情况下执行某段语句，而在另一种情况下执行另外的语句。

使用 if 条件语句，可选择是否要执行紧跟在条件之后的那个语句。关键字 if 之后是作为条件的“布尔表达式”，如果该表达式返回的结果为 true，则执行其后的语句；若为 false，则不执行 if 条件之后的语句。if 条件语句可分为简单的 if 条件语句、if...else 语句和 if...else if 多分支语句。

#### 1. 简单的 if 条件语句

语法如下：

```
if(布尔表达式){
    语句序列
}
```

- ☑ 布尔表达式：必要参数，表示最后返回的结果必须是一个布尔值。它可以是一个单纯的布尔变量或常量，也可以是使用关系或布尔运算符的表达式。
- ☑ 语句序列：可选参数。可以是一条或多条语句，当表达式的值为 true 时执行这些语句。若语句序列中仅有一条语句，则可以省略条件语句中的“{ }”。

【例 4.2】语句序列中只有一条语句，实例代码如下：

```
int a = 100;
if(a == 100)
    System.out.print("a 的值是 100");
```

#### 说明

虽然 if 后面的复合语句块只有一条语句，省略“{ }”并无语法错误，但为了增强程序的可读性最好不要省略。

条件语句后的语句序列省略时，则可以保留外面的大括号，也可以省略大括号。然后在末尾添加“;”。如下所示的两种情况都是正确的。

【例 4.3】省略了 if 条件表达式中的语句序列，实例代码如下：

```
boolean b = false;
if(b);
boolean b = false;
if(b){}
```

简单的 if 条件语句的执行过程如图 4.2 所示。

【例 4.4】在项目中创建类 Getif，在主方法中定义整型变量。使用条件语句判断两个变量的大小来决定输出结果。（实例位置：光盘\TM\sl\4.02）

```
public class Getif {
    public static void main(String args[]) {
        int x = 45;
        int y = 12;
        if (x > y) {
            System.out.println("变量 x 大于变量 y");
        }
        if (x < y) {
            System.out.println("变量 x 小于变量 y");
        }
    }
}
```

//创建类  
//主方法  
//声明 int 型变量 x，并赋给初值  
//声明 int 型变量 y，并赋给初值  
//判断 x 是否大于 y  
//如果条件成立，输出的信息  
//判断 x 是否小于 y  
//如果条件成立，输出的信息

运行结果如图 4.3 所示。

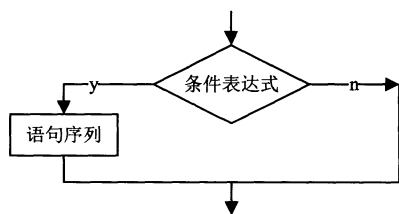


图 4.2 if 条件语句的执行过程

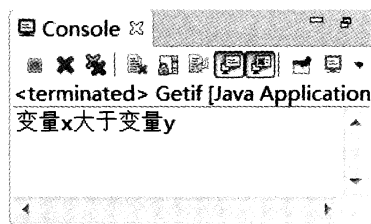


图 4.3 例 4.4 的运行结果

## 2. if...else 语句

if...else 语句是条件语句中最常用的一种形式，它会针对某种条件有选择地作出处理。通常表现为“如果满足某种条件，就进行某种处理，否则就进行另一种处理”。

语法如下：

```
if(表达式){
    若干语句
}
else{
    若干语句
}
```

if 后面“()”内的表达式的值必须是 boolean 型的。如果表达式的值为 true，则执行紧跟 if 语句的复合语句；如果表达式的值为 false，则执行 else 后面的复合语句。if...else 语句的执行过程如图 4.4 所示。

同简单的 if 条件语句一样，如果 if...else 语句的语句序列中只有一条语句（不包括注释），则可以

省略该语句序列外面的“{}”。有时为了编程的需要, else 或 if 后面的“{}”中可以没有语句。

【例 4.5】在项目中创建类 Getifelse, 在主方法中定义变量, 并通过使用 if...else 语句判断变量的值来决定输出结果。(实例位置: 光盘\TM\sl\4.03)

|                                                                                                                                                                                                                                                                                                                                                                                                                                    |                                                                                                                                                                        |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| <pre> public class Getifelse {     public static void main(String args[]) {         int math = 95;         int english = 56;         if (math &gt; 60) {             System.out.println("数学及格了");         } else {             System.out.println("数学没有及格");         }         if (english &gt; 60) {             System.out.println("英语及格了");         } else {             System.out.println("英语没有及格");         }     } } </pre> | <pre> //主方法 //声明 int 型局部变量, 并赋给初值 95 //声明 int 型局部变量, 并赋给初值 56 //使用 if 语句判断 math 是否大于 60 //条件成立时输出的信息  //条件不成立时输出的信息  //判断英语成绩是否大于 60 //条件成立时输出的信息 //条件不成立时输出的信息 </pre> |
|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------|

运行结果如图 4.5 所示。

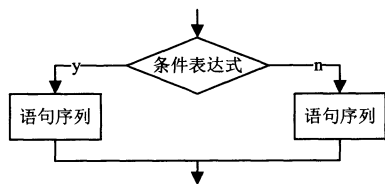


图 4.4 if...else 语句的执行过程

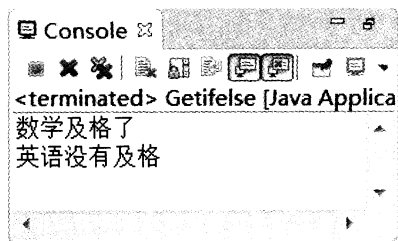


图 4.5 例 4.5 的运行结果



#### 技巧

对于 if...else 语句可以使用三元运算符对语句进行简化, 如下面的代码:

```

if(a > 0)
    b = a;
else
    b = -a;

```

可以简写成:

```

b = a > 0 ? a : -a;

```

上段代码为求绝对值的语句, 如果  $a > 0$ , 就把  $a$  的值赋值给变量  $b$ , 否则将  $-a$  赋值给变量  $b$ 。也就是问号前面的表达式为真, 则将问号与冒号之间的表达式的计算结果赋值给变量  $b$ , 否则将冒号后面的表达式的计算结果赋值给变量  $b$ 。使用三元运算符的好处是可以使代码简洁, 并且有一个返回值。

## 3. if...else if 多分支语句

if...else if 多分支语句用于针对某一事件的多种情况进行处理。通常表现为“如果满足某种条件，就进行某种处理，否则如果满足另一种条件则执行另一种处理”。

语法如下：

```
if(条件表达式 1){
    语句序列 1
}
else if(条件表达式 2){
    语句序列 2
}
...
else if(表达式 n){
    语句序列 n
}
```

- ☑ 条件表达式 1~条件表达式 n：必要参数。可以由多个表达式组成，但最后返回的结果一定要为 boolean 类型。
- ☑ 语句序列：可以是一条或多条语句，当条件表达式 1 的值为 true 时，执行语句序列 1；当条件表达式 2 的值为 true 时，执行语句序列 2，依此类推。当省略任意一组语句序列时，可以保留其外面的“{}”，也可以将“{}”替换为“;”。

if...else if 多分支语句的执行过程如图 4.6 所示。

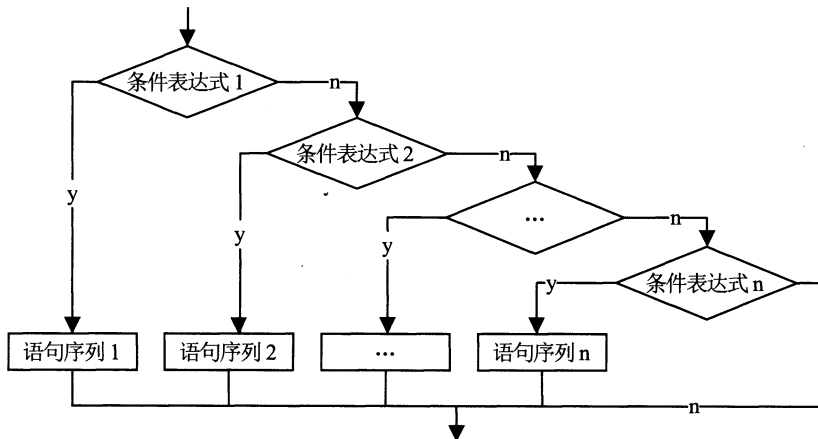


图 4.6 if...else if 多分支语句执行过程

**【例 4.6】** 在项目中创建类 GetTerm，在主方法中定义变量 x，使用 if...else if 多分支语句通过判断 x 的值决定输出结果。（实例位置：光盘\TM\sl\4.04）

```
public class GetTerm {
    public static void main(String args[]) {
        int x = 20;
        if (x > 30) {
            System.out.println("a 的值大于 30");
        }
    }
}
```

//创建主类  
//主方法  
//声明 int 型局部变量  
//判断变量 x 是否大于 30  
//条件成立时输出的信息



```

    } else if (x > 10) {
        System.out.println("a 的值大于 10, 但小于 30");
    } else if (x > 0) {
        System.out.println("a 的值大于 0, 但小于 10");
    } else {
        System.out.println("a 的值小于 0");
    }
}
}

```

//判断变量 x 是否大于 10  
//条件成立时输出的信息  
//判断变量 x 是否大于 0  
//条件成立时输出的信息  
//当以上条件都不成立时, 执行的语句块  
//输出信息

运行结果如图 4.7 所示。

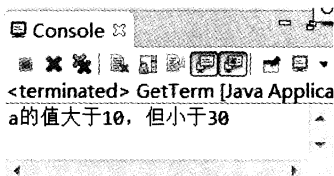


图 4.7 例 4.6 的运行结果

在本实例中由于变量 x 为 20, 所以条件  $x > 30$  为假, 程序向下执行判断下面的条件; 条件  $x > 10$  为真, 所以执行条件  $x > 10$  后面的程序块中的语句。输出“a 的值大于 10, 但小于 30”, 然后退出 if 语句。



#### 注意

if 语句只执行条件为真的命令语句, 其他语句都不会执行。

## 4.2.2 switch 多分支语句



视频讲解: 光盘\TM\4\switch 多分支语句.mp4

在编程中一个常见的问题就是检测一个变量是否符合某个条件, 如果不符合, 再用另一个值来检测, 依此类推。当然, 这种问题使用 if 条件语句也可以完成。

【例 4.7】使用 if 语句检测变量是否符合某个条件, 关键代码如下:

```

if(grade == 'A'){
    System.out.println("真棒");
}
if(grade == 'b'){
    System.out.println("做得不错");
}

```

这个程序显得比较笨重, 程序员需要测试不同的值来给出输出语句。在 Java 中, 可以用 switch 语句将动作组织起来, 以一个较简单明了的方式来实现“多选一”的选择。

语法如下:

```
switch(表达式)
```



```

{
case 常量值 1:
    语句块 1
    [break;]
...
case 常量值 n:
    语句块 n
    [break;]
default:
    语句块 n+1;
    [break;]
}

```

switch 语句中表达式的值必须是整型、字符型或字符串类型，常量值 1~n 必须也是整型、字符型或字符串类型。switch 语句首先计算表达式的值，如果表达式的值和某个 case 后面的常量值相同，则执行该 case 语句后的若干个语句直到遇到 break 语句为止。此时如果该 case 语句中没有 break 语句，将继续执行后面 case 中的若干个语句，直到遇到 break 语句为止。若没有一个常量的值与表达式的值相同，则执行 default 后面的语句。default 语句为可选的，如果它不存在，且 switch 语句中表达式的值不与任何 case 的常量值相同，switch 则不做任何处理。



#### 说明

在 JDK 1.6 及以前的版本中，switch 语句中表达式的值必须是整型或字符型，常量值 1~n 必须也是整型或字符型，但是在 JDK 1.7 中，switch 语句的表达式值除了是整型或字符型，还可以是字符串类型。这是 JDK 7 新添加的特性。

**【例 4.8】** 要通过 switch 语句根据字符串 str 的值，输出不同的提示信息可以使用下面的代码。

```

String str="明日科技";
switch (str){
case "明日":                                //定义 case 语句中的常量 1
    System.out.println("明日图书网 www.mingribook.com"); //输出信息
    break;
case "明日科技":                            //定义 case 语句中的常量 2
    System.out.println("吉林省明日科技有限公司"); //输出信息
    break;
default:                                    //default 语句
    System.out.println("以上条件都不是。"); //输出信息
}

```



#### 注意

同一个 switch 语句，case 的常量值必须互不相同。

switch 语句的执行过程如图 4.8 所示。

**【例 4.9】** 在项目中创建类 GetSwitch，在主方法中应用 switch 语句将周一到周三的英文单词打印出来。（实例位置：光盘\TM\sl\4.05）

```

public class GetSwitch {
    public static void main(String args[]) {
        int week = 2;
        switch (week) {
            case 1:
                System.out.println("Monday");
                break;
            case 2:
                System.out.println("Tuesday");
                break;
            case 3:
                System.out.println("Wednesday");
                break;
            default:
                System.out.println("Sorry,I don't Know");
        }
    }
}

```

//创建类  
 //主方法  
 //定义 int 型变量 week  
 //指定 switch 语句的表达式为变量 week  
 //定义 case 语句中的常量为 1  
 //输出信息  
 //定义 case 语句中的常量为 2  
 //输出信息  
 //定义 case 语句中的常量为 3  
 //输出信息  
 //default 语句

运行结果如图 4.9 所示。

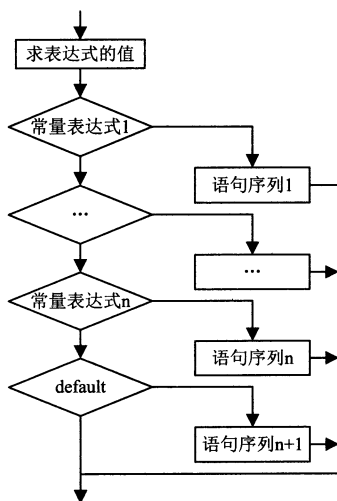


图 4.8 switch 语句的执行过程

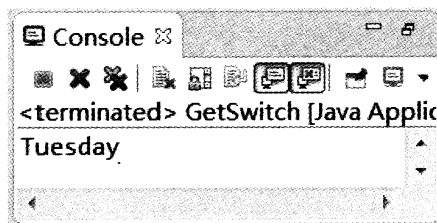


图 4.9 例 4.9 的运行结果

### 注意

在 switch 语句中，case 语句后常量表达式的值可以为整数，但绝不可以是实数。例如，下面的代码就是不合法的：

case 1.1:

## 4.3 循环语句

循环语句就是在满足一定条件的情况下反复执行某一个操作。在 Java 中提供了 3 种常用的循环语句，分别是 while 循环语句、do...while 循环语句和 for 循环语句，下面分别进行介绍。

### 4.3.1 while 循环语句

 视频讲解：光盘\TM\lx\4\while 循环语句.mp4

while 语句也称条件判断语句，它的循环方式为利用一个条件来控制是否要继续反复执行这个语句。语法如下：

```
while(条件表达式)
{
    执行语句
}
```

当条件表达式的返回值为真时，则执行“{}”中的语句，当执行完“{}”中的语句后，重新判断条件表达式的返回值，直到表达式返回的结果为假时，退出循环。while 循环语句的执行过程如图 4.10 所示。

【例 4.10】在项目中创建类 GetSum，在主方法中通过 while 循环将整数 1~10 相加，并将结果输出。（实例位置：光盘\TM\sl\4.06）

```
public class GetSum {           //创建类
    public static void main(String args[]) { //主方法
        int x = 1;               //定义 int 型变量 x，并赋给初值
        int sum = 0;             //定义变量用于保存相加后的结果
        while (x <= 10) {        //while 循环语句，当变量满足条件表达式时执行循环体语句
            sum = sum + x;
            x++;
        }
        System.out.println("sum = " + sum); //将变量 sum 输出
    }
}
```

运行结果如图 4.11 所示。

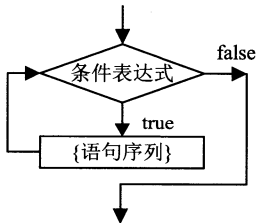


图 4.10 while 语句的执行过程

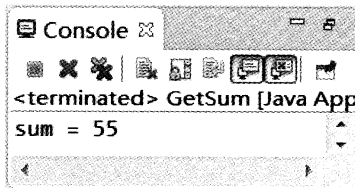


图 4.11 例 4.10 的运行结果



初学者经常犯的一个错误就是在 while 表达式的括号后加 “;”。如:

```
while(x == 5);
System.out.println("x 的值为 5");
```

这时程序会认为要执行一条空语句, 而进入无限循环, Java 编译器又不会报错。这可能会浪费很多时间去调试, 应注意这个问题。

### 4.3.2 do...while 循环语句

 视频讲解: 光盘\TM\lx\4\do...while 循环语句.mp4

do...while 循环语句与 while 循环语句类似, 它们之间的区别是 while 语句为先判断条件是否成立再执行循环体, 而 do...while 循环语句则先执行一次循环后, 再判断条件是否成立。也就是说, do...while 循环语句中 “{}” 中的程序段至少要被执行一次。

语法如下:

```
do
{
    执行语句
}
while(条件表达式);
```

do...while 语句与 while 语句的一个明显区别是, do...while 语句在结尾处多了一个分号 (;)。根据 do...while 循环语句的语法特点总结出的 do...while 循环语句的执行过程, 如图 4.12 所示。

【例 4.11】 在项目中创建类 Cycle, 在主方法中编写如下代码。通过本实例可看出 while 语句与 do...while 语句的区别。(实例位置: 光盘\TM\sl\4.07)

```
public class Cycle {
    public static void main(String args[]) {
        int a = 100;
        while (a == 60)
        {
            System.out.println("ok1");
            a--;
        }
        int b = 100;
        do {
            System.out.println("ok2");
            b--;
        } while (b == 60);
    }
}
```

//声明 int 型变量 a 并赋初值 100  
//指定进入循环体条件  
//while 语句循环体  
//声明 int 型变量 b 并赋初值 100  
//do...while 语句循环体  
//指定循环结束条件

运行结果如图 4.13 所示。

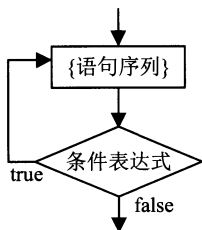


图 4.12 do...while 循环语句的执行过程

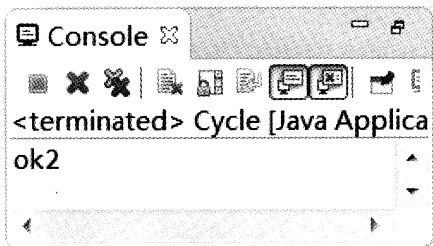


图 4.13 例 4.11 的运行结果

### 4.3.3 for 循环语句

视频讲解：光盘\TM\lx\4\for 循环语句.mp4

for 循环是 Java 程序设计中最有用的循环语句之一。一个 for 循环可以用来重复执行某条语句，直到某个条件得到满足。在 Java 5 以后新增了 foreach 语法，本节将对这两种 for 循环形式进行详细的介绍。

#### 1. for 语句

语法如下：

```
for(表达式 1;表达式 2;表达式 3)
{
    语句序列
}
```

- ☑ 表达式 1：初始化表达式，负责完成变量的初始化。
- ☑ 表达式 2：循环条件表达式，值为 boolean 型的表达式，指定循环条件。
- ☑ 表达式 3：循环后操作表达式，负责修整变量，改变循环条件。

在执行 for 循环时，首先执行表达式 1，完成某一变量的初始化工作；下一步判断表达式 2 的值，若表达式 2 的值为 true，则进入循环体；在执行完循环体后紧接着计算表达式 3，这部分通常是增加或减少循环控制变量的一个表达式。这样一轮循环就结束了。第二轮循环从计算表达式 2 开始，若表达式 2 返回 true，则继续循环，否则跳出整个 for 语句。

for 循环语句的执行过程如图 4.14 所示。

**【例 4.12】** 在项目中创建类 Circulate，在主方法中使用 for 循环来计算 2~100 之间所有偶数之和。（实例位置：光盘\TM\sl\4.08）

```
public class Circulate {           //创建类 Circulate
    public static void main(String args[]) { //主方法
        int sum = 0;               //声明变量，用于保存各数相加后的结果
        for (int i = 2; i <= 100; i += 2) { //指定循环条件及循环体
            sum = sum + i;
        }
        //将相加后的结果输出
        System.out.println("2 到 100 之间的所有偶数之和为：" + sum);
    }
}
```



```

    }
}

```

运行结果如图 4.15 所示。

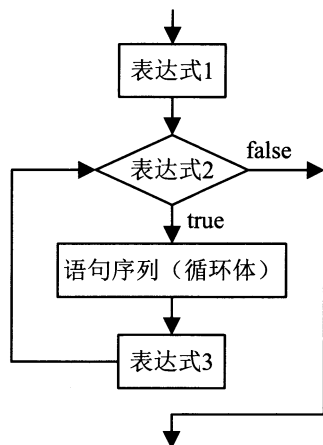


图 4.14 for 循环语句执行过程

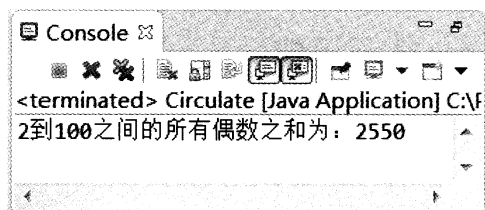


图 4.15 例 4.12 的运行结果

### 技巧

在编程时，有时会使用 for 循环的特殊语法格式来实现无限循环。语法格式为：

```

for(;;)
{
    ...
}

```

对于这种无限循环可以通过 break 语句跳出循环。例如：

```

for(;;)
{
    if(x < 20)
        break;
    x++;
}

```

## 2. foreach 语句

foreach 语句是 for 语句的特殊简化版本，不能完全取代 for 语句，但任何 foreach 语句都可以改写为 for 语句版本。foreach 并不是一个关键字，习惯上将这种特殊的 for 语句格式称为 foreach 语句。foreach 语句在遍历数组等方面为程序员提供了很大的方便（本书将在第 6 章对数组进行详细的介绍）。

语法如下：

```

for(元素变量 x：遍历对象 obj){
    引用了 x 的 java 语句;
}

```

foreach 语句中的元素变量 `x`，不必对其进行初始化。下面通过简单的例子来介绍 foreach 语句是如何遍历一维数组的。

**【例 4.13】** 在项目中创建类 `Repetition`，在主方法中定义一维数组，并用 foreach 语句遍历该数组。  
(实例位置：光盘\TM\sl\4.09)

```
public class Repetition {                                //创建类 Repetition
    public static void main(String args[]) {             //主方法
        int arr[] = { 7, 10, 1 };                       //声明一维数组
        System.out.println("一维数组中的元素分别为: "); //输出信息
        for (int x : arr) {
            //foreach 语句, int x 引用的变量, arr 指定要循环遍历的数组, 最后将 x 输出
            System.out.println(x);
        }
    }
}
```

运行结果如图 4.16 所示。

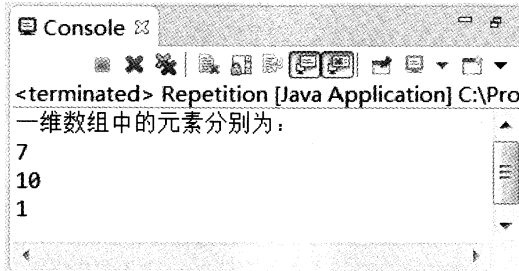


图 4.16 例 4.13 的运行结果

## 4.4 循环控制

 视频讲解：光盘\TM\lx\4\循环控制.mp4

循环控制包含两方面的内容，一方面是控制循环变量的变化方式，另一方面是控制循环的跳转。控制循环的跳转需要用到 `break` 和 `continue` 两个关键字，这两条跳转语句的跳转效果不同，`break` 是中断循环，`continue` 是执行下一次循环。

### 4.4.1 break 语句

使用 `break` 语句可以跳出 `switch` 结构。在循环结构中，同样也可用 `break` 语句跳出当前循环体，从而中断当前循环。

在 3 种循环语句中使用 `break` 语句的形式如图 4.17 所示。

|                             |                                             |                             |
|-----------------------------|---------------------------------------------|-----------------------------|
| <b>while(...)</b>           | <b>do</b>                                   | <b>for</b>                  |
| {                           | {                                           | {                           |
| ...<br><b>break;</b><br>... | ...<br><b>break;</b><br>...<br>}while(...); | ...<br><b>break;</b><br>... |
| }                           | }                                           | }                           |

图 4.17 break 语句的使用形式

【例 4.14】 使用 break 跳出循环。(实例位置: 光盘\TM\sl\4.10)

```
public class BreakTest {
    public static void main(String[] args) {
        for (int i = 0; i <= 100; i++) {
            System.out.println(i);
            if (i == 6) {                //如果 i 等于 6 则跳出循环
                break;
            }
        }
        System.out.println("---end---"); //显示程序结束
    }
}
```

运行结果如图 4.18 所示。

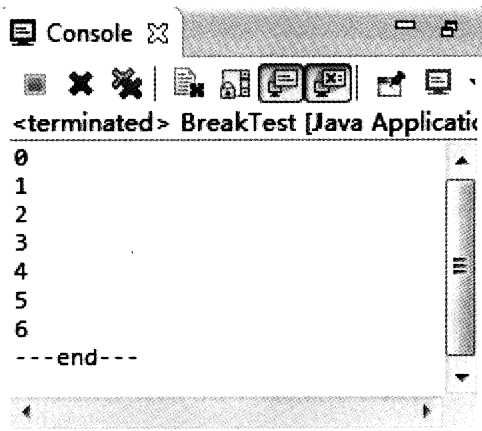


图 4.18 例 4.14 的运行结果

### ●注意

如果遇到循环嵌套的情况, break 语句将只会使程序流程跳出包含它的最内层的循环结构, 只跳出一层循环。

【例 4.15】 在嵌套的循环中使用 break 跳出内层循环。(实例位置: 光盘\TM\sl\4.11)

```
public class BreakInsideNested {
    public static void main(String[] args) {
        for (int i = 0; i < 3; i++) {
            for (int j = 0; j < 6; j++) {
                if (j == 4) {                //如果 j 等于 4 就结束内部循环
                    break;
                }
            }
        }
    }
}
```



```

        break;
    }
    System.out.println("i=" + i + " j=" + j);
}
}
}
}
}

```

运行结果如图 4.19 所示。

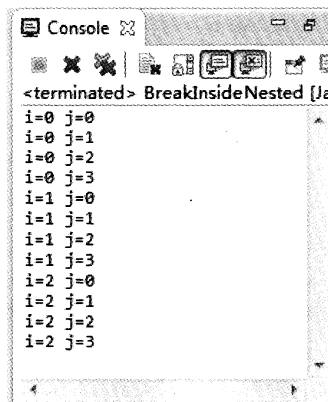


图 4.19 例 4.15 的运行结果

从这个运行结果我们可以看出：

(1) 循环中的 if 语句判断：如果 j 等于 4 时，执行 break 语句，则中断了内层的循环，输出的 j 值最大到 3 为止。

(2) 外层的循环没有受任何影响，输出的 i 值最大为 2，正是 for 循环设定的最大值。

如果想让 break 跳出外层循环，Java 提供了“标签”的功能，语法如下：

```

标签名：循环体{
    break 标签名;
}

```

- ☒ 标签名：任意标识符。
- ☒ 循环体：任意循环语句。
- ☒ break 标签名：break 跳出指定的循环体，此循环体的标签名必须与 break 的标签名一致。带有标签的 break 可以制定跳出的循环，这个循环可以是内层循环，也可以是外层循环。

**【例 4.16】** 用带有标签的 break 跳出外层循环。（实例位置：光盘\TM\sl\4.12）

```

public class BreakOutsideNested {
    public static void main(String[] args) {
        Loop: for (int i = 0; i < 3; i++) {           //在 for 循环前用标签标记
            for (int j = 0; j < 6; j++) {
                if (j == 4) {                          //如果 j 等于 4 就结束外层循环
                    break Loop;                       //跳出 Loop 标签标记的循环体
                }
            }
        }
    }
}

```

```

        System.out.println("i=" + i + " j=" + j);
    }
}
}
}

```

运行结果如图 4.20 所示。

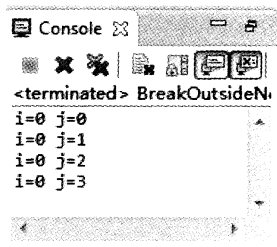


图 4.20 例 4.16 的运行结果

从这个结果我们可以看出，当  $j$  的值等于 4 时， $i$  的值没有继续增加，直接结束外层循环。

## 4.4.2 continue 语句

`continue` 语句是针对 `break` 语句的补充。`continue` 不是立即跳出循环体，而是跳过本次循环结束前的语句，回到循环的条件测试部分，重新开始执行循环。在 `for` 循环语句中遇到 `continue` 后，首先执行循环的增量部分，然后进行条件测试。在 `while` 和 `do...while` 循环中，`continue` 语句使控制直接回到条件测试部分。

在 3 种循环语句中，使用 `continue` 语句的形式如图 4.21 所示。

|                                                         |                                                            |                                                  |
|---------------------------------------------------------|------------------------------------------------------------|--------------------------------------------------|
| <pre>while(...) {     ...     continue;     ... }</pre> | <pre>do {     ...     continue;     ... }while(...);</pre> | <pre>for {     ...     continue;     ... }</pre> |
|---------------------------------------------------------|------------------------------------------------------------|--------------------------------------------------|

图 4.21 `continue` 语句的使用形式

【例 4.17】输出 1~20 之间的奇数，使用 `continue` 跳出循环。（实例位置：光盘\TM\sl\4.13）

```

public class ContinueTest {
    public static void main(String[] args) {
        for (int i = 1; i < 20; i++) {
            if (i % 2 == 0) {
                continue;           //如果 i 是偶数
                                   //跳到下一循环
            }
            System.out.println(i);   //输出 i 的值
        }
    }
}

```

运行结果如图 4.22 所示。

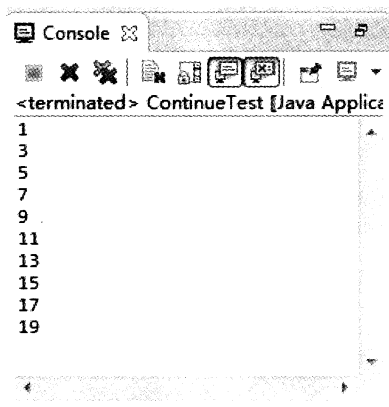


图 4.22 例 4.17 的运行结果

与 break 语句一样，continue 也支持标签功能，语法如下：

```
标签名：循环体{  
    continue 标签名;  
}
```

- ☒ 标签名：任意标识符。
- ☒ 循环体：任意循环语句。
- ☒ continue 标签名：continue 跳出指定的循环体，此循环体的标签名必须与 continue 的标签名一致。

## 4.5 小 结

本章介绍了流程控制语句（复合语句、条件语句和循环语句）；使用复合语句可以为变量定义一个有效区域；通过使用 if 与 switch 语句，可以基于布尔类型的测试，将一个程序分成不同的部分；通过 while、do...while 循环语句和 for 循环语句，可以让程序的一部分重复地执行，直到满足某个终止循环的条件。通过本章的学习，读者应该学会在程序中灵活使用流程控制语句。

## 4.6 实践与练习

1. 编写 Java 程序，实现判断变量 x 是奇数还是偶数。（答案位置：光盘\TM\sl\4.14）
2. 编写 Java 程序，应用 for 循环打印菱形。（答案位置：光盘\TM\sl\4.15）
3. 编写 Java 程序，使用 while 循环语句计算  $1+1/2!+1/3!+\cdots+1/20!$  之和。（答案位置：光盘\TM\sl\4.16）

# 第 5 章

---

## 字符串

(  视频讲解：1 小时 53 分钟 )

字符串是 Java 程序中经常处理的对象，如果字符串运用得不好，将影响到程序运行的效率。在 Java 中字符串作为 String 类的实例来处理。以对象的方式处理字符串，将使字符串更加灵活、方便。了解字符串上可用的操作，可以节省程序编写与维护的时间。

本章从创建字符串开始向读者介绍字符串本身的特性，以及字符串上可用的几个操作等。

通过阅读本章，您可以：

- » 掌握字符串的创建方式
- » 理解字符串连接的方式
- » 掌握获取字符串信息的方式
- » 掌握字符串的常用操作
- » 掌握字符串的格式化
- » 理解正则表达式
- » 掌握字符串生成器的用法

## 5.1 String 类

 视频讲解：光盘\TM\lx\5\String 类.mp4

前面的章节中介绍了 char 类型，它只能表示单个字符，不能表示由多个字符连接而成的字符串。在 Java 语言中将字符串作为对象来处理，可以通过 java.lang 包中的 String 类来创建字符串对象。

### 5.1.1 声明字符串

在 Java 语言中字符串必须包含在一对双引号 (") 之内。例如：

```
"23.23"、"ABCDE"、"你好"
```

以上这些都是字符串常量，字符串常量是系统能够显示的任何文字信息，甚至是单个字符。



在 Java 中由双引号 (") 包围的都是字符串，不能作为其他数据类型来使用，如 "1+2" 的输出结果不可能是 3。

可以通过以下语法格式来声明字符串变量：

```
String str;
```

- ☑ String：指定该变量为字符串类型。
- ☑ str：任意有效的标识符，表示字符串变量的名称。

【例 5.1】声明字符串变量 s，实例代码如下：

```
String s;
```



声明字符串变量必须经过初始化才能使用，否则编译器会报出“变量未被初始化错误”。

### 5.1.2 创建字符串

在 Java 语言中将字符串作为对象来管理，因此可以像创建其他类对象一样来创建字符串对象（关于类与对象，以及构造方法的详细介绍可参见第 7 章内容）。创建对象要使用类的构造方法。String 类的常用构造方法如下：

### (1) String(char a[])

用一个字符数组 a 创建 String 对象。

【例 5.2】 用一个字符数组 a 创建 String 对象，实例代码如下：

```
char a[] = {'g','o','o','d'};
String s = new String(a);
```

等价于

```
String s = new String("good")
```

### (2) String(char a[], int offset, int length)

提取字符数组 a 中的一部分创建一个字符串对象。参数 offset 表示开始截取字符串的位置，length 表示截取字符串的长度。

【例 5.3】 提取字符数组 a 中的一部分创建一个字符串对象，实例代码如下：

```
char a[] = {'s','t','u','d','e','n','t'};
String s = new String(a,2,4);
```

等价于

```
String s = new String("uden");
```

### (3) String(char[] value)

该构造方法可分配一个新的 String 对象，使其表示字符数组参数中所有元素连接的结果。

【例 5.4】 创建字符数组，将数组中的所有元素连接成一个 String 对象，实例代码如下：

```
char a[] = {'s','t','u','d','e','n','t'};
String s = new String(a);
```

等价于

```
String s = new String("student");
```

除通过以上几种使用 String 类的构造方法来创建字符串变量外，还可通过字符串常量的引用赋值给一个字符串变量。

【例 5.5】 引用字符串常量来创建字符串变量，实例代码如下：

```
String str1, str2;
str1 = "We are students";
str2 = "We are students";
```

此时 str1 与 str2 引用相同的字符串常量，因此具有相同的实体。内存示意图如图 5.1 所示。

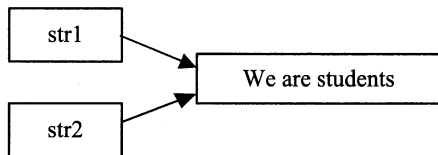


图 5.1 内存示意图

## 5.2 连接字符串

📺 视频讲解：光盘\TM\lx\5\连接字符串.mp4

对于已声明的字符串，可以对其进行相应的操作。连接字符串就是字符操作中较简单的一种。可以对多个字符串进行连接，也可使字符串与其他数据类型进行连接。

## 5.2.1 连接多个字符串

使用“+”运算符可实现连接多个字符串的功能。“+”运算符可以连接多个运算符并产生一个 String 对象。

**【例 5.6】** 在项目中创建类 Join，在主方法中创建 String 型变量，并将字符串连接的结果输出。（实例位置：光盘\TM\sl\5.01）

```
public class Join {  
    public static void main(String args[]) {  
        String s1 = new String("hello");  
        String s2 = new String("word");  
        String s = s1 + " " + s2;  
        System.out.println(s);  
    }  
}  
//创建类  
//主方法  
//声明 String 对象 s1  
//声明 String 对象 s2  
//将对象 s1 和 s2 连接后的结果赋值给 s  
//将 s 输出
```

运行结果如图 5.2 所示。

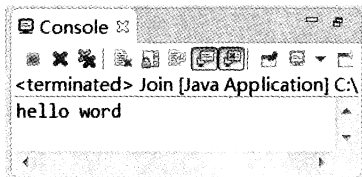


图 5.2 例 5.6 的运行结果



### 技巧

Java 中一句相连的字符串不能分开在两行中写。例如：

```
System.out.println("I like  
Java")
```

这种写法是错误的，如果一个字符串太长，为了便于阅读，可以将这个字符串分在两行上书写。此时就可以使用“+”将两个字符串联起来，之后在加号处换行。因此，上面的语句可以修改为：

```
System.out.println("I like"+  
"Java");
```

## 5.2.2 连接其他数据类型

字符串也可同其他基本数据类型进行连接。如果将字符串同这些数据类型数据进行连接，会将这些数据直接转换成字符串。

**【例 5.7】** 在项目中创建类 Link，在主方法中创建数值型变量，实现将字符串与整型、浮点型变量相连的结果输出。（实例位置：光盘\TM\sl\5.02）

```

public class Link {                                //创建类
    public static void main(String args[]) {        //主方法
        int booktime = 4;                          //声明的 int 型变量 booktime
        float practice = 2.5f;                     //声明的 float 型变量 practice
        //将字符串与整型、浮点型变量相连, 并将结果输出
        System.out.println("我每天花费" + booktime + "小时看书; "
            + practice + "小时上机练习");
    }
}

```

运行结果如图 5.3 所示。

本实例实现的是将字符串常量与整型变量 `booktime` 和浮点型变量 `practice` 相连后的结果输出。在这里 `booktime` 和 `practice` 都不是字符串, 当它们与字符串相连时会自动调用 `toString()` 方法, 将其转换成字符串形式, 然后参与连接。



只要“+”运算符的一个操作数是字符串, 编译器就会将另一个操作数转换成字符串形式, 所以应谨慎地将其他数据类型与字符串相连, 以免出现意想不到的结果。

如果将上例中的输出语句修改为:

```

System.out.println("我每天花费"+booktime+"小时看书;" + (practice+booktime)+
    "小时上机练习");

```

则例 5.7 修改后的运行结果如图 5.4 所示。

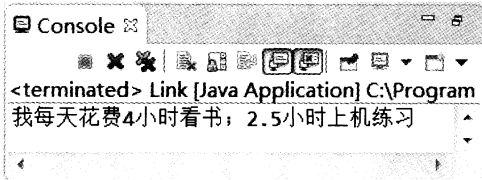


图 5.3 例 5.7 的运行结果

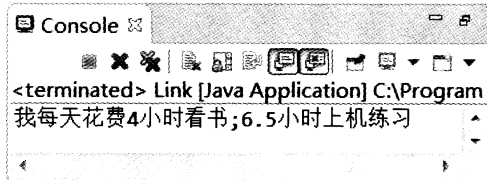


图 5.4 输出语句修改后例 5.7 的运行结果

为什么会这样呢? 这是由于运算符是有优先级的, 圆括号的优先级最高, 所以先被执行, 然后再将结果与字符串相连。

## 5.3 获取字符串信息

字符串作为对象, 可通过相应方法获取字符串的有效信息, 如获取某字符串的长度、某个索引位置的字符等。本节将介绍几种获取字符串的相关信息的方法。



### 5.3.1 获取字符串长度

 视频讲解：光盘\TM\lx\5\获取字符串长度.mp4

使用 `String` 类的 `length()` 方法可获取声明的字符串对象的长度。

语法如下：

```
str.length();
```

其中，`str` 为字符串对象。

【例 5.8】 获取字符串长度，实例代码如下：

```
String str = "We are students";  
int size = str.length();
```

上段代码是将字符串 `str` 的长度赋值给 `int` 型变量 `size`，此时变量 `size` 的值为 15，这表示 `length()` 方法返回的字符串的长度包括字符串中的空格。

### 5.3.2 字符串查找

 视频讲解：光盘\TM\lx\5\字符串查找.mp4

`String` 类提供了两种查找字符串的方法，即 `indexOf()` 与 `lastIndexOf()` 方法。这两种方法都允许在字符串中搜索指定条件的字符或字符串。`indexOf()` 方法返回的是搜索的字符或字符串首次出现的位置，`lastIndexOf()` 方法返回的是搜索的字符或字符串最后一次出现的位置。

#### (1) `indexOf(String s)`

该方法用于返回参数字符串 `s` 在指定字符串中首次出现的索引位置。当调用字符串的 `indexOf()` 方法时，会从当前字符串的开始位置搜索 `s` 的位置；如果没有检索到字符串 `s`，该方法的返回值是 -1。

语法如下：

```
str.indexOf(substr)
```

☑ `str`：任意字符串对象。

☑ `substr`：要搜索的字符串。

【例 5.9】 查找字符 `a` 在字符串 `str` 中的索引位置，实例代码如下：

```
String str = "We are students";  
int size = str.indexOf("a"); //变量 size 的值是 3
```

理解字符串的索引位置，要对字符串的下标有所了解。在计算机中 `String` 对象是用数组表示的。字符串的下标是 `0~length()-1`。例 5.9 中字符串 `str` 的下标如图 5.5 所示。

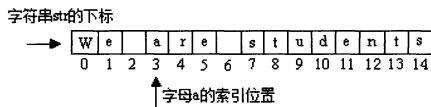


图 5.5 字符串 `str` 下标

## (2) lastIndexOf(String str)

该方法用于返回指定字符串最后一次出现的索引位置。当调用字符串的 `lastIndexOf()` 方法时，会从当前字符串的开始位置检索参数字符串 `str`，并将最后一次出现 `str` 的索引位置返回。如果没有检索到字符串 `str`，该方法返回 -1。

语法如下：

```
str.lastIndexOf(substr)
```

- ☑ `str`：任意字符串对象。
- ☑ `substr`：要搜索的字符串。

### 说明

如果 `lastIndexOf()` 方法中的参数是空字符串 ""（注意没有空格），则返回的结果与调用该字符串 `length()` 方法的返回结果相同。例 5.10 的程序就可说明这个问题。

**【例 5.10】** 在项目中创建类 `Text`，在主方法中创建 `String` 对象，使用 `lastIndexOf()` 方法查看字符串 `str` 中空字符串的位置，然后输出字符串的长度，看它们是否相同。（实例位置：光盘\TM\sl\5.03）

```
public class Text {                                //创建类
    public static void main(String args[]) {        //主方法
        String str = "We are students";            //定义字符串 str
        //将空字符串在 str 中的索引位置赋值给变量 size
        int size = str.lastIndexOf("");
        //将变量 size 输出
        System.out.println("空字符在字符串 str 中的索引位置是：" + size);
        //将字符串 str 的长度输出
        System.out.println("字符串 str 的长度是：" + str.length());
    }
}
```

运行结果如图 5.6 所示。

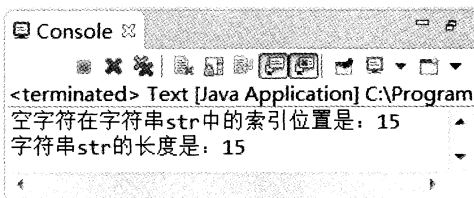


图 5.6 例 5.10 的运行结果

## 5.3.3 获取指定索引位置的字符

 视频讲解：光盘\TM\lx\5\获取指定索引位置的字符.mp4

使用 `charAt()` 方法可将指定索引处的字符返回。

语法如下：

```
str.charAt(int index)
```

- ☑ str: 任意字符串。
- ☑ index: 整型值, 用于指定要返回字符的下标。

【例 5.11】 在项目中创建类 Ref, 在主方法中创建 String 对象, 使用 charAt()方法查看字符串 str 中索引位置是 6 的字符。(实例位置: 光盘\TM\sl\5.04)

```
public class Ref {                                //创建类
    public static void main(String args[]) {      //主方法
        String str = "hello word";               //定义字符串 str
        char mychar = str.charAt(6);              //将字符串 str 中索引位置是 6 的字符返回
        System.out.println("字符串 str 中索引位置是 6 的字符为: " + mychar); //输出信息
    }
}
```

运行结果如图 5.7 所示。

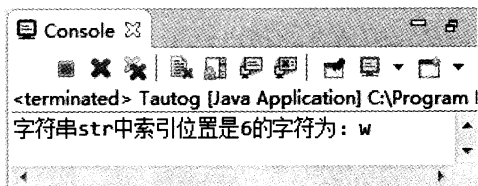


图 5.7 例 5.11 的运行结果

## 5.4 字符串操作

String 类中包含了很多方法, 允许程序员对字符串进行操作来满足实际编程中的需要。本节将介绍几种常见的字符串操作。

### 5.4.1 获取子字符串

📺 视频讲解: 光盘\TM\lx\5\获取子字符串.mp4

通过 String 类的 substring()方法可对字符串进行截取。这些方法的共同点就是都利用字符串的下标进行截取, 且应明确字符串下标是从 0 开始的。

substring()方法被两种不同的方法重载, 来满足不同的需要。

#### (1) substring(int beginIndex)

该方法返回的是从指定的索引位置开始截取直到该字符串结尾的子串。

语法如下:

```
str.substring(int beginIndex)
```

其中, beginIndex 指定从某一索引处开始截取字符串。

【例 5.12】 截取字符串, 实例代码如下:

```
String str = "Hello Word";           //定义字符串 str
String substr = str.substring(3);     //获取字符串, 此时 substr 值为 lo Word
```

使用 `substring(beginIndex)` 截取字符串的过程如图 5.8 所示。

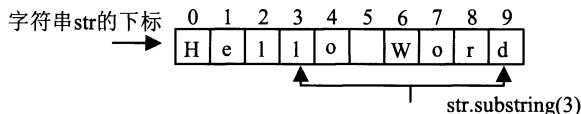


图 5.8 `substring(3)` 的截取过程

### 注意

在字符串中空格占用一个索引位置。

### (2) `substring(int beginIndex, int endIndex)`

该方法返回的是从字符串某一索引位置开始截取至某一索引位置结束的子串。

语法如下：

```
substring(int beginIndex, int endIndex)
```

- ☑ `beginIndex`: 开始截取子字符串的索引位置。
- ☑ `endIndex`: 子字符串在整个字符串中的结束位置。

【例 5.13】在项目中创建类 `Subs`，在主方法中创建 `String` 对象，实现使用 `substring()` 方法对字符串进行截取，并将截取后形成的新串输出。（实例位置：光盘\TM\sl\5.05）

```
public class Subs {           //创建类
    public static void main(String args[]) { //主方法
        String str = "hello word"; //定义的字符串
        String substr = str.substring(0, 3); //对字符串进行截取
        System.out.println(substr); //输出截取后的字符串
    }
}
```

运行结果如图 5.9 所示。

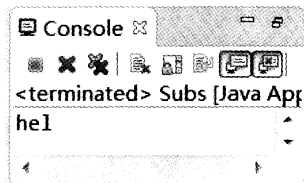


图 5.9 例 5.13 的运行结果

## 5.4.2 去除空格

📺 视频讲解：光盘\TM\lx\5\去除空格.mp4

`trim()` 方法返回字符串的副本，忽略前导空格和尾部空格。

语法如下：

```
str.trim()
```

其中，str 为任意字符串对象。

**【例 5.14】** 在项目中创建类 Blak，在主方法中创建 String 对象，将字符变量原来的长度与去掉前导和尾部空格后的长度输出。（实例位置：光盘\TM\sl\5.06）

```
public class Blak {                                //创建类
    public static void main(String args[]) {        //主方法
        String str = " Java class ";               //定义字符串 str
        System.out.println("字符串原来的长度: " + str.length()); //将 str 原来的长度输出
        //将 str 去掉前导和尾部的空格后的结果输出
        System.out.println("去掉空格后的长度: " + str.trim().length());
    }
}
```

运行结果如图 5.10 所示。

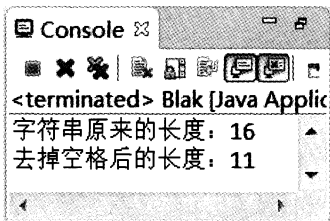


图 5.10 例 5.14 的运行结果

### 5.4.3 字符串替换

 视频讲解：光盘\TM\lx\5\字符串替换.mp4

replace()方法可实现将指定的字符或字符串替换成新的字符或字符串。

语法如下：

```
str.replace(char oldChar,char newChar)
```

- ☒ oldChar: 要替换的字符或字符串。
- ☒ newChar: 用于替换原来字符串的内容。

replace()方法返回的结果是一个新的字符串。如果字符串 oldChar 没有出现在该对象表达式中的字符串序列中，则将原字符串返回。

**【例 5.15】** 在项目中创建类 NewStr，在主方法中创建 String 型变量，将字符变量中的字母 a 替换成 A 后的结果输出。（实例位置：光盘\TM\sl\5.07）

```
public class NewStr {                                //创建类
    public static void main(String args[]) {        //主方法
        String str = "address";                     //定义字符串 str
        //字符串 str 中的字符 "a" 替换成 "A" 后返回的新字符串 newstr
        String newstr = str.replace("a", "A");
    }
}
```

```

        System.out.println(newstr);           //将字符串 newstr 输出
    }
}

```

运行结果如图 5.11 所示。

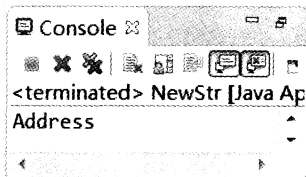


图 5.11 例 5.15 的运行结果

### 说明

如果要替换的字符 `oldChar` 在字符串中重复出现多次, `replace()` 方法会将所有 `oldChar` 全部替换成 `newChar`。例如:

```
String str = "java project";
String newstr = str.replace("j","J");
```

此时, `newstr` 的值为 `Java proJect`。

需要注意的是, 要替换的字符 `oldChar` 的大小写要与原字符串中字符的大小写保持一致, 否则不能成功地替换。例如, 上面的实例如果写成如下语句, 则不能成功替换。

```
String str = "java project";
String newstr = str.replace("P","t");
```

## 5.4.4 判断字符串的开始与结尾

 视频讲解: 光盘\TM\lx\5\判断字符串的开始与结尾.mp4

`startsWith()` 方法与 `endsWith()` 方法分别用于判断字符串是否以指定的内容开始或结束。这两个方法的返回值都为 `boolean` 类型。

### (1) `startsWith()` 方法

该方法用于判断当前字符串对象的前缀是否为参数指定的字符串。

语法如下:

```
str.startsWith(String prefix)
```

其中, `prefix` 是指作为前缀的字符。

### (2) `endsWith()` 方法

该方法用于判断当前字符串是否为以给定的子字符串结束。

语法如下:

```
str.endsWith(String suffix)
```

其中, suffix 是指作为后缀的字符串。

**【例 5.16】** 在项目中创建类 StartOrEnd, 在主方法中创建 String 型变量, 并判断变量的前导和后置字符串。(实例位置: 光盘\TM\sl\5.08)

```
public class StartOrEnd {                                //创建类
    public static void main(String args[]) {              //主方法
        String num1 = "22045612";                        //定义字符串 num1
        String num2 = "21304578";                        //定义字符串 num2
        boolean b = num1.startsWith("22");              //判断字符串 num1 是否以'22'开头
        boolean b2 = num1.endsWith("78");               //判断字符串 num1 是否以'78'结束
        boolean b3 = num2.startsWith("22");             //判断字符串 num2 是否以'22'开头
        boolean b4 = num2.endsWith("78");               //判断字符串 num2 是否以'78'结束
        System.out.println("字符串 num1 是以'22'开始的吗? " + b);
        System.out.println("字符串 num1 是以'78'结束的吗? " + b2); //输出信息
        System.out.println("字符串 num2 是以'22'开始的吗? " + b3);
        System.out.println("字符串 num2 是以'78'结束的吗? " + b4);
    }
}
```

运行结果如图 5.12 所示。

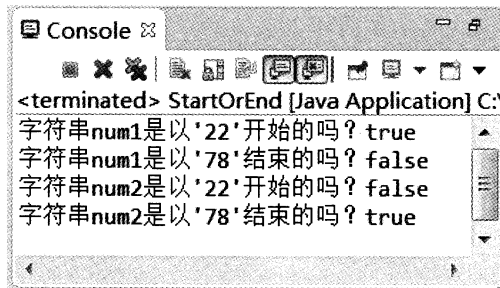


图 5.12 例 5.16 的运行结果

### 5.4.5 判断字符串是否相等

 视频讲解: 光盘\TM\lx\5\判断字符串是否相等.mp4

对字符串对象进行比较不能简单地使用比较运算符“==”, 因为比较运算符比较的是两个字符串的地址是否相同。即使两个字符串的内容相同, 两个对象的内存地址也是不同的, 使用比较运算符仍然会返回 false。

**【例 5.17】** 使用比较运算符比较两个字符串, 实例代码如下:

```
String tom = new String("I am a student");
String jerry = new String("I am a student");
boolean b = (tom == jerry);
```

此时, 布尔型变量 b 的值为 false, 因为字符串是对象, tom、jerry 是引用。内存示意图如图 5.13 所示。



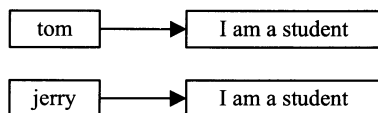


图 5.13 内存示意图

因此, 要比较两个字符串内容是否相等, 应使用 `equals()` 方法和 `equalsIgnoreCase()` 方法。

#### (1) `equals()` 方法

如果两个字符串具有相同的字符和长度, 则使用 `equals()` 方法进行比较时, 返回 `true`。

语法如下:

```
str.equals(String otherstr)
```

其中, `str`、`otherstr` 是要比较的两个字符串对象。

#### (2) `equalsIgnoreCase()` 方法

使用 `equals()` 方法对字符串进行比较时是区分大小写的, 而使用 `equalsIgnoreCase()` 方法是在忽略了大小写的情况下比较两个字符串是否相等, 返回结果仍为 `boolean` 类型。

语法如下:

```
str.equalsIgnoreCase(String otherstr)
```

其中, `str`、`otherstr` 是要比较的两个字符串对象。

通过下面的例子可以看出 `equals()` 方法和 `equalsIgnoreCase()` 方法的区别。

**【例 5.18】** 在项目中创建类 `Opinion`, 在主方法中创建 `String` 型变量, 实现判断两个字符串是否相等, 并将结果输出。(实例位置: 光盘\TM\sl\5.09)

```

public class Opinion {                                //创建类
    public static void main(String args[]) {           //主方法
        String s1 = new String("abc");                 //创建字符串对象 s1
        String s2 = new String("ABC");                 //创建字符串对象 s2
        String s3 = new String("abc");                 //创建字符串对象 s3
        boolean b = s1.equals(s2);                     //使用 equals()方法比较 s1 与 s2
        //使用 equalsIgnoreCase()方法比较 s1 与 s2
        boolean b2 = s1.equalsIgnoreCase(s2);
        System.out.println(s1 + " equals " + s2 + " : " + b); //输出信息
        System.out.println(s1 + " equalsIgnoreCase " + s2 + " : " + b2);
    }
}
  
```

运行结果如图 5.14 所示。

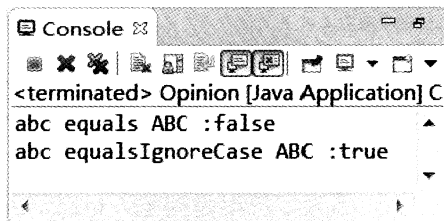


图 5.14 例 5.18 的运行结果



### 5.4.6 按字典顺序比较两个字符串

`compareTo()`方法为按字典顺序比较两个字符串，该比较基于字符串中各个字符的 Unicode 值，按字典顺序将此 `String` 对象表示的字符序列与参数字符串所表示的字符序列进行比较。如果按字典顺序此 `String` 对象位于参数字符串之前，则比较结果为一个负整数；如果按字典顺序此 `String` 对象位于参数字符串之后，则比较结果为一个正整数；如果这两个字符串相等，则结果为 0。

语法如下：

```
str.compareTo(String otherstr)
```

其中，`str`、`otherstr` 是要比较的两个字符串对象。

#### 说明

`compareTo()`方法只有在 `equals(Object)`方法返回 `true` 时才返回 0。

【例 5.19】在项目中创建类 `Wordbook`，在主方法中创建 `String` 变量，使用 `compareTo()`方法将字符串变量进行比较，并将比较结果输出。（实例位置：光盘\TM\5.10）

```
public class Wordbook {           //创建类
    public static void main(String args[]) { //主方法
        String str = new String("b");
        String str2 = new String("a");
        String str3 = new String("c"); } //用于比较的 3 个字符串
        System.out.println(str + " compareTo " + str2 + ":"
            + str.compareTo(str2)); //将 str 与 str2 比较的结果输出
        System.out.println(str + " compareTo " + str3 + ":"
            + str.compareTo(str3)); //将 str 与 str3 比较的结果输出
    }
}
```

运行结果如图 5.15 所示。

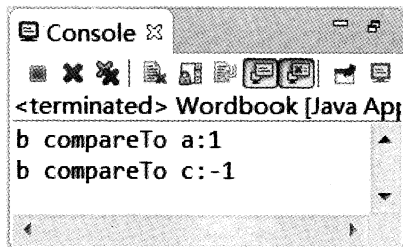


图 5.15 例 5.19 的运行结果

## 5.4.7 字母大小写转换

 视频讲解：光盘\TM\lx\5\字母大小写转换.mp4

字符串的 `toLowerCase()` 方法可将字符串中的所有字符从大写字母改写为小写字母，而 `toUpperCase()` 方法可将字符串中的小写字母改写为大写字母。

### (1) `toLowerCase()` 方法

该方法将 `String` 转换为小写。如果字符串中没有应该被转换的字符，则将原字符串返回；否则返回一个新的字符串，将原字符串中每个该进行小写转换的字符都转换成等价的小写字符。字符长度与原字符串长度相同。

语法如下：

```
str.toLowerCase()
```

其中，`str` 是要进行转换的字符串。

### (2) `toUpperCase()` 方法

该方法将 `String` 转换为大写。如果字符串中没有应该被转换的字符，则将原字符串返回；否则返回一个新字符串，将原字符串中每个该进行大写转换的字符都转换成等价的大写字符。新字符串长度与原字符串长度相同。

语法如下：

```
str.toUpperCase()
```

其中，`str` 是要进行转换的字符串。



### 说明

使用 `toLowerCase()` 方法和 `toUpperCase()` 方法进行大小写转换时，数字或非字符不受影响。

**【例 5.20】** 在项目中创建类 `UpAndLower`，在主方法中创建 `String` 型变量，实现字符变量的大小写转换，并将转换后的结果输出。（实例位置：光盘\TM\sl\5.11）

```
public class UpAndLower {
    public static void main(String args[]) {
        String str = new String("abc DEF");
        String newstr = str.toLowerCase();
        String newstr2 = str.toUpperCase();
        System.out.println(newstr);
        System.out.println(newstr2);
    }
}
```

//创建类  
//主方法  
//创建的字符串 str  
//使用 `toLowerCase()` 方法实行小写转换  
//使用 `toUpperCase()` 方法实行大写转换  
//将转换后的结果输出

运行结果如图 5.16 所示。

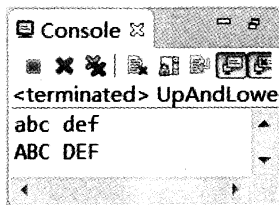


图 5.16 例 5.20 的运行结果

## 5.4.8 字符串分割

视频讲解：光盘\TM\lx\5\字符串分割.mp4

使用 `split()` 方法可以使字符串按指定的分割字符或字符串对内容进行分割，并将分割后的结果存放在字符串数组中。`split()` 方法提供了以下两种字符串分割形式。

### (1) `split(String sign)`

该方法可根据给定的分割符对字符串进行拆分。

语法如下：

```
str.split(String sign)
```

其中，`sign` 为分割字符串的分割符，也可以使用正则表达式。

### 说明

没有统一的对字符串进行分割的符号。如果想定义多个分割符，可使用符号“|”。例如，“`;,|=`”表示分割符分别为“`;`”和“`=`”。

### (2) `split(String sign,int limit)`

该方法可根据给定的分割符对字符串进行拆分，并限定拆分的次数。

语法如下：

```
str.split(String sign,int limit)
```

☑ `sign`：分割字符串的分割符，也可以使用正则表达式。

☑ `limit`：限制的分割次数。

**【例 5.21】** 在项目中创建类 `Division`，在主方法中创建 `String` 型变量，并将字符串变量进行分割，将分割后的结果输出。（实例位置：光盘\TM\sl\5.12）

```
public class Division {
    public static void main(String[] args) {
        //创建字符串
        String str = "192.168.0.1";
        //按照 "." 进行分割，使用转义字符 "\\."
        String[] firstArray = str.split("\\.");
        //按照 "." 进行两次分割，使用转义字符 "\\."
        String[] secondArray = str.split("\\.", 2);
    }
}
```

```

//输出 str 原值
System.out.println("str 的原值为: [" + str + "]");
//输出全部分割的结果
System.out.print("全部分割的结果: ");
for (String a : firstArray) {
    System.out.print "[" + a + ""];
}
System.out.println();// 换行
//输出分割两次的结果
System.out.print("分割两次的结果: ");
for (String a : secondArray) {
    System.out.print "[" + a + ""];
}
System.out.println();
}
}

```

运行结果如图 5.17 所示。

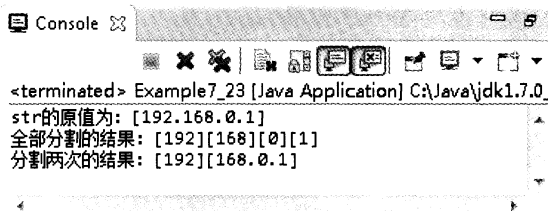


图 5.17 例 5.21 的运行结果

## 5.5 格式化字符串

String 类的静态 format()方法用于创建格式化的字符串。format()方法有两种重载形式。

(1) format(String format, Object...args)

该方法使用指定的格式字符串和参数返回一个格式化字符串，格式化后的新字符串使用本地默认的语言环境。

语法如下：

```
str.format(String format, Object...args)
```

- ☑ format: 格式字符串。
- ☑ args: 格式字符串中由格式说明符引用的参数。如果还有格式说明符以外的参数，则忽略这些额外的参数。此参数的数目是可变的，可以为 0。

(2) format(Locale l, String format, Object...args)

- ☑ l: 格式化过程中要应用的语言环境。如果 l 为 null，则不进行本地化。

- ☑ **format**: 格式字符串。
- ☑ **args**: 格式字符串中由格式说明符引用的参数。如果还有格式说明符以外的参数, 则忽略这些额外的参数。此参数的数目是可变的, 可以为 0。

## 5.5.1 日期和时间字符串格式化

 视频讲解: 光盘\TM\lx\5\日期和时间字符串格式化.mp4

在应用程序设计中, 经常需要显示时间和日期。如果想输出满意的日期和时间格式, 一般需要编写大量的代码经过各种算法才能实现。`format()`方法通过给定的特殊转换符作为参数来实现对日期和时间的格式化。

### 1. 日期格式化

先来看下面的例子。

**【例 5.22】** 返回一个月中的天数, 实例代码如下:

```
Date date = new Date();           //创建 Date 对象 date
String s = String.format("%te", date); //通过 format()方法对 date 进行格式化
```

上述代码中变量 `s` 的值是当前日期中的天数, 如今天是 15 号, 则 `s` 的值为 15; `%te` 是转换符。常用的日期格式化转换符如表 5.1 所示。

表 5.1 常用的日期格式化转换符

| 转 换 符            | 说 明               | 示 例                         |
|------------------|-------------------|-----------------------------|
| <code>%te</code> | 一个月中的某一天 (1~31)   | 2                           |
| <code>%tb</code> | 指定语言环境的月份简称       | Feb (英文)、二月 (中文)            |
| <code>%tB</code> | 指定语言环境的月份全称       | February (英文)、二月 (中文)       |
| <code>%tA</code> | 指定语言环境的星期几全称      | Monday (英文)、星期一 (中文)        |
| <code>%tA</code> | 指定语言环境的星期几简称      | Mon (英文)、星期一 (中文)           |
| <code>%tc</code> | 包括全部日期和时间信息       | 星期二 三月 25 13:37:22 CST 2008 |
| <code>%tY</code> | 4 位年份             | 2008                        |
| <code>%tj</code> | 一年中的第几天 (001~366) | 085                         |
| <code>%tm</code> | 月份                | 03                          |
| <code>%td</code> | 一个月中的第几天 (01~31)  | 02                          |
| <code>%ty</code> | 2 位年份             | 08                          |

**【例 5.23】** 在项目中创建类 `Eval`, 实现将当前日期信息以 4 位年份、月份全称、2 位日期形式输出。(实例位置: 光盘\TM\sl\5.13)

```
import java.util.Date;           //导入 java.util.Date 类
public class Eval {              //新建类
    public static void main(String[] args) { //主方法
        Date date = new Date();           //创建 Date 对象 date
    }
}
```

```

String year = String.format("%tY", date);    //将 date 进行格式化
String month = String.format("%tB", date);
String day = String.format("%td", date);
System.out.println("今年是: " + year + "年"); //输出信息
System.out.println("现在是: " + month);
System.out.println("今天是: " + day + "号");
    }
}

```

运行结果如图 5.18 所示。

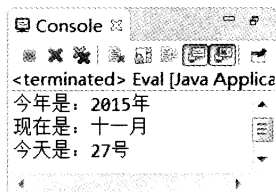


图 5.18 例 5.23 的运行结果

## 2. 时间格式化

使用 `format()` 方法不仅可以完成日期的格式化，也可以实现时间的格式化。时间格式化转换符要比日期转换符更多、更精确，它可以将时间格式化为时、分、秒、毫秒。格式化时间的转换符如表 5.2 所示。

表 5.2 时间格式化转换符

| 转 换 符 | 说 明                            | 示 例           |
|-------|--------------------------------|---------------|
| %tH   | 2 位数字的 24 时制的小时（00~23）         | 14            |
| %tI   | 2 位数字的 12 时制的小时（01~12）         | 05            |
| %tk   | 2 位数字的 24 时制的小时（0~23）          | 5             |
| %tI   | 2 位数字的 12 时制的小时（1~12）          | 10            |
| %tM   | 2 位数字的分钟（00~59）                | 05            |
| %tS   | 2 位数字的秒数（00~60）                | 12            |
| %tL   | 3 位数字的毫秒数（000~999）             | 920           |
| %tN   | 9 位数字的微秒数（000000000~999999999） | 062000000     |
| %tp   | 指定语言环境下上午或下午标记                 | 下午（中文）、pm（英文） |
| %tz   | 相对于 GMT RFC 82 格式的数字时区偏移量      | +0800         |
| %tZ   | 时区缩写形式的字符串                     | CST           |
| %ts   | 1970-01-01 00:00:00 至现在经过的秒数   | 1206426646    |
| %tQ   | 1970-01-01 00:00:00 至现在经过的毫秒数  | 1206426737453 |

**【例 5.24】** 在项目中创建类 `GetDate`，实现将当前时间信息以 2 位小时数、2 位分钟数、2 位秒数形式输出。（实例位置：光盘\TM\sl\5.14）

```

import java.util.Date;    //导入 java.util.Date 类
public class GetDate {    //新建类
    public static void main(String[] args) { //主方法

```

```

Date date = new Date();           //创建 Date 对象 date
String hour = String.format("%tH", date); //将 date 进行格式化
String minute = String.format("%tM", date);
String second = String.format("%tS", date);
//输出的信息
System.out.println("现在是: " + hour + "时" + minute + "分"
                  + second + "秒");
}

```

运行结果如图 5.19 所示。

### 3. 格式化常见的日期时间组合

格式化日期与时间的转换符定义了各种日期时间组合的格式，其中最常用的日期和时间的组合格式如表 5.3 所示。

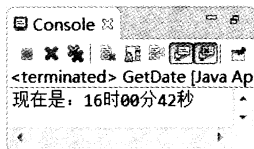


图 5.19 例 5.24 的运行结果

表 5.3 常见的日期和时间组合的格式

| 转 换 符 | 说 明                      | 示 例                         |
|-------|--------------------------|-----------------------------|
| %tF   | “年-月-日”格式（4 位年份）         | 2008-03-25                  |
| %tD   | “月/日/年”格式（2 位年份）         | 03/25/08                    |
| %tc   | 全部日期和时间信息                | 星期二 三月 25 15:20:00 CST 2008 |
| %tr   | “时:分:秒 PM (AM)”格式（12 时制） | 03:22:06 下午                 |
| %tT   | “时:分:秒”格式（24 时制）         | 15:23:50                    |
| %tR   | “时:分”格式（24 时制）           | 15:25                       |

【例 5.25】在项目中创建类 DateAndTime，在主方法中实现将当前日期时间的全部信息以及指定格式的日期输出。（实例位置：光盘\TM\sl\5.15）

```

import java.util.Date;           //导入 java.util.Date 类
public class DateAndTime {       //创建类
    public static void main(String[] args) { //主方法
        Date date = new Date();           //创建 Date 对象 date
        String time = String.format("%tc", date); //将 date 格式化
        String form = String.format("%tF", date);
        //将格式化后的日期时间输出
        System.out.println("全部的时间信息是: " + time);
        System.out.println("年-月-日格式: " + form);
    }
}

```

运行结果如图 5.20 所示。

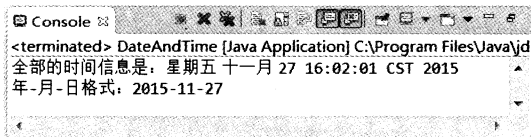


图 5.20 例 5.25 的运行结果



## 5.5.2 常规类型格式化

 视频讲解：光盘\TM\lx\5\常规类型格式化.mp4

常规类型的格式化可应用于任何参数类型，可通过如表 5.4 所示的转换符来实现。

表 5.4 常规转换符

| 转 换 符 | 说 明                     | 示 例                 |
|-------|-------------------------|---------------------|
| %b、%B | 结果被格式化为布尔类型             | true                |
| %h、%H | 结果被格式化为散列码              | A05A5198            |
| %s、%S | 结果被格式化为字符串类型            | "abcd"              |
| %c、%C | 结果被格式化为字符类型             | 'a'                 |
| %d    | 结果被格式化为十进制整数            | 40                  |
| %o    | 结果被格式化为八进制整数            | 11                  |
| %x、%X | 结果被格式化为十六进制整数           | 4b1                 |
| %e    | 结果被格式化为用计算机科学记数法表示的十进制数 | 1.700000e+01        |
| %a    | 结果被格式化为带有效位数和指数的十六进制浮点值 | 0X1.C000000000001P4 |
| %n    | 结果为特定于平台的行分隔符           |                     |
| %%    | 结果为字面值'%'               | %                   |

**【例 5.26】** 在项目中创建类 General，在主方法中实现不同数据类型到字符串的转换。（实例位置：光盘\TM\sl\5.16）

```

public class General {                                //新建类
    public static void main(String[] args) {          //主方法
        String str = String.format("%d", 400 / 2);    //将结果以十进制格式显示
        String str2 = String.format("%b", 3 > 5);     //将结果以 boolean 型显示
        String str3 = String.format("%x", 200);       //将结果以十六进制格式显示
        System.out.println("400 的一半是：" + str);   //输出格式化字符串
        System.out.println("3>5 正确吗：" + str2);
        System.out.println("200 的十六进制数是：" + str3);
    }
}

```

运行结果如图 5.21 所示。

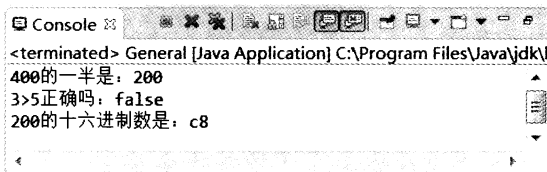


图 5.21 例 5.26 的运行结果



## 5.6 使用正则表达式

 视频讲解：光盘\TM\lx\5\使用正则表达式.mp4

正则表达式通常被用于判断语句中，用来检查某一字符串是否满足某一格式。正则表达式是含有一些具有特殊意义字符的字符串，这些特殊字符称为正则表达式的元字符。例如，“\d”表示数字0~9中的任何一个，“\d”就是元字符。正则表达式中元字符及其意义如表5.5所示。

表 5.5 正则表达式中的元字符

| 元 字 符     | 正则表达式中的写法  | 意 义                                     |
|-----------|------------|-----------------------------------------|
| .         | .          | 代表任意一个字符                                |
| \d        | \\d        | 代表0~9的任何一个数字                            |
| \D        | \\D        | 代表任何一个非数字字符                             |
| \s        | \\s        | 代表空白字符，如'\t'、'\n'                       |
| \S        | \\S        | 代表非空白字符                                 |
| \w        | \\w        | 代表可用作标识符的字符，但不包括“\$”                    |
| \W        | \\W        | 代表不可用于标识符的字符                            |
| \p{Lower} | \\p{Lower} | 代表小写字母a~z                               |
| \p{Upper} | \\p{Upper} | 代表大写字母A~Z                               |
| \p{ASCII} | \\p{ASCII} | ASCII 字符                                |
| \p{Alpha} | \\p{Alpha} | 字母字符                                    |
| \p{Digit} | \\p{Digit} | 十进制数字，即0~9                              |
| \p{Alnum} | \\p{Alnum} | 数字或字母字符                                 |
| \p{Punct} | \\p{Punct} | 标点符号：!"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~ |
| \p{Graph} | \\p{Graph} | 可见字符：[\p{Alnum}\p{Punct}]               |
| \p{Print} | \\p{Print} | 可打印字符：[\p{Graph}\x20]                   |
| \p{Blank} | \\p{Blank} | 空格或制表符：[\t]                             |
| \p{Cntrl} | \\p{Cntrl} | 控制字符：[\x00-\x1F\x7F]                    |



### 说明

在正则表达式中“.”代表任何一个字符，因此在正则表达式中如果想使用普通意义的点字符“.”，必须使用转义字符“\”。

在正则表达式中可以使用方括号括起若干个字符来表示一个元字符，该元字符可代表方括号中的任何一个字符。例如，`reg = "[abc]4"`，这样字符串a4、b4、c4都是和正则表达式匹配的字符串。方括号元字符还可以为其他格式。如：

- ☒ `[^456]`：代表4、5、6之外的任何字符。
- ☒ `[a-r]`：代表a~r中的任何一个字母。

- ☑ [a-zA-Z]: 可表示任意一个英文字母。
- ☑ [a-e[g-z]]: 代表 a~e, 或 g~z 中的任何一个字母（并运算）。
- ☑ [a-o&&[def]]: 代表字母 d、e、f（交运算）。
- ☑ [a-d&&[^bc]]: 代表字母 a、d（差运算）。

在正则表达式中允许使用限定修饰符来限定元字符出现的次数。例如，“A\*”代表 A 可在字符串中出现 0 次或多次。限定修饰符的用法如表 5.6 所示。

表 5.6 限定修饰符

| 限定修饰符 | 意 义      | 示 例    |
|-------|----------|--------|
| ?     | 0 次或 1 次 | A?     |
| *     | 0 次或多次   | A*     |
| +     | 一次或多次    | A+     |
| {n}   | 正好出现 n 次 | A{2}   |
| {n,}  | 至少出现 n 次 | A{3,}  |
| {n,m} | 出现 n~m 次 | A{2,6} |

【例 5.27】 在项目中创建类 Judge，在主方法中实现使用正则表达式来判断指定的变量是否为合法的 E-mail 地址。（实例位置：光盘\TM\sl\5.17）

```
public class Judge {
    public static void main(String[] args) {
        //定义要匹配 E-mail 地址的正则表达式
        String regex = "\\w+@\\w+(\\.\\w{2,3})\\.\\w{2,3}";
        String str1 = "aaa@";           //定义要进行验证的字符串
        String str2 = "aaaaa";
        String str3 = "1111@111ffyu.dfg.com";
        if (str1.matches(regex)) {      //判断字符串变量是否与正则表达式匹配
            System.out.println(str1 + "是一个合法的 E-mail 地址格式");
        }
        if (str2.matches(regex)) {
            System.out.println(str2 + "是一个合法的 E-mail 地址格式");
        }
        if (str3.matches(regex)) {
            System.out.println(str3 + "是一个合法的 E-mail 地址格式");
        }
    }
}
```

运行结果如图 5.22 所示。

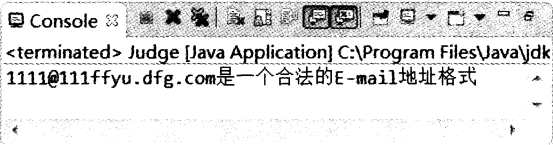


图 5.22 例 5.27 的运行结果

正则表达式分析:

通常情况下 E-mail 的格式为 “X@X.com.cn”。字符 X 表示任意的一个或多个字符, @为 E-mail 地址中的特有符号, 符号@后还有一个或多个字符, 之后是字符 “.com”, 也可能后面还有类似 “.cn” 的标记。总结 E-mail 地址的这些特点, 因此可以书写正则表达式 “\\w+@\\w+(\\.\\w{2,3})\*\\.\\w{2,3}” 来匹配 E-mail 地址。字符集 “\\w” 匹配任意字符, 符号 “+” 表示字符可以出现 1 次或多次, 表达式 “(\\.\\w{2,3})\*” 表示形如 “.com” 格式的字符串可以出现 0 次或多次。而最后的表达式 “\\.\\w{2,3}” 用于匹配 E-mail 地址中的结尾字符, 如 “.com”。

## 5.7 字符串生成器

 视频讲解: 光盘\TM\lx\5\字符串生成器.mp4

创建成功的字符串对象, 其长度是固定的, 内容不能被改变和编译。虽然使用 “+” 可以达到附加新字符或字符串的目的, 但 “+” 会产生一个新的 String 实例, 会在内存中创建新的字符串对象。如果重复地对字符串进行修改, 将极大地增加系统开销。而 J2SE 5.0 新增了可变的字符序列 String- Builder 类, 大大提高了频繁增加字符串的效率。

【例 5.28】在项目中创建类 Jerque, 在主方法中编写如下代码, 验证字符串操作和字符串生成器操作的效率。(实例位置: 光盘\TM\sl\5.18)

```
public class Jerque {
    public static void main(String[] args) {
        String str = "";
        //定义对字符串执行操作的起始时间
        long starTime = System.currentTimeMillis();
        for (int i = 0; i < 10000; i++) {
            str = str + i;
        }
        long endTime = System.currentTimeMillis();
        long time = endTime - starTime;
        System.out.println("String 消耗时间: " + time);
        StringBuilder builder = new StringBuilder("");
        starTime = System.currentTimeMillis();
        for (int j = 0; j < 10000; j++) {
            builder.append(j);
        }
        endTime = System.currentTimeMillis();
        time = endTime - starTime;
        System.out.println("StringBuilder 消耗时间: " + time);
    }
}
```

//新建类  
//主方法  
//创建空字符串  
  
//利用 for 循环执行 10000 次操作  
//循环追加字符串  
  
//定义对字符串操作后的时间  
//计算对字符串执行操作的时间  
//将执行的时间输出  
//创建字符串生成器  
//定义操作执行前的时间  
//利用 for 循环进行操作  
//循环追加字符  
  
//定义操作后的时间  
//追加操作执行的时间  
//将操作时间输出

运行结果如图 5.23 所示。

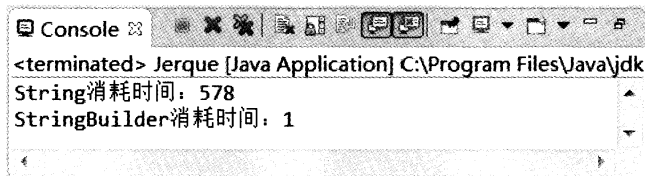


图 5.23 例 5.28 的运行结果

通过这一实例可以看出, 两种操作执行的时间差距很大。如果在程序中频繁地附加字符串, 建议使用 `StringBuilder`。新创建的 `StringBuilder` 对象初始容量是 16 个字符, 可以自行指定初始长度。如果附加的字符超过可容纳的长度, 则 `StringBuilder` 对象将自动增加长度以容纳被附加的字符。若要使用 `StringBuilder` 最后输出字符串结果, 可使用 `toString()` 方法。利用 `StringBuilder` 类中的方法可动态地执行添加、删除和插入等字符串的编辑操作。该类的常用方法如下。

#### (1) `append()` 方法

该方法用于向字符串生成器中追加内容。通过该方法的多个重载形式, 可实现接受任何类型的数据, 如 `int`、`boolean`、`char`、`String`、`double` 或者另一个字符串生成器等。

语法如下:

```
append(content)
```

其中, `content` 表示要追加到字符串生成器中的内容, 可以是任何类型的数据或者其他对象。

#### (2) `insert(int offset, arg)` 方法

该方法用于向字符串生成器中的指定位置插入数据内容。通过该方法的不同重载形式, 可实现向字符串生成器中插入 `int`、`float`、`char` 和 `boolean` 等基本数据类型或其他对象。

语法如下:

```
insert(int offset arg)
```

☑ `offset`: 字符串生成器的位置。该参数必须大于等于 0, 且小于等于此序列的长度。

☑ `arg`: 将插入至字符串生成器的位置。该参数可以是任何的数据类型或其他对象。

**【例 5.29】** 向字符串生成器中指定的位置添加字符, 实例代码如下:

```
StringBuilder bf = new StringBuilder("hello");           //创建字符生成器  
bf.insert(5, "word");                                   //添加至字符生成器的内容  
System.out.println(bf.toString());                     //此时输出信息为 helloworld
```

#### (3) `delete(int start, int end)` 方法

移除此序列的子字符串中的字符。该子字符串从指定的 `start` 处开始, 一直到索引 `end - 1` 处的字符, 如果不存在这种字符, 则一直到序列尾部。如果 `start` 等于 `end`, 则不发生任何更改。

语法如下:

```
delete(int start, int end)
```

☑ `start`: 将要删除的字符串的起点位置。

☑ `end`: 将要删除的字符串的终点位置。

【例 5.30】 删除指定位置的子字符串，实例代码如下：

```
StringBuilder bf = new StringBuilder("StringBuilder");    //创建字符串生成器  
bf.delete(5, 10);    //删除的子字符串  
System.out.println(bf.toString());    //此时输出的信息为 Strinder
```

说明

想要了解更多的 `StringBuilder` 类方法，可查询 `java.lang.StringBuilder` 的 API 说明。

## 5.8 小 结

精通 Java 中的字符串处理技术是学习 Java 语言的必修课。本章向读者介绍了字符串的创建和连接方式，以及获取字符串信息、常用的字符串操作等。这些对字符串的常规操作在实际编程中经常会遇到，因此应熟练掌握。另外还介绍了一些高级的字符串处理技术，如格式化字符串、使用正则表达式和字符串生成器等，这些也是字符串处理技术的重点，读者应该熟练掌握。

## 5.9 实践与练习

1. 使用 `String` 类的 `toUpperCase()` 方法和 `toLowerCase()` 方法来实现大小写的转换。（答案位置：光盘\TM\sl\5.19）
2. 分别截取字符串 `str1` 和字符串 `str2` 中的部分内容，如果截取后的两个子串相同（不区分大小写）会输出“两个子串相同”，否则输出“两个子串并不相同”。（答案位置：光盘\TM\sl\5.20）
3. 使用正则表达式来判断字符串 `text` 是否为合法的手机号。（答案位置：光盘\TM\sl\5.21）
4. 使用字符串生成器，将字符串 `str` 追加 1~10 这 10 个数字。（答案位置：光盘\TM\sl\5.22）

# 第 6 章

---

## 数组

(  视频讲解：1 小时 21 分钟 )

数组是最为常见的一种数据结构，是相同类型的、用一个标识符封装到一起的基本类型数据序列或对象序列。可以用一个统一的数组名和下标来唯一确定数组中的元素。实质上，数组是一个简单的线性序列，因此访问速度很快。本章将介绍有关数组的知识。

通过阅读本章，您可以：

- » 掌握一维数组创建和使用的方法
- » 掌握二维数组创建和使用的方法
- » 了解如何遍历数组
- » 了解如何填充替换数组中的元素
- » 了解如何对数组进行排序
- » 了解如何复制数组
- » 了解查询数组的方法

## 6.1 数组概述

数组是具有相同数据类型的一组数据的集合。例如，球类的集合——足球、篮球、羽毛球等；电器集合——电视机、洗衣机、电风扇等。在程序设计中，可以将这些集合称为数组。数组中的每个元素具有相同的数据类型。在 Java 中同样将数组看作一个对象，虽然基本数据类型不是对象，但由基本数据类型组成的数组却是对象。在程序设计中引入数组可以更有效地管理和处理数据。可根据数组的维数将数组分为一维数组、二维数组……

## 6.2 一维数组的创建及使用

 视频讲解：光盘\TM\lx\6\一维数组.mp4

一维数组实质上是一组相同类型数据的线性集合，当在程序中需要处理一组数据，或者传递一组数据时，可以应用这种类型的数组。本节将介绍一维数组的创建及使用。

### 6.2.1 创建一维数组

数组作为对象允许使用 `new` 关键字进行内存分配。在使用数组之前，必须首先定义数组变量所属的类型。一维数组的创建有两种形式。

#### 1. 先声明，再用 `new` 运算符进行内存分配

声明一维数组有下列两种方式：

```
数组元素类型 数组名字[];  
数组元素类型[] 数组名字;
```

数组元素类型决定了数组的数据类型。它可以是 Java 中任意的数据类型，包括简单类型和组合类型。数组名字为一个合法的标识符，符号“`[]`”指明该变量是一个数组类型变量。单个“`[]`”表示要创建的数组是一个一维数组。

【例 6.1】 声明一维数组，实例代码如下：

```
int arr[];           //声明 int 型数组，数组中的每个元素都是 int 型数值  
String str[];        //声明 String 数组，数组中的每个元素都是 String 型数值
```

声明数组后，还不能立即访问它的任何元素，因为声明数组只是给出了数组名字和元素的数据类型，要想真正使用数组，还要为它分配内存空间。在为数组分配内存空间时必须指明数组的长度。为数组分配内存空间的语法格式如下：

```
数组名字 = new 数组元素的类型[数组元素的个数];
```

- ☑ 数组名字：被连接到数组变量的名称。
- ☑ 数组元素的个数：指定数组中变量的个数，即数组的长度。

通过上面的语法可知，使用 `new` 关键字分配数组时，必须指定数组元素的类型和数组元素的个数，即数组的长度。

【例 6.2】为数组分配内存，实例代码如下：

```
arr = new int[5];
```

以上代码表示要创建一个有 5 个元素的整型数组，并且将创建的数组对象赋给引用变量 `arr`，即引用变量 `arr` 引用这个数组，如图 6.1 所示。

在图 6.1 中 `arr` 为数组名称，方括号 “[ ]” 中的值为数组的下标。数组通过下标来区分数组中不同的元素。数组的下标是从 0 开始的。由于创建的数组 `arr` 中有 5 个元素，因此数组中元素的下标为 0~4。

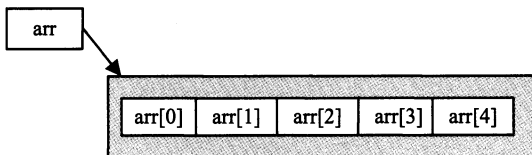


图 6.1 一维数组的内存模式



#### 说明

使用 `new` 关键字为数组分配内存时，整型数组中各个元素的初始值都为 0。

## 2. 声明的同时为数组分配内存

这种创建数组的方法是将数组的声明和内存的分配合在一起执行。

语法如下：

```
数组元素的类型 数组名 = new 数组元素的类型[数组元素的个数];
```

【例 6.3】声明并为数组分配内存，实例代码如下：

```
int month[] = new int[12]
```

上面的代码创建数组 `month`，并指定了数组长度为 12。这种创建数组的方法也是 Java 程序编写过程中普遍的做法。

## 6.2.2 初始化一维数组

数组与基本数据类型一样可以进行初始化操作。数组的初始化可分别初始化数组中的每个元素。数组的初始化有以下两种形式：

```
int arr[] = new int[]{1,2,3,5,25};    //第一种初始化方式
int arr2[] = {34,23,12,6};            //第二种初始化方式
```

从中可以看出，数组的初始化就是包括在大括号之内用逗号分开的表达式列表。用逗号 (,) 分割数组中的各个元素，系统自动为数组分配一定的空间。用第一种初始化方式，将创建 5 个元素的数组，依次为 1、2、3、5、25。第二种初始化方式，会创建 4 个元素的数组，依次为 34、23、12、6。



### 6.2.3 使用一维数组

在 Java 集合中一维数组是常见的一种数据结构。下面的实例是使用一维数组将 1~12 月各月的天数输出。

【例 6.4】 在项目中创建类 GetDay，在主方法中创建 int 型数组，并实现将各月的天数输出。（实例位置：光盘\TM\sl\6.01）

```
public class GetDay {                                //创建类
    public static void main(String[] args) {          //主方法
        //创建并初始化一维数组
        int day[] = new int[] { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 };
        for (int i = 0; i < 12; i++) {                //利用循环将信息输出
            System.out.println((i + 1) + "月有" + day[i] + "天"); //输出的信息
        }
    }
}
```

运行结果如图 6.2 所示。

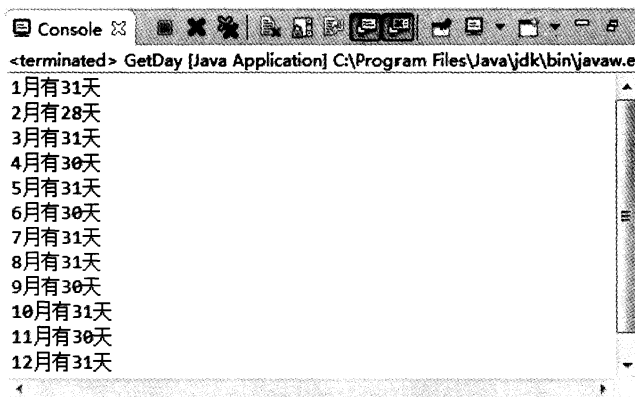


图 6.2 例 6.4 的运行结果

## 6.3 二维数组的创建及使用

 视频讲解：光盘\TM\lx\6\二维数组.mp4

如果一维数组中的各个元素仍然是一个数组，那么它就是一个二维数组。二维数组常用于表示表，表中的信息以行和列的形式组织，第一个下标代表元素所在的行，第二个下标代表元素所在的列。

## 6.3.1 二维数组的创建

二维数组可以看作是特殊的一维数组，因此，二维数组的创建同样有两种方式。

### 1. 先声明，再用 new 运算符进行内存分配

声明二维数组的语法如下：

```
数组元素的类型 数组名字[ ];  
数组元素的类型[ ][ ] 数组名字;
```

【例 6.5】 声明二维数组，实例代码如下：

```
int myarr[ ][ ];
```

同一维数组一样，二维数组在声明时也没有分配内存空间，同样要使用 new 关键字来分配内存，然后才可以访问每个元素。

对于高维数组，有两种为数组分配内存的方式：

(1) 直接为每一维分配内存空间

【例 6.6】 为每一维数组分配内存，实例代码如下：

```
a = new int[2][4];
```

上述代码创建了二维数组 a，二维数组 a 中包括两个长度为 4 的一维数组，内存分配如图 6.3 所示。

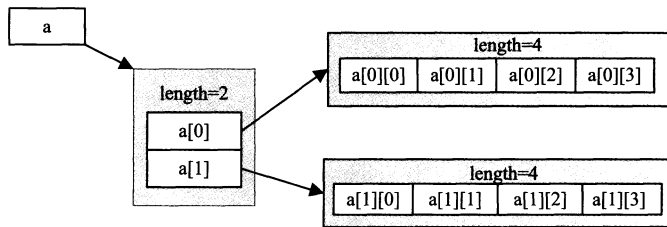


图 6.3 二维数组内存分配（第一种方式）

(2) 分别为每一维分配内存

【例 6.7】 分别为每一维分配内存，实例代码如下：

```
a = new int[2][ ];  
a[0] = new int[2];  
a[1] = new int[3];
```

### 2. 声明的同时为数组分配内存

第二种方式同第一种实现的功能相同。使用这种方式为二维数组分配内存时，首先指定最左边维数的内存，然后单独地给余下的维数分配内存。通过第二种方式为二维数组分配内存，如图 6.4 所示。

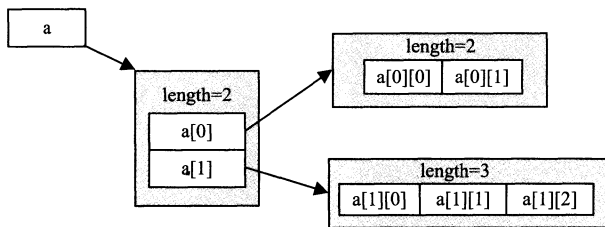


图 6.4 二维数组内存分配（第二种方式）

### 6.3.2 二维数组初始化

二维数组的初始化与一维数组初始化类似，同样可以使用大括号完成。

语法如下：

```
type arrayname[][] = {value1,value2...valuen};
```

- ☑ type: 数组数据类型。
- ☑ arrayname: 数组名称，一个合法的标识符。
- ☑ value: 数组中各元素的值。

【例 6.8】 初始化二维数组，实例代码如下：

```
int myarr[][] = {{12,0},{45,10}};
```

初始化二维数组后，要明确数组的下标都是从 0 开始。例如，上面的代码中 myarr[1][1] 的值为 10。

int 型二维数组是以 int a [][] 来定义的，所以可以直接给 a[x][y] 赋值。例如，给 a[1] 的第 2 个元素赋值的语句如下：

```
a[1][1] = 20
```

### 6.3.3 使用二维数组

二维数组在实际应用中用得非常广泛。下面的实例就是使用二维数组输出一个 3 行 4 列且所有元素都是 0 的矩阵。

【例 6.9】 在项目中创建类 Matrix，在主方法中编写代码实现输出一个 3 行 4 列且所有元素都为 0 的矩阵。（实例位置：光盘\TM\sl\6.02）

```
public class Matrix {                                //创建类
    public static void main(String[] args) {          //主方法
        int a[][] = new int[3][4];                  //定义二维数组
        for (int i = 0; i < a.length; i++) {          //循环遍历数组中的每个元素
            for (int j = 0; j < a[i].length; j++) {    //将数组中的元素输出
                System.out.print(a[i][j]);
            }
            System.out.println();                      //输出空格
        }
    }
}
```

```

    }
}

```

运行结果如图 6.5 所示。

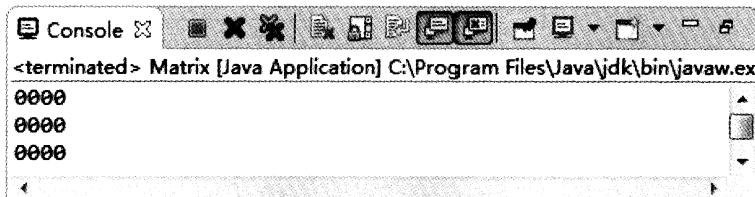


图 6.5 例 6.9 的运行结果



### 说明

对于整型二维数组，创建成功之后系统会赋给数组中每个元素初始值 0。

## 6.4 数组的基本操作

java.util 包的 Arrays 类包含了用来操作数组（如排序和搜索）的各种方法，本节就将介绍数组的基本操作。

### 6.4.1 遍历数组

 视频讲解：光盘\TM\lx\6\遍历数组.mp4

遍历数组就是获取数组中的每个元素。通常遍历数组都是使用 for 循环来实现。遍历一维数组很简单，也很好理解，下面详细介绍遍历二维数组的方法。

遍历二维数组需使用双层 for 循环，通过数组的 length 属性可获得数组的长度。

**【例 6.10】** 在项目中创建类 Trap，在主方法中编写代码，定义二维数组，实现将二维数组中的元素呈梯形输出。（实例位置：光盘\TM\sl\6.03）

```

public class Trap {
    public static void main(String[] args) {
        int b[][] = new int[][]{{ 1},{ 2, 3},{ 4, 5, 6 }};
        for (int k = 0; k < b.length; k++) {
            for (int c=0;c<b[k].length; c++){
                System.out.print(b[k][c]);
            }
            System.out.println();
        }
    }
}

```

//创建类  
//主方法  
//定义二维数组  
//循环遍历二维数组中的每个元素  
//将数组中的元素输出  
//输出空格