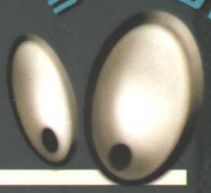
Java 2 实用编程百例



Programmer to Programmer

资深程序员的力作

施 铮 编著

清华大学出版社

实用百例

Java 2 实用编程百例

施　铮　编著

清华大学出版社

北　京

内 容 简 介

Java 语言是 Sun 公司推出的具有开放性、跨平台性和面向网络的交互性软件开发平台。本书通过 100 个实例，全面介绍了 J2EE 中所有实际应用涉及到的技术。全书共分为 10 章，即语言基础篇、用户界面篇、数据库篇、Web 篇、组件篇、图形篇、网络篇、邮件篇、无线篇和模式篇。

本书的每个实例都给出了实例说明、详细编程步骤和关键代码分析。某些实例的关键操作步骤和运行结果均给出了实际运行图示。实例源代码可通过 http://www.tupwk.com.cn/downpage/index.asp 下载。

本书内容全面、图文并茂，解释详尽，既可以作为 Java 编程初学者的基础教程，也可以作为 Java 程序开发人员的参考书。

图书在版编目(CIP)数据

Java 2 实用编程百例/施铮编著. —北京：清华大学出版社，2005.4

(实用百例)

ISBN 7-302-10403-4

Ⅰ. J…　Ⅱ. 施…　Ⅲ. JAVA 语言—程序设计　Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2005)第 010318 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：胡伟卷　　**文稿编辑**：刘金喜

封面设计：张海滨　　**版式设计**：康　博

印 刷 者：北京嘉实印刷有限公司　　**装 订 者**：三河市金元装订厂

发 行 者：新华书店总店北京发行所

开　　本：185×260　**印张**：29.5　**字数**：681 千字

版　　次：2005 年 4 月第 1 版　2005 年 4 月第 1 次印刷

书　　号：ISBN 7-302-10403-4/TP・7072

印　　数：1～5000

定　　价：42.00 元

前　言

众所周知，Java 以其独有的开放性、跨平台性和面向网络的交互性席卷全球，以其安全性、易用性和开发周期短的特点，迅速从最初的编程语言发展成为全球第二大软件开发平台。这些优点已引起国内外计算机界的极大关注，Java 技术已成为当今世界信息技术三大要点之一。

本书是清华大学出版社《实用百例》丛书中的一本。书中共包含了 100 个实例，这 100 个实例涵盖了目前 J2EE 中所有实际应用涉及到的技术，每个示例由“实例说明”、“编程步骤”、“代码分析”等部分构成。全书共分为以下 10 章：

第 1 章“语言基础篇”。介绍了类型之间的转换、运算符的使用、控制结构的使用、异常处理、多线程应用等实例。并且演示了如何在 JBuilder 中新建一个工程、新建一个 Class、运行代码。通过本章实例的学习，读者可以迅速对 Java 基本的语法知识有个比较全面的了解，为后续几章的学习打好基础。

第 2 章“用户界面篇”。介绍了 Java 的两个主要图形界面工具：AWT 包和 Swing 包。Swing 包是构建 Java 图形界面标准的 API(应用程序接口)。本章重点讲解了 AWT 和 Swing 主要控件的使用方法。

第 3 章“数据库篇”。由于数据库编程是 Java 编程的核心，掌握好 Java 数据库编程是程序员应该掌握的基本技能之一。Java 提供了很多数据库编程的接口和方法。在本章的实例中讲述如何将 Java 同 SQL Server 数据库连接，并进行相关操作。

第 4 章“Web 篇”。Web 技术是目前发展最快的技术，本章利用 10 个实例介绍了用于 Web 开发的基本 Java 技术，其中包括 JSP 的使用、JavaBean 的使用和 Servlet 的使用。

第 5 章“组件篇”。EJB 是一种让开发者快速开发大规模企业应用的组件体系结构，EJB 定义了如何编写服务器组件，并且为服务器端组件和管理这些组件的应用服务器之间提供了标准的协议。在本章的实例中讲述有关 EJB 技术的知识。

第 6 章“图形篇”。Java 2D API 支持复杂、灵活的 2D 图形的开发，并且 Java 3D 可用于开发跨平台的三维应用，包括三维图形、游戏等，目前广泛应用于各个领域。本章利用 10 个实例介绍了从简单的 2D 图形的处理到复杂的 3D 动画的制作。

第 7 章“网络篇”。在网络快速发展的今天，网络应用越来越多。基于网络环境的软件得到了越来越多的应用，网络编程应该是每个程序开发人员掌握的技能。本章通过 10 个简单的实例介绍了网络编程的基本使用方法，涵盖了简单的字符和字节流的使用、套接字的使用、数据包的使用等。

第 8 章“邮件篇”。介绍了关于邮件的操作，包括信息的发送和接收、附件的发送和

接收、信息的查找和删除等。本章的实例用到了 JavaMail API 中常用的功能。

第 9 章“无线篇”。J2ME 是为那些使用有限资源、有限网络连接以及有限图形用户界面能力的设备而开发的一种以广泛的消费性产品为目标的高度优化的 Java 运行环境，包括寻呼机、移动电话、可视电话、数字机顶盒和汽车导航系统等。J2ME 为小型设备带来了 Java 语言的跨平台功能，允许无线设备共享应用程序。本章利用 10 个实例向读者介绍了用于 J2ME 开发的基本技术，主要包括用户界面的使用、持久数据的使用和联网的使用。

第 10 章“模式篇”。介绍了 Java 开发中常用的 10 种设计模式。通过本章的学习，读者不仅能够学会编程，还能够学会如何有效编程。

本书中的实例源代码均经过笔者上机采用 JBuilder 调试过，希望读者能亲自上机试验，以更好地掌握实际编程技巧。实例源代码读者可通过 http://www.tupwk.com.cn/downpage/index.asp 下载。

由于编者水平有限，书中难免有不当之处，欢迎广大读者批评指正。

编　者

2005 年 2 月于西安

xuanxuan_boys@126.com

目　　录

第1章　语言基础篇

本章包括 10 个实例：类型之间的转换、使用运算符、使用控制结构、使用方法分解、面向对象编程、异常处理、使用封装类、使用字符串类、综合使用集合类、多线程应用。本章通过这 10 个基础实例，演示了如何在 JBuilder 中新建一个工程，如何新建一个 Class，如何运行代码。通过本章实例的学习，读者可以迅速对 Java 的基本语法有个比较全面的了解，为后续章节的学习打好基础。

第 1 例　类型之间的转换

【实例说明】

数据类型指明了变量和表达式的状态和行为。通过介绍如何将一个类型的数据转换成其他各类型的数据，带领读者深入学习 Java 的各个基本数据类型。

【步骤】

(1) 启动 JBuilder 编辑器，新建一个工程，给工程命名为 test.jpx，接下来在 Project 面板中右击 test.jpx，在弹出的快捷菜单中选择 New→Package 命令，新建一个包，命名为 ch01.section01。

(2) 在这个包下面新建一个类，我们将其命名为 TypeSwitch.java。在代码编辑窗口中输入下面的代码：

```
package ch01.section01;

public class TypeSwitch {
  public TypeSwitch() {
  }
/*
方法 charSwitch(char x)表明字符型(char)可以转换为整型(int)、长整型(long)、单精度型(float)和双精度型(double)等取值范围宽的类型。
*/
  private static void charSwitch(char x) {

    int i = x;
    long l = x;
```

```
    float f = x;
    double d = x;
    if(x==i&&x==l&&x==f&&x==d)
    System.out.println("字符型(char)可以转换为整型(int)、长整型(long)、"+
                       "单精度型(float)和双精度型(double)");
  }
/*
方法 byteSwitch(byte x)表明字节型(byte)可以转换为短整型(short)、整型(int)、长整型(long)、单
精度型(float)和双精度型(double)等取值范围宽的类型
*/
  private static void byteSwitch(byte x) {

    short s = x;
    int i = x;
    long l = x;
    float f = x;
    double d = x;
    if(x==s&&x==i&&x==l&&x==f&&x==d)
    System.out.println("字节型(byte)可以转换为短整型(short)、整型(int)、"+
                       "长整型(long)、单精度型(float)和双精度型(double)");
  }
/*
方法 shortSwitch(short x)表明短整型(short)可以转换为整型(int)、长整型(long)、单精度型(float)
和双精度型(double)等取值范围宽的类型
*/
  private static void shortSwitch(short x) {

    int i = x;
    long l = x;
    float f = x;
    double d = x;
   if(x==i&&x==l&&x==f&&x==d)
     System.out.println("短整型(short)可以转换为整型(int)、"+
                       "长整型(long)、单精度型(float)和双精度型(double)");
  }
/*
方法 intSwitch(int x) 表明整型(int)可以转换为长整型(long)、单精度型(float)和双精度型(double)
等取值范围宽的类型
*/
  private static void intSwitch(int x) {
    long l = x;
    float f = x;
    double d = x;
```

```
        if(x==l&&x==f&&x==d)
          System.out.println("整型(int)可以转换为长整型(long)、单精度型(float)和双精度型
(double)");
      }
    /*
    方法 longSwitch(long x)表明长整型(long)可以转换为单精度型(float)和双精度型(double) 等取值
    范围宽的类型
    */
      private static void longSwitch(long x) {

        float f = x;
        double d = x;
        if(x==f&&x==d)
          System.out.println("长整型(long)可以转换为单精度型(float)和双精度型(double)");
      }
    /*
    floatSwitch(float x)表明单精度型(float)可以转换为双精度型(double) 等取值范围宽的类型
    */
      private static void floatSwitch(float x) {

        double d = x;
        if(x==d)
        System.out.println("单精度型(float)可以转换为双精度型(double)");
      }

      public static void main(String[] args) {
        char char_x = 'F';
        byte byte_x = 1;
        short short_x = 045;
        int int_x = 78;
        long long_x = 4350L;
        float float_x = 2424.23F;
        double double_x = 230.56D;
        charSwitch(char_x);
        byteSwitch(byte_x);
        shortSwitch(short_x);
        intSwitch(int_x);
        longSwitch(long_x);
        floatSwitch(float_x);
        int_x=~short_x;
        --short_x;
        double_x = short_x+double_x;
        int_x = short_x+byte_x;
```

```
        System.out.println(int_x);
    }
}
```

(3) 按 F9 快捷键，或者选择 Run→Run Project 命令运行当前的程序，运行结果如图 1-1 所示。

```
字符型(char)可以转换为整型(int)、长整型(long)、单精度型(float)和双精度型(double)
字节型(byte)可以转换为短整型(short)、整型(int)、长整型(long)、单精度型(float)和双精度型(double)
短整型(short)可以转换为整型(int)、长整型(long)、单精度型(float)和双精度型(double)
整型(int)可以转换为长整型(long)、单精度型(float)和双精度型(double)
长整型(long)可以转换为单精度型(float)和双精度型(double)
单精度型(float)可以转换为双精度型(double)
37
```

图 1-1　实例运行结果

【代码分析】

(1) 本例主要向读者讲述数据类型之间的转换。

(2) long long_x = 4350L 表明声明一个长整型时需要加后缀 l 或 L 来表示。如果没有这个后缀，就默认成 int 型了。

(3) 程序中的 float float_x = 2424.23F 表明声明一个单精度浮点型变量时必须加后缀 f 或 F 来表示。如果没有这个后缀，就默认成了 double 型了。

注意：

double 型的变量声明时可以不加任何后缀，但最好加上后缀 d 或 D，这样程序的可读性就更强。

(4) 将 int_x=~short_x 改为 short_x=~short_x 会产生语法错误。一般情况下当运算符为取正运算符(+)、取负运算符(-)或按位取反运算符(~)时，如果操作数为字节型(byte)、短整型(short)或字符型(char)，则先转换为整型(int)，再参与算术运算，所以当要将一个短整型(short)的变量取反时，应先转换成整型(int)再取反，结果得到的是 int 型的值，但不能把一个 int 型的数值转换成 short 型的了。

(5) 当运算符为自动递增运算符(++)或自动递减运算符(--)时，如果操作数为字节型(byte)、短整型(short)或字符型(char)，则不用先被转换为整型(int)，而是直接参与算术运算，且运算结果类型也不变，在本例中有--short_x，我们可以说当自减完成后，short_x 还是 short 型，而没有被转换成 int 型。

(6) 本例中的 double_x = short_x+double_x 不能写成 short _x = short_x+double_x，是因为如果操作数之一为双精度型(double)，则另一个操作数要先被转换为双精度型(double)，再参与算术运算。我们可以引申为一般情况下双操作数运算符参与算术运算时，如果两个操作数均不为双精度型(double)、单精度型(float)或长整型(long)，则两个操作数先被转换为整型(int)，再参与算术运算；如果两个操作数中有一个为双精度型(double)、单精度型(float)或长整型(long)中的一种，取值范围窄的类型的变量必须先被转换成取值范围宽的类型再参

与运算，结果的类型是取值范围宽的变量的类型。

(7) 当试图把 int_x = short_x+byte_x 改写成 short_x = short_x+byte_x 时就会产生语法错误，因为必须先把 short_x 和 byte_x 都转换成 int 再参与运算，其结果也是 int 型，所以当将一个类型为 int 的变量转换成 short 型时就会出现语法错误。

注意：

数据类型转换只能是取值范围窄的类型向取值范围宽的类型转换，不能从取值范围宽的类型向取值范围窄的类型转换。

第 2 例　使用运算符

【实例说明】

本实例主要讲述如何使用运算符，以及运算符之间的优先级关系。使用 Java 运算符以一个或多个自变量为基础，可生成一个新值。加号(+)、减号或负号(-)、乘号(*)、除号(/)以及等号(=)的用法与其他所有编程语言类似。

【步骤】

(1) 启动 JBuilder 编辑器，在 Project 面板中右击 test.jpx，在弹出的快捷菜单中选择 New→Package 命令，新建一个包，命名为 ch01.section02。

(2) 在这个包下面新建一个类，将其命名为 Operator.java。在代码编辑窗口中输入下面的代码：

```
package ch01.section02;

public class Operator {
  private static void f(boolean b) {
    System.out.println(b);
  }

  public static void main(String[] args) {
    int x = 1;
    int y = 1;
    // 算术运算符
    x = x * y;
    x = x / y;
    x = x % y;
    x = x + y;
    x = x - y;
```

```
        x++;
        x--;
        x = +y;
        x = -y;
        // 比较运算符
        f(x > y);
        f(x >= y);
        f(x < y);
        f(x <= y);
        f(x == y);
        f(x != y);
        f(! (x > y));
        f( (x > y) && (x != y));
        f( (x > y) || (x > y));
        //按位运算符
        x = ~y;
        x = x & y;
        x = x | y;
        x = x ^ y;
        //移位运算符
        x = x << 1;
        x = x >> 1;
        x = x >>> 1;
        x += y;
        x -= y;
        x *= y;
        x /= y;
        x %= y;
        x <<= 1;
        x >>= 1;
        x >>>= 1;
        x &= y;
        x ^= y;
        x |= y;
        System.out.println(x);
    }
}
```

(3) 按 F9 快捷键，或者选择 Run→Run Project 命令运行当前的程序，运行结果如下：

```
false
false
true
true
```

```
false
true
true
false
false
1
```

【代码分析】

(1) 运算符的优先级决定了存在多个运算符时一个表达式各部分的计算顺序。Java 对计算顺序做出了特别的规定。其中，最简单的规则就是乘法和除法在加法和减法之前完成。程序员经常会忘记其他优先级规则，所以最好用括号明确规定计算顺序。Java 中运算符的优先级由高到低排列如下：

```
  [ ]()
++  --  !  ~     instanceof
new(type)
*/ %
+-
>>  >>>  <<
<>  <=   >=
==  !=
&
^
|
&&
||
?:
=  +=  -=  *=  /+     %=  ^=
&=  \=  <<=  >>=  >>>=
```

(2) 运算符用来表明对操作数所进行的运算。运算符主要有两种分类方式：按照操作数的数目来分，可以分为一元运算符、二元运算符和三元运算符；按照运算符的功能来分，可以分为算术运算符、赋值运算符、关系运算符、逻辑运算符、位运算符、移位运算符和条件运算符。

(3) 算术运算符用在数学表达式中，其运算数必须为数字类型。算术运算符不能用在布尔类型上，但它可以用在字符类型上，因为在 Java 中，char 类型实际上是 int 类型的一个子集。

(4) 赋值运算符用来将具体的值赋给变量。普通的赋值运算符用“＝”表示。此外，Java 中还提供了一些将算术运算符和赋值结合起来的特殊赋值运算符。

(5) 关系运算符用来对两个值进行比较，返回值为 boolean 类型。Java 中的任何类型的值，如整数、浮点数、字符、布尔值，都可以用“＝＝”来判断是否相等，用“！＝”来

判断是否不等。但只有整数浮点数和字符才能进行大小的比较。

(6) 逻辑运算符的运算数只能是布尔型的值，其运算结果也为布尔类型。逻辑与、逻辑或和条件与、条件或在通常情况下的运算结果是一样的。它们的区别在于：逻辑与和逻辑或对两个运算数都要求值，然后再进行逻辑运算；而使用条件与和条件或时，如果根据第一个运算数的值就能确定运算结果，则它就不在对第 2 个运算数求值。因此，有时也把条件与和条件或称为“短路与”和“短路或”。这种短路的运算特性在某些情况下是非常有用的。

(7) 位运算符对 byte、int、long、float、double 以及 char 型的值以位为单位进行逻辑运算。

(8) 移位运算符用来将被操作数移动规定的位数。其使用格式通常如下：

```
value operator num
```

其中 value 为被操作数，num 为规定的移动次数。“>>”和“>>>”的区别在于：使用“>>”时，符号位自动扩展；而使用“>>>”时，左边空出的位以 0 填充。例如，我们看下面两条语句：

```
int x = 0x80000000;
x = x >> 4;
```

0x80000000 写成二进制形式如下：

```
1000 0000 0000 0000 0000 0000 0000 0000
```

右移 4 位后为

```
1111 1000 0000 0000 0000 0000 0000 0000
```

因此 x 的值由 - 2 147 483 648 变成了 - 134 217 728。

现在让我们来看看使用“>>>”情况又如何：

```
int x = 0x80000000;
x = x >>> 4;
```

由于此时右移后空出来的位用 0 填充，所以

```
1000 0000 0000 0000 0000 0000 0000 0000
```

移位后变为

```
0000 1000 0000 0000 0000 0000 0000 0000
```

即 x 的值由 - 2 147 483 648 变为 134 217 728。

移位运算符也可以和赋值相结合，简写为移位赋值运算符。例如，x = x >> 4 和 x = x >>> 4 就可以分别写成 x >>= 4 和 x >>>= 4。