国外经典教材　国外数十所著名院校首选教材

PEARSON Addison Wesley

Java by Dissection

Java解析教程

THE ESSENTIALS OF JAVA PROGRAMMING 层层解析，揭示Java语言之精华

Ira Pohl，Charlie McDowell 著

王晓光 译

清华大学出版社

国外经典教材

Java 解析教程

（美） Ira Pohl
Charlie McDowell 著

王晓光 译

清华大学出版社
北京

内 容 简 介

本书借助于精心设计的示例程序，采用颇受学生欢迎的“解析法”，揭示了 Java 的主要特性，并着重突出了编程风格和编程方法。书中首先介绍了所有基本数据类型和控制语句的习惯用法，然后循序渐进，过渡到 Java 语言面向对象的特性以及这一特性对程序设计的重要性。本书用一半的篇幅深入讨论了一些高级主题，如多线程、GUI、异常处理和文件操作等。本书既可用作相关专业初级教材，也适合打算提高编程技能的读者自修和参考。

图书在版编目（CIP）数据

Java 解析教程/（美）波尔（Pohl, I），（美）迈克道威尔（McDowell, C）著；王晓光译.—北京：清华大学出版社，2003.7
（国外经典教材）
书名原文：Java by Dissection
ISBN 7-302-07392-9
Ⅰ.J… Ⅱ.①波… ②迈… ③王… Ⅲ.JAVA 语言—程序设计 Ⅳ.TP312

中国版本图书馆 CIP 数据核字（2003）第 092332 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

文稿编辑：文开棋
封面设计：立日新设计公司
印 刷 者：世界知识印刷厂
装 订 者：化甲屯小学装订二厂
发 行 者：新华书店总店北京发行所
开　　本：185×260　**印张**：28.25　**字数**：687 千字
版　　次：2003 年 10 月第 1 版　2003 年 10 月第 1 次印刷
书　　号：ISBN 7-302-07392-9/TP・5360
印　　数：1～4000
定　　价：49.00 元

前 言

本书深入浅出地向读者讲解 Java 编程，是为那些没有任何编程经验的读者编写的。本书在透彻介绍 Java 现代化编程技术的同时，展示了所有基本数据类型和控制语句的传统用法，并介绍了 Java 的面向对象特性及其对程序设计的重要性。此外，本书详细解释了 Java 中比较复杂的特性，例如线程(thread)、图形用户界面(Graphical User Interface，GUI)和文件操作功能。本书既可作为专业编程课程的主修教材，也可作为数据结构课程、软件方法学、可比性语言(如 C++)等课程的补充教材，当然，任课教师也可将本书用于将 Java 选作备选语言的其他课程。

Java 问世于 20 世纪 90 年代中期的 Sun Microsystems 公司，它功能强大，是继 C 和 C++之后的现代化程序设计语言。类似于 C++，Java 也在 C 的基础上增加了面向对象的编程概念，例如类(class)、继承(inheritance)和运行时类型绑定(run-time type binding)。类机制提供了用户自定义类型(也称“抽象数据类型”)。虽然 Java 的很多语义特性与 C 和 C++相同，但它增加了大量改进，其中包括名为垃圾回收(garbage collection)的自动内存回收、数组边界检查和强制类型转换。另外，Java 的标准类库(也就是 Java 中的包，即 package)为分布式编程、多线程处理和图形用户界面提供了独立于平台的支持。

虽然 Java 与 C++一样，大部分语义类似于 C。但与 C++不同的是，Java 不是 C 的超集。这使得 Java 的缔造者可以对语义进行大量的改进，以使 Java 比 C 更安全。其结果是，Java 更适合作为首选编程语言。

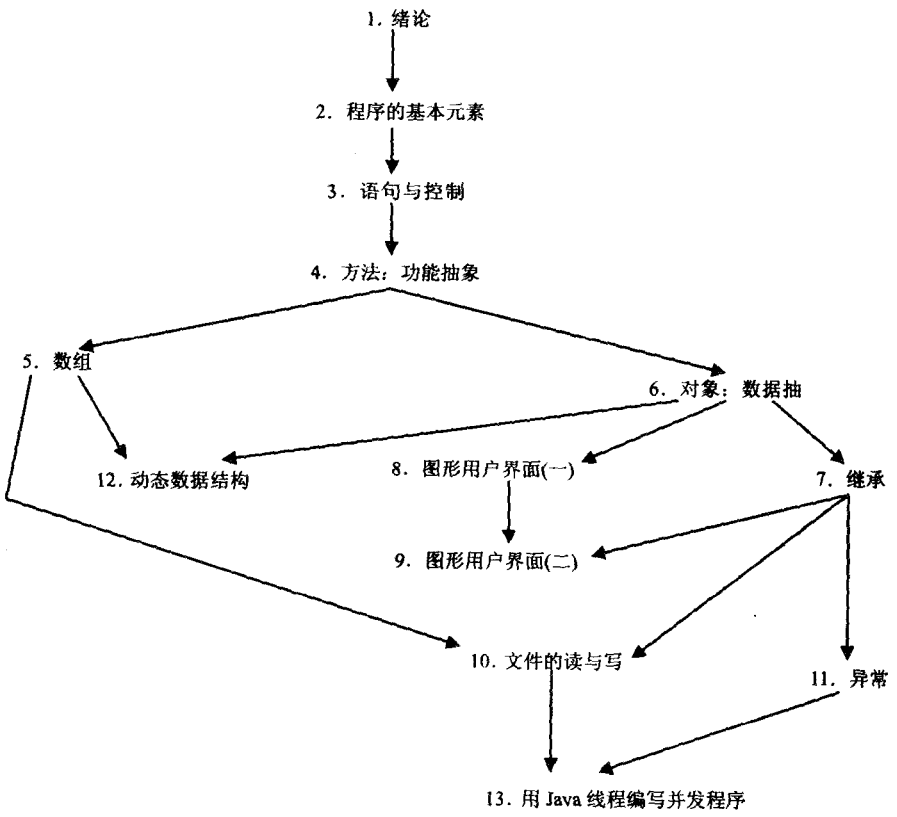
本书以一种经典的编程风格展开介绍，它首先把程序看作指令的简单序列，然后在上面增加控制流和功能抽象。

书中接着介绍数组和使用类的数据抽象，这两个概念的介绍顺序不分先后，既可以先介绍数组，也可以先介绍类的数据抽象。然后讨论继承和图形用户界面，同样地，这两个概念的介绍顺序也不分先后，继承可以在图形用户界面之前或之后介绍。书中最后几章着重介绍高级主题，下面的图形象地展示了本书各章内容之间的相互关系。

本书突出了工作代码。我们将解析对特别能说明当前章主题的一个或多个程序，这种解析过程如同代码的结构化走查。在解析过程中，还将为读者解释新出现的编程元素和习惯用法。

因为 Java 包含一个相对好用、用于创建图形用户界面的标准包，所以入门级编程教材中要介绍如何使用 GUI。目前，编写带有 GUI 的程序就像能创建美观的格式化文本输出一样，属于每个 Java 程序员必“炼”的“基本功”。为了全面理解 Java 中的 GUI 包，读者有必要了解一下面向对象编程(Object-Orentied Programming，OOP)和继承。所以，本书在介绍如何构建 GUI 之后，接着介绍了对象和继承。对于那些希望尽早了解 Java 图形特性的学生，我们在前几章末尾提供了一系列补充练习题(其中介绍了 GUI 和 applet)。这些练习题尽管没有完整解释所需的语言特性，但提供了一些简单 applet 模板。

书中还为没有 C 语言基础的 Java 程序员提供了附录 E“关于 C 语言”，帮助他们补充 C 语言的基础知识。



本书的主要特色：

- **示例教学** 本书是一本强调工作代码的教材，一开始就介绍完整的工作程序。练习题与鼓励读者动手实验的例子紧密结合，从而避免了解释编写工作代码所涉及的大量“废话”。书中各章都有几个重要的示范程序。这些程序的主要元素都将在解析过程中解释。
- **面向对象编程(OOP)** 逐步引导读者去理解面向对象的编程风格。第 2 章将介绍数据值的对象。第 6 章将表明程序员如何从 Java 和面向对象编程中受益。另外，第 6 章还将定义面向对象的一些概念，以及 Java 支持这些概念的方式。第 7 章则要介绍面向对象范例中的两个重要元素——继承和动态方法指定。
- **终端输入** 作为一种现有的、使用较广的语言，Java 仍然不支持简单的终端输入。这样一来，读者将被迫立即构建和使用 GUI，或者使用标准 I/O 包中的不太方便的结构。直接使用 GUI 将要求读者使用一些他们还不理解的语言特性。同样地，使用功能较强而灵活的标准 I/O 包来解决简单的文本输入和数字数据输入，也需要读者尚未掌握的一些语言特性。为了解决这一问题，我们提供了一个 tio 包，这个包支持数字和文本数据的简单输入以及简单的格式输出。附录 C 提供了这个包

的完整代码，其源代码的电子版本可以从 Addison-Wesley 的站点(*ftp://ftp.awl.com/cseng/authors/pohl-mcdowell*)下载。这个包中的主要类简化了很多常见的输入处理需求，并且可用于很多实际应用程序中。

- **Java 中的数据结构** 本书涵盖了计算机科学中的很多标准数据结构，具体涉及到了栈、安全数组、动态分配的多维数组、链表和字符串。每章后面的练习题有助于学生理解如何实现和使用这些结构。这些数据结构的实现与抽象类型和软件的面向对象方法是一致的。
- **图形用户界面** Java 的一个重要部分是支持以独立于平台的方式创建图形用户界面和基于 Web 的程序(称为 applet)。第 8 章和第 9 章将介绍如何用标准的 Java 包 Swing 来创建图形用户界面。这两章详细解释了如何创建有用又有趣的 applet 和 GUI。另外，附录 D 还提供了几个额外的 GUI 组件。针对急于动手编写 applet 的读者，我们从第 2 章的练习题就开始介绍简单的 applet。这些练习题是可选的。第 8 章和第 9 章中 applet 和 GUI，并不要求读者必须完成或阅读前几章的 applet 补充练习题。
- **线程** 入门级教材中一般不涉及多线程编程。但是，对理解事件驱动的、基于 GUI 的程序而言，起码应具备一点线程方面的知识。另外，我们认为，基于线程的编程在各个级别的编程课程中越来越重要。第 13 章将解释线程，并介绍客户端/服务器计算。本书为那些已经掌握了前几章内容的编程人员提供了很好的训练。
- **已通过课程试验** 本书是作者所授课程的基础，自 1997 年以来，作者在各种讲座中一直在用本书内容培训学生。所以，本书内容是通过课程试验的，反映了作者丰富的教学和咨询经验。
- **代码示例** 所有主要的代码段都经过测试，而且自始至终都采用一致而又恰当的编程风格，这一编程风格也是 Java 社区内专业人员的选择。这些代码可以从 Addison Wesley 的站点下载，网址是 *ftp://ftp.awl.com/cseng/authors/pohl-mcdowell*。
- **练习题** 练习题用于测试并巩固学生对 Java 语言的掌握程度。很多练习题都要求读者在阅读过程中交互完成。
- **站点** 本书中的代码和 Addison-Wesley 站点的代码都展现了良好的编程风格。Addison-Wesley 站点提供了本书的示例程序和用于说明本书部分要点的相关程序。

怎样使用本书

- 本书适合作为编程入门基础的第一门课程(类似那些使用 C、Pascal 或 C++的相应课程)。第 1～8 章阐述了类似的一种课程。
- 本书适合概述面向对象的第二门课程或高级课程。
- 已经掌握面向过程编程语言(例如 C 和 Pascal)的读者，可略过第 2～5 章。

- 已具备 OOP 知识的读者，可略过第 6 章和第 7 章。
- 第 8～13 章提供了一系列有趣的高级话题，初级编程课程一般不涉及此类知识。在初级课程中，教师可使用 tio 包和传统的文本输入/输出方法，或者从第 2 章开始布置可选的基于 applet 的练习题，以便读者尽快使用 applet。

目　录

第1章 绪 论

Java 是万维网(World Wide Web, WWW)塑造的第一种主要编程语言。您可以用它进行传统编程，还可以用它所提供的一些特殊特性和库，方便地编写可使用 Web 资源的程序。Java 所提供的特性和库包括，对图形用户界面(Graphics User Interface, GUI)的广泛支持，支持，在 Web 文档中嵌入 Java 程序、支持与世界各地计算机之间的通信以及编写并发行运行，或在多台机器上同时运行的程序。

本章将给出一个如何用计算机来解决问题的完整过程。在这一过程中，首先必须设计一个解决问题的办法，然后将这个菜谱转化成若干个详细的、计算机能遵循的步骤，最后必须选择一种语言(比如 Java)，将这些步骤按计算机能够理解的步骤表达出来。然后，一个叫做编译器(compiler)的程序会把用 Java 语言写的这个解决方案翻译成底层操作，这些底层操作是计算机能够直接执行的。

然后，我们将讨论为什么 Java 会在计算机世界里引起这么大的反响。概括地讲，我们会解释 Web 计算的重要性以及 GUI 在其中所扮演的角色，这些都是促成程序员们转而使用 Java 编程的部分原因。

书中将提供很多例子，其中大部分是完整的程序。我们常常会对这些程序进行解析，以说明每个 Java 编程结构是如何工作的。本章所提到的话题还会出现在后续的各章中，并进行详细说明。本章提供的代码和示例，只是让您感觉一下如何编程，所以读者不必过分关注示例中的一些细节，它们只是给出一个梗概。如果您已经知道了何为程序，并且想立刻开始接触 Java 编程的本质，则可跳过本章，或者快速浏览本章。

1.1 解 决 办 法

计算机程序是用于执行特定任务或解决特定类型问题的详细指令列表。用于解决问题或执行任务的、程序化的指令列表，有时又称为算法(algorithm)。它们在日常生活中很常见，例如毛衣编织说明、服装剪裁说明、烹饪说明、大学课程注册说明、外出旅游说明、自动贩卖机使用说明，它们都是算法，随便检查其中一个即可说明问题。

考虑以下用于烤肉的解决办法：

给肉块撒上盐和胡椒，在肉块中插入一只温度计，把肉块放入烤炉，预加热到 150℃，直至温度计显示的温度达到 80℃～85℃。为肉块配上肉汁(可从肉店直接买或在烤肉时渗出的)。

这个解决办法很不严密——它没有指出“撒”这个词是什么意思，没有指出温度计应该插在哪里，没有指出锅里足够的汁是多少。

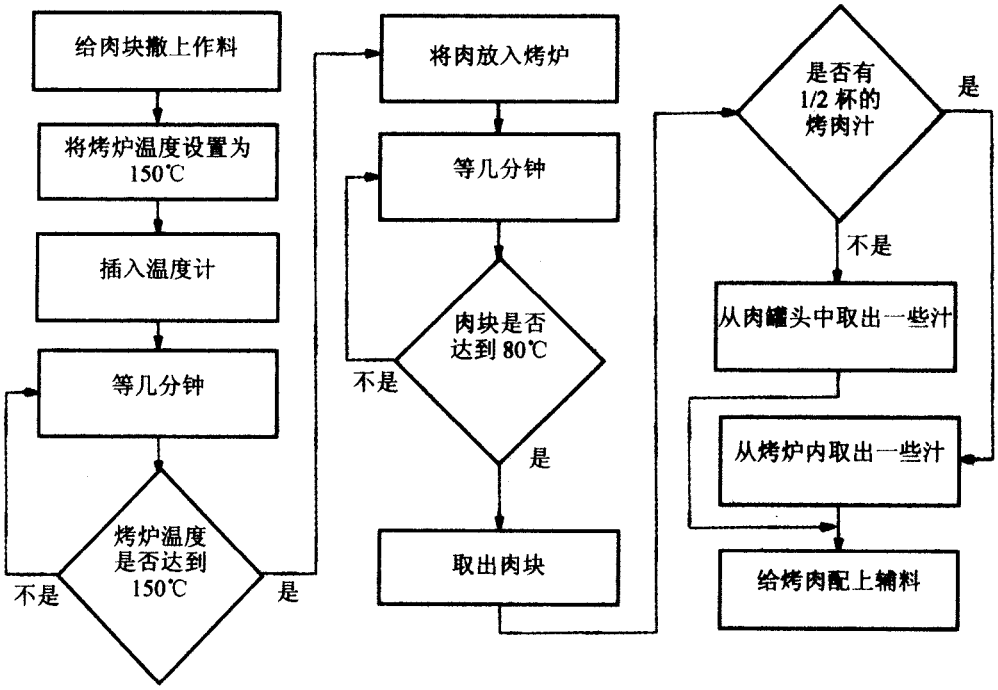
但是，这个解决办法可以进一步简化，简化成指令列表。

做烤肉

1. 在肉块上撒 1/8 茶匙的盐和胡椒。
2. 将烤炉的温度设置为 150℃。
3. 在肉的正中心插入一只温度计。
4. 等几分钟。
5. 如果烤炉的温度还没有达到 150℃，就转到步骤 4。
6. 将肉放入烤炉里。
7. 等几分钟。
8. 检查插在肉里的温度计，如果温度低于 80℃，就转到步骤 7。
9. 从烤炉中取出肉。
10. 如果在烤的过程中渗出的汁达到 1/2 杯，就转到步骤 12。
11. 从肉罐头中取一些汁，转到步骤 13。
12. 从烤炉中取一些汁。
13. 在肉上浇汁。

这些步骤由 3 种指令和行为组成——修改成分和操作装置的指令，检查和测试系统状态的指令及转到其他步骤的指令。第 1 步和第 6 步是第一种指令，第 8 步关于温度的测试和第 10 步关于肉汁的测试是第二种指令，第 5 步和第 8 步的转移(“转到步骤 x”)是第 3 种指令。

用特定的图形符号表示每一种指令，可得出这个烤肉算法的二维表示，如下图所示：



我们称这样的图为流程图(flowchart)。如果要执行程序(准备烤肉)，就按箭头方向执行

每个框里的指令。操作性指令在方框里，测试性指令在菱形框里。流程的转移由箭头决定，因为这样看起来既清晰又明确，所以人们通常用流程图描述程序，而不用指令列表，甚至一些烹饪书作者也在书中采用流程图。本书第 3 章描述 Java 语言的一些特性时，也会用到流程图。

1.2 算法——力求准确

对于上一节提到的烤肉菜谱，计算机是无法执行的，因为各条指令太松散，太不明确了。让我们来考虑另一个例子，一个对数字而不是食物进行操作的例子。您用 1 美元买一些东西，会找回一些零钱(分币和角币，即 penny 和 dime)。问题是在保证找回的零钱数正确的前提下，尽量减少分币的个数。日常生活中，大多数人都会不加思考地完成这件事，但我们如何准确地描述这个算法呢？

解决这个问题时，采用一个具体的例子会有所帮助。假设要支付 77 美分，而您给了 1 美元，就可以轻松算出 1 美元减去 77 美分应该是 23 美分，要想保证分币和角币最少，应该找回 2 个角币，3 个分币。角币的个数是 23 除以 10 后舍去小数的整数。分币的个数是 23 除以 10 的余数。找零钱算法的步骤如下：

找零钱算法

1. 假设价格写在一个标签名为“price”的框里。
2. 从 100 中减去“price”的值，将结果放入一个标签名为“change”的框。
3. 用 change 除以 10，舍去余数，将结果放入一个标签名为“dimes”的框。
4. 取“change”除以 10 的余数，将结果放入一个标签名为“pennies”的框。
5. 打印出“dimes”框和“penny”框内的值。
6. 终止。

这个算法用到了 4 个框，分别是“price”，“change”，“dimes”和“pennies”。我们用给定的值执行程序，假设价格是 77 美分，从第一条指令开始，在指令执行的不同阶段，4 个框的内容如下表所示。

框名	第 1 步	第 2 步	第 3 步	第 4 步	第 5 步
price	77	77	77	77	77
change		23	23	23	23
dimes			2	2	2
pennies				3	3

执行第 1 步时，将 77 放入“price”框。在第 2 步末尾，100 减 23 的结果 77 就放入“change”框中。算法的每一步都执行了一部分计算。到了第 5 步，每个正确值都在相应的框里，并且被打印了出来。仔细研究这个例子，直至您确定这个算法在 price 小于 1.00 美元的情况下均能正常运行。一个好办法是按菜谱中的描述执行几步。这种方法称为手工模拟(hand simulation)或基准测试(bench testing)。它也是用来查找算法或程序错误的一种好方法。在计

算机专业里，这些错误叫做 bug。找到错误并修改错误的过程称为排错或调试(debugging)。

我们像代理一样执行找零钱算法，机械化地按照指令列表执行操作，由代理来执行指令集的过程称为计算(computation)。通常情况下，代理就是计算机。此时，指令集合就是计算机程序。本书后文中，除非特殊指明，所提到的程序都是指计算机程序。

找零钱算法中，有几个重要特征是所有算法都具有的：

- 指令序列最终会终止。
- 指令简练，每条指令不会含混不清，它只有一个解释。
- 指令易于执行。每条指令都负责执行一定的功能，而且花费的时间是有限的，这样的指令称为有效的指令。
- 有输入和输出。每个算法都有一个和多个输出(答案)。这些输出取决于输入。找零钱算法的输入是所买东西的价格，输出是分币和角币的个数。

虽然我们对找零钱算法的描述很简单，但毕竟不是按正式的编程语言书写的，这种非正式的表示叫做伪代码(pseudocode)。而真实代码是指适合计算机使用的代码。我们可以用伪代码描述算法，这样就不用解释计算机所需要的细节。

算法(algorithm)这个词有很长的演化历史，它是 19 世纪一位著名的阿拉伯数学家 Abu Jafar Muhammed Musa Al-Khwarizmi 提出的。后来，它与一些算术过程联系在一起，尤其和计算两个整数的最大公因数的 Euclid 算法联系起来。后来，随着计算机的发展，这个词有了更简洁的含义，它把一台实际的或抽象的计算机定义成执行代理——任何由计算机完成的可终止的计算都是算法，任何算法都可以编为程序。

1.3 用 Java 实现我们的算法

这一节，我们将用 Java 编程语言来实现前面的找零钱算法。此时无需过于关注 Java 程序的细节，我们会在后两章详细讲解，2.10.1 节将再次讨论这个例子。现在，只需比较下面的 Java 程序和前面的非正式算法。您不但要明确地表达菜谱使之算法化，而且要用计算机语言表达出来。

```
// MakeChange.java - change in dimes and pennies
import tio.*; // use the package tio

class MakeChange {
  public static void main (String[] args) {
    int price, change, dimes, pennies;
    System.out.println("type price (0 to 100):");
    price = Console.in.readInt();
    change = 100 - price;      //how much change
    dimes = change / 10;       //number of dimes
    pennies = change % 10;     //number of pennies
    System.out.print("The change is :");
    System.out.print(dimes);
    System.out.print(" dimes ");
    System.out.print(pennies);
    System.out.print(" pennies.\n");
  }
}
```

MakeChange 程序解析

- `import tio.*; // use the package tio`
  包(package)是一个可以使用的库(library)或一个在此之前写的程序部件。这行代码告诉 Java 编译器，程序 MakeChange 使用了包 tio 的信息。我们特意为本书开发了这个包(源代码见附录 C)，以帮助您简化键盘输入，利用它，可直接写 Console.in.readInt()。我们稍后会解释这个函数。
- `int price, change, dimes, pennies;`
  这行代码声明了 4 个变量。这些变量用来存储待操作的值。
- `System.out.println("type price (0 to 100):");`
  这一行用于提示用户输入价格。程序希望用户下一步做什么时，都应该在屏幕上显示提示信息，提醒用户怎么做，程序执行时，双引号中的内容会出现在用户屏幕上。
- `price = Console.in.readInt();`
  Console.in.readInt()用于获得键盘输入，读入的值存放在变量 price 里。符号“=”叫做赋值运算符，这行可解释为“price 被赋值为从 Console.in.readInt()读入的值”。此时，必须输入一个整数价格，例如输入 77，并按回车。
- `change = 100 - price;  //how much change`
  这一行代码计算要找回多少零钱。
- `dimes = change / 10;  //number of dimes`
  `pennies = change % 10;  //number of pennies`
  角币的个数是“change”除以 10 的整数部分，符号“/”位于两个整数之间时，它表示计算除法的整数部分。分币的个数是“change”除以 10 的余数部分，符号“%”是 Java 中的求余运算符，也就是求模运算符，所以如果“change”是 23，23/10 的结果值为 2，23%10 的结果值为 3。
- ```
  System.out.print("The change is :");
  System.out.print(dimes);
  System.out.print(" dimes ");
  System.out.print(pennies);
  System.out.print(" pennies.\n");
  ```
  本例中，System.out.print()语句会将括号内的值打印在计算机屏幕上。第一行只是打印双引号所包含的字符，第二行将 dimes 的值转化成数字的序列，并把这些数字打印出来，其他打印语句类似。如果输入值是 77，输出就是：
  ```
  The change is :2 dimes 3 pennies
  ```
  最后一条打印语句的“\n”表示要向控制台发送一个“换行符”(newline)，终止输出。

1.4 为什么要学 Java

编程语言种类繁多，但最有用的语言应当既适合机器阅读又适合人阅读，本书中，我们选择了这样一种语言——Java。
  ```

Java 是一种相对较新的编程语言，它产生于 1995 年。本书要描述的语言应能使算法易于理解、设计、分析并实现为计算机程序。下面这段话摘自 Java 问世时发表的原始论文：

Robert A. Heinlein 在他的科幻小说 *The Rolling Stone* 中提到：

每种技术都会经历 3 个阶段：首先是一个粗糙、简单、令人不满意的小玩艺，然后是一个庞大而又复杂的组，这些组由那些设计用于弥补第一个版本缺点的部分组成，从而通过极其复杂的组成获得一些令人满意的性能，第三个阶段就有了一个恰当的设计版本。

Heinlein 的评论很好地描述了很多编程语言的演化过程。Java 在编程语言演化史中提出一种新视角——创建了一个小而简单的语言，但仍然足以用于软件程序的开发。虽然 Java 和 C、C++十分相似，但是 Java 通过从其先辈中移除系统相关特性，得以大大简化。

我们赞同 Java 创建者所说的，Java 在很大程度上是一种小巧、简单但又广泛的编程语言。同样地，Java 既是一个适合开发实际程序的优秀编程语言，又是一个优秀的首选编程语言。

Java 最成功的一点是，它适合开发分布在因特网上的程序。这些程序称为 applet，可在 Internet 浏览器(例如 Netscape Navigator 和 Internet Explorer)中执行。

1.5 网络计算和 Web

大部分现代化计算都是在联网环境下完成的，也就是说，计算机与一个网络中的其他计算机彼此相连。这种连接使计算机彼此之间可以交换信息。Java 是以网络计算为日常环境开发的，它有很多支持网络的特性和程序库。早期的语言和系统将计算机视为一种可单独完成大量计算的仪器，计算结果通常输出到屏幕或打印机。通过网络，不同计算机可彼此连接，并动态传递结果，以协同完成大规模的计算，基于此，Java 正成为网络计算的重要语言。

最大的网络是名为因特网(Internet)的全球性网络。通过使用因特网，欧洲粒子物理实验室(the European Particle Physics Laboratory，CERN)的研究员开发了一种方法，能通过一种名为超文本标记语言(hyper-text markup-language，HTML)的格式化语言来共享信息。计算机之间利用一种名为超文本传输协议(hyper-text transfer protocol，HTTP)来交换 HTML 文档。协议(protocol)如同计算机用于与其他计算机“交谈”的语言。HTML 使得一个电子文档可以连接到另一台电脑上的电子文档。用于查看 HTML 文档，并跟踪链接转移到其他文档的程序称为浏览器(browser)。单击鼠标按钮，用户就可以从一台计算机上的文档转移到另一台计算机的相关文档。因为有了这些链接，不同文档所组成的网络就被称为万维网(World Wide Web)，一般简称 Web。如今，很多人将因特网(Internet)和万维网(Web)混为一谈，但实际上，万维网只是因特网技术的一个应用。

Java 程序称为 applet，用户可跟踪 Web 上的一个链接自动载入并开始执行这类程序。这种可以将 Java 程序嵌入 HTML 文档的能力是 Java 取得成功的重要因素。自从有了 Java applet，Web 文档就不再是静态的文本、图像或视频片断。Web 文档可以提供任何程序的所有交互，有关的图形用户界面和 applet，我们将在第 8 章中讨论。