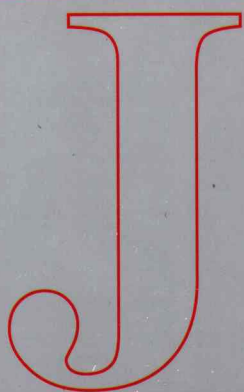


21世纪高等学校计算机**专业**实用规划教材

Java程序设计教程

江红 余青松 主编



清华大学出版社

21世纪高等学校计算机**专业**实用规划教材

Java程序设计教程

江红 余青松 主编

清华大学出版社
北京

内 容 简 介

本书主要基于 Java SE 6 SDK/Eclipse/NetBeans,讲述 Java SE 6 语言的基础知识,以及使用 Java SE 6 语言的 实际开发应用实例。本书具体内容包括 Java 语言概述,数据类型、变量和常量,运算符、表达式和语句,程序流程和异常处理,数组,类和对象,继承和多态,枚举类型和注解类型,泛型,多线程编程技术,数值、日期和字符串处理,输入/输出流和文件,集合和数据结构,数据库访问技术,网络编程和通信,图形用户界面应用程序等。

本书作者结合多年的程序设计、开发及授课经验,精选大量的实例,由浅入深、循序渐进地介绍了 Java 程序设计语言,让读者能够较为系统、全面地掌握程序设计的理论和应用。

本书可作为高等学校各专业的计算机程序设计教程,也可作为广大程序设计开发者、爱好者的自学参考书。

本书配有实验和辅导教材《Java 程序设计实验指导与习题测试》,提供了大量的思考与实践练习,让读者从实践中巩固和应用所学的知识。

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 程序设计教程 江红等主编. —北京:清华大学出版社,2012.11

(21 世纪高等学校计算机专业实用规划教材)

ISBN 978-7-302-28819-0

I. ①J… II. ①江… III. ①Java 语言—程序设计—高等学校—教材 IV. ①TP312

中国版本图书馆 CIP 数据核字(2012)第 103094 号

责任编辑:魏江江 王冰飞

封面设计:傅瑞学

责任校对:梁毅

责任印制:张雪娇

出版发行:清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址:北京清华大学学研大厦 A 座 邮 编:100084

社总机:010-62770175 邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质量反馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

课 件 下 载: <http://www.tup.com.cn>,010-62795954

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:185mm×260mm 印 张:31.5 字 数:784千字

版 次:2012 年 11 月第 1 版 印 次:2012 年 11 月第 1 次印刷

印 数:1~2500

定 价:49.50 元

产品编号:045675-01

前 言

目前,国内外众多的大学本科、专科、高职高专均采用 Java 作为程序设计课程的必修语言。Java 是较流行的编程语言之一,具有高度的安全性、可移植性和代码可重用性。

本书主要基于 Java SE 6 SDK/Eclipse/NetBeans,讲述 Java SE 6 语言的基础知识,以及使用 Java SE 6 语言的 actual 开发应用实例。

本书内容共分 16 章,分别讲述 Java 语言概述,数据类型、变量和常量,运算符、表达式和语句,程序流程和异常处理,数组,类和对象,继承和多态,枚举类型和注解类型,泛型,多线程编程技术,数值、日期和字符串处理,输入/输出流和文件,集合和数据结构,数据库访问技术,网络编程和通信,图形用户界面应用程序等。

本书配有实验和辅导教材《Java 程序设计实验指导与习题测试》,提供了本书的上机实验指导,以及本书各章节的习题测试和习题参考解答。

本书特点:

- (1) 由浅入深、循序渐进、重点突出、通俗易懂。
- (2) 理论与实践相结合,通过大量的实例,讲述程序设计的基本原理,使读者不仅掌握理论知识,同时掌握大量程序设计的实用案例。
- (3) 提供了大量的思考与实践练习,让读者从实践中巩固和应用所学的知识。

本书涉及的各章节所有的源程序代码和相关素材以及供教师参考的教学电子文稿和教学文档(包括 JDK 的安装与使用、Eclipse SDK 简介)均可以通过清华大学出版社的教学资料网站(www.tup.tsinghua.edu.cn)下载。

本书由华东师范大学江红和余青松共同编写。在此衷心感谢本书的编辑魏江江老师,敬佩他的高瞻远瞩和敬业精神。由于时间和编者学识有限,书中不足之处在所难免,敬请诸位同行、专家和读者指正。

编 者

2012 年 3 月

目 录

第1章 Java语言概述	1	2.4.4 布尔类型常量	22
1.1 Java语言及其特点	1	2.4.5 字符类型常量	22
1.1.1 Java语言简介	1	2.4.6 字符串类型常量	23
1.1.2 Java的特点和开发应用范围	2	2.4.7 null类型常量	23
1.2 Java语言的编译和运行环境	3	2.4.8 用户声明常量	23
1.2.1 Java语言与Java平台	3	2.5 预定义数据类型	24
1.2.2 Java SE	3	2.5.1 整数类型	24
1.2.3 Java的运行环境	4	2.5.2 浮点类型	26
1.2.4 Java的开发环境	4	2.5.3 布尔类型	27
1.3 创建简单的Java程序	5	2.5.4 字符类型	28
1.3.1 Hello World程序	5	2.5.5 String数据类型	29
1.3.2 代码分析	5	2.6 类型转换	30
1.3.3 编译和运行结果	5	2.6.1 自动类型转换	30
1.4 Java程序的基本结构	6	2.6.2 强制类型转换	32
1.4.1 程序结构	6	第3章 运算符、表达式和语句	33
1.4.2 包	7	3.1 运算符	33
1.4.3 类和对象	11	3.1.1 算术运算符	34
1.4.4 main方法	12	3.1.2 关系运算符	37
1.4.5 注释	13	3.1.3 逻辑运算符	39
1.4.6 Java编码规则	14	3.1.4 字符串运算符	41
第2章 数据类型、变量和常量	15	3.1.5 位运算符	41
2.1 标识符	15	3.1.6 赋值运算符	43
2.1.1 Java标识符和关键字	15	3.1.7 条件运算符	45
2.1.2 Java命名规则	16	3.1.8 其他运算符	46
2.2 数据类型	16	3.1.9 运算符优先级	47
2.2.1 简单类型	17	3.2 表达式	49
2.2.2 引用类型	17	3.2.1 表达式的组成	49
2.2.3 装箱和拆箱	17	3.2.2 表达式的书写规则	49
2.3 变量	18	3.3 语句	50
2.3.1 变量的声明和赋值	18	第4章 程序流程和异常处理	52
2.3.2 变量的作用域	20	4.1 顺序结构	52
2.4 常量	21	4.2 选择结构	53
2.4.1 文本常量	21	4.2.1 if语句	53
2.4.2 整型常量	21	4.2.2 switch语句	61
2.4.3 浮点数据类型常量	22	4.3 循环结构	63

4.3.1 for 循环	63	第 6 章 类和对象	113
4.3.2 while 循环	66	6.1 面向对象概述	113
4.3.3 do...while 循环	67	6.1.1 对象	113
4.3.4 for each 循环	69	6.1.2 封装	113
4.3.5 循环的嵌套	70	6.1.3 继承	114
4.4 跳转语句	71	6.1.4 多态性	114
4.4.1 break 语句	71	6.2 类和对象概述	114
4.4.2 continue 语句	72	6.2.1 类的声明	114
4.4.3 return 语句	73	6.2.2 创建和使用对象	117
4.5 异常处理	74	6.3 类的成员	119
4.5.1 异常处理概述	74	6.3.1 数据成员	119
4.5.2 异常类	75	6.3.2 函数成员	119
4.5.3 抛出异常	78	6.3.3 静态成员和实例成员	119
4.5.4 捕获处理异常	79	6.3.4 this 关键字	121
4.6 Java 断言处理	85	6.4 字段	121
4.6.1 Java 断言处理概述	85	6.4.1 字段的声明和访问	121
4.6.2 assert 语句和 AssertionError 类	86	6.4.2 静态字段和实例字段	122
4.6.3 启用/禁用断言	87	6.4.3 常量字段	123
4.6.4 断言编译注意事项	87	6.4.4 volatile 字段和 transient 字段	124
第 5 章 数组	90	6.5 方法	124
5.1 数组概述	90	6.5.1 方法的声明和调用	124
5.1.1 数组的声明	90	6.5.2 参数的传递	125
5.1.2 数组的实例化和初始化	91	6.5.3 方法的重载	128
5.1.3 数组的基本访问操作	91	6.5.4 静态方法和实例方法	129
5.2 一维数组	92	6.5.5 strictfp 方法	130
5.2.1 一维数组的声明、实例化和 初始化	92	6.5.6 递归	131
5.2.2 一维数组的基本访问操作	93	6.6 对象构造	131
5.3 多维数组	94	6.6.1 构造方法	131
5.3.1 多维数组的声明、实例化和 初始化	94	6.6.2 私有构造方法	133
5.3.2 多维数组的基本访问操作	96	6.6.3 静态初始化代码块	134
5.4 交错数组	98	6.6.4 实例初始化代码块	134
5.5 匿名数组	100	6.6.5 字段的初始化顺序	134
5.6 数组的应用举例	101	6.7 类成员访问修饰符	136
5.6.1 数组元素的求和、最值	101	6.8 嵌套类	139
5.6.2 数组的排序	102	6.8.1 嵌套顶级类	139
5.6.3 数组元素的插入	105	6.8.2 实例内部类	140
5.6.4 数组元素的删除	106	6.8.3 本地内部类	142
5.6.5 矩阵的基本操作	107	6.8.4 匿名内部类	142
5.7 Java 类库中操作数组的类和方法	108	6.9 Object 类和 Class 类	143
5.7.1 java.util.Arrays	108	6.9.1 Object 类和通用方法	143
5.7.2 System.arraycopy	110	6.9.2 Class 类和反射技术	144
5.7.3 数组名.clone()	111	6.10 对象的生命周期	146
		6.10.1 对象的创建	146
		6.10.2 对象的使用	146

6.10.3 对象的销毁	147	9.1.3 泛型概述	193
第7章 继承和多态	148	9.2 泛型的定义	194
7.1 继承概述	148	9.2.1 泛型类	194
7.1.1 继承的概念	148	9.2.2 泛型接口	196
7.1.2 继承的类型	148	9.2.3 泛型方法	196
7.1.3 继承的层次关系	149	9.2.4 泛型参数的约束	199
7.2 继承	149	9.2.5 泛型与子类	200
7.2.1 派生类的声明	149	9.2.6 通配符	200
7.2.2 super 关键字	149	9.3 泛型和继承	203
7.2.3 类成员的继承	150	9.3.1 泛型类的继承设计准则	203
7.2.4 构造方法的继承和调用	152	9.3.2 泛型接口的继承设计准则	205
7.2.5 类成员变量的隐藏	154	9.4 泛型的使用	206
7.2.6 类方法的重写	155	9.4.1 创建泛型类的对象	206
7.2.7 抽象类和抽象方法	157	9.4.2 泛型类的类型检查	206
7.2.8 密封类和密封方法	160	9.4.3 泛型类的类型转换	207
7.3 接口	161	9.4.4 类型擦除和泛型类对象的类	207
7.3.1 接口的声明	161	9.4.5 与遗留代码交互	208
7.3.2 接口的成员	162	第10章 多线程编程技术	209
7.3.3 接口的实现	162	10.1 线程处理概述	209
7.3.4 接口的继承	163	10.1.1 进程和线程	209
7.4 多态	165	10.1.2 线程的优缺点	209
7.4.1 多态的概念	165	10.2 创建多线程应用程序	210
7.4.2 通过继承实现多态性	165	10.2.1 Java 应用程序主线程	210
7.4.3 通过方法重载实现多态性	167	10.2.2 创建和启动新线程	210
7.4.4 通过方法重写实现多态性	168	10.3 线程状态和生命周期	213
7.4.5 多态性综合举例	170	10.3.1 线程状态及其转换图	213
第8章 枚举类型和注解类型	175	10.3.2 线程的创建	214
8.1 枚举	175	10.3.3 线程的属性设置和获取	214
8.1.1 枚举类型概述	175	10.3.4 线程的启动、停止、挂起和 唤醒	214
8.1.2 枚举的声明和使用	175	10.3.5 休眠线程 sleep()	214
8.1.3 枚举类的成员方法	178	10.3.6 线程让步 yield()	214
8.1.4 枚举类综合举例	178	10.3.7 线程加入 join()	215
8.2 注解类型	179	10.3.8 中断线程 interrupt()	216
8.2.1 注解类型概述	179	10.3.9 终止线程	216
8.2.2 预定义注解类型	180	10.4 用户线程和 Daemon 线程	217
8.2.3 自定义注解类型	184	10.5 线程优先级和线程调度	219
8.2.4 使用反射访问注解类型	187	10.6 线程组	220
8.2.5 注解类型综合举例	188	10.7 线程同步	221
第9章 泛型	191	10.7.1 线程同步处理	221
9.1 泛型的基本概念	191	10.7.2 使用 synchronized 同步 方法	222
9.1.1 引例 ArrayList	191	10.7.3 使用 synchronized 同步 代码块	223
9.1.2 引例 ArrayList < E >	192		

10.7.4 线程间通信 wait()、notify() 和 notifyAll()	225	12.4 字符流的写入和读取	288
10.8 java.util.Timer 和 java.util .TimerTask	228	12.4.1 文本文件的写入/读取	288
第 11 章 数值、日期和字符串处理	232	12.4.2 字符数据的写入/读取	290
11.1 数值处理	232	12.4.3 使用字符缓冲流提高 写入/读取效率	293
11.1.1 Math 类和数学函数	232	12.5 随机文件的访问	296
11.1.2 Random 类和随机函数	235	12.5.1 RandomAccessFile 概述	296
11.1.3 BigInteger 类和任意精度 整数	237	12.5.2 创建 RandomAccessFile 对象	296
11.1.4 BigDecimal 类和任意精度 浮点数	241	12.5.3 随机文件的读取	296
11.1.5 数值格式化输出 NumberFormat 和 DecimalFormat	245	12.5.4 随机文件的写入	297
11.2 日期和时间处理	248	12.5.5 随机文件的定位	297
11.2.1 java.util.Date 类	248	12.5.6 随机文件的关闭	297
11.2.2 java.util.Calendar 类和 GregorianCalendar 类	249	12.6 对象序列化	298
11.2.3 java.text.DateFormat 类和 SimpleDateFormat 类	251	12.6.1 对象序列化概述	298
11.3 字符串处理	253	12.6.2 对象输出流	298
11.3.1 String 类	253	12.6.3 对象输入流	299
11.3.2 StringBuilder 类和 StringBuffer 类	257	12.7 控制台 I/O	300
11.4 正则表达式	260	12.7.1 System.in/System.out/ System.err	300
11.4.1 正则表达式语言	261	12.7.2 I/O 重定向	301
11.4.2 正则表达式类及应用举例	263	12.7.3 java.util.Scanner	302
第 12 章 输入/输出流和文件	267	第 13 章 集合和数据结构	305
12.1 输入/输出流概述	267	13.1 Java 平台集合框架	305
12.1.1 流的基本概念	267	13.2 集合框架中的接口	306
12.1.2 java.io 中主要类的继承 关系	268	13.2.1 接口的继承关系	306
12.1.3 I/O 流的四大抽象类	270	13.2.2 Collection 接口	307
12.2 磁盘、目录和文件的基本操作	270	13.2.3 List 接口	308
12.2.1 java.io.File 类概述	270	13.2.4 Set 接口	309
12.2.2 磁盘分区的基本操作	271	13.2.5 Queue 接口	309
12.2.3 文件和目录的基本操作	272	13.2.6 Map 接口	311
12.3 字节流的写入和读取	279	13.2.7 对象排序	311
12.3.1 二进制文件的写入/读取	279	13.2.8 SortedSet 接口	314
12.3.2 基本 Java 数据类型的 写入/读取	282	13.2.9 SortedMap 接口	314
12.3.3 使用字节缓冲流提高 写入/读取效率	285	13.3 集合框架中的算法	315
		13.3.1 Collections 类	315
		13.3.2 排序	315
		13.3.3 混排	315
		13.3.4 常规数据操作算法	316
		13.3.5 查找	317
		13.3.6 极值	318
		13.3.7 其他算法	319
		13.3.8 封装器	319
		13.4 列表	320

13.4.1	数组列表	320	15.1.1	网络基础知识	370
13.4.2	链表	322	15.1.2	TCP/IP 简介	370
13.5	集	324	15.1.3	IP 地址和域名	371
13.5.1	散列集	324	15.1.4	统一资源定位器	372
13.5.2	树集	325	15.2	InetAddress	372
13.5.3	链表散列集	326	15.2.1	创建 InetAddress 对象	372
13.6	队列	328	15.2.2	获取 InetAddress 的信息	373
13.7	映射表	331	15.3	基于 URL 的网络编程	374
13.7.1	散列映射表	331	15.3.1	创建 URL 对象	374
13.7.2	树映射表	332	15.3.2	解析 URL 对象	374
13.7.3	链表散列映射表	334	15.3.3	从 URL 读取网络资源	375
13.8	遗留的集合类	335	15.3.4	创建 URLConnection 并读取 内容	375
13.8.1	向量	335	15.4	基于 Socket 的网络编程	376
13.8.2	堆栈	335	15.4.1	Socket 概述	376
13.8.3	哈希表	337	15.4.2	Socket 类	377
13.9	创建自定义集合类	337	15.4.3	ServerSocket 类	378
第 14 章	数据库访问技术	339	15.4.4	简单的 Client/Server 程序 设计	379
14.1	关系数据库和 SQL 语言	339	15.4.5	支持多客户的 Client/Server 程序设计	381
14.1.1	数据库概念	339	15.5	基于 Datagram 的网络编程	384
14.1.2	关系数据库	339	15.5.1	DatagramSocket 和 DatagramPacket	384
14.1.3	SQL 语言基础	341	15.5.2	基于 DatagramSocket 的 Client/Server 程序设计	386
14.1.4	本书使用的样例数据库	342	第 16 章	图形用户界面应用程序	389
14.2	JDBC 概述	343	16.1	Java 图形用户界面概述	389
14.2.1	JDBC 的基本概念	343	16.1.1	AWT 简介	389
14.2.2	JDBC 的结构	344	16.1.2	Swing 简介	389
14.2.3	JDBC 驱动程序分类	344	16.1.3	SWT 简介	390
14.2.4	JDBC API	346	16.2	Swing 概述	390
14.3	使用 JDBC 访问数据库	347	16.2.1	Swing 组件	390
14.3.1	加载 JDBC 驱动程序	347	16.2.2	JFrame 类	391
14.3.2	创建与数据源的连接	349	16.2.3	创建 Swing 应用程序的 一般步骤	393
14.3.3	执行数据库操作	351	16.2.4	创建简单的 Swing 应用 程序	394
14.3.4	处理 SQL 命令结果	356	16.3	布局管理器	395
14.4	使用 JDBC 访问数据库的示例	358	16.3.1	布局管理器概述	395
14.4.1	查询数据库表数据	358	16.3.2	FlowLayout	396
14.4.2	插入数据库表数据	359	16.3.3	GridLayout	397
14.4.3	更新数据库表数据	360	16.3.4	GridBagLayout	397
14.4.4	删除数据库表数据	362	16.3.5	BorderLayout	401
14.4.5	使用存储过程访问数据库	363			
14.4.6	创建、删除、修改表结构	365			
14.4.7	查询数据库的结构信息 ——元数据	366			
第 15 章	网络编程和通信	370			
15.1	网络编程的基本概念	370			

16.3.6	BoxLayout	402	16.7.15	JTree	446
16.3.7	CardLayout	404	16.7.16	JEditorPane 和 JTextPane	449
16.3.8	null 布局	406	16.8	通用对话框	451
16.4	AWT 事件处理	407	16.8.1	JOptionPane 对话框	451
16.4.1	事件处理机制	407	16.8.2	JFileChooser 对话框	453
16.4.2	事件类	409	16.8.3	JColorChooser 对话框	454
16.4.3	事件监听器	410	16.8.4	通用对话框应用举例	455
16.5	Swing 组件概述	418	16.8.5	自定义对话框	458
16.6	面板容器	421	16.9	菜单和工具栏	460
16.6.1	JPanel	421	16.9.1	菜单相关组件	460
16.6.2	JScrollPane	421	16.9.2	创建主菜单	464
16.6.3	JTabbedPane	421	16.9.3	创建上下文菜单	467
16.7	常用 Swing 组件	423	16.9.4	JToolBar	468
16.7.1	JLabel	423	16.9.5	菜单和工具栏应用举例: 文本编辑器	470
16.7.2	JButton	423	16.10	图形绘制	474
16.7.3	JTextField	425	16.10.1	图形绘制概述	474
16.7.4	JPasswordField	426	16.10.2	图形上下文设置	474
16.7.5	JTextArea	426	16.10.3	绘制字符串	475
16.7.6	JRadioButton	429	16.10.4	绘制图形和图像	475
16.7.7	JCheckBox	429	16.10.5	Java 2D 简介	478
16.7.8	JList	432	16.11	Swing 与线程	484
16.7.9	JComboBox	435	16.11.1	Swing 事件派发线程	484
16.7.10	Timer	436	16.11.2	SwingUtilities 的 invokeLater 和 invokeAndWait 方法	486
16.7.11	JSlider	438	16.12	Swing 外观	488
16.7.12	JSpinner	440	16.13	Toolkit 实用工具包	490
16.7.13	JProgressBar	443			
16.7.14	JTable	444			

Java 语言是一种通用的面向对象的编程语言,被广泛用于构建各种跨平台的、安全可靠的网络应用程序。

本章要点:

- Java 语言及其特点
- Java 语言的编译和运行环境
- Java 程序的创建、编译和运行
- Java 程序的基本结构

1.1 Java 语言及其特点

1.1.1 Java 语言简介

1991 年 6 月,美国 Sun Microsystems 公司(该公司已于 2009 年被甲骨文公司收购)的 James Gosling 等人组成开发小组,旨在开发一种用于家用电器(如电视机、冰箱、烤箱、电话等)的控制和通信的逻辑控制系统。项目最初拟采用 C 语言,但发现 C 语言和可用的 API 存在许多问题,为此,Gosling 等结合 C 语言和 Mesa 语言的特点,创建了一种全新的语言 Oak,即 Java 语言的前身。由于这些智能化家电的市场需求不足,该项目最后被终止。

随着 Internet 的发展,Sun Microsystems 公司看到了 Oak 比较适合编写 Internet 应用程序,于是结合 WWW 的需要对 Oak 进行了改进和完善,并在 1995 年 5 月以 Java 的名称正式发布。Java 伴随着互联网的迅猛发展而发展,逐渐成为重要的网络编程语言。

Java 编程语言的风格与 C++ 语言近似。Java 继承了 C++ 语言面向对象技术的核心,但舍弃了 C++ 语言中容易引起错误的指针、运算符重载、多重继承等特性,增加了垃圾回收器功能,用于回收不再被引用的对象所占据的内存空间。Java SE 1.5 中又引入了泛型编程、类型安全的枚举、不定长参数和自动装/拆箱等语言特性。

与编译执行语言(例如 C++)不同,Java 是解释执行语言。Java 源代码首先编译成字节码,然后依赖各种不同平台上的虚拟机来解释执行字节码,从而实现“一次编译、到处执行”的跨平台特性。Java 语言的解释执行特点在一定程度上会降低 Java 程序的运行效率,但在 J2SE 1.4.2 发布后,Java 的执行速度有了大幅提升。

Java 作为广泛使用的语言,其主要发展历史如表 1-1 所示。

表 1-1 Java 主要发展历史

发布时间	Java 版本	说 明
1995/05		Java 语言诞生
1996/01	JDK 1.0	JDK 1.0 发布

发布时间	Java 版本	说 明
1997/02	JDK 1.1	JDK 1.1 发布
1998/12	J2SE 1.2	J2SE SDK 1.2 发布。自版本 1.2 至 1.5,通常称为 Java 2。JDK 被更名为 J2SE SDK。Sun 公司发布了 Java 的 3 个版本: J2SE (Java 2 Platform, Standard Edition)、J2EE (Java 2 Platform, Enterprise Edition) 和 J2ME (Java 2 Platform, Micro Edition)
2000/05	J2SE 1.3	J2SE SDK 1.3 发布
2002/02	J2SE 1.4	J2SE SDK 1.4 发布
2004/09	J2SE 5.0	J2SE SDK 5.0 发布。J2SE 1.5 被更名为 J2SE 5.0。内部版本号为 1.5
2006/12	Java SE 6	JDK 6 发布。Java 2 被更名为 Java SE 6。内部版本号为 1.6
2011/7	Java SE 7	JDK 7 发布

本书主要基于目前使用最广泛的 Java SE 6,讲述 Java 语言的基础知识,以及使用 Java 语言的实际开发应用实例。

注: 2009 年 4 月,Oracle 公司宣布收购 Sun 公司。

1.1.2 Java 的特点和开发应用范围

1. Java 语言的特点

Java 是一种现代的、面向对象的、类型安全的编程语言。Java 具有下列特点:

(1) 简单。Java 语言的语法与 C/C++ 语言很接近,使得大多数程序员很容易学习和使用。Java 简化了 C/C++ 中许多复杂的特性,如运算符重载、多继承等。特别地,Java 语言不使用指针,并提供了自动的垃圾收集,从而避免了直接操作内存的复杂性和风险性。

(2) 面向对象。Java 支持数据封装、继承、多态和接口。所有的变量和方法,包括 main 方法(应用程序的入口点),都封装在类定义中。类可以直接从一个父类继承(不支持多重继承),可以实现任意数量的接口。

(3) 分布式。Java 语言提供了用于网络应用编程的类库,可以处理 TCP/IP 及其他网络协议,广泛用于开发各种分布式的网络应用程序。

(4) 健壮性。Java 的强类型机制、异常处理、垃圾自动收集等是 Java 程序健壮性的重要保证。Java 的安全检查机制使得 Java 更具健壮性。

(5) 安全性。Java 通常被用在网络环境中,为此,Java 提供了一个安全机制以防恶意代码的攻击。

(6) 体系结构中立。Java 程序源文件(扩展名为 .java 的文件)在 Java 平台上被编译为体系结构中立的字节码格式(扩展名为 .class 的文件),然后可以在实现该 Java 平台的任何系统中运行,从而实现“一次编译、到处执行”的跨平台特性。

(7) 可移植性。Java 的可移植性来源于体系结构中立性。Java 系统本身也具有很强的可移植性,Java 编译器是用 Java 实现的,Java 的运行环境是用 ANSI C 实现的,从而保证了整个 Java 系统的可移植性。

(8) 解释型。Java 程序在 Java 平台上被编译为字节码格式,然后可以在实现该 Java 平台的任何系统中运行。在运行时,Java 平台中的 Java 解释器对这些字节码进行解释执行,执行过程中需要的类在连接阶段被载入到运行环境中。

(9) 高性能。解释型语言的缺点是性能低,但随着 JIT(Just-In-Time)编译器技术的发展,Java 的运行速度越来越接近于 C++。

(10) 多线程机制。Java 语言支持多个线程的同时执行,并提供多线程之间的同步机制。Java 线程可以设定优先级。使用多线程机制,可以编写各种高性能的应用程序,例如,使用线程处理同时进行用户输入的监视、后台任务的执行,以及并发输入流的处理等。

(11) 动态性。Java 语言的设计目标之一是适应动态变化的环境。Java 程序需要的类能够动态地载入到运行环境,也可以通过网络来载入所需要的类,有利于软件的升级。另外,Java 中的类有一个运行时刻的表示,能进行运行时刻的类型检查。

2. Java 语言开发应用范围

Java 语言主要用来构建在 Java 的运行环境(Java Runtime Environment,JRE)上运行各种安全、可靠的应用程序。使用 Java,可以创建下列类型的应用程序和服务:

- 桌面应用程序;
- Java Applet 小应用程序;
- Web 应用程序;
- 企业应用程序;
- 移动应用程序。

1.2 Java 语言的编译和运行环境

1.2.1 Java 语言与 Java 平台

Java 语言的一个重要特性是跨平台特性,即“一次编译、到处执行”。要实现该目标,必须提供相应的 Java 运行平台,在实现相应 Java 平台的任何系统中运行 Java 程序。根据应用范围,Java 运行平台分为 3 个体系:

(1) Java SE(Java Platform, Standard Edition, Java 平台标准版)。Java SE(旧版本称为 J2SE)提供标准的 SDK(Software Development Kit)开发平台(旧版本称为 JDK)。Java SE 允许开发和部署 Java 桌面应用程序、Java Applet 程序、低端服务器应用程序。Java SE 程序开发环境提供了开发与运行 Java 软件的编译器等开发工具、类库及 Java 虚拟机。

(2) Java EE(Java Platform, Enterprise Edition)。Java EE(旧版本称为 J2EE)是在 Java SE 的基础上构建的。Java EE 用于开发和部署可移植、健壮、可伸缩且安全的服务器端应用程序。Java EE 提供 Web 服务、组件模型、管理和通信 API,可以用来实现企业级的面向服务体系结构(Service-Oriented Architecture, SOA)和 Web 2.0 应用程序。

(3) Java ME(Java Platform, Micro Edition)。Java ME(旧版本称为 J2ME)是一种精简的 Java 运行环境,用于各种嵌入式设备,如移动电话、PDA、电视机顶盒等。Java ME 包括灵活的用户界面、健壮的安全模型、许多内置的网络协议,以及对可以动态下载的应用程序的支持。

1.2.2 Java SE

Java SE 程序开发环境包括 3 个部分:开发工具、类库及 Java 虚拟机。

1. Java SE 开发工具

Java SE 6 程序开发环境提供了开发与运行 Java 软件的编译器等开发工具,主要的组件包括以下内容。

- (1) javac: 编译器。将源程序转成字节码。
- (2) jar: 打包工具。将相关的类文件打包成一个文件。
- (3) javadoc: 文档生成器。从源码注释中提取文档。
- (4) jdb: 调试工具(Java debugger)。用于调试 Java 应用程序。

2. Java 类库

Java 提供了强大的应用程序编程接口 (Application Programming Interface, API), 即 Java 类库。Java 类库是系统提供的已实现的标准类的集合, 包括大量类和接口, 可以帮助程序员进行字符串处理、绘图、数学计算、网络应用等方面的工作。合理充分地利用 Java 类库, 可以高效率地开发各种应用程序。

Java 类库中的类和接口大多封装在特定的包里, 每个包具有自己的功能。Java 类库的主要包如表 1-2 所示。

表 1-2 Java 类库的主要包

包 名	主要功能
java.applet	提供用于创建 Applet 所需的类
java.awt.*	提供用于创建用户界面以及绘制和管理图形、图像的类
java.beans.*	提供用于开发 beans 的类
java.io	提供通过数据流、序列化和文件系统实现系统输入和输出的类
java.lang	提供 Java 编程语言的基础类
java.math	提供用于执行任意精度整数、小数算法的类
java.net	提供用于实现网络应用程序的类
java.nio.*	提供用于高效输入和输出的非堵塞型 I/O 类
java.rmi.*	提供用于远程方法调用的类和接口
java.security.*	提供用于安全框架的类和接口
java.sql	提供处理 Java 数据源中的数据的类
java.text.*	提供以与自然语言无关的方式来处理文本、日期、数字和消息的类和接口
java.util.*	提供各种常用工具包(包括集合框架、国际化等)
javax.swing.*	提供一组“轻量级”组件构建图形用户界面应用程序
javax.xml.*	提供用于 XML 处理的类
javax.naming.*	提供用于访问命名服务的类

3. Java 虚拟机

Java 虚拟机 (Java Virtual Machine, JVM) 是可运行 Java 字节码的假想计算机, 在实际的计算机上通过软件模拟来实现。Java 虚拟机包括一套字节码指令集、一组寄存器、一个栈、一个垃圾回收堆和一个存储方法域。Java 虚拟机是一种用于计算设备的规范, 可用不同的方式 (软件或硬件) 加以实现。编译虚拟机的指令集与编译微处理器的指令集非常类似。

Java 程序源文件被编译为体系结构中立的字节码格式。执行 Java 程序时, Java 虚拟机将加载并解释执行字节码程序。只要根据 JVM 规格描述将解释器移植到特定的计算机上, 就能保证经过编译的任何 Java 代码能够在该系统上运行, 从而实现“一次编译、到处执行”的跨平台特性。

1.2.3 Java 的运行环境

Java 的运行环境 (Java Run Environment, JRE) 为 Java 应用程序提供运行环境, JRE 包括 Java 虚拟机 (JVM) 和标准类库 (Class Library)。

注: Java SE SDK 包含一个 JRE, 也可安装独立的 JRE。

1.2.4 Java 的开发环境

要开发 Java 应用程序, 可以使用文本编辑器 (如 Notepad) 编写代码, 并使用 Java SE 6 中的编译器进行编译、运行, 也可以使用集成开发工具 (如 Eclipse、NetBeans)。

1. JDK

JDK(Java Development Kit)包括开发人员编写、生成、测试和部署 Java 应用程序时所需要的一切,如编译器、运行调试工具、文档及示例等。有关 JDK 的下载和安装配置,请参见附录 A。

2. Eclipse

Eclipse 是著名的免费且跨平台的集成开发环境(Integrated Development Environment, IDE),集设计、编辑、运行、调试等多种功能于一体。使用 Eclipse,可以高效地开发各种 Java 应用程序。

Eclipse 本身只是一个框架平台,通过插件,可以实现不同的开发功能。Eclipse 主要用于 Java 语言开发,但通过插件使其作为 C++、Python、PHP 等其他语言的开发工具。许多软件开发商以 Eclipse 为框架开发自己的 IDE。Eclipse 是目前使用最广泛的 Java 集成开发环境。有关 Eclipse 的下载与安装配置,请参见附录 B。

3. NetBeans

NetBeans 是一款开源集成开发环境,可用于 Java、C/C++、PHP、Python、Ruby 等程序的开发。NetBeans 是一个开放框架,是可扩展的开发平台,可以通过扩展插件来扩展功能。NetBeans 也是目前使用十分广泛的 Java 集成开发环境,许多 Java 文档和示例使用的就是 NetBeans 集成开发环境。

1.3 创建简单的 Java 程序

1.3.1 Hello World 程序

使用任意编辑软件(如记事本 Notepad. exe)创建程序文件 HelloWorld. java(Java 源文件的扩展名通常是. java)。

【例 1.1】 创建 Hello World 程序。

```
01 //javabook\chapter01\HelloWorld. java A "Hello World!" program
02 public class HelloWorld {
03     public static void main(String[] args) {
04         System.out.println("Hello World!");           //显示字符串
05     }
06 }
```

1.3.2 代码分析

行 01 为注释。

行 02 定义了用户自定义类 HelloWorld。

行 02 和行 06 的花括号对定义了代码块,其中的代码为类 HelloWorld 的实现。

行 03 定义了类 HelloWorld 的一个成员,即名为 main 的方法。main()方法是使用 static 修饰符声明的静态方法,将作为程序的入口点。

行 03 和行 05 的花括号对定义了代码块,其中的代码为 main()方法的实现。

行 04 通过调用 System. out. println(" Hello World!"),在控制台上输出字符串: Hello World!。System. out 是 System 类中 out 变量的全限定名称,代表标准输出设备,即控制台屏幕。System 是 Java 常用的类,本书范例也大量使用。

1.3.3 编译和运行结果

可以使用命令行“javac HelloWorld. java”调用 Java 编译器编译 HelloWorld 程序,如图 1-1

所示。编译后将产生一个名为 HelloWorld.class 的文件,即字节码的类文件。如果程序中产生错误(例如如拼写错误。特别强调,Java 源代码大小写敏感,如果上例中的 public 误拼写为 Public,则会产生编译错误),则编译器会产生错误信息。

可以使用命令行“java HelloWorld”调用 Java 解释器运行这个程序,如图 1-2 所示。当此应用程序运行时,输出结果为:Hello World!。

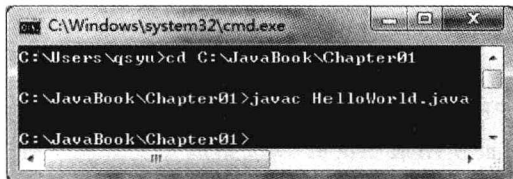


图 1-1 编译 HelloWorld 程序

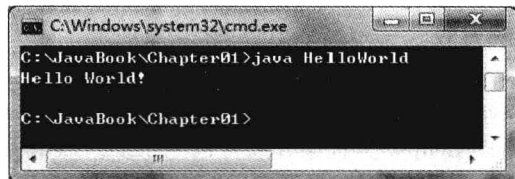


图 1-2 运行 HelloWorld 程序

使用 Java 解释器运行程序时,Java 解释器会在类路径(CLASSPATH 系统环境)中查找类的字节码文件。例如:执行命令“java HelloWorld”将在当前目录中查找 HelloWorld.class 字节码文件。如果找不到字节码文件,则会产生错误。

```
java HelloWorld.java
```

上述命令行会产生如下错误(Java 解释器会查找 HelloWorld.java.class):

```
Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorld/java
```

再如,执行命令“java HelloWorld.class”会产生如下错误信息(Java 解释器会查找 HelloWorld.class.class):

```
Exception in thread "main" java.lang.NoClassDefFoundError: HelloWorld/class
```

有关 Java 程序的编译器的详细信息,请参见附录 A。

注:一个源文件(扩展名为.java)可以包含多个类,编译时每个类都将生成一个字节码的类文件。另外,各嵌套类也将分别生成对应的字节码的类文件(请参见 6.8 节)。

1.4 Java 程序的基本结构

1.4.1 程序结构

Java 程序由一个或多个源文件(扩展名为.java)组成,每个源文件又称为编译单元。在编译单元(源文件)中可声明类和接口,类包含成员,例如成员变量和方法等。通过 Java 编译器,编译单元(Java 源文件)被编译成字节码文件(扩展名为.class),也可选择通过工具 JAR 把字节码文件打包为 JAR。

编译单元(源文件)中声明的类型按包组织成层次结构。Java 程序的基本结构如下所示:

```
//Java 程序的基本结构
import packagename;           //导入包
package yourpackagename;     //声明包
public class YourClass {     //声明类
//类体 声明字段、方法等
}
interface IYourInterface {   //声明接口
//接口体
}
```


一般情况下,一个编译单元(源文件)声明一个类(或接口),如果类的访问修饰符为 public,则源文件的文件名必须和文件中声明的 public 类的名称保持一致。

注: 在一个编译单元(Java 源文件)中,只能声明一个 public 类(与 Java 源文件名相同);如果在同一个源文件中声明了多个类,则其他类不能为 public。

1.4.2 包

Java 类库包含大量的类型,用户也可以自定义类型。为了有效地组织 Java 程序中的类型并保证其唯一性,Java 引入了包的概念,从而最大限度地避免类型重名错误。

在编译单元(Java 源文件)中定义类或接口时,可以声明其所在的包。Java 程序中,类型由指示逻辑层次结构的完全限定名(fully qualified name)描述。例如,java.util.Date 表示 java 包的子包 util 中的 Date 类。

1. 定义包

Java 程序中使用 package 关键字声明包。声明格式如下:

```
package 包名称;
```

其中,包名称的一般格式如下:

```
<company>. (<product >|< technology >)[.< feature >][.< subPackageName >]
```

包名称一般约定为小写字母,这样就可以与类的名称(一般约定为首字母大写)区分开来。例如,Acme 公司(假设其域名为 acme.com)的 ERP 项目中关于数据访问的类型可以组织到包 com.acme.erp.data 中。使用域名作为包命名的一部分,可以有效地组织项目中定义的类和接口,并与其他公司提供的类库区分开来。

package 语句必须是源程序文件中第一个非注释、非空白行的语句。

包的成员可以包含子包、类、接口。例如,java 包中包含子包 awt、applet、io、lang、net 和 util。java.util 包中包含 java.util.jar 等子包、java.util.Date 等类、java.util.Formatter 等接口。

一个包(子包)的成员不能重名。例如,如果包 javabook 定义了子包 test,则不能定义名称为 test 的类或接口。

注意: 如果源代码中没有指定包,则使用默认包。默认包一般用于编写实验性质的临时程序,例如 Java 学习程序。在实际项目开发中,除非简单的小程序,一般不推荐使用默认包。

一般情况下,包对应于文件系统的目录,子包对应相应目录下的子目录,而类文件(.class)则对应目录下的字节码文件。

【例 1.2】 包示例 1: 默认包。

```
//假设当前路径为: C:\javabook\chapter01
//源文件路径: C:\javabook\chapter01\HelloWorld1.java
//编译: javac HelloWorld1.java -> 编译结果: HelloWorld1.class
//运行: java HelloWorld1
public class HelloWorld1 {
    public static void main(String[] args) {
        System.out.println("Hello World!");    //显示字符串
    }
}
```

【例 1.3】 包示例 2: 简单包。

```
//假设当前路径为: C:\javabook\chapter01
//源文件路径: C:\javabook\chapter01\qsyu\HelloWorld.java
```