第一章 Java 简介

Java 语言是美国 Sun 公司于 1995 年 5 月推出的新一代面向对象的计算机通用编程语言，是继 World Wide Web 之后又一个轰动世界的产品，其影响甚至超过了同年宣布的 Windows95。Java 除了能像 C、C++ 等高级语言一样进行用途广泛的编程外，还能对 Internet 和 World Wide Web 编程，仅这一点就超越了其他编程语言。

本章将介绍 Java 的产生背景、Java 的特点和 Java 应用程序与小应用程序的区别。

§1.1 Java 的产生背景

Java 和 World Wide Web 的研制几乎都在 90 年代初开始。最初互不相识，在大西洋两岸彼此独立地开发，最后却融合到了一起，成为 Internet 上最受欢迎的技术。

World Wide Web（WWW）产生于地处瑞士日内瓦的欧洲基本粒子物理实验室（CERN）. 1989 年 3 月，当时在 CERN 工作的英国年轻科学家 Tim Berners - Lee 提出了 WWW 的概念。1990 年 12 月，第一个 WWW 软件在 Next 计算机上实现，这是一个文本模式的浏览程序。1992 年，CERN 正式发表了 WWW。

再看大西洋的另一边。

1991 年 4 月，美国 Sun 公司成立了“Green”小组，研究开发商业电子产品，需要一种独立于平台的开发环境。开发者之一 James Gosling 发觉当时最流行的 C++ 也很难满足这种要求，他决定为 Green 项目研制一种新语言。由于他看见工作室窗外有棵橡树，所以他把新语言称为 Oak。

最初研制出来的 Oak 语言应用于电话装置等电子产品中，然而由于商业上的种种原因，他们的产品并没有能投入市场，Oak 几乎前功尽弃。

此时 WWW 却越发展越壮大。1993 年，美国 Illinois 大学的 National Center for Supercomputing Applications（NCSA）为 WWW 研制了图形接口 Mosaic，使原来较深奥的 WWW 变为大众容易理解的东西，大大普及了 Web 在 Internet 中的使用。

WWW 的影响使 Sun 的开发者 Naughton 和 Jonathan Payne 意识到 Oak 和 WWW 的潜力，他们在 1994 年 9 月完成了后来称为 HotJava 的 WebRunner。由于用 Oak 作为商标没有成功，研究小组反复讨论之后，最后将 Oak 改名为 Java。Java 这个名字来自印度尼西亚爪哇岛上种植的一种同名咖啡，据说 Sun 公司 Java 语言的开发者们喜欢喝这种咖啡，因此将他们的产品命名

为 Java和 HotJava。不久，Arthur Van Hoff 用 Java 本身实现了 Java 编译器，表明 Java 已经是一个完整的语言了。

1995 年 5 月 23 日，Sun 公司在美国圣·佛朗西斯召开的 SUN WORLDS'95 上正式向全世界宣布了 Java 和 HotJava。Netscape 公司当即宣布支持 Java，要在其浏览器 Navigator 中采用 Java 技术，结果改变了 Web 页面静态不变的现象，也使得 Netscape Navigator成为最受欢迎的 WWW 浏览器。

如果把 WWW 称为 Internet 的第一次革命，则 Java 可称为 Internet 的第二次革命。Java 经过 4 年的开发，终于进入 Internet 的主流。Tim Berners - Lee 预言，计算机事业发展的下一个浪潮就是 Java。

§ 1.2 Java 的特点

Java语言一经正式发表就立刻受到欢迎，除了它适应形势发展的趋向之外，必然还有其优秀的特点。Java 语言的特点主要体现在简单（simple）、面向对象（object - oriented）、分布式（distributed）、解释性（interpreted）、强壮（robust）、安全（secure）、结构中立（architecture - neutral）、可移植（portable）、高性能（high - performance）、多线程（multithreaded）、动态（dynamic）和通用（general - purpose）等。

1. 简单

由于设计者的周密考虑，使得 Java 语言简单有效，使用者可以很容易地进行程序设计而不需要接受严格的训练。表现在：① Java的语法格式与 C + +相似。C + +应用很广，大多数程序员都熟悉 C + +，所以让 Java 程序的外观看起来很像 C + +的风格，这样大家自然容易接受 ② Java 中去掉了 C + + 中应用不方便以及设计不周全的部分，如指针（pointer）和内存管理（memory management）等；③Java 的程序规模较小，其基本解释程序和类支持功能只有 40K 字节左右，很容易在小机器上运行。Java 的类库提供的基本功能使编程人员能快速有效地开发应用程序，从而简化开发过程。

2. 面向对象

Java最显著的特点是它是真正面向对象的语言。C + +是在 C 的基础上改造成面向对象的，而 Java 从开始设计时就用面向对象技术来设计，它的封装性（Encapsulation）、继承性（Inheritance）和多态性（Polimorphism）等都很好。还实现了 C + + 无法实现的动态联编（Dynamic binding），使得 Java 具有代码重用（Reusability of code）、扩展性（Extensibility）和动态应用（Dynamic application）等优点。

在 Java 中，面向对象编程的基本元素是对象（object），它把变量（variable）和方法（method）从功能上封装成一个可重用、可动态装载的整体。类（class）是对象的集合，同一类的对象有同样的操作方法和性质。一个类建立后，只要增加一些功能就可以产生新类，这就是封装性、继承性等性质的含义。

3.分布式

Java语言是很适合分布式应用的。通过类库（class library)提供的方法，Java可方便地适应各种TCP/IP协议(Transmission Control Protocol/Internet Protocol,这是Internet的通用网络协议），如HTTP(HyperText Transfer Protocol,用于WWW)和FTP(File Transfer Protocol,用于Internet传送文件）等。并可通过URL(Universal Resouce Locator)存取网络对象,比如你可以让一个Java程序同时运行取自两个不同地方的声音文件。

4.解释性

Java是解释执行的。但它与Basic语言的解释不同,Basic没有编译阶段,语句输入后,一边检查错误一边运行，效率很低,Java则是经过编译后产生字节码(Bytecode),然后由各种支持Java解释器(interpreter）的机器来解释执行字节码。由于Java的字节码比较接近汇编指令，所以可以很快解释为实际机器指令，这种做法比Basic的解释效率高得多。Java编译后直接解释运行,比C++少了一个连接阶段，缩短了开发周期。

5.强壮

Java的强壮性保证代码有好的表现，即稳定可靠地运行。因为Java的发明者发现了C++的不稳定之处:C++用指针直接访问存储器位置，而且没有自动内存管理，即使程序员写出句法和语义都正确的程序，也会由于访问内存出错等原因而造成系统崩溃，因此，Java用消除指针和提供自动内存管理来保证强壮的运行环境。

消除指针后，程序员就不能用指针算法在内存中任意移动，可防止对操作系统的内存空间进行误写。另外，Java增加了数组边界检查功能，这样一来，程序就不能访问数组没有分配的寻址空间。

Java的自动内存管理功能一方面减轻了程序员注意内存情况的费神劳动，另一方面提高了内存使用效率，同时还防止人为干预内存而造成破坏。

此外,Java在编译和运行程序时都对程序进行检查，防止错误隐藏。Java的编译检查比C++严格,它有异常(exception)处理机制，在编译时会提示可能出现但未采取措施处理的异常情况，提醒程序员采取措施。

6.安全

Java是应用于网络或分布环境下的语言，其安全性是极其重要的。因为Java代码可以在网络上发布和传递，若带有病毒，就会危害执行这些代码的所有客户(client)的机器,因此Java的设计者充分考虑了安全问题，提供了多道关卡防止病毒的入侵。

从Java的强壮性我们知道它取消了指针，并对数组的下标进行检查，这样就防止非法改写系统内存。另外,对于与系统资源有关的对象,可以用Java语言的封装性来保护，这一道关卡我们称为语言关，它防止从编程上破坏系统。

在网络上传送的Java代码是编译后的字节码，会不会在传送过程中受病毒感染呢？这点语言关控制不了。因此，Java的设计者在编译后运行前增加了一次字节码检验，不管是从哪

里得到的字节码，在它进入解释器之前，先由字节码检验器(bytecode verifier)检查它的安全性，这一道关我们称为字节码关。

到了真正运行的时候，Java 解释器决定各个类的存储器安排（类是 Java 的基本执行单元),这样,网络黑客(hacker)不可能了解某个类在内存的什么地方从而非法访问。类装载器(class loader ）把从网络下载的各个类放入各自的存储空间，防止它们访问文件系统。这一道关我们称为解释关。

所有这些措施,使得 Java 对任何系统都很安全，完全胜任网络上的应用。

7.结构中立

Java 的结构中立性是要达到软件界一直追求的一个目标，即平台无关性，这也是当初开发 Java 的原因之一。我们知道，在一台计算机的不同操作系统上开发的应用程序是不同的，有时即使是用同一种高级语言编写的程序，在不同的机器上编译产生的代码也不一样。这就是平台有关性。

Java 如何做到结构中立呢？就是编译后产生一种与平台无关的字节码，称为 Java 虚拟机(Virtual Machine ）的指令代码。这种代码不能直接运行，但可以传送，由网络送到各种不同的计算机上,由该机上支持的 Java 解释器(也就是 Java 虚拟机的仿真运行程序）来执行，真正实现了一机编译，多机执行。这样看来，Java 的字节码很像一种计算机界的‘ 世界语 ”,可在 Internet 上到处传送,然后由各地懂‘ 世界语 ’的‘ 翻译 ’翻成本地语言,让大家‘ 听懂 ”,所以 Java 近乎完美地解决了平台无关性。

Java 的这种结构中立性不仅对网络十分有用，对软件开发也有很大意义。

8.可移植

除了采用虚拟机器代码的形式使 Java 语言程序可以在不同机器上运行外，Java 语言还采用其他措施提高可移植性。

C 语言随着机器硬件和操作系统的变化而变化。比如其整数（ integer),在不同机器上位数不同 在 DEC Alpha 机上是 64 位,在 Intel 486 上是 32 位；在同一机器上操作系统不同时位数也不同 在 Windows 3.1 上取 16 位,在 Windows95 上取 32 位。这些变化使 C 语言程序的移植性大为减少。

Java 语言则规定数据类型只有一种标准，不依赖于机器，避免了上述不可移植性。各种类库也规定了可移植的界面,如 Window 类就能适应 Unix 、Windows 和 Macintosh 环境。

不但 Java 语言编写的应用程序可移植，连 Java 系统本身也具有可移植性。Java 的编译器是用 Java 语言实现的，运行系统用标准 C 实现，它们都有很好的移植性。

9 高性能

由于兼顾了可移植性、安全性、强壮性、结构中立等特点,其代价是 Java 的性能有可能降低，特别是解释执行字节码的速度显然比不上 C + + 执行机器码的速度。因此， Java 的设计者采取一些技术来保证高性能。

其一是内建多线程，能提高 Java 程序的性能。下一点会详细解释它。

其二是有效的字节码，编译后的字节码很接近机器码，可以在任何具体平台上有效地解释它。

其三是在运行期间将字节码译成当地机器码，不过要花一定的延迟时间才能运行。

其四是链接到本地的 C 语言代码，这样效率很高，但可能失去移植性。

10. 多线程

所谓多线程是指在一个程序中可以同时执行一个以上的线程（thread），也就是通常所说的并行执行多个任务。

线程与进程(process）相似，也是执行中的程序，但线程数据较少，多个线程共享一组系统资源。系统处理线程的负荷要比处理进程小。

多线程的优点是可以合理调配多个任务，交互式响应性能较好，并有实时特性。比如打印任务很花时间，如果程序是单线程的，那么不打印完就不能做其他事情。采用多线程方法后，一个线程负责打印，另一线程继续干其他事，就不会浪费时间了。

实现多线程比单线程难，所以 C++ 都没能实现多线程。Java 则加入了多线程功能，它完成了其他语言难以实现或实现得不好的数据同步化（synchronization）过程，避免了资源冲突，这是 Java 的又一个突出的优点。

11. 动态

Java 的动态性是其面向对象设计的扩展。它提供运行时刻的扩展性，即在后期才建立各模块间的互连，各个库可以自由地增加新的方法和实例（instance）变量，这意味着现有的应用程序可以增加功能，只须链接新类封装有的所需的方法。

C++ 是多重继承(multiple inheritance)的，若某个超类(superclass）改变了某个方法或变量，其子类(subclass）必须重新编译。Java 则用接口(interface）来实现多级继承，使用起来比 C++ 的多重继承更灵活。

Java 语言的动态性使它能够胜任分布式系统环境下的应用，位于各地的类可以自由地升级，而不影响原 Java 应用程序的运行。

12. 通用

Java 语言是通用的编程语言，不像 Lisp、Prolog 等语言只适用于某一领域，这一点它与 C++ 类似。Java 语言利用应用编程接口 API（Application Programming Interface）扩充其编程方面的范围。比如图形处理和多媒体 API、网络 API 和数据库接口 JDBC（Java DataBase Connectivity）API 等，使 Java 可以应用于网络、多媒体、数据库等领域。SUN 公司还不断推出新的 API 系列，进一步拓展 Java 的应用领域。

以上简略介绍了 Java 语言的特点，在后面各章中还会对某些特点作更详细的介绍。

§1.3 Java 应用程序和小应用程序

Java 语言可以开发出两种运行方式不同的程序：一种称为独立（stand-alone）的应用程序(application)。用它可以编制较大型的 Java 程序。它常用于不联网的情况，故称为独立。它的用法与 C++等编程语言基本一样，只是编译后要用 Java 解释器来运行；另一种称为小应用程序(applet)，它编译后必须嵌入 HTML(Hyper Text Markup Language)文件中，由浏览 HTML 文件的网络浏览器来执行，而且该浏览器要内嵌 Java 解释器才能显示小应用程序所产生的结果。这种方式专门用于对 Internet 和 WWW 的编程，也是 Java 语言轰动世界之处。

下面我们用简单的例子说明这两种不同方式的 Java 程序的区别。

1. Java 应用程序

Java 应用程序由类组成，其中有一个类必须包含一个 main()方法，让解释器从此开始执行。我们把含有 main()方法的类称为主类，其他类则不能有 main()方法。

首先用普通的文本编辑器编写下列程序：

例 1-1

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

这是一个非常简单的应用程序，只有一个类。第一行是类的声明，类名 HelloWorldApp。第二行声明类中的方法 main()，关键字 public 表明 main()方法是公用的，可以被任何对象调用；static 表明 main()方法是类方法，而不是类的某个实例；void 表明 main()方法不返回任何值。String 是 Java 用于表示字符串的类，args[]是 String 类的一个对象，向程序传递命令行参数。第三行将 Hello,World ！这串字符送到标准输出（通常就是屏幕），System 类调用 out 方法，out 又调用 println 方法，都是用点操作(.)调用的，这些会在后面的章节详细介绍。

编辑好后存入一个扩展名为 .java 的文件中，记住其文件名要与主类名一致。本例的文件名应为 HelloWorldApp.java，字母大小写也不能错。

然后用 javac 编译器编译这个文件：

```
C:\>javac HelloWorldApp.java
```

编译成功后会生成一个 HelloWorldApp.class 文件，这就是所谓的字节码。现在可以用 java 解释器运行这个文件，后缀 class 可以不键入。

```
C:\>java HelloWorldApp
```

运行结果将出现：

```
Hello,World!
```

这就是开发一个应用程序的完整过程，所用到的编译器 javac 和解释器 java 都在 Java 开发工具 JDK 中提供，详见本书附录的介绍。

2. 应用程序的命令行参数

Java语言支持命令行参数(command line argument) ，在用解释器执行应用程序时在命令行输入一些参数给程序，使用户可以影响应用程序的操作。

Java应用程序可以接受任何数量的命令行参数，它们跟在程序名后，用空格隔开，运行系统用一个 Strings数组把命令行参数送到程序的 main()方法。如：

例 1 - 2

```
class Argu {
    public static void main(String args[]) {
        for(int i = 0; i < args.length; i + + )
            System.out.println(args[i]);
    }
}
```

本程序把输入的命令行参数每行打印一个， args.length 是参数个数。如果不输入参数，则没有任何输出。例如：

C: \>java Argu

无输出。

C: \>java Argu 1 2 3

输出结果：

```
1
2
3
```

注意 Java 与 C语言的区别，C 语言把程序名作为第一个参数，而 Java 把程序名后第一个字符串作为第一个参数。上例的 args[0] = 1, args[1] = 2, args[2] = 3。

C: \>java Argu I am a student.

将分四行输出：

```
student.
```

若希望在一行输出，则可以用双引号括起来：

C: \>java Argu "I am a student."

输出：I am a student.

3. Java小应用程序

Java 小应用程序也是由类组成，其中有一个类必须是由 java.applet.Applet 类派生的子类，

由 applet 浏览器调用执行。我们把这个类称为小应用程序的主类，其他类则不是 Applet 类派生的子类。

首先用普通的文本编辑器编写下列程序：

例 1－3

```
import java.awt.Graphics;
public class HelloWorld extends java.applet.Applet {
    public void init() {
      resize(150,25);
    }
    public void paint(Graphics g) {
      g.drawString("Hello,World!",50,25);
    }
}
```

这个小应用程序比刚才的应用程序稍微复杂一点，它有一个类，类中有两个方法。第一行的 import 语句引入 Java 类库中已生成的类，为本程序所用。java.awt.Graphics 是用于完成图形方式的输出功能的类，paint()方法需要这类对象。第二行声明一个类名 HelloWorld，public 表明该类是公用的，extends java.applet.Applet 表明 HelloWorld 是从标准类 Applet 派生出来的一个子类，这样它就可以继承 Applet 的所有特性和方法。第三行引用 Applet 类的 init()方法，该方法公用，且无返回值。第四行告诉浏览器将显示区域重新定义为 150×25，然后再引用 Applet类的 Paint() 方法。方法中的参数声明 Graphics 类的一个对象 g，下一行则由 g 调用 Graphics 类的方法 drawString()，把方法中的参数的字符串部分 Hello, World！在显示区的 x=50,y=25 位置处显示出来。

同样，编辑好的文件以扩展名 .java 存放，文件名要与主类名一样，用 HelloWorld.java 保存。

用 javac 编译器编译这个文件：

```
C:\>javac HelloWorld.java
```

同样会产生一个字节码文件 HelloWorld.class，但这个文件不能用 java 解释器执行，也不能直接由浏览器运行，需要编写一个 HTML 文件将这个 class 文件嵌入其中。

用文本编辑器编写一个名为 HelloWorld.html 的文件如下：

例 1－4

```
<HTML>
<HEAD>
<TITLE>A Simple Program</TITLE>
</HEAD>
<BODY>
<APPLET CODE = "HelloWorld.class" WIDTH = 150 HEIGHT = 25>
</APPLET>
</BODY>
```

〈/HTML〉

该文件的第一行〈HTML〉与第九行〈/HTML〉标志 HTML 文本的开始与结束。第二行〈HEAD〉与第四行〈/HEAD〉标志 HTML 文本的标题部分，标题 A Simple Program 由〈TITLE〉和〈/TITLE〉包含。第五行〈BODY〉与第八行〈/BODY〉标志 HTML 文本的正文部分，这部分就是嵌入 HelloWorld.class 的地方。由〈APPLET CODE = 开始，〈/APPLET〉结束，其参数 WIDTH = 150 HEIGHT = 25 定义显示区域为 150 × 25。

这个 HTML 文件比较正规，简单的只要：

〈APPLET CODE = "HelloWorld.class" WIDTH = 150 HEIGHT = 25〉〈/APPLET〉

一句就行了。若要内嵌其他小应用程序，只需在 APPLET CODE = 后换上其他 .class 文件，再调整 WIDTH 和 HEIGHT 参数值就行了。

编辑好的 HTML 文件可以用支持 Java 的 WWW 浏览器或 Java 开发工具 JDK 中的 appletviewer 来调用它。我们以后者为例：

C: \〉appletviewer HelloWorld.html

运行后会在屏幕上出现一个 applet 小窗口，显示 Hello, World!和 applet started 等信息，表明这个小应用程序已执行，并输出正确结果。

4. 小应用程序的其他事项

前面我们提到，用 HTML 文件包装小应用程序主类的 .class 文件的语句是：

〈APPLET CODE = AppletSubclass.class WIDTH = anInt HEIGHT = anInt〉〈/APPLET〉

这种格式意味着小应用程序的主类与 HTML 文件存放在相同的目录中。若两者位于不同目录，要加一个 CODEBASE 属性，表明小应用程序主类的位置：

〈APPLET CODE = AppletSubclass.class CODEBASE = aURL WIDTH = anInt HEIGHT = anInt〉

〈/APPLET〉

aURL 是 AppletSubclass.class 的 URL 地址，若是绝对 URL，意味着小应用程序在其他 HTTP 服务器；若是相对 URL，意味着小应用程序在 HTML 文件位置附近的其他目录中。有关 URL 地址的详细内容请看本书第七章。

小应用程序也可以输入用户参数（与应用程序的命令行参数一样），是在 HTML 文件中通过〈PARAM〉标签引入的：

〈APPLET CODE = AppletSubclass.class CODEBASE = aURL WIDTH = anInt HEIGHT = anInt〉

〈PARAM NAME = parameter1Name VALUE = aValue〉

〈PARAM NAME = parameter2Name VALUE = anotherValue〉

〈/APPLET〉

上面两个 VALUE 值就是对两个 NAME 指定的参数赋值，即用户可以用这种方式输入参数给小应用程序。

由于小应用程序可以在 Internet 和 WWW 上运行，为了网络安全，从网络下载的小应用程序有下列限制：

①不能装载函数库或定义 native 方法；

② 不能在执行它的主机中读写文件；

③不能进行网络连接，除非是连接它所在的主机；

④不能在执行它的主机中启动任何程序；

⑤不能读任何系统属性；

⑥ 小应用程序的窗口会带有警告性信息，提醒用户这不是可靠的应用程序的窗口。

对于从本地文件系统下载的小应用程序则没有上述限制。有时用 WWW 浏览器调用小应用程序有某种限制，而用 Appletviewer 调用它则没有限制。这是因为浏览器有个安全管理器(Sercurity Manager)对象，检查小应用程序是否遵从安全限制。

现在，可以看出 Java 应用程序与小应用程序的不同了。前者的程序中一定有一个 main( ) 方法，后者则有一个 Applet 类的子类。前者编译后直接用 java 解释器运行，后者要嵌入 HTML 文件再由 WWW 浏览器或 appletviewer 执行。前者主要单独运行，后者则主要在 WWW 页面内运行。但是，应用程序和小应用程序的本质是一样的。

通过本章的介绍，你对 Java 语言的特点应该有所了解。以下各章将讲述 Java 语言的结构和应用，同时会把 Java 语言的特点充分描述透彻。

习题一

1. Java 语言是何时何地产生的？
2. Java 语言有什么特点？
3. Java 语言为何简单？
4. Java 的解释性与 Basic 的解释性有何不同？
5. Java 语言采用哪几道关卡保证安全性？
6. 什么是 Java 的字节码，它如何运行？
7. Java 语言如何保证高性能？
8. 什么是 Java 应用程序？什么是 Java 小应用程序？
9. Java 程序如何取名？
10. main( )方法出现在哪种 Java 程序？
11. Java编译器产生什么文件？
12. 应用程序如何接收命令行参数？
13. 哪种 Java 程序必须有 Applet 类派生的子类？
14. HTML 文件如何内嵌 .class 文件？
15. 如何执行包含小应用程序的 HTML 文件？
16. 如何向小应用程序输入参数？
17. 从网络下载的小应用程序有什么限制？

第二章　语言基础

Java语言的设计者采用 C 和 C++ 语言的语法格式来设计 Java 语言，目的是减轻大多数编程人员重新学习新语言的劳动。但 Java 也并非全盘沿袭 C 或 C++ 的特点，其有所割舍亦有所添新。比如，Java 取消了结构(structure)和联合(union)、宏替换(#define)、指针、多重继承、单独的函数(function)、goto 语句、操作符重载(operator over loading)、自动强制转型(automatic coercion)等，而采用接口代替头文件(header file)、常量(constants)代替宏替换、类取代结构和联合、函数封装到类中、用接口实现多重继承、必须用明显的语句说明类型转换等。由此看来，不管你原来有没有 C 语言的基础，都要仔细了解 Java 语言的语法规范。

本章将对 Java 的数据类型、表达式和控制流程等逐一进行详细介绍，这些内容是各种编程语言的基础，必须首先了解清楚。

§ 2.1　数据类型

数据类型是在高级语言中广泛使用的一个概念，表明这种语言中有什么数据结构，是构成编程语言极为重要的部分。

Java 语言的数据类型如下：

- 数据类型
  - 基本类型
    - 整型：byte(8位)，short(16 位)，int(32 位)，long(64 位)
    - 浮点型：float(32 位)，double(64 位)
    - 字符型：char(16 位)
    - 布尔型：boolean
  - 复合类型
    - 数组类型
    - 类／接口类型

基本(primitive)类型是最简单的类型，不能由别的类型来构造。复合(complex)类型则是由基本类型构造而成的。本章介绍基本类型和数组类型，而把类与接口放到下一章。

在介绍基本类型之前先要谈谈 Java 的标记(token)集，也称为 Java 的词法，这是 Java 编译基本单元。注意 Java 语言的代码从 8 位 ASCII 码扩充到 16 位 Unicode，以支持非拉丁语言的字符。

1. Java 的词法

Java 的标记共有 5 种 标识符(identifier)、关键字(keywork)、字面量(literal)、运算符(operator)和分隔符(separator)。Java 程序就是用这 5 种词再加上不属于标记集的注释(comment)来编写的。

1)标识符

标识符是赋给变量、类和方法的词,如我们前面见到过的 HelloWorld、main 等。Java 的标识符必须是由字母、下划线'_'及美元符号'$'开头的字符数字串组成。字符数字串的数字是 0~9,字母包括大小写 A~Z 以及编码小于十六进制数 00C0 的 Unicode 字符。

以下是合法的标识符:

x, y, HelloWord, text_1 等 (建议不用'$',不用'_'开头,不用全部大写字母)。

以下则是不合法的标识符:

5days, book-2, room# 等(用关键字做标识符也不合法)。

2)关键字

关键字是 Java 语言本身使用的特殊词,不能作为标识符赋给变量、类和方法。下表列出了 Java 使用的和保留未用(带 * 号者)的关键字。

表 2-1 Java 语言关键字

abstract	const *	finally	interface	return	throws
boolean	continue	float	long	short	transient
break	default	for	native	static	true
byte	do	goto *	new	super	try
byvalue *	double	if	null	switch	void
case	else	implements	package	synchronized	volatile
catch	extends	import	private	this	while
char	false	instanceof	protected	threadsafe	
class	final	int	public	throw	

3)字面量

字面量表示 Java语言中数与字符的明显值,即常量。由于从字面即可判定它们是哪一类的常量,所以称为字面量,共有 5 种,分别为整数字面量、浮点数字面量、布尔字面量、字符字面量和串字面量。

整数字面量通常有十进制、十六进制和八进制格式。十进制就是常见的 63、100、82986 等;十六进制以 0x(或 0X)开头,如 0x10、0x5AF7 等;八进制以 0 开头,如 02,077 等。整数字面量通常表示 32 位二进制数范围内的数,若上述字面量后加字母 l 或 L,则表示 64 位。

浮点数字面量用于表示带小数部分的十进制数。如:

4.. .3, 70.16, 5.29E8, 1.7e-19 (后两个是科学记数法)

浮点数的单精度(single precision)占 32 位存储空间,用字母 f 或 F 跟在数后;双精度(double precision)占 64 位存储空间,用 d 或 D 跟在数后。如:

5.29E8F,1.7e-19D

布尔字面量有 true 和 false 两种,代表逻辑真和假,不能用数字 1 和 0 来代替。这是 Java 与 C 的又一区别。

字符字面量用单引号' '对括起,可以取 Unicode 的任何单个字符,如'a','B'等。再用反斜线序列支持转义字符,如'\n'等。常用的反斜线序列如下表:

表 2-2 常用反斜线序列

序列	描述
\	继续下行
\n	换行
\t	横向跳格
\b	退格
\r	回车
\f	走纸
\\	反斜线
\'	单引号
\"	双引号
\ddd	3 位八进制数
\xdd	2 位十六进制数
\udddd	Unicode 字符

串字面量是用双引号" "对括起来的任意数量的字符。在 Java 中,串是用 String 类实现的,每个串字面量都是 String 类的一个新实例。以下是一些串字面量:

""(空串)、"A"、"\","、"This is a class."、"One line\nTwo line" 等。

4)运算符

Java 语言用一些符号表示一些基本运算形式,这些符号叫运算符或操作符。下表按运算优先级从高到低列出所有运算符:

表 2-3 运算符

后缀(postfix)运算:	. [ ] ( ) expr++ expr--
一元(unary)运算:	++expr --expr +expr -expr !

建立(creation)或转型(cast): new (type)expr

乘(multiplicative): * / %

加(additive): + -

移位(shift): 《 》 》》

关系(relational): < > <= >= instanceof

相等(equality): == !=

按位与(bitwise AND): &

按位异或(bitwise exclusive OR): ^

按位或(bitwise inclusive OR): |

逻辑与(logical AND): &&

逻辑或(logical OR): ||

条件(conditional): ?:

赋值(assignment): = op=

5 分隔符

分隔符用于告诉编译器代码从哪里分开和组合,常用的有()、{}、;,等。

6注 释

Java有3种注释格式:

/* ... */用于多行注释。

// ... 用于单行注释,到行尾自动终止。

/** ... */用于自动文档产生器(javadoc)产生注释文档,它必须位于声明(declaration)部分之前。

2. 基本数据类型的变量与声明

变量用于表示Java在内存中存储的一个数据,它用带有类型的标识符表示,需要用声明语句来建立:

type identifier [, identifier];

该语句告诉编译器建立名为identifier、类型为type的一个变量。若声明多个同类的变量,则用逗号隔开,最后用分号结束。声明了变量之后就可以赋值给它,并可以操作。Java允许在声明变量时对变量赋初值:

type identifier = value;

下面分别介绍4种基本类型变量的声明。

1)整型变量

整型变量有 4种,分别是 8 位的 byte 类、16 位的 short 类、32 位的 int 类和 64 位的 long 类。全部是有符号数(其中 byte 的范围是 -256 到 255,即最高位也可以用于表示正数),其声明为:

```
byte op1;
short op2 = 2;
int op3;
long op4 = 4;
```

2)浮点型变量

浮点型变量有 float 型和 double 型两种,前者是 32 位单精度浮点数,后者是 64 位双精度浮点数,其声明为:

```
float op1;
double op2 = 0.5;
```

如果对 float 型数据赋初值,必须在初值后加个 f(或 F),如:

```
float op1 = 30.0f;
```

而在 double 型数据后可不加 d/D,因为 Java 自动把浮点数看成 double 型。

3)字符型变量

Java 的字符类型变量 char 是 16 位无符号整数,用于表示 16 位的 Unicode。字符变量只存储单个字符,其声明为:

```
char op1;
char op2 = 'A'. op3 = '\n';
```

4)布尔型变量

布尔型变量 boolean 可取逻辑真和假两种值,但不代表数字 1 或 0,其声明为:

```
boolean op1;
boolean op2 = false, op3 = true;
```

变量声明之后,它的作用范围(scope)就确定了,从变量声明的位置开始到它所在的代码块(block)结束。所谓块就是由花括号对{}包含的一段代码。举例如下:

```
class myApp {
  public static void main(String args[]) {
      int x;

  }
  public void mymethod() {
      char y;
      ...
  }
}
```

整数变量 x 的作用范围只能用于 main 的方法,不能用在 mymethod 方法中;而字符变量 y 只能用于 mymethod 中,不能用于 main 中。

如果在一个大范围内嵌套了一个小范围，而两个范围的某个变量同名，当程序进入小范围时，大范围的同名变量会被隐藏不用，直到退出小范围才恢复使用。因此，在定义变量名时要注意。

3. 数组类型

数组是 Java 的一种复合类型，是由同类型的对象组成的，这些对象可由索引（indexing）来引用。数组中的对象也可以是数组，即数组的嵌套，但并不像 C++ 那样称为多维数组。声明数组用方括号对[ ]加在标识符后(或类型后)。如：

```
int A[ ] 或 int[ ]A;
char B[ ][ ];
float C[ ][ ][ ];
```

可以用{ }赋初值，如：

```
int op1[ ] = {1,2,3} ;
char[ ]op2 = {'a','b','c','d'};
```

但这种赋值不能用于非声明的场合。

注意，声明数组时不能像 C++ 那样在方括号内加数字表示数组长度，而必须用 new 运算来分配，如：

```
int op3[ ][ ] = new int[5][3];
```

可以只声明长度 5，另一长度由程序来分配，如：

```
int op3[ ][ ] = new int[5][ ] ;
```

Java 语言会严格检查下标变量，如：

```
int a[ ] = new int[10];
```

则 a[5] = 1；和 a[1] = a[0] + a[2] 都是合法的，但 a[-1] = 4；和 a[10] = 2 则出错。

数组的长度可以用 .length 来取得，如：

```
int a[ ][ ] = new int[10][3];
println(a.length) 将会打印 10，而 println(a[0].length) 将会打印 3。
```

其中 length 是类 Array 的一个实例变量，所有的数组都是 Array 的子类，而 Array 是 Java 的最高类 Object 的直接子类。

下面是一个应用数组的例子。这个程序首先定义一个二维的 4 * 4 的 double 型矩阵 m，其中对角线上元素初始化为 1，其他的元素为 0。

例 2 - 1

```
class myArray {
   public static void main(String args[ ]) {
      double m[ ][ ];
      m = new double[4][4];
      m[0][0] = 1;
      m[1][1] = 1;
      m[2][2] = 1;
```

```
        m[3][3] = 1;
        System.out.println(m[0][0]+""+ m[0][1]+""+ m[0][2]+""+ m[0][3]);
        System.out.println(m[1][0]+""+ m[1][1]+""+ m[1][2]+""+ m[1][3]);
        System.out.println(m[2][0]+""+ m[2][1]+""+ m[2][2]+""+ m[2][3]);
        System.out.println(m[3][0]+""+ m[3][1]+""+ m[3][2]+""+ m[3][3]);
    }
}
```

程序运行结果为：

```
1 0 0 0
0 1 0 0
0 0 1 0
0 0 0 1
```

§2.2 表达式

Java 的表达式由标识符、关键字、字面量、运算符、分隔符以及变量等元素构成，对这些元素执行运算并返回某个值。表达式可用于对变量赋值，也可以作为程序控制的条件。

表达式进行的运算取决于构成的各种元素的类型，以及运算的优先顺序，按先高后低，先左后右的规则进行，加了括号()的部分则首先计算。

表达式的运算按运算符的功能来分类，可以分为算术运算（+，-，*，/，%，++，--），关系运算(>，<，>=，<=，==，!=)，布尔逻辑运算(!，&&，||)，位运算(>>，<<，>>>，&，|，^，~)，赋值运算(=，op=)，条件运算(?:)，强制类型转换(type)expr，和其他运算〔如实例运算 instanceof，分量运算(.)，内存分配运算 new，方法调用运算()等〕。下面我们将介绍常用的运算。

1. 算术运算

算术运算对整型和浮点型数据进行操作，分为一元运算和二元（binary）运算两种。见下表：

表 2-4 一元算术运算

一元运算	用法	描述
+	+op	正值
-	-op	负值
++	++op，op++	加一
--	--op，op--	减一