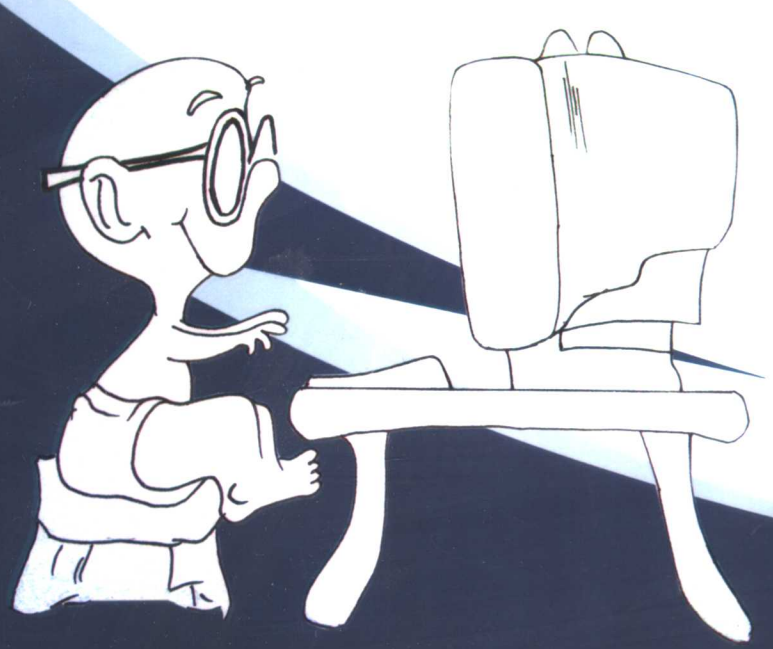
内附光盘

Java 语言编程 基础教程

宋振会 编著



清华大学出版社

Java 语言编程基础教程

宋振会　编著

清华大学出版社
北　京

内 容 简 介

Java 是美国 SUN 公司在 1996 年正式推出的纯面向对象的编程语言，主要用于 Internet 网络编程，目前为 Java 2 版本。本书按照此标准为基础，对 Java 进行了全面、详细的介绍。

Java 是在 C++的基础上发展起来的，因此其基本语法和 C++类似。为了让没有 C++基础的读者也能读懂本书，本书在前面章节安排了 C++的基础内容。概括起来本书内容主要包括：从 C++编程转到 Java 编程；常量、变量和内存；运算符、优先级和结合律；面向对象的编程方法（类）；条件判定和循环；创建窗体界面的组件；使用布局管理器布局界面；创建基于 Web 的 Applet 应用；窗体界面的交互和事件处理；程序运行中的异常处理；多线程编程；文件管理和输入/输出流；基于 Web 的网络编程。本书编写时参考了大量的国际软件工程师培训教程，又借鉴了作者多年的编程经验和教学经验，采用符合国际性标准的编程方法和惯例，将一些高深、抽象的理论，通过大量的程序案例进行讲述，使读者阅读起来通俗易懂。

本书是学习 Java 语言编程的优秀教程，内容丰富，讲述清楚，实例典型而丰富，适用于 Java 培训学员、高等院校及职业院校的学生、其他 Java 编程爱好者。

图书在版编目（CIP）数据

Java 语言编程基础教程/宋振会编著. —北京：清华大学出版社，2005.5

ISBN 7-302-10648-7

Ⅰ.J…　Ⅱ.宋…　Ⅲ.JAVA 语言-程序设计-教材　Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2005）第 020118 号

出 版 者：清华大学出版社　　**地　　址**：北京清华大学学研大厦
http://www.tup.com.cn　　**邮　　编**：100084
社 总 机：010-62770175　　**客户服务**：010-62776969

组稿编辑：欧振旭
文稿编辑：刘　丽
封面设计：姜凌娜
版式设计：冯彩茹
印 刷 者：北京四季青印刷厂
装 订 者：三河市新茂装订有限公司
发 行 者：新华书店总店北京发行所
开　　本：203×260　**印张**：22　**字数**：498 千字
版　　次：2005 年 5 月第 1 版　2005 年 5 月第 1 次印刷
书　　号：ISBN 7-302-10648-7/TP·7210
印　　数：1～5000
定　　价：32.00 元（附光盘 1 张）

前言

编写目的

目前，由于软件产业的美好前景和诱人薪酬，使软件培训和学习如火如荼，热遍大江南北。诸如 NIIT（印度国家信息技术学院）、北大青鸟等知名的培训机构在国内就有 1000 多家，培训学员达 200 多万。本书作者为 NIIT 特聘教师，在 Java 培训和教学过程中，深切体会到需要一本内容翔实，并且通俗易懂的教程作为 Java 培训学员或自学人员的教材或参考书。为了满足这些读者的需求，作者编写了本书。

编写本书的另一个目的是，作者试图以新的角度来探讨培训和自学教程的写作模式，给现有相关书籍的千篇一律带来一些新思想。作者希望通过本书，能将复杂的问题简单化，能使读者掌握 Java 编程的捷径。

主要内容

本书内容主要包括：从 C++编程转到 Java 编程；常量、变量和内存；运算符、优先级和结合律；面向对象的编程方法（类）；条件判定和循环；创建窗体界面的组件；使用布局管理器布局界面；创建基于 Web 的 Applet 应用；窗体界面的交互和事件处理；程序运行中的异常处理；多线程编程；文件管理和输入/输出流；基于 Web 的网络编程。

读者对象

本书明确定位于学习 Java 编程的入门与提高人员。主要适合以下读者：

- 参加 Java 培训的学员。
- 高等院校学习 Java 编程的学生。
- 中职、高职学校学习 Java 编程的学生。
- 其他 Java 爱好者或自学人员。

特色提示

本书是一本与众不同的书籍，它将带给读者耳目一新的感觉。本书具有以下特色：

- 本书结合 NIIT 软件培训教材的特点编写，在内容的选取、讲解、实例及课后实践等方面都力求有代表性。
- 本书探讨了快速授课的教学模式，是一本很好的 Java 培训教材和自学教材。

- 选材准确，不讲废话。本书的内容是任何一个初学 Java 编程的读者所必须掌握的最基本、最常见、最实用的内容。不符合 Java 初学者的内容一概不涉及。
- 形式新颖，适于阅读。本书摈弃了传统图书编排方式的呆板，代之以活泼而清新的风格，让读者阅读时有一种轻松感。

阅读建议

作为一本基础教程，作者建议读者按照章节的顺序阅读，并且注意阅读每章后的小结，完成独立实践。对于书中所有范例程序的源代码，希望读者能在读懂代码的基础上，亲自上机调试，看能否正常运行。只有这样，才能真正深刻理解所学的内容。

配书光盘

本书附带一张光盘，内容为本书所有范例程序的源代码。其中，部分代码为片段代码，部分代码为有意设置的需要读者更正的错误代码，这些代码在书中都有详细讲述，并不能完全编译运行。其他完整的范例程序都是在 JDK 1.4 编译环境下编译，能正常运行，无任何错误。

配书光盘中的源代码按章收录，并且文件名和编号与书中完全对应。例如，书中第 1 章的“程序 J01_Array.java”对应的源代码在光盘中的目录为：/第 1 章　从 C++编程转到 Java 编程/J01_Array.java。请读者按此方法在本书的配书光盘中检索本书范例程序的源代码。

作者情况

本书主要由 NIIT 特聘教师宋振会教授编写。作者长期讲授 Java、C++、SQL 等课程，在教学过程中积累了丰富的经验，了解初学人员的学习特点，熟悉教学的重点与难点。NIIT 软件培训基地的李秋芬、张瑾、孔祥国等老师在素材提供、代码调试、后期审阅等方面做了大量工作。在此一并表示衷心的感谢！

由于作者水平所限，加之写作时间比较仓促，书中可能存在疏漏之处，恳请广大读者批评与指正。

E-mail：0532-2906053@tom.com

tel2906053@tom.com

编著者

2005 年 2 月

目 录

第1章

从C++编程转到Java编程

目标：

本章中，你将学习：

- ☑ Java概述
- ☑ Java与C++的对比
- ☑ 把C++代码转换为Java代码

Java 概述

Java 的历史

Java 是由 Sun Microsystems 公司开发的面向对象的编程语言。Java 语言是在 C++语言的基础上发展起来的，但和 C++是两种不同的语言。Java 语言的设计者模仿 C++语言，使语言短小而简单，而且易于在不同操作平台之间移植。

Java 的创始者为 Sun Microsystems 公司的首席程序员 James Gosling。他和他的工作小组开始创建一个控制电子电器的软件项目。因为其面向对象的特点，Gosling 开始使用 C++语言开发此项目。但是他发现程序中出现的许多问题都是与 C++语言的复杂性有关。例如，包括指针错误和内存泄漏之类的程序错误。就在这个时候，Gosling 决定创建一个新的语言以克服 C++语言中的问题。

Gosling 喜欢 C++语言的基本语法和面向对象的特点。因此，他设计的新语言以 C++语言为主线，但克服了 C++的缺陷。结果产生了一个称为 Oak 的新的编程语言，后来被重新命名为 Java。

如今，Java 被广泛应用于网络编程，它最大限度地利用了网络，也提供了大量的、丰富的类库，以满足网络化、多线程、面向对象系统的需要，使程序员可以非常方便地创建自己的系统。因此，学习 Java 语言，主要是学习使用 Java 类库。

Java 程序

Java 程序有两类：应用程序（application）和小应用程序（applet）。

应用程序

应用程序是可在任何操作系统提示下执行的程序。单独的应用程序可以是基于窗口的应用程序或基于控制台的应用程序。

- 基于窗口的应用程序一般是图形用户界面，有一些操作平台，如 Microsoft Windows、Macintosh、Motif 和 OS/2 等。
- 基于控制台的应用程序是基于字符的应用程序，没有图形用户界面。

单独的应用程序使用单机资源，它驻留在本地计算机的硬盘上。当需要执行此应用程序时，把它从硬盘调入到内存中并执行。

小应用程序

小应用程序是在 Web 页面内执行的 Java 程序，它不是驻留在本地计算机的硬盘上，而必须在网络上通过 Web 浏览器装入 Web 页时被调入和执行，常用的浏览器有：Microsoft Internet Explorer、Netscape Navigator 和 HotJava 等。

小应用程序使用简单，但必须启动 Internet 访问 Web 页面。它们驻留在远程计算机上，当本地计算机需要执行小应用程序时，小应用程序从远程计算机装入本地计算机的内存中，通过浏览器解释小应用程序，与本地计算机链接并执行。

Java 的性质

Java 是一个 Internet 编程语言

Web 提供了来自世界上任何地方的计算机的可访问性，可用 Java 程序跨网络地访问数据，不管其源平台是什么。

Java 是安全的

跨 Internet 下载到计算机上的程序可能会带有病毒。由于 Java 在用户计算机上作了强类型检查，对程序所作的任何变动都标识为错误，且程序不被执行。因此，Java 是安全的。

Java 是独立于语言的平台

Java 编译器把 Java 代码编译成被 JVM 环境理解的中间字节代码。Java 解释器或任何的 Java 使能的 Internet 浏览器便可执行这些字节代码。

由于字节代码的编译过程和浏览器对它的解释，使 Java 程序可在不同的硬件和操作系统上执行，只要求该系统有 Java 使能的 Internet 浏览器或解释器就行。

Java 是高性能的语言

Java 程序的执行速度可与 C 和 C++之类的基于编译的语言相媲美。Java 程序比其他基于解释的语言所编写的程序的执行速度要快，如 Basic 语言。

Java 是简单的

Java 是一种简单的编程语言，即使您没有任何 Java 基础也可很快学会它的使用。Java 程序员不必知道 Java 的内部功能。Java 的语法类似于 C++的语法，但 Java 不支持指针、多重继承性、goto 语句及运算符重载。

使应用开发周期慢下来的 C++缺陷已在 Java 中删去。例如，Java 防止由于不合适的内存用法而引起的错误，因为在 Java 中，程序员不必操纵内存。

Java 的简单性还表现在其精简的系统，它力图使用最小的系统实现足够多的功能。其基本解释器只有 40KB 左右，加上标准类库和线程的支持，也只有 210KB 左右。

Java 与 C++的对比

因为 Java 是在 C++的基础上发展起来的，因此 Java 与 C++有许多共同点。之所以有 Java，是由于 C++的某些不适应性和复杂性，所以这二者又存在一些区别。Java 是一个纯面向对象语言，因为 Java 中每个语句都写在类内。C++中，main()方法总是写在类外，一些全局变量和全局函数也是写在类外。

Java 中，除了初等数据类型外，其他的数据类型都是对象。即使初等数据类型也可封装在类内。Java 与 C++的对比如下。

数据类型

- Java 中除了 C++中有的数据类型外，还有如下两个附加的初等数据类型：
  - byte 数据类型，占有一个字节的内存空间，可存储整数。
  - boolean 数据类型，可存储两个布尔值（true 或 false）之一。
- Java 省略了 C++中有的 pointers（指针）和 structs（结构）。
- Java 中，字符数据类型存储 Unicode 格式的字符，不像 C++中为 8 位的 ASCII 格式。Unicode 可存储亚洲语言字母表的 16 位字符格式。
- Java 中，数据类型有固定的大小，而不管所用的是什么操作系统。不像在 C++中，对于不同的操作系统和不同位数的计算机其数据类型有不同的大小。

Java 的数据类型如表 1-1 所示。

表 1-1　Java 的数据类型列表

数据类型	大　小	范　围
布尔型（boolean）	1bit	布尔值（ture 或 false）
字符型（char）	16bit	存放一个字符
字节型（byte）	8bit	有符号：-128~127 即-2^7~（2^7-1） 无符号：0~255
短整型（short）	16bit	-32768~32767 即-2^{15}~（$2^{15}-1$）
整型（int）	32bit	-231~231-1
长整型（long）	64bit	-263~263-1
单精度浮点型（float）	32bit	-231~231-1
双精度浮点型（double）	64bit	-263~263-1

为了让没有 C++基础的读者，更好地理解本书内容，将在第 2 章中详细讲述数据类型的相关知识。

运算符和构造

Java 中支持 C++中所有的运算符。Java 中诸如 if...else、while 和 do...while 构造的表达式都以布尔值为其结果。switch 构造和 for 循环都类似于 C++中相应的构造。

本书将在第 3 章中详细讲述运算符的相关内容。

继承性

Java 不支持多重继承，这意味着不可继承一个以上的类，但是可以通过接口来实现。

Java 中所有的类都是从 Object 类派生出来的。

为继承一个类，要使用 extends 关键字。例如，由 Book 类派生出 TextBook 类。

在 C++中代码如下：

```
class TextBook : public Book
{
    //类体
};
```

在 Java 中代码如下：

```
class TextBook extends Book
{
    //类体
}
```

注意

在 C++中类以分号（;）结束，否则编译器会报错。在 Java 中类后面没有分号（;）。若有分号，编译器也可以通过。

方法和方法重载

在 C++中，为了兼容 C，可以有独立的函数。因此 C++是在面向过程的编程语言基础上，增加了面向对象的机制。

Java 是一个纯面向对象的编程语言，在类之外不允许有任何语句（包括函数）；在 Java 中，在类中的函数称为方法，Java 允许方法（函数）重载，但不支持 C++中的运算符重载。

数组和 String 对象

C++中的数组是元素的集合，而 Java 中的数组是实在的对象，因为可用 new 运算符分配内存给数组。

Java 中需检查数组访问以确保其下标落在数组的范围内。语法如下：

```
<数据类型> [ ]<数组名>;
<数据类型> <数组名> [ ];
```

上面给出的第一个语法是 Java 中惯用的。声明数组之后，就必须为它分配内存。

数组长度存储在变量 length 中，该变量是 length 所有数组的一个成员变量。

Java 中数组用法举例如下：

```
/*程序 J01_Array.java：Java 中数组的用法*/
public class J01_Array
{
    int [ ]marksEnglish;                    //声明整型数组
    int [ ]marksHistory={70,80,90};         //声明同时初始化
    String [ ]subjects;                     //声明串对象数组
    public J01_Array()                      //声明构造函数
    {
        marksEnglish=new int[3];            //创建包含 3 个整型数据数组
        marksEnglish[0]=30;
        marksEnglish[1]=40;
        marksEnglish[2]=50;
```

```
        subjects=new String[2];          //创建包含 2 个串对象的数组
        subjects[0]=new String("English");
        subjects[1]=new String("History");
    }
}
```

Java 中，串 String 是一个实在的对象，而不像 C++那样是一个以空为终结的字符数组。因为 String 是一个实在的对象，它与 C++中的串相比有以下两个主要优点：

- Java 中，String 对象是一致的。获取串和访问串的方式对所有系统都是一致的，它们有良好定义的编程接口。
- Java 中，String 对象是可靠的，它们不会引起程序中的内存泄漏。

main()方法

Java 中不支持单独的方法，包括 main()方法在内，所有函数方法都在类声明中定义。

Java 中 main()方法的用法如下：

```
/*程序 J02_Welcome.java：Java 中 main()方法的用法*/
public class J02_Welcome
{
    public static void main(String[] args)
    {
        System.out.println("**************************");
        System.out.println("* 欢迎来到 Java 世界! *");
        System.out.println("**************************");
    }
}
```

- Java 中 main()方法和它所在的类应声明为 public，因为 Java 运行环境必须访问 main()方法以执行程序。
- main()方法应声明为 static，因为此类的任何对象在创建前必须存在。
- 其命令行参数是 String 类型数组变量 main(String[] args)。参数的个数由 String 类对象确定。
- 文件名必须和 main()方法所在的公共类名相同。

类、对象和方法

在 C++中，包含 iostream 类，该类封装了 cout 对象，用于在屏幕上显示数据，cin 对象用于接受键盘输入。

在 Java 中，包含 System 类，该类封装了 PrintStream 类的 out 对象和 InputStream 类的 in 对象。out 对象和标准输出设备（显示器）相连，in 对象标准和输入设备（键盘）相连。

- System 类，Java 使用了所有系统资源，如在 System 类帮助下的显示输出和接受键盘输入。此类包含处理系统资源所需的所有方法。

- out 对象，是封装在 System 类中的一个 PrintStream 类对象，表示标准的输出设备（显示器）。
- println(String x)方法，PrintStream 类 println(String x)方法用于在屏幕上显示一个字符串。例如：

  System.out.println("* 欢迎来到 Java 世界! *");
- in 对象，是封装在 System 类中的一个 InputStream 类对象，表示标准的输入设备（键盘）。
- InputStream 类，是一个抽象类，但不能创建该类的对象，而只能用于继承。
- InputStreamReader 类，是抽象类 Reader 类的子类，它的构造符以 InputStream 类对象为参数，将字节流包装成 Unicode 字符流。例如：

  new InputStreamReader(System.in);
- BufferedReader 类，是抽象类 Reader 类的另一个子类，它的构造符以 Reader 所有子类对象为参数，将 Unicode 字符流包装成缓冲流，使输入速度更快。例如：

  BufferedReader inObj=new BufferedReader(new InputStreamReader(System.in));
- readLine()方法，每次从输入流中读取一行数据，返回一个字符串。例如：

  String sName=inObj.readLine();

因为 Java 使用 System 类以便与显示器和键盘设备接口，所以用 System.out.println()替代 C++中 ucout 对象。用（new BufferedReader(new InputStreamReader(System.in))）对象的 readLine()方法替代 C++中的 cin 对象。

注意

以上内容，初学者只需简单了解即可，详细内容将在第 12 章中讲述。

- Number 类，是一个抽象类，其子类包含各种基本数据类型的相关类，如下：

```
java.lang
Class Number
java.lang.Object
|
+—java.lang.Number
All Implemented Interfaces:
    Serializable
Direct Known Subclasses:
    BigDecimal, BigInteger, Byte, Double, Float, Integer, Long, Short
```

各个类对应的 parseXXX(String s)方法可以将字符串转换为相对应的各种数据类型。该方法是抽象方法，可以用类名直接调用该方法。例如：

```
String str1="123";                  //123 是一个字符串
int iNum1= parseInt(str1)           // iNum1=123，整型
String str2="100.5";                //100.5 是一个字符串
float fNum2=parseFloat(str2);       // iNum2=100.5，浮点型
```

执行 Java 程序

- 称为 Java 虚拟机 JVM 的程序执行 Java 程序。JVM 包含运行环境和类加载器。因此要运行 Java 程序必须先安装 Java 虚拟机（JVM）。
- 必须保存以*.java 为扩展名的 Java 源文件。该文件名应与包含 main()方法的公共类名一样。当编译 *.java 文件时，便创建了*.class 文件。
- 为了编译 java 文件，应使用 java 实用程序编译成*.class 文件。
- 为了执行 java 程序，应使用 java 实用程序执行该*.class 文件。

为了编译和执行上面有公共类 J02_Welcome 类的 J02_Welcome.java 文件，应按以下步骤操作：

1. 在 C 盘根目录保存 Java 文件为 J02_Welcome.java。
2. 启动 DOS 窗口（选择“开始”|“运行”命令，执行 cmd 命令）。
3. 在命令提示处，输入 javac　J02_Welcome.java 命令，按回车键。
4. 在此命令提示处，输入 java　J02_Welcome 命令，按回车键。
5. 编译、执行程序及输出结果如图 1-1 所示。

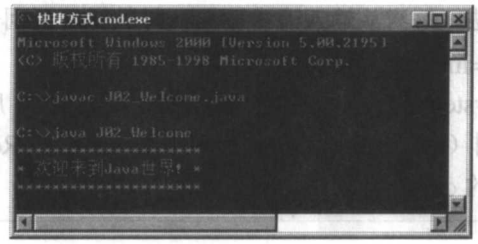


图 1-1　程序 Jo2_welcome.java 输出结果

注意：即使安装了 Java 虚拟机（JVM），也可能无法运行 Java 程序，因为还需进一步设置系统特性。如下：

1. 右击“我的电脑”，从打开的右键菜单中选择“属性”，打开“系统特性”对话框，选择“高级”选项卡，如图 1-2 所示。
2. 在“高级”选项卡中，单击“环境变量”按钮。打开“环境变量”对话框，如图 1-3 所示。
3. 在“系统变量”列表框中，选择 Path 变量，然后单击“编辑”按钮。在变量值编辑框中，添加 java 虚拟机的安装路径，如 C:\j2sdk1.4.，如图 1-4 所示。

Java 包

Java 使用包来组织相关的类。通过使用 import 语句，可在程序中使用 Java 包。这有点类似于 C++的 include 语句，包含头文件。差别是 Java 的包中只包含类，而 C++的头文件中可包含独立的方法。另一个区别是，包有类似于目录结构的层次结构。