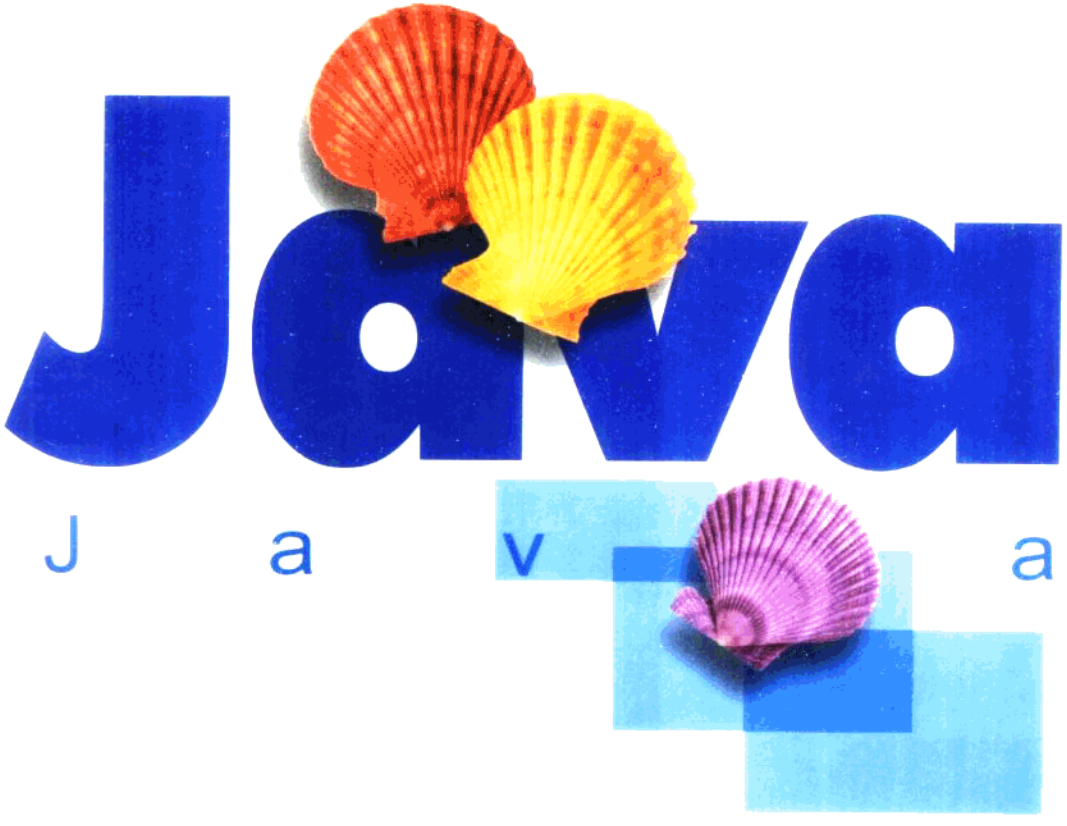
◎实用程序设计开发丛书

Shiyong Chengxu
Sheji Kaifa Congshu



Java 从入门到精通

◎罗光春 主编

◎电子科技大学出版社

Program

前　言

Java 程序是一门新兴的程序设计语言，它吸收了当今计算机技术领域中的一系列技术革新成果中最有价值的部分，从而在一个体现从量变到质变的计算机技术发展过程中，标志了一个里程碑性质的新成果的出现。它代表了面向对象程序设计方法在目前的最高适用水平。新一代的计算机模式——网络计算模式，也应为得到 Java 语言的支持，从而大大地向前迈进了一步。Java 语言的确是计算机领域中一项非常有价值的新成果。

另一个让人感叹之处，是在当今计算机行业竞争如此激烈的时期，由一个计算机硬件公司研究出的 Java 语言，居然一下子得到了几乎全部世界上的各大计算机软、硬件公司的支持。Java 语言很可能成为计算机程序设计语言的标准语言，用它编写的程序可以运行在各种计算机上。如果真能如此，这将是计算机行业里破天荒的一件大事。

对于 Java 语言这样一个新颖而重要的题材，自然会有不少对此有兴趣人来学习。本书包含了 Java 语言程序设计的基本概念及语法说明，又有一些高级的应用。本书另外一点就是强调实用性，凡是重要的概念和知识点，都有适当的实例和文字解释补充，使读者加深了对概念和基础知识的了解，又掌握了 Java 语言的基本方法。此书是一本很好的 Java 语言入门学习书。

计算机技术发展非常快，加上作者水平有限，编写当中难免有所疏漏和不足，希望广大读者能给予批评和指正。

编者

2000 年 10 月

目　录

第一章 Java 语言简介

1.1 Java 的历史

在计算机技术快速发展的今天，Internet 网络成为现代信息高速公路的代名词。Java 伴随着 Internet 的流行发展起来，Java 为 Internet 提供了适应计算机网络环境的编程语言，其 Applet 应用程序是专门用于 WWW（World Wide Web）环境，Java 被誉为“长时间以来最卓越的语言”，Java 语言将成为新的开发环境标准。

Java 语言是由 Sun Microsystem 公司从 1991 年开始开发，它的早期产品是一种交互式操作软件，取名为 Oak。虽然在技术上是一个成功的系统，但在商业上遭到失败。此时正值 WWW 处在不断发展的新阶段，虽然 WWW 页面内容丰富，能够做到对声音、图像、文字并茂处理，但是它们是静态的，不能动态交互处理，大大限制了 WWW 的发展。WWW 需要一种具有动态嵌入式的编程语言，而具有动态交互处理能力的 Oak 软件正好能满足这一要求，它语言简洁、精炼，又有较强的安全性，能支持在各种处理器上运行 Java 编译器。Sun 公司正是看到越来越受欢迎的 WWW 更适合 Oak 语言开发的潜力，开始对 Oak 语言进行不断的改进和完善，以便适应 Internet 网络环境；另一方面用 Oak 语言开发编制一个真正的应用程序。在 1995 年，Oak 语言被改名为 Java 语言，而应用程序被作为 WWW 浏览器取名叫 HotJava。

1.2 Java 的特点

Java 语言是一种新型的编程设计语言，广泛地应用于 Internet 网络编程设计。它是跨平台的适用于分布计算机环境的面向对象程序设计语言。它具有简洁、安全、面向对象、动态、体系结构中立、可移植、高性能、多线程、解释执行和分布式等特性。

1.2.1 简洁性

Java 语言是一种面向对象语言，它为用户提供了最基本方法来完成某种任务。只要理解了这些相关的概念，就能够理解并编制出适合于实际情况的程序。

Java 语言简单有效，这与原设计的初衷有关。在所设计的系统中必须具有简洁性，一方面尽可能采用面向对象的编程特点；另一方面要使编程者尽量避免可能的麻烦和错误，

Java 的这一特性是以增加系统的复杂性为代价的。

1.2.2 安全性

Java 语言的一切对内存访问都是通过对象实例变量实现的，防止用户在网络系统或分布系统环境下使用特洛伊木马等手段访问对象的私有成员。Java 语言不支持指针，避免指针操作的错误，Java 语言提供了内存管理机制，有一个自动搜索“内存垃圾”的程序。Java 在字节码的传输过程中使用了公开密钥加密机制(PKC)，而在运行环境提供了四级安全性保障机制：

(1)字节码校验器(ByteCode Verifier)

(2)类装载器(Class Loader)

(3)运行时内存布局

(4)文件访问限制

1.2.3 面向对象

Java 语言具有真正的面向对象语言的特点，它支持静态和动态的代码继承及重用。所谓面向对象是一种计算机编程方法，它是将具有属性和行为的现实世界中的对象映射到计算机的编程上，属性表示数据，行为表示程序代码。

面向对象语言的仩何方面均是基于消息或对象的。而消息是传递支持对象间所有可能的互相作用，面向对象编程就是对对象和消息编程，它支持封装、多态性和继承在。封装就是将对象内的数据和代码联编起来，形成一个对象；多态性是指一个接口，有多个内在实现形式表示；继承是指某一对象直接使用另一对象所有属性和方法的过程，它可以简化对象的定义，即在定义某个对象时可以只定义有别于父对象的属性或行为，其他部分则可以从父对象那里继承来重用。

1.2.4 动态性

Java 语言的设计能够适应于发展变化的环境。由于 Java 语言编程的基本组成单元是类，在运行中 Java 的类是动态装载的，只要 Java 在分布式系统中动态地维护应用程序和支持类库间一致性，就可以避免像类库升级这一类问题。

Java 语言滞后联编机制充分利用面向对象编程风格的优点，真正做到即插即用的模块功能。

1.2.5 体系结构中立和可移植性

Java 语言编程的程序可以不经任何改动在不同的硬件或软件平台上执行，即 Java 编译器所生成的可执行代码不是基于任何硬件平台的，它是基于一种抽象的处理器——Java 虚

拟机(Java Virtual Machine)实现的。

Java 程序的运行，首先要经过编译，再进行解释这两个过程实现的。

Java 编译器所生成的代码不是针对某一具体的硬件体系结构的，而是一种叫做字节码指令代码，它与硬件体系结构无关。在运行过程中，则由针对于运行系统硬件体系结构的 Java 解释器，将字节码转换成与该系统相对应的指令。

Java 的可移植性使人们可在网络上的各种硬件平台运行同一 Java 程序。

1.2.6 高性能性

虽然 Java 是解释执行语言，但它的字节码在编译生成时，带有许多的编译信息，在 Java 字节码格式设计中充分考虑到它与机器码的执行效率，很容易直接转换成对应于特定处理器的高性能机器码。

Java 字节码的执行效率非常接近 C 或 C++生成的机器码执行效率。

1.2.7 多线程

使用多线程，可以用不同的线程完成特定的行为，而不需要采用全局事件循环机制，这样利于实现网络上实时交互操作。

Java 多线程支持表现在两个方面，一方面是它自身的多线程，可以利用系统的空闲执行一些常规处理等；另一方面，提供对多线程的语言级支持，提高程序执行效率。

1.2.8 解释执行

Java 解释器直接对 Java 字节码进行解释执行。由于其所带的编译信息，使得执行连接过程更加简单。

1.2.9 分布式

Java 是一个适用于网络的语言。它提供的类库支持对 TCP/IP 协议处理，可以通过 URL 地址访问网络上其他的对象。

Java 支持 WWW 的客户机/服务器计算机网络模型，它可以支持分布式的数据分布和操作分布。它是一门非常适合 Internet 网络和分布式环境的编程语言。

1.3 Java Applet 和 Java Application 的介绍

Java 程序在使用中分为两种类型：一种是独立应用程序，称谓 Application，不依赖于任何浏览器而独立执行，它是通过调用类的 main()方法开始的；另一种是小应用程序，称

谓 Applet，用于嵌入到 WWW 页面，在支持 Java 的 WWW 浏览器上执行。

Java 语言的 main()方法类似 C 和 C++语言的 main()函数。执行 C 或者 C++程序时，系统首先调用它的 main()函数，然后再由 main()函数调用其他的函数。相类似，解释器在解释执行 Application 时，通过调用类的 main()方法，再由 main()方法调用其他的方法。所以，Java 程序中的每一个 Application 必须包含一个 main()方法。

main()方法的声明形式是：

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

其中：

public	表示 main()方法能被任何对象调用
static	表示 main()方法是一个类
void	表示 main()方法不返回值
String args[]	用于接收传递给 main()参数的 String 类型数组

Applet 则不需要一个 main()方法，因为 Applet 程序是嵌入 WWW 页面的 HTML 文件中。在 WWW 浏览器上加载 HTML 文件，就可以执行嵌入的 Applet 程序。Applet 是专门基于 WWW 环境的应用程序。

Java 应用程序 Application 和 Applet 的执行，都首先要经过 Java 编译器 Javac 对它们的源程序进行编译，生成字节码程序.class，然后，Application 的字节码程序由 Java 解释器 java 解释执行；而 Applet 字节码程序，在 HTML Script 中被调用，由客户机端解释执行。嵌入在 HTML 中的 Applet 使 WWW 增添了交互性和动态性。

1.3.1 Application 程序编写与运行步骤

要实现 application 程序从编写到运行目的，则需要按下列步骤进行：

1．首先创建一个 Java 的 Application 源程序(.java 文件)

为创建一个名为 HelloWorldApp.java 的文件，则可在任何字符编辑器上输入并保存下列 Java 源程序代码：

```
class HelloWorldApp{
    public static void main(String args[])
    {
        System.out.println("Hello,World!");
    }
}
```

其实质是创建一个名为 HelloWorldApp 类，并把它保存与它相同名字的文件中(即 HelloWorldApp.java 文件)。

2．第二步是对已创建好的 Java 源程序(.java)进行编码

该步骤是用 Java 编译器对 Java 源程序(.java)进行编译，生成对应的字节代码程序(.class)。如果编译成功，会得到一个有相同文件名的带.class 扩展名的字节代码文件。

例如，对上面例中的 HelloWorldApp.java 文件编译，对基于不同的运行平台，有如下命令格式：

在 UNIX 系统下

```
javac HelloWorldApp.java
```

在 Windows 95/NT 的 DOS 方式下

```
javac HelloWorldApp.java
```

如果编译中不出现错误，将会得到一个名为 HelloWorldApp.class 文件。编译选项在这里使用的是缺省方式。

3．最后就可以解释执行已编译成功的字节代码程序(.class)

用 Java 解释器 java 对 Java 字节代码程序(.class)解释执行。

在上例得到的 HelloWorldApp.clas 文件，现在可以用 java 解释执行了，对基于不同的运行平台，有如下命令格式：

在 UNIX 系统下

```
java HelloWorldApp
```

在 Windows 95/NT 的 DOS 方式下

```
jave HelloWorldApp
```

运行的结果，将会在标准输出设备上输出：

```
Hello,World!
```

Java 解释器在解释执行时，解释处理的是类名，而不是文件名，所以在解释器 java 后面跟随的是类名，而不能写成文件名的形式(HelloWorldApp.class)。其选项也使用缺省方式，这就是不依赖浏览器而独立运行的 Java Application 应用程序，从编写到运行的基本步骤，这里只作了简单介绍。

1.3.2 Applet 程序编写与运行步骤

Java 的 Applet 应用程序不同于 Application 应用程序，它是在支持 Java 浏览器上执行的，它的创建和使用操作步骤，有别于 Application 的操作步骤。

1．首先创建保存 HTML 页的目录

创建的这个目录是用来保存 HTML 页的，取名为 HTML。对基于不同的运行平台，有如下命令格式：

在 UNIX 系统下

```
mkdir    /HTML
cd       /HTML
```

在 Windows 95/NT 的 DOS 方式下

```
md   \HTML
cd   \HTML
```

如果已经有了存放 HTML 页的目录，就不需要这一步骤了。

2．第二步创建 Java 的 Applet 源程序(.java 文件)

Applet 源程序编写同 Application 一样，用字符编辑器输入相应的 Applet 的源程序。但它的程序中并不需一个 main()方法。

下面的例子是创建并保存在 HTML 目录中的一个 Applet 小应用程序，取文件名为 HelloWorldApp.java 的程序。

3．第三步是对已创建好的 Applet 源程序(.java)进行编译

用 Java 编译器对 Java 的 Applet 源程序(.java)进行编译生成对应的字节代码程序(.class)。如果编译成功，会得到一个有相同文件名的带.class 扩展名的字节代码文件。这与 Application 文件的编译是一样的。

对 HelloWorldApp.java 文件的编译，对基于不同的运行平台，有如下命令格式：

在 UNIX 系统下

```
javac HelloWorldApp.java
```

在 Windows 95/NT 的 DOS 方式下

```
javac HelloWorldApp.java
```

得到一个名为 HelloWorldApp.class 文件。编译选项在这里使用的是缺省方式。

4．第四步创建 Apple 嵌入到 HTML 文件中

把编好的 Applet 小应用程序，嵌入到 HTML 文件中，并保存在一个文件中。

下例是创建一个名为 HelloWorld.HTML 文件，并且将 Java 的 Applet 小应用程序 HelloWorldApp.class 嵌入进去。

5．最后加载 HTML 文件

在 HotJava 窗口或支持 Java 的 WWW 浏览器上的 URL 处，按下例格式输入新的 HTML 文件：

```
file://HTML/HelloWorld.html
```

在标准输出设备上显示出：

```
Here is the output of my program          Hello,World!
```

从上面的简单的介绍，可以建立对 Java 的 Application 和 Applet 操作概念。为后面章节的学习打下了基础。

1.4 Java 虚拟机原理

Java 虚拟机（JVM ： Java Virtual Machine ）是虚拟运行 Java 代码的假想计算机。Java 编译程序是将 Java 的源程序编译成 JVM 可执行代码，即字节代码（byte-code），而不是像 C 或 C++编译器，生成直接能运行于某种特定硬件平台的可执行代码。后者，在编译过程中就确定了内存分配情况；而前者，是由解释器在运行过程中创立内存布局的，这样，更加有效地保证了 Java 的可移植性和安全性。

1.4.1 Java 生成可执行代码的过程

在以往许多程序设计语言是通过对源程序编译目标文件，在经连接处理直接产生可执行的代码程序。而 Java 程序则不同，它首先经过 Java 的编译器对 Java 的源程序编译，生成的是不同的目标代码程序的字节代码程序，然后，再由 Java 的编译器来运行这个字节代码程序。

解释器在 Java 字节代码运行中,分三个阶段：

（1）代码的装入，是由类装载器完成；

（2）代码的校验，发现各种可能出现的错误；

（3）代码的运行，在代码校验后就可以执行了。

代码的运行有两种执行方式：

（1）即时编译方式：由 Java 先将字节代码转换成机器码，而直接运行该机器码；

（2）解释执行方式：由 Java 通过翻译并执行一小段代码，直至完成。

在具有 Java 解释器的特定平台上，只要根据 JVM 的规格描述，就能够在该系统上运行。就两种执行方式而言，解释执行的速度较慢，而采用编译方式，速度将基本上于 C 语言相当。目前，Sun 公司还未推出 Java 字节代码编译器。

1.4.2 Java 虚拟机规范

Java 虚拟机规范是保证 Java 代码在任何系统上都能运行的前提。它包括：

1．Java 虚拟机体系结构直接支持 Java 语言的基本数据类型：byte、short、long、int、boolean、float、double 和 char，它们的定义是不针对某个具体的平台的。

2．Java 虚拟机规范不对 Object 的内部结构有特殊的要求。

3．Java 虚拟机规范只有面向堆栈的体系结构，只有少量的寄存器。

4．Java 虚拟机规范局部变量和操作数都采用 32 位的，long 和 double 类型则需要 64 位。

5．Java 虚拟机规范还包括有运行环境、无用单元收集堆、方法区、指令集和限制等

方面。

6. Java 语言的源程序储存在以.Java 为扩展名的文件中，Java 虚拟机的字节代码程序储存在以.class 为扩展名的类文件中。

7. Java 虚拟机的指令由操作码和操作数组成。操作码是 8 位的，目前在使用的约 200 条。操作数数目由指令决定。除跳转指令为 4 个字节外，其他指令均为 2 字节。

采用 Java 虚拟机，对 Java 平台独立性和安全性是有很大的意义的。Java 虚拟机体系是非常开放的，能够为编程者提供解决 Java 编程中的问题和其他方法。可以让编程者根据自身的爱好来制作一个扩展库。甚至可以开发附加的语句，表达出所需要解决的问题，控制所希望的应用程序的性能。正是 Java 虚拟机体系结构的这些特性，使之能够简单、高效的编程，且又能用于所有的计算机、所有的 Java 解释器、所有的 Java 应用程序。

第二章 Java 语言基础

Java 语言正如前面所描述的是一种面向对象程序设计语言。从本章开始我们将逐一介绍它的各个方面，首先它是一门编程语言，具有编程语言的最基本特点，像关键字、标识符、基本数据类型、操作符与表达式，以及流程控制语句等。

Java 语言编程在很多方面类似 C 或 C++语言，其语言基础部分内容更接近 C 或 C++。对于熟悉 C 或 C++语言的编程人员，就容易理解和编写 Java 语言程序了。在本章中，通过列举一些编程的例子，以期更加形象地帮助读者理解这些概念。我们所列举的例子，将使用 Java 的独立应用程序 Application 进行编程，并用 Java 解释器 java 解释执行，而不采用 Applet 编程。这是基于 Application 进行编程的运行环境，要比需要浏览器环境的 Applet 对系统开销需求更小些而考虑的。事实上，Java 语言的这两种应用程序在概念上是一致的。

在本章所使用的程序实例，将会是简单、易懂并具备一定功能的程序。它们能够体现 Java 语言的纯面向对象性，一个 Java 程序就是一个类的概念；每个程序实例中都有结果输出模块(例如 System.out 方法实现)或数据输入模块(例如 System.in 方法实现)。力求其免使用未定义的名词或概念。当然在学习过程中，仍然可能遇到一些难理解的内容，尤其是有关 Java 类的概念。

2.1 Java 程序的基本结构

Java 语言的源程序是一个或多个以.java 为扩展名的文件，这些文件就是 Java 编译器 java c 的编译单元。而每个单元又由 package 语句、import 语句、类声明或接口 interface 声明语句构成。

1．包(package)是类和接口的集合，即为类库。Java 语言使用类库来管理类，这样能够方便管理，减少类名间的竞争。Java 的类都包含在类库中，package 语句可用来指定类所属的类库。

2．import 语句类似 C 或 C++语言中的包含语句——include 语句，它为程序装载类或包，使程序能够使用 Java 环境下的其他类。

3．接口(interface)声明语句是用来声明接口的各种属性的。类(class)声明语句是用来声明类的名字及相关属性等内容。

例如，有如下代码，并存入文件 ClassName.java 中：

```
package Name_of_Package;
```

```
import OtherClassName;
class ClassName
{
    public static void main(String args[])
    {
        ...
    }
}
```

其中，Name_of_Package 表示包名，OtherClassName 表示某类名，ClassName 表示正在创建的类名。

这里 package 语句将正在创建的类 ClassName 放到包 Name_of_Package(即类库)中；

import 语句装载了一个名为 OtherClassName 的类；

class 声明语句则声明了一个类名为 ClassName 的类及其相关属性等内容；

public static void main(String args[])是 ClassName 类的 main()方法的标识，其中 main()方法的方法标记有三个修饰符，其含义分别为：

- public　表示 main()方法能被任何对象调用
- static　表示 main()方法是一个类方法
- void　表示 main()方法不返回任何值

另外，main()方法的参数 args[]是一个 String 类型的数组。…是有关 main()方法的方法体内容，留在后面的章节中讨论。

2.2 Java 符号集

符号是构成程序的基本单位，不同的语言所采用的符号标准有所不同，在 C 和 C++等一些语言中，一般都采用 ASCII 码，而 Java 则采用的是 Unicode 字符集，又称统一码字符集，它可以支持多种语言。

由 Unicode 字符集中的字符，按照一定的使用规则，就可以构成 Java 的符号，在 Java 中符号分为五种类型：

关键字(KeyWords)

标识符(Identifiers)

常量(Literals)

运算符(Operands)

分隔符(Separator)