计算机语言函数应用丛书

Java

计算机语言函数应用

李 明 袁晓君 编著

科学出版社

计算机语言函数应用丛书

Java

计算机语言函数应用

李 明 袁晓君 编著

科学出版社
2000

内 容 简 介

Java 自发布以来，取得了巨大的成功，它具有面向对象和网络编程的优点，已成为当今的主流编程工具。Internet 和万维网用户可以访问来自 Internet 上任意节点平台的独立的应用程序。Web 浏览器的能力也因 Java 作为其扩充语言而变得几乎无限，程序员只需依次编写 Applet，就可以运行在任何地方的任意机器上。

本书是 Java 应用程序和 Applet 程序员详尽的参考手册，描述了 Java 应用程序接口（API），即编写 Java 程序的一套标准库函数，包括 Java. lang 类库，Java. io 类库，Java. util 类库，Java. net 类库，Java. awt 类库，Java. awt. image 类库，Java. awt. peer 类库，Java. applet 类库和 Javal.2 新增类库的内容等。

图书在版编目（CIP）数据

Java 计算机语言函数应用/李明等编. -北京:科学出版社,2000.1
ISBN 7-03-007761-X

Ⅰ. 计… Ⅱ. 袁… Ⅲ. Java 语言-库函数 Ⅳ. TP311.13

中国版本图书馆 CIP 数据核字（1999）第 30853 号

科学出版社出版
北京东黄城根北街 16 号
邮政编码:100717
北京科普印刷厂 印刷
新华书店北京发行所发行 各地新华书店经售
*
2000 年 1 月第 一 版 开本：787×1092 1/16
2000 年 1 月第一次印刷 印张：48 1/2
印数：1—5 100 字数：1 127 000

定价：65.00 元

(如有印装质量问题，我社负责调换(环伟))

序　言

1. 起源

当人们追述计算机软件的发展时，会说B导致了C，C演化出C++，C++则又引发了Java。任何一种计算机语言的革新与改进不外乎两个原因，一是为适应环境和应用的改变，二是为了实现编程艺术的精细化。Java语言的产生同样也脱不开这两个原因。C语言的强大使它至今仍是一门十分通用的语言，但当一个程序超过25 000乃至1 000 000行时，它的复杂性已使C语言难以驾驭。因而产生了面向对象技术、更为强大的C++语言。而随着WWW与Internet的蓬勃发展，新的要求又不断提出，Java语言也就于此时应运而生了。

Java最初是由James Gosling，Patrick Naughton，Chris Warth，EdFrank和MikeSheridan于1991年在SunMicrosystems开发的。他们用18个月的时间开发出了第一个版本。这门语言最初叫“OAL”，在1995年改名为Java。

Java语言最初并非是为了Internet而设计的，它的最初动力源于为各种各样的家用电器设计一种可移植的且与平台无关的语言。GOSLING和他的小组就致力发展Java以使它所产生的编码能在各种各样的CPU上运行。幸运的是，此时Internet和WWW也迅猛发展，从而把Java推向计算机语言设计的前沿。Internet把各种各样的计算机操作系统连接在一起，可是用户却希望他们能够在这不同的平台上运行同样的程序。因此如何开发出一门可移植的语言已迫在眉睫。在1993年，Java小组的人已经认识到Java被最初设计所要解决的问题在Internet中也会同样遇到，而且会应用到更大的规模上。因此他们把Java语言的研究方向从面向消费电器转向网络编程。应该说，是Internet推动了现在Java的诞生，并把它推向了大规模的成功。从它于1995年第一次发布到今天，Java在Internet与WWW上找到越来越多的应用并仍在迅速向前发展。

2. 特点

Java之所以取得今日的成功是与它的许多特点分不开的，尤其是一些针对Internet的特点更使它能被得心应手地应用到网络编程中去。

(1) 安全性

每当从网上下载一个程序时，往往会冒着被病毒感染的危险，有时还要提防一些恶意的程序会刺探私人数据，如信用卡号、银行帐号和密码等。而Java则通过提供网络程序和私人计算机之间的防火墙而解决了这些问题。这是因为Java编译器的输出并非可执行代码，而是BYTECODE。BYTECODE是一个高度优化的指令集，在Java虚拟机器上运行，Java RUN_TIMES SYSTEM就是一个BYTECODE的解释器。

这可能有点奇怪，因为目前绝大多数语言出于效率的考虑均采用编译执行，而Java却采用了解释执行。这就保证了Java的安全性，因为Java程序的执行是在RUNTIMESYSTEM的控制之下，RUNTIMESYSTEM可以屏蔽外来系统的副作用。

(2) 可移植性

Java的解释执行同样也保证了它的可移植性。我们只需对不同的平台安装上不同的RUNTIMESYSTEM,Java程序即可在上面执行。尽管RUNTIMESYSTEM因平台的不同而不同，但它们均把Java程序解释为BYTECODE后再执行，从而我们很容易生成可移植的Java程序。

(3) 简单性

Java被特意设计得可以很容易地学会并有效使用。如果你有一些编程经历，将会发现学习，使用Java并不困难；而且，如果你已有一些面向对象的概念，学习Java会变得更加容易。而且Java继承了许多C/C++的概念，从而使大多数程序员在转向Java时毫无困难。另外，Java对一些基本的编程任务给出了明确的规范，并且尽力避免在编程中追求意想不到的效果，这也简化了Java的编程。

(4) 面向对象

面向对象技术对Java而言是一个不可缺少的部分，但不再像C++那样只作为一个选择。所有的Java程序都必须是面向对象的，通过面向对象技术的应用，一个复杂程序的各个部分可以被有效地结合在一起。

(5) 强壮性

由于Java要在多种多样的平台环境下运用，因此在设计Java时，强壮性是一个必须要考虑的问题。Java可使你在开发程序时更早更容易地发现错误。Java会在编译时检查错误，许多对其他语言十分隐蔽的运行错误会在Java编译时被发现。而且Java仍会在执行时检查错误。在一般程序中常见的错误如内存管理错误，你可能忘记释放已分配的空间，或者更糟的，释放一个其他程序正在使用的空间，Java会通过垃圾收理器自动地进行内存的分配与释放工作，从而避免了这类问题的发生。

(6) 多线程

Java可满足现实世界中交互的网络编程需要，同时还支持多线程的编程，使你可以在编写程序的同时，还可处理其他的程序。

(7) 高性能

尽管Java是解释执行的，但这并没有降低它的效率。JAVABYTECODE经过精心的设计而可被很容易地转化成高性能的机器代码。JAVARUNTIMESYSTEM在执行这些代码时没有丢失任何平台无关代码的优点 。

3. 类库简介

全书覆盖了Java1.1和1.2的所有类库，它们构成了Java语言编程的基础。

Java 1.1.x 包含的类库有：

(1) Java.lang

Java.lang类库提供的类和接口构成了Java语言和Java虚拟机器的核心部分。它提供了诸如OBJECT，STRING和THREAD，这些基本上是在每个程序中都要用到的类，并定义了基本的异常和错

误处理类 。

(2) Java.io

Java.io 提供了一系列的输入输出类，也可以向文件或其他 I/O 资源进行读写操作 。

(3) Java.util

Java.util 包含了 UTILITY 类和相关的接口。

(4) Java.net

Java.net 包含了和网络相关的类及其接口。

(5) Java.awt

Java.awt 包含了标准的图象用户接口元素，如按钮列表菜单等。

(6) Java.awt.image

Java.awt.image 包含了复杂图象操作所需要的类及接口 。

(7) Java.awt.peer

Java.awt.peer 包含了连接 AWT 组员和它们的窗口系统所需要的接口。

(8) Java.applet

Java.applet 包含了生成 Applet 所需要的类和接口。

Java 1.2 新增的类库有:

(9) Java.awt.dnd

这个类库提供了支持拖放操作的类和界面。

(10) Java.awt.font

这个类库提供了与风格有关的类和界面。

(11) Java.awt.geom

这个类库提供了一些 Java 2D 类。这些类的定义与操纵与二维几何相关。

(12) Java.awt.im

这个类库提供的类和界面使所有编辑控件能够接收日文、中文和韩文文字。

(13) Java.awt.image.renderable

为了产生独立于“图象增强器”的图象，这个类库提供一些类和界面。所谓独立于“图象增强器”的图象就是指对某一图象所做的操作独立于任何对该图象的“增强”，如提供图象的草稿预览。

（14）Java.awt.print

这个类库的类和界面提供了与打印相关的 API。

（15）Java.beans.beancontext

这个类库的类和界面相关于“Bean 容器”。一个 Bean 容器能容纳 Java Beans 并且为它所包含的 Beans 定义了执行环境。

（16）Java.lang.ref

这个类库提供了一些“引用对象类”，这些类提供了与无用单元收集器的有限的交互。

（17）Java.rmi.activation

这个类库支持“RMI 对象激活”机制。

（18）Java.security.spec

这个类库提供了密匙和算法参数规格说明的类及界面。

（19）Java.util.jar

这个类库提供了创建及读写 JAR（Java ARchive）文件的类和界面。

（20）Javax.swing

这个类库提供了一个“轻量”控件集。所有 swing 控件均用 Java 写成，并且尽可能地实现平台无关性。

（21）Javax.swing.border

这个类库的类和界面能够在 swing 控件周围画出多种风格地边框。

（22）Javax.swing.colorchooser

这个类库包含了被 JcolorChooser 所使用的几个类与界面。（JcolorChooser 类提供了一个使用户可以操纵与选择颜色）。

（23）Javax.swing.event

这个类库提供了可以被 swing 控件触发的各类事件。

（24）Javax.swing.filechooser

这个类库包含了被 JFileChooser 所使用的几个类与界面。

（25）Javax.swing.plaf

这个类库提供了一个界面与许多抽象类使得 swing 控件有动态转变显示风格（如 Windows，Motif 或 Metal 风格）的能力。

（26）Javax.swing.plaf.basic

提供“基本”的显示风格（如，包含了 Windows 和 Motif 风格）。

（27）Javax.swing.plaf.metal

提供 Metal 显示风格（Look-and-Feel）。

（28）Javax.swing.plaf.multi

这个类库的界面与类允许用户通过基本的 Look-and-feel 提供一些辅助的 look-and-feel。

（29）Javax.swing.table

这个类库提供了丰富的类和界面处理 JTable 控件。

（30）Javax.swing.text

这个类库的类和界面主要处理可编辑或不可编辑的文本控件。

（31）Javax.swing.text.html

提供类（HTMLEditorKit）及其支持类去创建 HTML 编辑器。

（32）Javax.swing.tree

这个类库提供了丰富的类和界面去处理 JTree 控件。

（33）Javax.swing.undo

这个类库支持了在一个应用中的 undo/redo 操作。

目　录

Java 1.1.x 类库

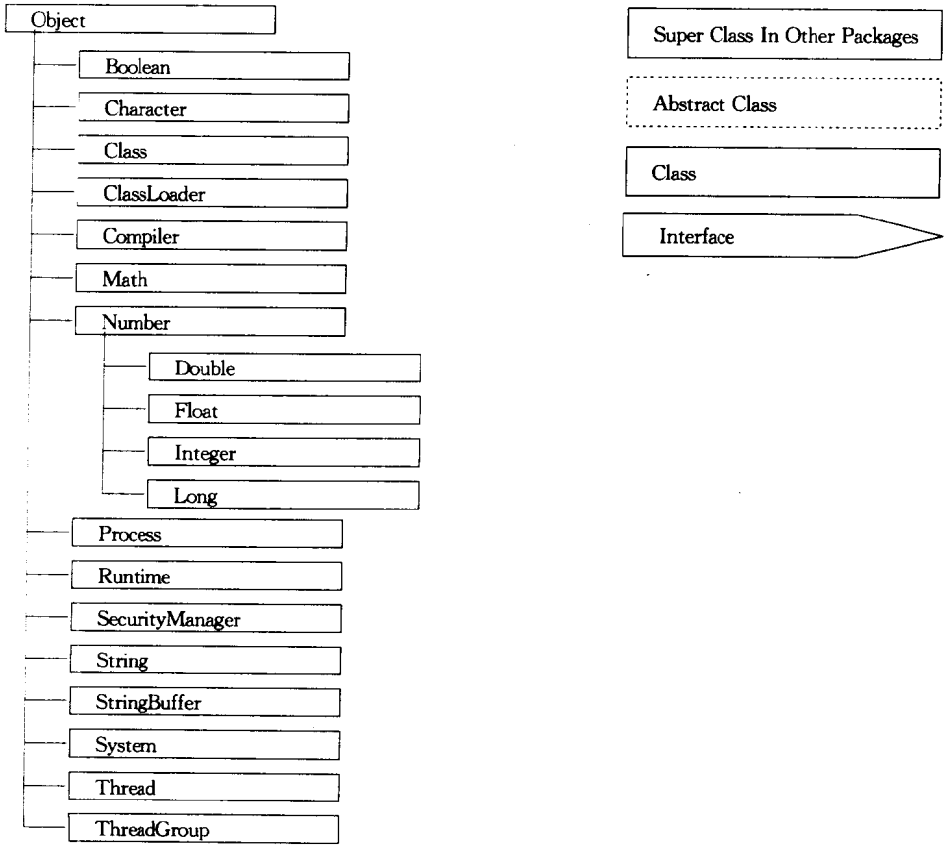
1 Java.lang 类库

1.1 类库介绍

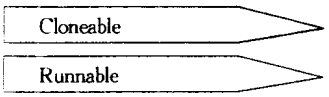
可以肯定地说，在所有的 JavaAPI 程序类库中，Java. lang 是最重要的，它提供 Java 语言中的诸如 Object、String 和 Thread 等核心类与接口。这些类中的任何一个丢失，运行时都不会启动。这些类自动导入到每个 Java 程序中，没有必要显示地导入它们。鉴于这些原因，由 Java 虚拟机抛出的错误和异常都出现在这个类库里。Java. lang 类库还包含基本类型包装器、访问系统资源的类、数学类和安全类等。

1.2 类库层次图

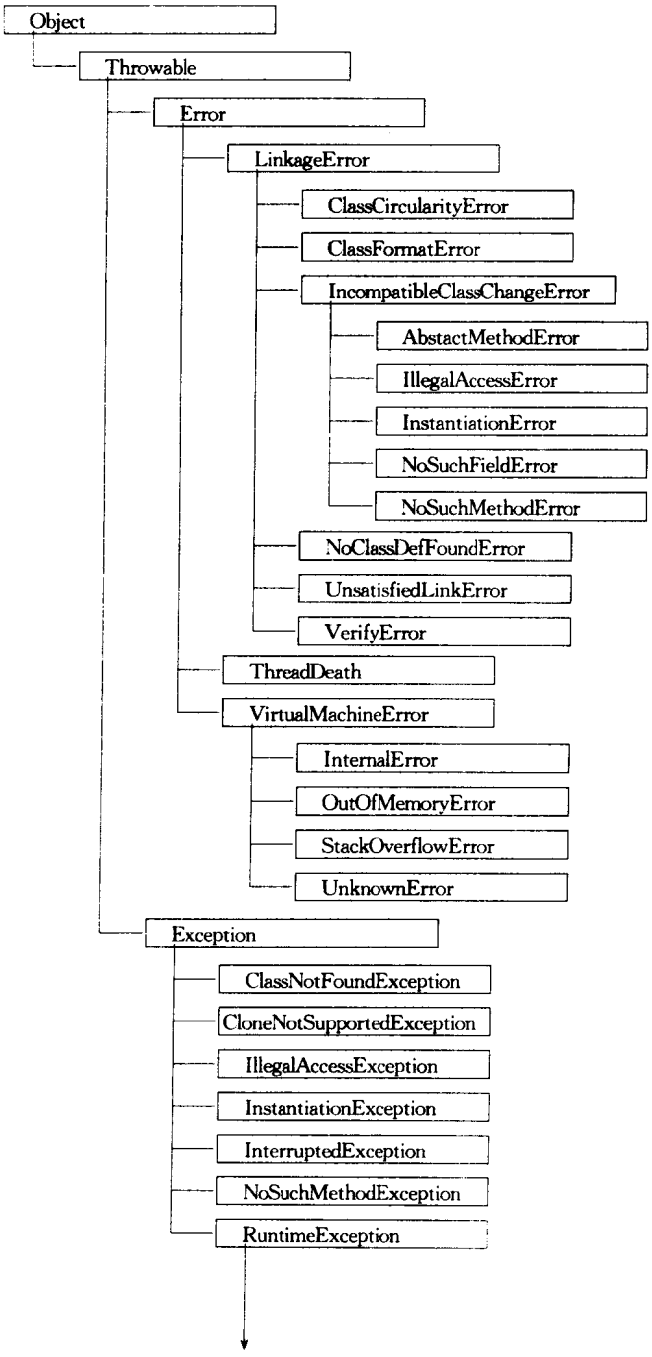
Classes

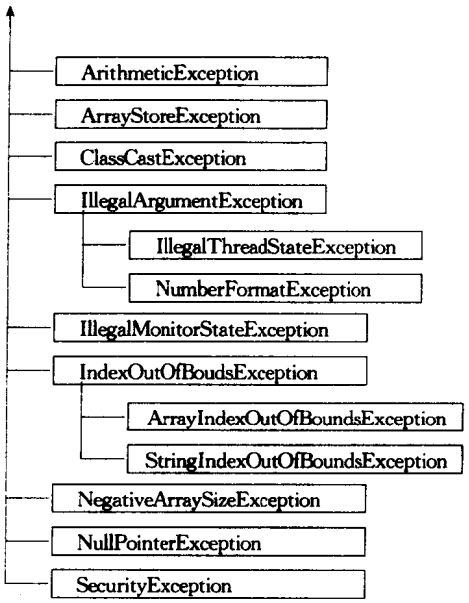


Interfaces



Errors and Exceptions





1.3 类库列表

1.3.1 基本类

Class 类

Class 类是封装了类和对象的属性特征的类，它包含解释一个 Java 类的信息。Class 类的实例表示一个运行 Java 应用程序的类和接口的信息。Class 对象能表示在程序运行的类，系统的任一对象都是一些 Class 类的实例。

Class 类没有公用的构造函数。Class 对象由 Java 虚拟机在通过调用类装载器中的 defineClass 方法装载类时自动构造。

语法：public final class Java. lang. Class

extends Java. lang. Object

参见：ClassLoader，Object. getClass()

类 方 法	描 述
forname ()	返回与指定类名的类相关联的 Class 对象。
getClassLoader ()	确定该类的类装载器。
getInterfaces ()	确定该对象表示的类或接口所实现的接口。
getName ()	返回所表示的类或接口的适当的全名。
getSuperClass ()	返回该对象所代表的类的超类。
isinterface ()	确定该对象是否表示一个接口。
newInstance ()	创建类的新实例。
toString ()	把对象转化为一个字符串。

forname ()

语法：public static native Class forName (String className)

throws ClassNotFoundException

功能：按字符串 className 所指定的类名返回相关的 Class 对象。

参数：className　所需类的全名。

返回：具有指定类名的类的 Class 描述符。

异常：ClassNotFoundException ()

没有找到该类。

参见：getName ()

getClassLoader ()

语法：public native ClassLoadeer getClassLoader ()

功能：确定该类的类装载器。

返回：创建当前类或接口的类装载器，如果类装载器没有创建该类，则为 null。

参见：ClassLoader

getInterfaces ()

语法：public native Class [] getInterfaces ()

功能：确定当前对象表示的类或接口所实现的接口，并将其返回。

如果当前类实现了多个接口，则返回值为一个元素，表示所有该类所实现接口的对象数组。它们在数组里的次序对应于该对象表示的类定义语句中的 implements 子句中接口名的次序。

如果该对象继承了多个父接口，则返回包含所有父接口的数组。接口对象在数组里的次序对应于该对象表示的接口定义语句中 extends 子句中的接口名次序。

如果类或接口没有实现接口，则返回长度为 0 的数组。

返回：该类所实现的接口数组。

参见：getSuperclass ()

getName ()

语法：Public String getName ()

功能：返回所表示的类或接口的全名。

返回：该对象所表示的类或接口的全名。

参见：toString ()，forName ()

getSuperClass ()

语法：public Class getSuperclass ()

功能：若该对象表示非 Object 类，则返回代表该类的父类的对象。

若该对象表示 Object 类或一个接口，则返回 null。

返回：该对象所代表的类的超类。

参见：Object. getClass ()

isinterface ()

语法：public native boolean isInterface ()

功能：确定该对象是否表示一个接口。

返回：若该对象表示一个接口，则为 true，否则为 false。

参见：toString ()

newInstance ()

语法：public native Object newInstance ()
throws InstantiationException, IllegalAccessException
功能：为当前类创建一个新实例。
返回：该对象所代表的类新分配的实例。类似于不带参数的 new 表达式的执行结果。
异常：InstantiationException ()
一个应用程序试图实例化一个抽象类或一个接口，但由于某种原因这种实例化失败。
IllegalAccessException ()
类或初始化器不可访问。

toString ()

语法：public String toString ()
功能：把对象转化为一个以字符串“Class”或“Interface”开头，后跟空格和该类适合的全名的字符串。
返回：该对象的字符串表示。
覆盖：Object 类中的 toString。
参见：getName ()，isInterface ()

Object 类

Object 类处于 Java 类层次结构的最上层，是所有类的父类。Java 语言中的所有类都是直接或间接继承 Object 类而得到的，Obect 类中定义的数据和方法均可用于子类。子类常常覆盖一些方法（比如，clone ()，equals ()）的实现。
语法：public class Java. lang. Object
参见：Class，Cloneable，System. gc ()，Thread，Hashtable

构造方法

Object ()

语法：Public Object ()
功能：分配 Object 类的一个新实例。

类 方 法	描 述
clone ()	创建对象的拷贝。
equals ()	比较两个对象是否相等。
finalize ()	清理对象。
getClass ()	获取同该对象有关的类的名称。
hashCode ()	计算该对象的哈希码。
notify ()	唤醒等待对象监视器线程。
notifyAll ()	唤醒所有等待进入监视器的线程。
toString ()	送回对象的字符串表示。
wait ()	等待来自另一线程调用 notify () 方法来唤醒当前线程。