<<Java 7入门经典>>

图书基本信息

书名：<<Java 7入门经典>>

13位ISBN编号：9787302289593

10位ISBN编号：730228959X

出版时间：2012-7-1

出版时间：清华大学出版社

作者：(美) 霍尔顿(Horton, I.)

译者：梁峰

前言

欢迎阅读《Java 7 入门经典》，本书通俗易懂，是学习Java语言以及Java平台API的综合性入门指南。
本书提供了在Java JDK 7或更高版本环境下进行程序开发的一些基础知识。
本书除了教授Java知识外，还会介绍与Java程序员有关的各种主题。
本书的组织结构经过了仔细设计，符合思维逻辑，让Java编程学习的各个阶段都环环相扣。
本书读者对象 Java编程应用范围广泛，而且随着语言本身以及关联的函数库的增长，Java的应用领域仍然在不断扩展。
自从发布以来，Java作为一门面向对象的语言已成为Internet编程、跨平台应用以及教学中的首选语言。
在笔者看来，这其中有三个原因：Java与生俱来的系统无关的特性、Java语言的简单性和强大，以及Java作为工具可以有效并轻松实现的编程任务的广泛性。
对于主要的应用程序开发，Java是许多程序员的选择。
与其他语言相比，Java能提供便捷的开发和维护优势。
此外，还有在各种计算机和操作系统中不用修改代码即可运行的内在能力。
使用Java可以做到更多、更快、更容易。
本书旨在提供对于Java语言的全面理解以及在一些Java应用程序上下文环境中进行编程的经验，希望读者能在其中的每个核心领域都打下坚实的基础。
书中介绍的Java各方面知识都通过实际的例子进行解释和说明，这些例子也可以自己创建并运行，而且也应该自己进行试验。
每章末尾的练习有助于对所学知识进行尝试。
在理解本书介绍的主题内容之后，就能开始编写有特色而且有效果的Java程序了。
书名中的“入门”更多是指本书的编写风格而不是指读者的能力水平，所以也可以称为“直通Java”，这是因为本书的结构设计合理，不论是已有其他语言编程经验的程序员还是刚入门的新手，本书都适合阅读。
本书假定您至少有一点编程的基础，比如应该至少理解程序运行的基本概念。
但是在阅读本书之前并不需要太多的预备知识。
本书讲解的进度虽然有些快，但却覆盖了对Java运行工作机制的所有必要解释。

内容概要

无论学习Java是为了编写嵌入网页的定制applet，还是为了编写大型应用程序，《Java7入门经典》都十分适合阅读。
《Java7入门经典》是关于Java语言讲解最为细致、全面的入门书籍，介绍了使用最新的Java JDK 7开发程序所需要的所有基础知识。
书中包含大量的示例，清晰地解释了涉及的关键概念，演示了Java开发的复杂细节。
在阅读各个章节的过程中，您还能获得无价的编程经验，并逐步开始编写功能全面的Java程序。
Ivor Horton在编写入门类编程图书方面独具心得，著作深受好评。
Java SE 7相对之前发布版本更新了超过30%的语言特性，本书从基础知识入门，介绍了使用这一最新发布版本编写Java程序的方方面面，涵盖了Java的所有语言新特性，可为读者打下坚实的基础。

作者简介

作者：(美国)Ivor Horton 译者：梁峰Ivor Horton是Java、C和C++编程语言方面的杰出作者，由他执笔的图书都非常有名，如《Visual C++ 2010入门经典(第5版)》、《C语言入门经典(第4版)》、《C++入门经典(第3版)》等。
他编写的图书十分适合初学者学习，讲解细腻、全面，示例丰富，深受读者好评。
Ivor Horton还是私人实践方面的系统顾问。

书籍目录

第1章Java简介 1.1 Java概览 1.2 Java语言的特性 1.3学习Java 1.3.1 Java程序 1.3.2 Java学习路线图 1.4 Java环境 1.5 Java中的面向对象编程 1.5.1什么是对象 1.5.2如何定义对象类 1.5.3对象操作 1.5.4 Java程序语句 1.5.5封装 1.5.6类与数据类型 1.5.7类与子类 1.5.8使用对象的优势 1.5.9标记 1.5.10泛型类 1.6 Java程序结构 1.6.1 Java的类库 1.6.2 Java应用程序 1.7 Java和Unicode 1.8 小结 1.9资源 第2章程序、数据、变量和计算 2.1数据和变量 2.1.1变量的命名 2.1.2变量名与Unicode 2.1.3变量与类型 2.2整数数据类型 2.3浮点数数据类型 2.3.1浮点数字面量 2.3.2声明浮点型变量 2.4固定变量的值 2.5算术运算 2.5.1整数计算 2.5.2整数除法和余数 2.5.3增量与减量运算符 2.5.4短整数类型的计算 2.5.5整数算术中的错误 2.5.6浮点计算 2.5.7其他浮点算术运算符 2.5.8浮点算术中的错误情况 2.5.9混合算术表达式 2.5.10显式转换 2.5.11赋值中的自动类型转换 2.6 op=运算符 2.7数学函数和常量 2.8存储字符 2.8.1字符转义序列 2.8.2字符算术 2.9位运算 2.9.1使用AND和OR运算符 2.9.2使用异或运算符 2.9.3位移操作 2.9.4位操作方法 2.10取值范围为固定整数值集合自变量 2.11布尔变量 2.12运算符的优先级 2.13程序注释 2.14 小结 第3章循环与逻辑 3.1 决策 3.1.1 比较 3.1.2 if语句 3.1.3嵌套的if语句 3.1.4比较枚举值 3.2逻辑运算符 3.2.1逻辑与操作 3.2.2逻辑或操作 3.2.3异或操作 3.2.4布尔非操作 3.2.5使用标准库方法测试字符 3.3条件运算符 3.4 switch语句 3.5变量的作用域 3.6循环 3.6.1循环的种类 3.6.2使用浮点值计数 3.6.3嵌套循环 3.6.4 continue语句 3.6.5带标签的continue语句 3.6.6在循环中使用break语句 3.7 断言 3.8 小结 第4章数组与字符串 4.1 数组 4.1.1数组变量 4.1.2定义数组 4.1.3数组长度 4.1.4访问数组元素 4.1.5重用数组变量 4.1.6初始化数组 4.1.7使用数组 4.1.8二维数组 4.1.9字符数组 4.2字符串 4.2.1字符串字面量 4.2.2创建String对象 4.2.3字符串数组 4.3字符串操作 4.3.1连接字符串 4.3.2比较字符串 4.3.3字符串排序 4.3.4访问字符串中的字符 4.3.5在字符串中查找字符 4.3.6查找子字符串 4.3.7提取子字符串 4.3.8 String对象的修改版本 4.3.9从String对象创建字符数组 4.3.10使用字符串执行基于集合的for循环 4.3.11在字符串中获取字节数组形式的字符 4_3.12从字符数组中创建String对象 4.4可变字符串 4.4.1创建StringBuffer对象 4.4.2 StringBuffer对象的容量 4.4.3为StringBuffer对象修改字符串的长度 4.4.4增长StringBuffer对象 4.4.5寻找子字符串的位置 4.4.6替换缓冲区中的子字符串 4.4.7插入字符串 4.4.8从可变字符串中提取字符 4.4.9可变字符串的其他操作 4.4.10从StringBuffer对象创建String对象 4.5 小结 第5章定义类 5.1类的定义 5.1.1类定义中的域 5.1.2类定义中的方法 5.1.3访问变量和方法 5.1.4 Final域 5.2定义类 5.3定义方法 5.3.1方法的返回值 5.3.2参数列表 5.3.3定义类方法 5.3.4访问方法中的类数据成员 5.3.5变量this 5.3.6初始化数据成员 5。
4构造函数 5.4.1默认构造函数 5.4.2创建类的对象 5.5定义和使用类 5.6方法重载 5.6.1多个构造函数 5.6.2使用构造函数复制对象 5.7使用对象 5.8递归 5.9理解包 5.9.1对类打包 5.9.2将类从包添加到程序中 5.9.3程序中的包和名称 5.9.4导入静态类成员 5.9.5标准包 5.10类成员的访问控制 5.10.1使用访问属性 5.10.2设定访问属性 5.10.3选择访问属性 5.11 嵌套类 5.1 1.1静态嵌套类 5.11.2使用非静态嵌套类 5.11.3使用非顶级类的嵌套类 5.11.4本地嵌套类 5.12 小结 第6章扩展类与继承 6.1使用已有的类 6.2类继承 6.2.1继承数据成员 6.2.2继承方法 6.2.3覆盖基类方法 6.3@Override标记 6.4选择基类访问属性 6.5 多态 6.6多级继承 6.7抽象类 6.8通用超类 6.8.1 toString0方法 6.8.2判定对象的类型 6.8.3复制对象 6.9接收可变数目参数的方法 6.10转换对象 6.10.1转换对象的时机 6.10.2识别对象 6.11枚举进阶 6.12设计类 6.13使用final修饰符 6.14接口 6.14.1在程序中封装常量 6.14.2用接口声明方法 6.14.3扩展接口 6.14.4使用接口 6.14.5将接口类型作为方法的参数使用 6.14.6在接口定义中嵌套类 6.14.7接口与真实环境 6.15匿名类 6.16 小结 第7章异常 7.1异常的基本思想 7.2异常类型 7.2.1 Error类型的异常 7.2.2 RuntimeException类型的异常 7.2.3 Exception类的其他子类 7.3处理异常 7.3.1 设定方法能够抛出的异常 7.3.2处理异常 7.3.3 try代码块 7.3.4 catch代码块 7.3.5在一个代码块中捕获多种异常类型 7.3.6 finally代码块 7.3.7构造方法 7.3.8执行顺序 7.3.9嵌套的try代码块 7.3.10重新抛出异常 7.4异常对象 7.4.1 Throwable类 7.4.2标准异常 7.5定义自己的异常 7.5.1定义异常类 7.5.2抛出自己的异常 7.5.3异常的抛出策略 7.6 小结 第8章理解流 8.1流与输入输出操作 8.2流的概念 8.2.1输入流与输出流 8.2.2二进制流与字符流 8.3输入输出类 8.3.1基本的输入流操作 8.3.2缓冲输入流 8.3.3基本的输出流操作 8.3.4 Reader和Writer 8.4标准流 8.4.1从键盘读入数据 8.4.2写到命令行中 8.4.3 printf（）方法 8.4.4将数据格式化为字符串 8.5 小结 第9章访问文件与目录 9.1访问文件系统 9.2使用Path对象 9.2.1访问系统属性 9.2.2设置系统属性 9.2.3测试和检查Path对象 9.2.4查询文件和目录 9.2.5获取文件属性 9.2.6其他的路

径操作 9.3创建与删除目录和文件 9.3.1创建目录 9.3.2创建文件 9.3.3删除文件和目录 9.4获取目录内容 9.5关闭流 9.6移动与复制文件和目录 9.6.1对文件或目录重命名 9.6.2复制文件和目录 9.6.3遍历文件树 9.7 小结 第10章写文件 10.1文件I／O基础 10.2文件输出 10.3通过输出流写文件 …… 第11章读文件 第12章序列化对象 第13章泛型 第14章集合框架 第15章一组有用的类 第16章线程 第17章创建窗口 第18章处理事件 第19章在窗口中绘图 第20章扩展GUI 第21章填充和打印文档 第22章Java和XML 第23章创建和修改XML文档

章节摘录

版权页： 插图： 1）十六进制字面量 Java中的十六进制字面量在开头都有o）【或0X，而且遵循使用字母A到F（或者a到f）分别表示数字10到15的约定。
如果对十六进制不是很熟悉，这里有一些例子：如果不熟悉十六进制数，可以在附录8中找到有关它们工作方式的说明。
上面所有的十六进制字面量都是int类型。
如果想要设定一个long类型的十六进制字面量，就必须在字面量后面添加一个L，就像对十进制字面量一样。
例如，0xFL是一个与十进制值l5等同的十六进制字面量。
当然，可以将一个字面量，例如0xAABBCCD9L写成0xAABB_CCD9L。
这里的下划线字符将十六进制数字每4个分成一组。
每组的4个十六进制数对应内存中的两个字节。
与十进制整数字面量一样，下划线只能在十六进制字面量的数字之间出现，因此0x_3ABC和0x3ABC_都不对。
2）二进制字面量 有时，将整数字面量设置为二进制值更加方便。
在一个字面量的前面加上0b或0B就能将其标识为一个二进制数。
在这种情况下，数字只能是0或1。
例如，0b110010101011或0B110010101011和0xCAB以及十进制值3243一样。
也可以在二进制字面量中使用下划线字符，所以可以将值写为Ob1100 1010 1011，那样更容易阅读。
每组的4个二进制数对应一个十六进制数。
当然，二进制字面量也可以是long类型；只需要在数字后面附加一个L。
0b_1000和0b1000_都不正确，因为下划线只能在数字之间出现。
3）八进制字面量 写八进制数字字面量时要以零开头，所以035和067都是int类型的八进制数，而0777777L是long类型的八进制字面量，后者也可以写成0777_777L。
八进制数只能使用数字0～7，而且每个八进制数都定义为3比特。
在以前机器都用3比特倍数的长字来存储数字时，经常使用八进制数。
现在很少有必要使用八进制数，但是必须注意不要意外使用它们。
如果在一个整数字面量的前面添加一个0，Java编译器就会认为在指定一个八进制值。
除非其中一个数字比7大，导致编译器将其标记为错误，否则将无法知道该错误，而这个数字也不会是想象中的值。
2.2.1声明整型变量 如前所述，可以使用如下语句声明long类型的变量：long bigOne；该语句是对变量bigOne的声明，指定变量bigOne存储一个long类型的值。
编译该语句时，会为变量bigOne分配8字节的内存。
Java不会自动初始化一个这样的变量。
如果想要变量有一个初始值，而不是使用内存上次使用后留下来的无效值，就必须在声明中指定自己的值。
为了声明变量bigOne并初始化为2，999，999，999，只需要这样写：long bigOne=2_999_999_999L；这会将该变量设置为等号后面的值。
在声明变量时总是初始化是一个很好的实践。
在字面量中插入下划线是为了使它易读。
注意如果在计算中使用没有赋值的变量，程序将无法编译。
有时候如果不在声明变量时对其初始化，编译器就不能在使用该变量之前判断出它是否已经初始化，即使看起来很明显已经初始化。
这也会被标记为一个错误，但是如果能在声明这些变量时总是初始化它们，就可以避免这些问题。

编辑推荐

《Java 7入门经典》为编程导师Ivor Horton最新力作。

版权说明