

Java 语言 及类库详解

薛刚 刘禹 刘尧 胡立新 等编

潘金贵 顾铁成 审

南京大学出版社

内 容 简 介

本书是对 Java 语言的全面介绍,说明了 Java 语言的特点、程序运行的基本概念、Internet 网上的各种 Java 资源、Java 语言的语法以及 Java 与 C/C++ 语言的接口,介绍了如何编写 Java 小应用程序,如何在自己的 Home Page 上加入图像、动画、声音等。此外,本书还按字母顺序列出了 Java 类及相应的构造函数、方法和变量,说明了其功能和用法。

本书列举了很多完整的程序实例,读者根据实际可运行它们。本书可供所有对 Java 感兴趣的人员参考,也是所有 Java 应用程序员和小应用程序员的必备参考书。

JS 496 / 18

Java 语言及类库详解

薛 刚 刘 禹 刘 尧 胡立新 等编
潘金贵 顾铁成 审

*

南京大学出版社出版

(南京大学校内 邮码:210093)

江苏省新华书店发行 南京通达彩印厂印刷

*

开本:787×1092 1/16 印张:27.75 字数:692千

1999年1月第1版 1999年1月第1次印刷

印数:1—3 000

ISBN 7-305-03195-x/TP·180

定价:38.00元

前 言

目前,Java 不仅在 Internet 网络环境上很流行,就是不经常上网的人,也喜欢把 Java 从网络上下载(Download)下来,用于编写一些动画程序或交叉平台应用程序。

Java 建立在 Web 和基于 CGI(Common Gateway Interface,通用网关接口)应用程序的基础上,同时又提供了更多的功能,如可把动画加到 Web 应用程序中,相同的代码可在各种不同的平台上运行,而且,Java 看起来像 C/C++。因此,访问过 Internet 的 C/C++ 程序员都发现很容易掌握 Java,并开始着手编写 Java 小应用程序和应用程序。

当然,Java 与 C/C++ 之间也有很大的区别。因此,要编写可正常运行的 Java 应用程序和小应用程序,不掌握 Java 语言的语法结构和小应用程序的运行机理是不行的。这正是作者编写本书的目的之一。

尽管 Java 表面上与 C/C++ 相似,几乎任何人都容易编写执行简单计算或重复循环的程序,但仍需要掌握 Java 的应用程序接口(API)来创建有用的小应用程序和应用程序。熟悉 Macintosh,Windows 或 X-Window 的程序员将会很快掌握其实质,但如果 API 文档不完整,那么要使用 Java Development Kit (Java 开发工具包,JDK),就要作出艰苦的努力。本书也试图解决这个问题,其中提供了清楚的材料和相应的例子来说明怎样调用每个 API 类的方法。

本书在章节的安排上,基本上可分成三部分:

第一篇为 Java 语言详解。本篇是对 Java 语言的全面介绍,除了说明 Java 语言的特点之外,还介绍了 Java 程序运行的基本概念和 Internet 网上的各种 Java 资源。本篇也完整地论述了整个 Java 语言的语法,并列举了很多完整的程序实例。读者可以从本篇了解到 Java 语言设计的思路。如果读者对 C,C++ 或 Pascal 很熟悉的话,本篇阅读的速度将会加快。本篇的最后几章是为有志深入学习 Java 的读者所设计的,这几章的内容包括异常处理、I/O 流、多线程程序设计以及 Java 与 C/C++ 语言程序的接口,对读者开发大型 Java 程序将有很大的帮助。

第二篇为 Java 小应用程序详解。利用小应用程序来完成极有吸引力的功能是广大 Java 用户喜爱 Java 的原因之一。本篇将配合较多的实际例子,逐步带领读者在自己的 Home Page 上加入动画、图像、声音等小应用程序。除此之外,配合 AWT 和 Java 的网络功能,读者的 Home Page 不但可以产生出一个个的窗口,还可以与 WWW 服务器之间随时交流信息。

第三篇为 Java 类库详解。本篇的内容是从 Internet 网上收集整理出来的,这是一个完整、系统的 Java 文档,讲解了 Java 类及相应的构造函数、方法和变量,说明了其功能和用法;对于较复杂的方法,举例作了说明。在本篇中,类是按照字母顺序列出的,对每个类先给出一般的说明,然后列出其构造函数、方法、变量和常量及其语法。大部分构造函数和方法说明都带有代码例子,用以说明其用法。

本书第一篇由薛刚、刘禹、刘尧执笔,第二篇由胡立新、任中华、李志义、刘明、赵文斌、马军、刘闯执笔,第三篇由何胜利、陈辉、李东、高辉、张贵林执笔。全书由潘金贵教授和顾铁成审阅定稿。

限于水平,书中的不足,希望读者批评指正。

编 者

1998.9

目 录

第一篇 Java 语言详解

第一章 Java 语言的特点与运行环境	3
1.1 面向对象的特性	3
1.2 与平台无关的特性	4
1.2.1 严格的语言定义	4
1.2.2 Bytecode 中介结构	5
1.2.3 解释和编译的比较	6
1.3 多线程特性	6
1.4 Java 与 C 及 C++ 的关系	7
1.5 关于 Java 小应用程序	10
1.6 Java 程序开发环境 JDK	11
1.7 编译和运行第一个 Java 应用程序	12
1.7.1 编译 Java 应用程序	13
1.7.2 执行 Java 应用程序	13
1.8 编译和运行第一个 Java 小应用程序	13
1.8.1 编译 Java 小应用程序	14
1.8.2 Java 小应用程序浏览器 appletviewer 的使用	14
1.8.3 将 Java 小应用程序加到自己的主页中	15
1.9 Internet 上的 Java 联机资源	15
第二章 Java 的数据类型	19
2.1 布尔类型	19
2.2 字符类型	21
2.3 整型	23
2.4 浮点型	27
2.5 基本数据类型变量的预设值	32
2.6 数组类型	33
2.6.1 数组的定义和初始化	34
2.6.2 数组的使用	36
2.7 关键字	38
第三章 Java 运算符	40
3.1 单目运算符	40
3.1.1 递加和递减运算符	40

3.1.2	正负号运算符	42
3.1.3	自反运算符	42
3.1.4	类型转换表达式	42
3.2	加法运算符	43
3.3	乘法运算符	43
3.4	移位运算符	43
3.4.1	左移运算 < <	43
3.4.2	带符号数右移运算 > >	44
3.4.3	无符号数右移运算 >>>	44
3.5	相等性的运算	47
3.5.1	布尔值类型相等性	47
3.5.2	整数类型和浮点数类型相等性	47
3.5.3	引用相等性	47
3.6	关系运算符	49
3.7	按位取反运算符	49
3.8	逻辑运算符	50
3.9	三元运算符	53
3.10	赋值运算符	54
第四章	Java 程序的流程	55
4.1	if	55
4.2	switch 说明	56
4.3	while 说明	59
4.4	do 说明	59
4.5	for 说明	59
4.6	break 语句	61
4.7	continue 语句	61
第五章	引用、字符数组与字符串	65
5.1	Java 的动态内存机制	65
5.2	引用	66
5.3	字符数组与字符串	68
5.3.1	字符串常数	68
5.3.2	类 String	69
5.3.3	类 StringBuffer	74
5.4	命令行参数	77
第六章	类	78
6.1	类的继承关系	79
6.2	类的严格定义	80
6.2.1	abstract	81
6.2.2	final	81
6.2.3	public	82

6.3	变量和方法	83
6.3.1	变量域	83
6.3.2	方法域	83
6.4	public、protected 和 private	84
6.4.1	public	84
6.4.2	protected	85
6.4.3	private	86
6.5	再论类继承	88
6.6	final 关键字	90
6.6.1	final 变量	91
6.6.2	final 方法	91
6.7	static 关键字	91
6.8	关键字 abstract	93
第七章	接口与程序包	97
7.1	接口	97
7.2	程序包	101
第八章	Java 的动态内存管理	104
8.1	动态配置及垃圾回收	104
8.1.1	堆的概念(传统程序的内存空间图)	104
8.1.2	用动态内存配置的目的	104
8.1.3	何谓“垃圾回收”	105
8.2	new	107
8.3	构造函数和 finalizer	109
8.4	super 和 this	110
第九章	异常处理	113
9.1	异常处理的基本概念	113
9.1.1	传统的错误处理	113
9.1.2	Java 异常处理	114
9.2	Java 的异常处理机制	117
9.2.1	什么是“异常”	118
9.2.2	try 和 catch	119
9.2.3	finally	123
9.2.4	用 throw 产生异常	126
9.2.5	生成自己的异常	129
第十章	Java 的输入输出与数据流	131
10.1	输出数据流	131
10.1.1	类 ByteArrayOutputStream	133
10.1.2	类 FileOutputStream	136
10.1.3	类 PipedOutputStream	136
10.1.4	类 FilterOutputStream	136

10.1.5	类 BufferedOutputStream	137
10.1.6	类 DataOutputStream	138
10.1.7	类 PrintStream	139
10.2	输入数据流	140
10.2.1	类 InputStream	140
10.2.2	类 ByteArrayInputStream	143
10.2.3	类 FileInputStream	143
10.2.4	类 PipedInputStream	144
10.2.5	类 SequenceInputStream	145
10.2.6	类 StringBufferInputStream	147
10.2.7	类 FilterInputStream	147
10.2.8	类 BufferedInputStream	147
10.2.9	类 DataInputStream	148
10.2.10	类 LineNumberInputStream	150
10.2.11	类 PushbackInputStream	152
10.3	其他相关的类	152
10.3.1	类 File	152
10.3.2	类 FilenameFilter	152
10.3.3	类 RandomAccessFile	152
第十一章	多线程程序设计	154
11.1	什么是多线程	154
11.2	Java 与多线程	155
11.3	如何建立线程	156
11.3.1	继承类 Thread	156
11.3.2	类 Thread	157
11.3.3	实现接口 Runnable	160
11.4	资源的协调	162
11.5	锁定对象数据	163
11.6	锁定类数据	166
第十二章	Java 与 C 的接口	167
12.1	在 Java 中调用 C 语言函数	167
12.1.1	编写并编译 Java 程序代码	168
12.1.2	利用 javah 生成 ShowMessage.h	169
12.1.3	利用 javah-stubs 生成 ShowMessage.c	170
12.1.4	编写原生方法的程序代码	170
12.1.5	编译并执行	171
12.2	原生方法的参数与返回值	172
12.2.1	自动参数	172
12.2.2	简单数据类型的参数与返回值	173
12.2.3	复杂数据类型的参数与返回值	174

第二篇 Java 小应用程序详解

第十三章 小应用程序简介	177
13.1 小应用程序的生命周期	178
13.1.1 public void init()	179
13.1.2 public void start()	179
13.1.3 public void stop()	179
13.1.4 public destroy()	179
13.2 加入自己的代码	180
第十四章 日历小应用程序实例	185
14.1 如何在主页中加入日历程序	186
14.1.1 Applet 标签	186
14.1.2 CODE、WIDTH 和 HEIGHT	186
14.1.3 CODEBASE	187
14.1.4 ALIGN	187
14.1.5 HSPACE 和 VSPACE	188
14.2 向小应用程序传递参数	189
14.3 类 Date	191
14.3.1 构造函数	192
14.3.2 类方法	193
14.3.3 一般的设置方法和取值方法	193
14.3.4 其他方法	194
第十五章 图形与动画	196
15.1 图形坐标系统	196
15.2 字形与颜色	197
15.2.1 字型的设置	197
15.2.2 显示的方法	197
15.2.3 取得字型的数据	199
15.2.4 颜色的调整	201
15.3 绘图指令	202
15.3.1 画线	202
15.3.2 矩形	202
15.3.3 圆角矩形	203
15.3.4 立体矩形	205
15.3.5 椭圆形	206
15.3.6 画弧	206
15.3.7 多边形	208
15.4 限定作图的区域	209
15.5 复制图形	210
15.6 载入现成的图形文件	211

15.7	输出图形	214
15.8	动画与线程	215
15.8.1	时钟小应用程序版本 1.0	215
15.8.2	时钟小应用程序版本 2.0	217
15.8.3	时钟小应用程序版本 3.0	219
15.8.4	解决闪烁问题	223
第十六章	建立动画主页	225
16.1	建立一个简单的主页	225
16.2	建立动画主页	228
16.2.1	动画原理简述	228
16.2.2	如何载入图像数据	229
16.2.3	控制放映的速度	229
16.2.4	实例一	230
16.2.5	配音	233
16.2.6	实例二	235
16.2.7	某些改进工作	236
第十七章	截获鼠标事件与键盘事件	241
17.1	鼠标事件	241
17.1.1	mouseDown 和 mouseUp	242
17.1.2	mouseMove 和 mouseDrag	244
17.1.3	mouseEnter 和 mouseExit	246
17.2	键盘事件 keyDown 和 keyUp	248
17.3	事件处理程序 handleEvent	250
第十八章	Java 的网络功能	252
18.1	什么是 URL	252
18.2	URL 对象的构造法	253
18.2.1	绝对 URL 位置构造法	253
18.2.2	相对 URL 位置构造法	253
18.2.3	指定域的 URL 对象构造法	254
18.2.4	处理 MalformedURLException 异常	255
18.3	URL 类的基本方法	255
18.3.1	取出 URL 域的数据	256
18.3.2	将 URL 对象的内容转成字符串	257
18.3.3	比较两个 URL 的数据	257
18.4	直接读入 URL 的数据	258
18.5	Java 主页计数器	260
18.5.1	HTML 文件	260
18.5.2	服务器端:简单的 CGI 计数程序	261
18.5.3	客户端:Java 小应用程序	262
18.6	URL 的双向通信	263

第三篇 Java 类库详解

1	Applet 类	269
2	BorderLayout 类	274
3	Button 类	276
4	Canvas 类	278
5	CardLayout 类	279
6	Checkbox 类	283
7	CheckboxGroup 类	286
8	CheckboxMenuItem 类	287
9	Choice 类	289
10	Color 类	292
11	Component 类	297
12	Container 类	313
13	Dialog 类	318
14	Dimension 类	321
15	Event 类	322
16	FileDialog 类	331
17	FlowLayout 类	334
18	Font 类	337
19	FontMetrics 类	341
20	Frame 类	346
21	Graphics 类	350
22	GridBagConstraints 类	361
23	GridBagLayout 类	365
24	GridLayout 类	369
25	Image 类	371
26	Insets 类	373
27	Label 类	374
28	List 类	376
29	MediaTracker 类	382
30	Menu 类	388
31	MenuBar 类	391
32	MenuComponent 类	393
33	MenuItem 类	395
34	Panel 类	397
35	Point 类	398
36	Polygon 类	400
37	Rectangle 类	402
38	Scrollbar 类	407

39	TextArea 类	411
40	TextComponent 类	414
41	TextField 类	416
42	Toolkit 类	419
43	Window 类	424
附录	Java 类的结构	427

第一篇 Java 语言详解

第一章 Java 语言的特点与运行环境

1.1 面向对象的特性

Java 是一种面向对象的编程语言,它具有:

- (1) 封装性(Encapsulation):必须有模块化的性质以及信息隐藏的功能。
- (2) 多态性(Polymorphism):对同一种信息,不同的对象可以按对象本身的性质加以响应。
- (3) 继承性(Inheritance):可以定义一组对象之间的层次关系,下层的对象继承上层对象的特性,借此可实现程序代码的重复利用,并且有效地组织整个程序。
- (4) 动态联编(Dynamic binding):一个对象一旦生成之后,要使用这个对象,只需简单地把信息传递给它,不再需要去参考对象当初设计时的规格。只有在程序运行时,才会真正锁定需要的对象,这样的方式可以使程序设计具有最大的灵活性。

Java 语言对类的定义方式与 C++ 基本相同,参见程序 1-1 即可得知。

程序 1-1

// Java 的类实例

```
class Car {  
    int color_number;    //车的颜色编号  
    int door_number;    //有多少车门  
    int speed;          //车速  
    ...  
    push_break () {    //踩煞车的反应  
        ...  
    }  
    Add_oil() {        //加油的反应  
        ...  
    }  
}
```

类的作用之一,就是提供了“继承性”的实现方法。在面向对象程序设计中,所谓继承,是指被继承的类中所定义的性质和方法,都直接被后来的类所承接,同时后来的类还可以加入自己新的定义。就拿“汽车”类来说,可以定义出一种新的类叫“垃圾车”,“垃圾车”类直接继承自“汽车”类,所以原来“汽车”类中已经定义过的性质和对外界信息的处理方法,在“垃圾车”类上同样有效。即垃圾车有颜色,也有车门,踩了油门一样会跑。但是垃圾车的特殊用途——装垃圾,则必须再独立定义。

从这里可以看出继承的最大好处——程序代码的重复使用。原先在“汽车”类里写过的定

义、写过的程序,在“垃圾车”类中都不需要再次重写。另外一个好处是可以帮助编程人员在写程序时,很自然地运用面向对象的概念去组织所要处理的数据,使程序和数据的组织更加完善。

程序 1-2

```
// Java 的类继承实例
class Trash_Car extends Car {
    double amount;        //垃圾数量
    fill_trash()          //装垃圾的方法
    ...
}
```

1.2 与平台无关的特性

如果要让程序可以在各种不同的机器及操作系统上运行,那么在编写程序代码时,就不能使用编程语言中由编译器或平台本身定义的功能。例如,在 C 语言中,int 类型并没有定义它的实际长度是 2 个字节或 4 个字节,必须由所在的机器或操作系统确定。如果在使用时,只把它想象成 2 个字节长度的数据空间,那么当这个程序移植到别的平台时,很难保证不会发生问题。因此,编程语言严格的定义,是确保程序可以在各种平台上工作的第一步。在 Java 的语言定义中,看不到任何取决于工作平台或编译器的功能或特性。

另一个问题是,各种机器都有不同的低级汇编语言,在一种机器上编译好的机器码,不一定能直接拿到其他机器上使用。即使是同一种机器上的机器码,由于与程序的执行过程及操作系统相关,只要是不同的操作系统,编译好的机器码往往也有很大的差异,而不能互用,必须重做一次编译工作。例如,在同一台 PC 下,Linux 操作系统下的机器码并不能直接在 DOS 下使用。为了解决这个难题,Java 的策略是采取半编译、半解释的方式,定义出 Java 自己的虚拟机(Virtual Machine)。

1.2.1 严格的语言定义

为了确保程序代码不受各种平台的限制,在 Java 的语言定义中,所有部分都是经过严格定义的。也就是说,在 Java 的语言定义文档中,不会看到“这部分依各个机器不同而不同”或者“这部分由你所使用的编译程序决定”等之类的字眼。每个部分都是确定的,所以不管使用的机器及使用的编译器如何不同,最后出来的目标码都相同。

以基本的数据类型做例子,可以看到每一种数据类型的长度及表示方式都是确定的,如表 1.1 所示。

表 1.1 基本数据类型及其长度和表示方式

数据类型	数据长度	数据表示方式
byte	8 位	2 的补码
short	16 位	2 的补码
int	32 位	2 的补码

续表

数据类型	数据长度	数据表示方式
long	64 位	2 的补码
float	32 位	IEEE 754 浮点数标准
double	64 位	IEEE 754 浮点数标准
char	16 位	单码字符集

因为上面的数据类型都有标准的定义,程序设计者在使用这些类型时,也不会出现意义上的混淆。

另外一个例子是变量初值的设置。在许多语言中,变量的初值是未定义的,假如定义了一个整数变量 a,则一开始 a 的数值就是所谓的初值。就程序的设计概念上来看,初值是什么其实并不重要,因为变量的初值应该由设计程序的人来决定,而不是由系统或编译器来决定。但是如果写出下面一小段 C 语言程序:

```
int a,b;  
b = a + 5;
```

一般的编译器虽然会给出一些警告信息,但是程序依然可以通过编译并执行。至于程序执行时,a 会是多少,就不是程序设计者所能知道的了。

但是在 Java 中,即使像上面的程序,程序设计者依然可以知道最后 a 和 b 的值各是多少,因为在 Java 语言定义中对各种数据类型的变量初值,也有完整的定义,如表 1.2 所示。

表 1.2 各种数据类型及其变量的初始值

数据类型	初始值
byte	(byte)0
short	(short)0
int	0
long	0L
float	0.0f
double	0.0d
char	' \ u0000'
boolean	false
reference	null

所以执行上面的程序,最后 a 是 0,b 是 5。

1.2.2 Bytecode 中介结构

Java 解决各机器不同机器码限制的方法是定义出自己的一套虚拟机,以及这套虚拟机上所使用的机器码——Java Bytecode。在讨论 Bytecode 之前,先来看一个 Java 程序从编译到执行的整个过程,见图 1.1。

看了这个示意图以后,就能了解为什么要执行的 Java 程序是经过半编译、半解释的过程。一个编写好的 Java 程序源代码,先通过 Java 编译器编译,产生出 Java 虚拟机的机器码——Bytecode,再经过 Java 解释器将 Bytecode 转换成实际使用的机器和操作系统上的机器码去