高等学校计算机教材

Core Java /Java应用程序设计教程

刘甲耀 郑小川 严桂兰 编著

WUHAN UNIVERSITY PRESS
武汉大学出版社

Core Java / Java应用程序设计教程

高等学校计算机教材

Core Java /Java应用程序设计教程

刘甲耀 郑小川 严桂兰 编著

WUHAN UNIVERSITY PRESS
武汉大学出版社

图书在版编目(CIP)数据

Core Java/Java 应用程序设计教程/刘甲耀,郑小川,严桂兰编著. —武汉:武汉大学出版社,2009.1
高等学校计算机教材
ISBN 978-7-307-06705-9

Ⅰ.C… Ⅱ.①刘… ②郑… ③严… Ⅲ.JAVA 语言—程序设计—高等学校—教材 Ⅳ.TP312

中国版本图书馆 CIP 数据核字(2008)第 194962 号

责任编辑:黄金文　　责任校对:黄添生　　版式设计:支　笛

出版发行:**武汉大学出版社**　(430072　武昌　珞珈山)
(电子邮件:cbs22@whu.edu.cn 网址:www.wdp.com.cn)
印刷:武汉中科兴业印务有限公司
开本:787×1092　1/16　印张:29.5　字数:707 千字
版次:2009 年 1 月第 1 版　2009 年 1 月第 1 次印刷
ISBN 978-7-307-06705-9/TP·320　定价:42.00 元

内 容 简 介

本书《Core Java/Java 应用程序设计教程》，首先阐述 Core Java/Java 应用程序设计的方法与技巧，然后列举一题多解的示例，在每条语句后加以注解，说明其含义与作用，并对同一示例的不同方案说明其使用的方法与手段，使读者能通过具体的对比及方法与手段的反复使用，熟练地掌握 Core Java/Java 的编程方法。本书内容包括：基本 Core Java/Java(含基本编程模式，基本数据类型，基本数据输入/输出，基本运算符，条件与循环语句，方法)；引用(含引用的含义与操作，对象与引用的基础，字符串， 数组， 异常处理，文件)；对象与类(含面向对象程序设计的含义，javadoc， 基本方法， 软件包， 附加的构造)；继承(含继承的含义， 继承的基本语法， 多重继承，接口， 通用组件的实现)。全书所有示例均 Core Java2(使用 Textpad 工具)环境通过，一题多解的示例充分体现了 Core Java/Java 编程的灵活性、多样性、实用性与趣味性。本书提供了 Core Java/Java 应用程序设计的最基本的理论与方法，若与《Core Java/Java 应用程序设计案例》一书配套使用，会加快掌握使用 Core Java/Java 的编程能力。

本书可作为大专院校计算机及相关专业的教材，并可供各行各业从事计算机应用的人员参考使用。书中附录提供了 Text 与 JDK 的使用步骤及 Core Java 安装步骤和习题参考答案。

前 言

Core Java/Java 是基于网络的纯面向对象编程语言，适用于编写各式各样的软件，适用于各种平台与操作系统，编译后的代码能在互联网上传递，并确保用户安全运行，因而是当前最富有生命力的计算机编程语言之一。

本书具有如下特点：

（1）在阐述基本编程方法的基础上，列举一题多解方案示例，并对每条语句加上注解，说明其含义与作用，对初学者十分合适；

（2）本书是根据作者多年的教学经验的总结和在学生实践中出现的问题，有针对性地编写的，特别适用于作为大专院校的教材和读者入门自学使用。

（3）一题多解的示例，形式新颖，经试用，这种教材能培养学生举一反三的思维方法与能力，反应很好。

（4）Core Java 是 Java 的高级版本，因此，它既适用 Core Java 课程也适用 Java 课程，且在实用上优于 Java，目前正受到广大读者的喜爱与使用。

因此，Core Java 除了包含 Java 的所有功能外，其最大的优越性是数据的输入与输出（尤其格式化输出）特别简单，使用 Core Java 像 C 语言一样容易。为适应当前 Internet 的迅猛发展及各行各业学习 Core Java 的需要，特别是为大专院校研究生和本科生甚至专科生开设面向对象程序设计课程的需要，我们根据多年对 Java 和 Core Java 教学和科研的实践以及 Java 版本的升级，并根据 Core Java 具有的功能来描述编程基本方法，然后列举一例题的多种解决方案，通过一例多解的方式说明 Core Java 编程的灵活性、多样性、实用性与趣味性。

在本书的编写中，承蒙美国某公司 CEO 刘涌博士、美国休斯敦大学（UH）教授冯千妹博士，以及美国贝勒医学院（BCM）教授刘浩博士提供了大量资料，私立华联大学校长侯德富教授给予了关心和支持，在此表示感谢。

本书不足之处，敬请读者指正。

作者 E-mail：yg10501@sina.com。

作 者

2008 年 9 月

目 录

第1章 基本 Core Java/Java

本章介绍基本 Core Java/Java，内容涉及：Core Java/Java 的基本编程模式、基本数据类型、Core Java 的基本输入与输出、基本运算符、条件与循环语句、方法。

1.1 Core Java/Java 的基本编程模式

1.1.1 注解

Core Java/Java 有三种注解形式：

第一种形式是继承 C 的多行注解，以 /*开始，*/ 结束。例如：

```
/* This is a two line comment */
```

这种形式注解不能嵌套，即不能在一个注解中含有另一个注解。

第二种形式是继承 C++，以 // 开始，没有结束标志，是从注解开始扩展到行结束为止。例如：

```
// This is a line comment
```

第三种形式是以/**开始，以*/结束。这种形式的注解可提供 javadoc 实用程序的信息，其将根据注解产生文档。实际上，javadoc 是一个能自动生成文字的工具，它从程序中抽取方法原型及特定格式的注解，生成优质的超文本文档。例如：

```
/** Java documentation comment */
```

注解不会使计算机产生任何操作，而被 Java 编译器所忽略。编程时，加上适当的注解，能提高程序的可读性。

1.1.2 基本编程模式

1. Core Java 独立应用程序的基本编程模式为：

```
import corejava.*;  //引入 corejava 包中本程序用到的类
public class  //用户定义的类
{
    public static void main(String[ ] args)  //主方法
    {
        方法体
    }
}
```

2. Java 独立应用程序的基本编程模式为：

```
import java.io. *;  //引入 java.io 包中本程序用到的类
```

```
public class   //用户定义的类
{
    public static void main(String[ ] args)   throws IOException
    //主方法，并实现 IO 异常
    {
        方法体
    }
}
```

注意：
- 关键字 class 前面的 public 可省略。
- 用户定义的类名可作为文件名，并以 java 为扩展名。
- Java 与 Core Java 基本编程模式不同，主要是程序中用到的包不同。在 Java 中，输入和输出是使用 java.io 包，而在 Core Java 中，输入和输出是使用 corejava.包来完成的，并必须在程序开头引入该包，方可在程序中实现输入/输出。即

```
import java.io.*;  //引入 java.io.*包中本程序用到的类
```

或

```
import corejava.*;  //引入 corejava 包中本程序用到的类
```

[例 1.1.1] 输入您的姓名,并在屏幕上显示所输入的姓名。

方法一：用 Java Application 编程

方案一：

```
//NameJava.java    //注解本程序的文件名为 NameJava.java
import java.io.*;  //引入 java.io.*包中本程序用到的类
public class NameJava    //定义类
{
    public static void main(String[] args)throws IOException
    //主方法，并实现 IO 异常
    {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        //创建 BufferedReader 类对象
        System.out.print("Please input your name: ");  //提示语句
        String name=br.readLine();
        //输入语句，输入一字符串并赋给字符串变量 name
        System.out.println("Your name is: "+name);
        //输出语句，使用 System.out 类调用 println 方法，输出所输入的字符串
        }
}
```

运行结果：

```
Please input your name: Annie Liu
Your name is: Annie Liu
```

方案二：

```
//NameJava1.java    //注解本程序的文件名为 NameJava1.java
import java.io.*;  //引入 java.io.*包中本程序用到的类
```

```
public class NameJava1    //定义类
{
    public static void main(String[] args)throws IOException
    //主方法，并实现 IO 异常
    {
        BufferedReader br=new BufferedReader(new InputStreamReader(System.in));
        //创建缓冲区读入器流类对象
        System.out.print("Please input your name: ");  //提示语句
        System.out.println("Your name is: "+br.readLine());
        //输出语句，使用 br 对象调用 readLine()方法，输出所输入的字符串
    }
}
```

运行结果与方案一相同，两方案主要是采用不同的输入输出方法。

方法二：用 Core Java Application 编程

方案一：使用 System.out.println 语句输出

```
//NameCoreJava.java  //注解本程序的文件名为 NameCoreJava.java
import corejava.*;  //引入 corejava 包中本程序用到的类
public class NameCoreJava  //定义类，类名为 NameCoreJava
{
    public static void main(String[] args)  //主方法
    {
        String name=Console.readLine("Please input your name: ");
        //输入语句，输入一字符串并赋给字符串变量 name
        System.out.println("Your name is: "+name);
        //输出语句，使用 System.out 类调用 println 方法，输出所输入的字符串
    }
}
```

运行结果:

```
Please input your name:  Annie Liu
Your name is: Annie Liu
```

方案二：使用 Format.printf 语句输出

```
//NameCoreJava1.java  //注解本程序的文件名为 NameCoreJava1.java
import corejava.*;  //引入 corejava 包中本程序用到的类
public class NameCoreJava1  //定义类
{
  public static void main(String[] args)  //主方法
  {
    String name=Console.readLine("Please input your name: ");
    //输入语句，输入一字符串并赋给字符串变量 name
    Format.printf("Your name is: %s\n",name);
```

```
        //输出语句，使用 Format 类调用 printf 方法，输出所输入的字符串
    }
}
```

运行结果与方案一相同，输出方法与方案一不同。

[例 1.1.2] 输入两个整数求其和。

方法一：用 Java Application 编程

方案一：一次输入两个整数，输出结果（求和表达式放在输出语句中）

```
//SumJava.java  //注解本程序的文件名为 SumJava.java
import java.io.*;  //引入 java.io 包中本程序用到的类
import java.util.*;  //引入 java.util 包中本程序用到的类
class SumJava  //定义类
{
    public static void main(String[] args)throws IOException
    //主方法，并实现 IO 异常
    {
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
        //创建缓冲区读入器流类对象
        System.out.print("Enter two int numbers: ");  //提示语句
        String s=in.readLine();
        //输入语句，通过对象 br 调用 readLine 方法输入两个整数，并存入变量 s 中
        StringTokenizer str=new StringTokenizer(s);
        //创建 StringTokenizer 类对象
        int a=new Integer(str.nextToken()).intValue();
        //从字符串中取出第一个字符串，转换为整型数赋给整型变量 a
        int b=new Integer(str.nextToken()).intValue();
        //从字符串中取出第二个字符串，转换为整型数赋给整型变量 b
        System.out.println("Sum="+(a+b));
        //输出语句(含计算表达式)，输出计算结果
    }
}
```

运行结果:

```
Enter two int numbers: 5 10
Sum=15
```

方案二：分别输入两个整数，输出结果（求和表达式放在输出语句中）

```
//SumJava1.java  //注解本程序的文件名为 SumJava1.java
import java.io.*;  //引入 java.io 包中本程序用到的类
import java.util.*;  //引入 java.util 包中本程序用到的类
class SumJava1  //定义类
{
    public static void main(String[] args)throws IOException
```

```
    //主方法，并实现 IO 异常
    {
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
        //创建缓冲区读入器流类对象
        System.out.print("Enter an int: ");  //提示语句
        String s=in.readLine();
        //输入语句，通过对象 br 调用 readLine 方法输入字串，并存入变量 s 中
        int a=Integer.parseInt(s);
        //从字符串中取出一个字符串，转换为整型数赋给整型变量 a
        System.out.print("Enter an int : ");  //提示语句
        String s1=in.readLine();  //输入语句
        int b=Integer.parseInt(s1);
        //从字符串中取出一个字符串，转换为整型数赋给整型变量 b
        System.out.println("Sum="+(a+b));
        //输出语句(含计算表达式)，输出计算结果
    }
}
```

运行结果:

```
Enter an int:  5
Enter an int:  10
Sum=15
```

*方案三：分别输入两个整数，输出结果（求和表达式放在输出语句中）

```
//SumJava3.java  //注解本程序的文件名为 SumJava3.java
import java.io.*;  //引入 java.io 包中本程序用到的类
import java.util.*;  //引入 java.util 包中本程序用到的类
class SumJava3  //定义类
{
    public static void main(String[] args)throws IOException
    //主方法，并实现 IO 异常
    {
        BufferedReader in=new BufferedReader(new InputStreamReader(System.in));
        //创建缓冲区读入器流类对象
        System.out.print("Enter an int: ");  //提示语句
        int a=Integer.parseInt(in.readLine());
        //输入语句，输入一个整数并赋给整型变量 a
        System.out.print("Enter an int: ");  //提示语句
        int b=Integer.parseInt(in.readLine());
        //输入语句，输入一个整数并赋给整型变量 b
        System.out.println("Sum="+(a+b));
        //输出语句(含计算表达式)，输出计算结果
```

```
    }
}
```

运行结果与方案二相同。

方法二：用 Core Java Application 编程

方案一：分别输入两个整数，求其和，并输出结果。

```
//Sum.java  //注解本程序的文件名为 Sum.java
import corejava.*;  //引入 corejava 包中本程序用到的类
class Sum  //定义类
{
    public static void main(String[] args)  //主方法
    {
        int a=Console.readInt("Enter an int: ");
        //输入语句，输入一个整数并赋给整型变量 a
        int b=Console.readInt("Enter an int: ");
        int sum=a+b;  //计算语句，将 a,b 相加的值赋给整型变量 sum
        Format.printf("Sum=%d\n",sum);
        //输出语句,Format 类调用 printf 方法，输出计算结果
    }
}
```

运行结果:

```
Enter an int:  5
Enter an int:  10
Sum=15
```

方案二：分别输入两个整数，输出结果（求和表达式放在输出语句中）

```
//Sum1.java  //注解本程序的文件名为 Sum1.java
import corejava.*;  //引入 corejava 包中本程序用到的类
class Sum1  //定义类
{
    public static void main(String[] args)  //主方法
    {
        int a=Console.readInt("Enter an int: ");
        //输入语句，输入一个整数并赋给整型变量 a
        int b=Console.readInt("Enter an int: ");
        //输入语句，输入一个整数并赋给整型变量 b
        Format.printf("Sum=%d\n",a+b);
        //输出语句(含计算表达式)，Format 类调用 printf 方法，输出计算结果
    }
}
```

运行结果与方案一相同。

方案三：一次输入两个整数，输出结果（求和表达式放在输出语句中）

```
//Sum2.java
import corejava.*;  //引入 corejava 包中本程序用到的类
class Sum2  //定义类
{
    public static void main(String[] args)  //主方法
    {
        String str=Console.readLine("Enter two int numbers: ");
        //输入语句，输入两个整数，以字符串存入字符串变量中
        String[] s=str.split(" ");  //字符串之间用空格分隔，存入字符串数组中
        int a=Integer.parseInt(s[0]);
        //从字符串数组元素中取出第一个字符串，转换为整型数赋给整型变量 a
        int b=Integer.parseInt(s[1]);
        //从字符串数组元素中取出第二个字符串，转换为整型数赋给整型变量 b
        Format.printf("Sum=%d\n",a+b);  //输出语句(含计算表达式)
    }
}
```

运行结果:

```
Enter two int numbers:  5 10
Sum=15
```

方案四：一次输入两个整数，求其和，然后输出结果

```
//Sum3.java
import corejava.*;  //引入 corejava 包中本程序用到的类
class Sum3  //定义类
{
    public static void main(String[] args)  //主方法
    {
        String[] s=Console.readLine("Enter two int numbers: ").split(" ");
        //输入语句，输入两个整数，并以字符串（中间用空格分隔）存入字符串数组 s
        int a=Integer.parseInt(s[0]);
        //从字符串数组元素中取出第一个字符串，转换为整型数赋给整型变量 a
        int b=Integer.parseInt(s[1]);
        //从字符串数组元素中取出第二个字符串，转换为整型数赋给整型变量 b
        int sum=a+b;  //计算语句
        Format.printf("Sum=%d\n",sum);  //输出语句
    }
}
```

运行结果与方案三相同。

方案五: 分别输入两个整数，求其和，然后输出结果，与方案一不同的是先说明程序用到的变量

```
//Sum4.java
```