国家教育委员会《面向二十一世纪计算机类教学内容和课程体系改革研究》项目推荐书籍

JAVA语言系列丛书

JAVA语言程序设计教程

许满武 主编
张宏 睢安 严悍 编著

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
URL:http://www.phei.com.cn

JAVA 语言程序设计教程

许满武 主编

张宏 睢安 严悍 编著

電子工業出版社

PUBLISHING HOUSE OF ELECTRONICS INDUSTRY

内容简介

Java语言在1995年推出，是适合于网络为中心计算的程序设计语言，它是纯对象式的，并具有独立于平台的特性，本书第一、二篇介绍Java语言的对象机制和基本计算与控制成分，它们是对象式语言的核心部分。第三篇介绍应用编程中不可缺少的异常、字符串和线程。第四篇介绍系统级类库。

本书为国家教育委员会《面向二十一世纪计算机类教学内容和课程体系改革研究》项目推荐书籍，可作为计算机专业本科与研究生教学用书，并且可供软件开发人员参考。

书　　名：JAVA语言程序设计教程
主　　编：许满武
编 著 者：张宏　睢安　严悍
责任编辑：邓露林
印 刷 者：民族印刷厂
装 订 者：三河市金马印装有限公司
出版发行：电子工业出版社出版、发行　URL：http://www.phei.com.cn
　　北京市海淀区万寿路173信箱　邮编 100036　发行部电话 68214070
经　　销：各地新华书店经销
开　　本：787×1092 1/16 印张：14.25　字数：364.8千字
版　　次：1998年1月第一版　1998年1月第一次印刷
书　　号：ISBN 7-5053-4358-0
　　TP·2001
定　　价：18.00元

前　言

以网络为中心的计算是当今计算机应用发展的主流。随着计算机系统规模的扩大和软硬件体系结构的日趋复杂，软件的质量已成为突出的问题。软件危机自六十年代被明确认识以来，计算机科学工作者一直寻求有效的软件开发方法。纯形式化的软件开发方法取得了很大的进展，但人类问题求解的智能思维过程与问题求解的形式化表述之间存在着一条难以跨越的鸿沟，因此无法根本解决软件质量问题。而非形式化的软件开发途径已在实际的软件开发中日趋成熟，其中尤为突出的是面向对象技术，在问题分析和软件设计中，对象建模已形成完整的方法。在本书中将要讨论新近推出的 Java 语言，以其纯对象式和独立于平台的特性，为软件的实现提供了卓越的编程工具。

本书分为四篇，第一篇介绍 Java 语言的对象机制。为保证语言定义的结构清晰，在 Java 语言中，类只允许单一继承，为了获得多重继承的益处，提出了接口的概念，类和接口组成了 Java 语言的对象机制。第二篇介绍 Java 语言的表达式和语句，表达式是用来计算值的语言成份，由于表达式每次只能计算出结构相对简单的值，因此控制流用来存放中间结果以及参与复杂的计算。第三篇介绍 Java 语言的异常、字符串和线程，本篇从纯对象式的角度来讨论应用编程中不可缺少的成份。第四篇介绍系统级类库，严格地说 Java 语言设计中只考虑基本计算与控制流和对象机制两个方面，类库则给编程人员提供了灵活有力的支持。类库可划分为系统级和应用级，由于应用级与具体问题求解领域密切相关，本书仅讨论系统级类库，其中的类是在一般的编程过程中经常用到的。

计算机是当代发展最为迅速的学科领域之一，Java 语言可以说是程序设计语言发展中最新的突破。

在本书编写中，郝小卫、章萍同志做了大量的工作，在此致谢！

作　者

一九九七年九月　于南京

目　录

第一篇　Java语言的对象机制

第一章　Java 概述

本章概要介绍 Java 编程语言，以便很快了解并使用 Java。这里仅介绍语言的主要特性，而没有论及细节。以后的各章节将详细讨论 Java 语言的特色。

1.1　引　　言

Java 程序是由类构造的。从一个类的定义可以创建任意多个对象，这些对象被称为这个类的实例。可以把类看作是一个生产零部件工厂的蓝图，生产出来的零部件就是对象。

一个类包含两种成份：域(field)和方法(method)。域既是类的数据，也是该类生成的对象的数据。域构成了对象或相应类的状态。方法是语句的集合；方法对域进行操作，从而处理对象的状态。

多数语言总以打印"Hello World"为其第一个简例，下面是其 Java 语言的示例：

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

使用文本编辑器把源程序输入到一个文件中，然后运行 Java 编译器编译这段源代码，生成 Java 字节码，Java 字节码是运行在 Java 虚拟机上的"机器语言"。源代码的编辑和编译，各个系统不尽相同，在此从略。当你运行此程序时，将显示：

```
Hello,World
```

下面我们具体分析一下这个程序的含义。

这个程序说明了一个称为 HelloWorld 的类，这个类只有一个方法 main。类的成员用花括号{}括起来，并置于类名之后。这个类没有域。

main 方法唯一的参数是一个由 String 对象构成的数组，这些 String 对象可在命令行调用该类时得到。

由于 main 方法不返回值，所以说明为 void。main 方法是 Java 中特殊的几个方法之一。只要进行如上的说明，当你将这个类作为应用来运行时，该类的 main 方法将被执行。在运行时 main 方法可以创建对象、计算表达式、调用其它方法等等，完成应用中所定义的行为。

在上面这个例子中，main 只有一个语句，它调用了在 System 类的 out 对象中的一个方法。方法的调用通过指定一个对象的引用和一个方法名来进行，中间用点符号隔开。HelloWorld 类调用了 out 对象的 println 方法，在一个标准输出流上显示一个以换行结束的字符串。

1.2 变　量

Java 有一些内置的基本数据类型：整型、浮点型、布尔型和字符型。对于这些基本类型的数据，在 Java 中无须再行定义，而程序员定义的对象类型则不然。Java 没有“缺省”类型；每个变量的类型必须显式定义。Java 的基本数据类型是：

boolean	true 真 / false 假
char	16 - 位 Unicode 1.1 字符
byte	8 位整数(带符号)
short	16 位整数(带符号)
int	32 位整数(带符号)
long	64 位整数(带符号)
float	32 位浮点数(IEEE 754 - 1985)
double	64 位浮点数(IEEE 754 - 1985)

下一个例子是打印无穷级数 $\sum_{i=1}^{\infty} i^2$ 的部分和 $\sum_{i=1}^{n} i^2$，给出 $n \leqslant 50$ 的所有值。

该打印程序比较简单，但是可以揭示如何在程序中说明变量，如何写简单的循环，如何做简单的算术运算。程序如下：

```
class PartialSum {
/** Print out the Partial Sum for n≤50 */
    public static void main(String[] args) {
        int s = 0;
        int i = 1;
        while (i <= 50) {
            s = s + i * i;          /* New PartialSum is
                                       old PartialSum plus i^2 */
            i = i + 1;              // new term number
            System.out.println(s);
        }
    }
}
```

这个简单的例子说明了一个 PartialSum 类，类中有一个 main 方法。在 main 方法的头两行说明了两个变量：s 和 i。被说明的每个变量必须有一个类型放在变量的前面。i 和 s 具有 int 类型，即一个 32 位带符号的整数类型，变化范围：-2^{32}到 $2^{32}-1$。

在上面的程序中，说明了 s 和 i 的初始值分别为 0 和 1。当变量被说明时，用"="操作符和初始化表达式设置变量的初始值。“=”操作符的左面是被设置变量的名称，右面是用表达式表示的值。

变量在初始化之前是没有定义的。若试图使用一个尚未赋值的变量，Java 编译器将指出错误。

while 语句是 Java 的一种循环方式。首先计算 while 表达式的值，如果为真，就执行循环体，再测试 while 表达式的值，以决定循环体执行与否，如此重复，直到 while 表达式的值为假。如果表达式不能为假，程序将永远执行，除非有某些可以脱离循环的功能起作用，比如 break

语句,或者一个异常发生。

while 表达式是一个布尔表达式,在上面程序中,这个表达式判断序列的数量是否小于等于 50,如果小于等于 50,则打印出来并计算下一个值;如果大于 50,那么控制将转到 while 循环体后面的第一条语句,在这个例子中 main 方法结束,于是程序终止。

注意到在 PartialSum 例子中 println 方法接受一个整数参数,而在 HelloWorld 例子中接受一个字符串作为参数。println 是一种名复用(overload)的方法,名复用方法可以接受不同类型的参数。

1.3 代码中的注释

散布在程序代码中的英语即为注释,Java 有三个注释的方式,在上面的例子中都使用了。

第一种注释是在//后面加上直到行末的一段文字,这段文字会被编译器略去。

第二种注释是在/* 和 */中间的一段文字,这种方式可以注释多行。

第三种注释是在/** 和 */中间的一段文字,这种注释被称为文本注释(doc comment)。文本注释一般用于描述其后说明中的内容,上例中描述了 main 方法。称为 Javadoc 的工具可以提取文本注释以生成 HTML 文本。

1.4 有名常量

像 12、17.9 和"String Like This"的值被称为常量。常量是你直接指出的值,不由计算得到的,它们在程序执行中保持不变。

程序员使用有名常量有两个原因。其一,常量的名字是一个文本形式,可以也应该描述特定值的含义。

其二,你可仅在程序中的一处定义,当常量需要改变或调整时,也只需在一个地方改变,这样使程序维护更方便。在 Java 中创建有名常量是用 static 和 final 来说明一个变量,并且在说明中提供一个初始化值,例如:

```
class CircleStuff {
    static final float π = 3.1416;
}
```

当我们发现 π 的五位数据精度不够时,只需要在一个地方改变它。把 π 定义为一个 double 类型,即一个 64 位的浮点数,这样我们可以改为一个更为精确的值:3.14159265358979323846。

可以把相关的常量组织起来放在一个类里。比如在扑克牌游戏中可以定义花色常量:

```
class Suit {
    final static int CLUBS = 1;
    final static int DIAMONDS = 2;
    final static int HEARTS = 3;
    final static int SPADES = 4;
}
```

这样,把所有的花色名字组织起来放在一个类 Suit 中,在程序中花色就可以用 Suit.HEARTS,Suit.SPADES 等来存取。

1.4.1 Unicode 字符

我们稍微注意一下就可以发现,在上面的例子中,使用了 π 字符作为常量的名字。而在大多数的程序设计语言中,标识符被严格限制在 ASCII 字符集中的字母和数字。

这是因为 Java 把我们带入到国际化软件的现代世界,你可以用 Unicode 来写 Java 代码。Unicode 是一个国际标准字符集,有 16 位宽度足可以满足世界上绝大多数语言的字符范围要求,这就是为什么我们可以使用 π 作为变量名的原因。π 是 Unicode 中希腊字母集中的一个字母,所以在 Java 源程序中是有效的。Java 现有的多数程序使用 ASCII (7 位的标准字符集),或 ISO - Latin - 1(8 位的标准字符集,通常称为 Latin - 1),但是在程序运行之前都被转变为 Unicode 字符,所以 Java 字符集总是 Unicode 标准。

1.5 控 制 流

控制流描述的是在程序运行时决定执行哪条语句的问题。在上面的程序中,while 循环就是其中的一种方式。其它的控制流语句包括 if/else, for, switch, do/while,还有"块",块是用{}封装起来的多个语句,也被称为"分程序"。下面是改进的 PartialSum 程序,在输出结果中我们标注了序列号,并且用 * 标注所有偶数的数字序列。

```
class ImprovedPartialSum {
/* * Print out the first few PartialSum
    numbers marking events with a "*" */
    static final int MAX-INDEX = 10 ;
    static void main(String[ ]xargs) {
        int s = 0;
        String mark;
        for (int i = 1 ; i < MAX-INDEX; i+ + ) {
            s = s + i * i;
            if (s % 2 = = 0)
                mark ="*";
            else
                mark ="";
            System.out.println(i + ": " + s + mark);
        }
    }
}
```

下面是程序运行的结果:

```
1: 1
2: 5
3: 14 *
4: 30 *
5: 55
6: 91
...
```