高等教育立体化精品教材

Java程序设计基础

谢　鹏　金　伟　胡　悦　主　编
胡远伟　周满满　朱丹蕾　参　编

汕頭大學出版社

图书在版编目（CIP）数据

Java 程序设计基础 / 谢鹏，金伟，胡悦主编．--汕头：汕头大学出版社，2020.8
ISBN 978-7-5658-4098-2

Ⅰ.①J… Ⅱ.①谢… ②金… ③胡… Ⅲ.①JAVA 语言－程序设计－高等学校－教材 Ⅳ.①TP312.8

中国版本图书馆 CIP 数据核字（2020）第 145330 号

Java 程序设计基础　　Java CHENGXU SHEJI JICHU

主　　编： 谢　鹏　金　伟　胡　悦
责任编辑： 汪艳蕾
责任技编： 黄东生
封面设计： 易　帅
出版发行： 汕头大学出版社
广东省汕头市大学路 243 号汕头大学校园内　邮政编码：515063
电　　话： 0754-82904613
印　　刷： 天津市蓟县宏图印务有限公司
开　　本： 787mm×1092mm　1/16
印　　张： 13
字　　数： 277 千字
版　　次： 2020 年 8 月第 1 版
印　　次： 2020 年 8 月第 1 次印刷
定　　价： 38.00 元
ISBN 978-7-5658-4098-2

版权所有，翻版必究
如发现印装质量问题，请与承印厂联系退换

PREFACE 前言

Java 具有面向对象、平台无关、可靠稳定、分布式及多线程等特点，是近年来较为流行的程序设计语言。目前，国内 Java 程序开发人员的需求缺口巨大。

本书主要介绍了 Java 程序设计语言的基本语法和特点，共分为 14 个章节。各章内容如下：

第 1 章不仅介绍了 Java 的发展历史、特点和语言规范，而且讲述了 Java 虚拟机的特点及使用主体。

第 2 章主要介绍了 Java 的开发环境，包括开发工具的介绍和 JDK 开发环境的配置。

第 3 章主要介绍了数据类型、关键字、标识符、常量、变量及运算符和表达式的使用方法。

第 4 章主要介绍了流程控制，包括顺序结构、分支结构和循环结构等。

第 5 章主要介绍了数组与字符串的基本概念和使用方法。

第 6 章主要介绍了面向对象程序设计的基本思想、类与对象的定义及使用方法、访问控制符、包、成员变量、成员方法、构造方法及 this 关键字。

第 7 章主要介绍了类的继承、方法的重载、多态、抽象类与接口等相关知识。

第 8 章主要介绍了异常的基本概念和处理机制。

第 9 章主要介绍了输入/输出方法流的基本概念，以及 InputStream、OutputStream、Reader、Writer 等类的使用方法和文件的处理。

第 10 章主要介绍了多线程的基本概念和使用方法。

第 11 章主要介绍了 AWT 和 Swing 的基本概念、按钮的类型、文本输入框、密码输入框及布局管理器等。

第 12 章主要介绍了事件的基本概念、分类及处理模型等内容。

第 13 章主要介绍了数据库的分类及特点、关系型数据库、SQL 语句及 JDBC 的使用方法。

第 14 章主要介绍了网络编程的基础知识及 Socket 的使用方法。

本书由谢鹏、金伟和胡悦担任主编。胡远伟、周满满和朱丹蕾等教师参与了本书资料整理和校对工作，在此一并致谢。

由于编者水平有限，书中疏漏之处敬请读者指正。

编　者

CONTENTS 目录

目录

第 14 章 Java 网络编程

参考文献

第 1 章　Java 语言概述

1.1 Java 语言的诞生与发展

人们使用文字处理程序编写文档，使用 Web 浏览器探索互联网，使用电子邮件程序收发电子邮件。这些文字处理程序、Web 浏览器及电子邮件程序都是在计算机上运行的软件产品。软件是用程序设计语言开发出来的。程序设计语言有很多种，为什么要选择学习Java程序设计语言呢？答案就是 Java 能够让用户开发运行于 Internet 的服务器、台式计算机和小型手持设备上的应用程序。互联网会对计算技术的未来产生深远的影响，而 Java 语言肯定会在未来计算技术中占很大的比例。所以，Java 是一种真正的互联网程序设计语言。

Java 是 James Gosling 领导开发的。Java 最初称为“Oak”（橡树），是 1991 年为消费类电子产品的嵌入式芯片设计的程序设计语言。1995 年更名为“Java”，并重新设计用于开发 Internet 应用程序。

1996 年 1 月，Sun 公司发布了 Java 的第一个开发工具包（JDK 1.0），这是 Java 发展过程中的里程碑，标志着 Java 已成为一种独立的开发工具。同年 9 月，约 8.3 万个网页应用了 Java 技术来进行制作。

1998 年 12 月 8 日，第二代 Java 平台的企业版 J2EE 发布。1999 年 6 月，Sun 公司发布了第二代 Java 平台（简称 Java 2）。Java 2 平台的发布是 Java 发展过程中最重要的里程碑，标志着 Java 的应用开始普及。

2009 年，Oracle 公司收购了 Sun 公司。2010 年，Java 的共同创始人之一詹姆斯·高斯林从 Oracle 公司辞职。2011 年，Oracle 公司举行了全球性活动，以庆祝 Java 7 的推出，随后 Java 7 正式发布。2014 年，Oracle 公司发布了 Java 8 正式版。

近年来，Java 变得非常流行。Java 的快速发展和被广泛接受都应归功于它的设计特性，特别是它的承诺——一旦编写了一个程序，在任何地方都可以运行。就像 Sun 公司声称的那样，Java 是简单（simple）的、面向对象（object-oriented）的、分布式（distributed）的、解释执行（interpreted）的、健壮（robust）的、安全（secure）的、结构中立（architecture-neutral）的、可移植（portable）的、高性能（high performance）的、多线程(multithreaded)的和动态（dynamic）的程序设计语言。

Java 是功能完善的通用程序设计语言，可以用来开发可靠的、要求严格的应用程序。现在，它不仅用于 Web 程序，而且用于在服务器、台式计算机和移动设备上开发跨平台

的独立应用程序。

1.2 Java 语言的特点

Java 程序可以在 Web 浏览器中运行。这种能在 Web 浏览器中运行的 Java 程序称为Java应用小程序（Applet）。Applet 使用目前流行的图形用户界面与网络用户交互，处理用户的需求，这些界面中包括按钮、文本字段、文本域、单选按钮等。Applet 使网页更加灵活、生动和易于交互。

1 简单性

Java 看起来设计得很像C++，但是为了使语言小且容易掌握，设计者们把C++中许多可用的特征去掉了，这些特征是一般程序员很少使用的。例如，Java 不支持 goto 语句，代之以 break 和 continue 语句及异常处理。Java 还剔除了C++的操作符过载（overload）和多继承特征，并且不使用主文件，免去了预处理程序。因为 Java 没有结构体，数组和字符串都是对象，所以不需要指针。Java 能够自动处理对象的引用和间接引用，实现自动的无用单元收集，使用户不必为存储管理问题烦恼，能将更多的时间和精力花在程序设计上。

2 面向对象

Java 是一个面向对象的语言。对程序员来说，这意味着要注意应用中的数据和操纵数据的方法（method），而不是严格的过程性思考。在一个面向对象的系统中，类（class）是数据和操作数据的方法的集合。数据和方法一起描述对象（object）的状态和行为。每一个对象均是其状态和行为的封装。类是按一定体系和层次安排的，使得子类可以从超类继承行为。在这个类层次体系中有一个根类，它是具有一般行为的类。Java 程序是用类来组织的。

Java 还包括一个类的扩展集合，分别组成各种程序包（package），用户可以在自己的程序中使用。例如，Java 提供产生图形用户界面部件的类（java.awt 包）、处理输入/输出的类（java.io 包）和支持网络功能的类（java.net 包）。

3 分布性

Java 被设计为支持在网络上应用，它是分布式语言。Java 支持各种层次的网络连接，Socket 类可靠地支持流网络连接，所以用户可以产生分布式的客户机和服务器。

4 编译和解释性

Java 编译程序生成字节码（bytecode），而不是通常的机器码。Java 字节码对体系结构中的目标文件格式提供支持，代码设计成可有效地传送程序到多个平台。Java 程序可以在任何实现了 Java 解释程序和运行系统（run-time system）的环境中运行。

在一个解释性的环境中，程序开发的标准“链接”阶段很大程度上消失了。如果说 Java 还有一个链接阶段，也只是把新类装进环境的过程，它是增量式的、轻量级的过程。因此，Java 支持快速原型和容易试验，它将导致快速程序开发。这是一个与传统的耗时的

“编译、链接和测试”形成鲜明对比的精巧的开发过程。

5 稳健性

Java原来是用于编写消费类家用电子产品软件的语言，所以它被设计成高可靠和稳健的软件。Java消除了某些编程错误，使得用它写可靠软件相当容易。

Java是一个强类型语言，它支持扩展编译时检查潜在类型不匹配问题的功能。Java要求以显式的方式声明，它不支持C语言风格的隐式声明。这些严格的要求可以保证编译程序能捕捉调用错误，这使程序更可靠。

可靠性方面最重要的增强之一是Java的存储模型。Java不支持指针，它消除重写存储和讹误数据的可能性。类似地，Java自动进行“无用单元收集”，预防存储泄漏和其他有关动态存储分配，以及消除分配的有害错误。Java解释程序也执行许多运行时的检查，诸如验证所有数组和字符串访问是否在界限之内。

异常处理是Java中使程序更稳健的另一个特征。异常是某种类似于错误的异常条件出现的信号。使用try/catch/finally语句，程序员可以找到出错的处理代码，这简化了出错处理和恢复的过程。

6 安全性

Java的存储分配模型是它防御恶意代码的主要方法之一。Java没有指针，所以程序员不能得到隐蔽起来的内幕和伪造指针去指向存储器。更重要的是，Java编译程序不处理存储安排决策，所以程序员不能通过查看声明去猜测类的实际存储安排。编译的Java代码中的存储引用在运行时由Java解释程序决定实际存储地址。

Java运行系统使用字节码验证过程来保证装载到网络上的代码不违背任何Java语法限制。该安全机制部分包括类如何从网上装载。例如，装载的类是放在分开的命名空间，而不是局部类，以预防恶意的小应用程序用它自己的版本来代替标准Java类。

7 结构中立

Java编译器生成的目标文件格式是结构中立的。只要存在Java运行环境，编译后的代码就能在不同的CPU上运行。因为Java编译器生成的字节码指令与特定的计算机结构无关。这样的设计使字节码指令既能够在任何机器上进行解释，也能够实时编译成本地机器码。

8 可移植性

Java使语言声明不依赖于实现的方面。例如，Java显式说明每个基本数据类型的大小和它的运算行为（这些数据类型由Java语法描述）。

Java环境本身对新的硬件平台和操作系统来说是可移植的。Java编译程序也用Java语言编写，而Java运行系统用ANSIC语言编写。

9 高性能

Java是一种先编译后解释的语言，所以它的执行速度不如全编译性语言快。但是有些情况下性能是很要紧的，为了支持这些情况，Java设计者制作了“及时”编译程序，它能在运行时把Java字节码翻译成特定CPU（中央处理器）的机器代码，这也就是实现了全编译。

Java字节码格式在设计时考虑到这些“及时”编译程序的需要，所以生成机器代码的过程相当简单，它能产生相当好的代码。

10 多线程性

Java是多线程语言，它提供支持多线程的执行（也称轻量级进程），能处理不同任务，使具有线程的程序设计相当容易。Java的lang包提供了一个Thread类，它支持开始线程、运行线程、停止线程和检查线程状态的方法。

Java的线程支持也包括一组同步原语。这些原语是基于监督程序和条件变量的，由C. A. R. Haore开发的广泛使用的同步化方案。通过关键字“synchronized”，程序员可以说明某些方法在一个类中不能并发地运行。这些方法在监督程序的控制之下，确保变量维持在一个一致的状态。

11 动态性

Java被设计成可适应于变化的环境，它是一个动态的语言。例如，Java中的类是根据需要载入的，甚至有些是通过网络获取的。

1.3 Java语言规范

计算机语言有严格的使用规范，如果编写程序时没有遵循这些规范，计算机就不能理解程序。Java语言规范和Java API定义了Java的标准。

Java语言规范是对语言的技术定义，它包括Java的语法和语义。

包名：全小写的名词，中间可由点分隔，如java. awt. event。

类名：首字母大写，由多个单词合成时，每个单词首字母大写，如HelloWorldApp。

接口名：规则同类名，如Collection。

方法名：由多个单词合成时，第一个单词通常为动词，首字母小写，中间的每个单词的首字母都要大写，如balanceAccount、isButtonPressed。

变量名：全小写，一般为名词，如length。

常量名：基本数据类型的常量名为全大写，如果由多个单词构成，可以用下划线隔开，如YEAR、WEEK _ OF _ MONTH。

对象类型的常量，则是大小写混合，由大写字母把单词隔开。

Java有3个版本：Java标准版（Java Standard Edition，Java SE）、Java企业版（Java Enterprise Edition，Java EE）、Java微型版（Java Micro Edition，Java ME）。Java SE可以用来开发客户端独立的应用程序或Applet。Java EE用来开发服务器端的应用程序，如Java Servlet和Java Server Pages。Java ME可以用来开发移动设备的应用程序，如手机App等。本书使用Java SE来介绍Java程序设计。

1.4 Java 虚拟机

1.4.1 简介

虚拟机是一种抽象化的计算机，通过在实际的计算机上模拟各种计算机功能来实现。Java 虚拟机（Java Virtual Machine，JVM）有自己完善的硬体架构，如处理器、堆栈、寄存器等，还具有相应的指令系统。

Java 虚拟机是运行所有 Java 程序的抽象计算机，是 Java 语言的运行环境，它是 Java 最具吸引力的特性之一。

Java 虚拟机是一种用于计算机设备的规范，可用不同的方式（软件或硬件）加以实现。编译虚拟机的指令集与编译微处理器的指令集非常类似。

Java 虚拟机是可运行 Java 代码的假想计算机。只要根据 Java 虚拟机规范描述将解释器移植到特定的计算机上，就能保证经过编译的任何 Java 代码都能够在该系统上运行。

1.4.2 Java 虚拟机的特点

Java 的一个非常重要的特点就是与平台的无关性，而使用 Java 虚拟机是实现这一特点的关键。一般的高级语言如果要在不同的平台上运行，至少需要编译成不同的目标代码。而引入 Java 虚拟机后，Java 在不同平台上运行时不需要重新编译。Java 虚拟机屏蔽了与具体平台相关的信息，使 Java 编译程序只需生成在 Java 虚拟机上运行的目标代码（字节码），就可以在多种平台上不加修改地运行。Java 虚拟机在执行字节码时，把字节码解释成具体平台上的机器指令执行。

1.4.3 Java 虚拟机的使用主体

Java 虚拟机是 Java 底层实现的基础。这有助于理解 Java 的一些性质，也有助于使用 Java。对于要在特定平台上实现 Java 虚拟机的软件人员、Java 编译器作者及要用硬件芯片实现 Java 虚拟机的人来说，必须深刻理解 Java 虚拟机的规范。另外，如果你想扩展 Java，或者把其他语言编写的程序编译成 Java 的字节码，你也需要深入了解 Java 虚拟机。

1.4.4 体系结构

Java 虚拟机由 5 个部分组成：一组指令集、一组寄存器、一个栈、一个无用单元收集堆、一个方法区域。这 5 个部分是 Java 虚拟机的逻辑成分，不依赖任何实现技术或组织方式，但它们的功能必须在真实机器上以某种方式实现。

1 指令集

Java 虚拟机支持大约 248 个字节码。每个字节码执行一种基本的 CPU 运算。例如，把一个整数加到寄存器、子程序转移等。Java 指令集相当于 Java 程序的汇编语言。

Java 指令集中的指令包含一个单字节的操作符，用于指定要执行的操作；还有 0 个或多个操作数，提供操作所需的参数或数据。许多指令没有操作数，仅由一个单字节的操作符构成。

Java 虚拟机的内层循环的执行过程如下：

```
do{
    取一个操作符字节；
    根据操作符的值执行一个动作；
}while(程序未结束)
```

Java 指令集的简单性使 Java 虚拟机执行的过程十分简单，从而有利于提高执行效率。指令中操作数的数量和大小是由操作符决定的。如果操作数比一个字节大，那么它存储的顺序是高位字节优先。例如，一个 16 位的参数存放时占用两个字节，其值为：第一个字节×256＋第二个字节，字节码指令流一般只是字节对齐的。指令 tabltch 和 lookup 是例外，在这两条指令内部强制要求 4 字节边界对齐。

2 寄存器

Java 虚拟机的寄存器用于保存机器的运行状态，与 CPU 中的某些专用寄存器类似。

Java 虚拟机的寄存器有以下 4 种。

- pc：Java 程序计数器。
- optop：指向操作数栈顶端的指针。
- frame：指向当前执行方法的执行环境的指针。
- vars：指向当前执行方法的局部变量区中第一个变量的指针。

3 栈

Java 虚拟机的栈有 3 个区域：局部变量区、运行环境区、操作数区。

局部变量区中每个 Java 方法使用一个固定大小的局部变量集。它们按照与 vars 寄存器的字偏移量来寻址。局部变量都是 32 位的。长整数和双精度浮点数占据两个局部变量的空间，却按照第一个局部变量的索引来寻址。Java 虚拟机规范并不要求局部变量中 64 位的值是 64 位对齐的。Java 虚拟机提供了把局部变量中的值装载到操作数栈的指令，也提供了把操作数栈中的值写入局部变量的指令。

运行环境区在运行环境中包含的信息用于动态链接、正常的方法返回及异常传播。

（1）动态链接。

运行环境包括对指向当前类和当前方法的解释器符号表的指针，用于支持方法代码的动态链接。方法的 class 文件代码在引用要调用的方法和要访问的变量时使用符号。动态链接把符号形式的方法调用翻译成实际方法调用，装载必要的类以解释还没有定义的符号，并把变量访问翻译成与这些变量运行时的存储结构相应的偏移地址。动态链接方法和变量使方法中使用的其他类的变化不会影响到本程序的代码。

（2）正常的方法返回。

如果当前方法正常结束了，在执行一条具有正确类型的返回指令时，调用的方法会得到一个返回值。运行环境在正常返回的情况下用于恢复调用者的寄存器，并把调用者的程

序计数器增加一个恰当的数值，以跳过已执行过的方法调用指令，然后在调用者的运行环境中继续执行下去。

（3）异常传播。

异常情况在 Java 中称为 Error（错误）或 Exception（异常），是 Throwable 类的子类，在程序中异常产生的原因有：动态链接错误，如无法找到所需的 class 文件；运行时错误，如对一个空指针的引用；程序使用了 throw 语句。

当异常发生时，Java 虚拟机采取如下措施。

①检查与当前方法相联系的 catch 子句表。每个 catch 子句包括其有效指令范围、能够处理的异常类型，以及处理异常的代码块地址。

②与异常相匹配的 catch 子句应该符合下面的条件：造成异常的指令在其指令范围之内，发生的异常类型是其能处理的异常类型的子类型。如果找到了匹配的 catch 子句，那么系统转移到指定的异常处理块执行；如果没有找到异常处理块，重复寻找匹配的 catch 子句的过程，直到当前方法嵌套的所有 catch 子句都被检查过。

③由于 Java 虚拟机从第一个匹配的 catch 子句处继续执行，所以 catch 子句表的顺序是很重要的。因为 Java 代码是结构化的，因此总可以把某个方法所有的异常处理块都按序排列到一个表中，对任意可能的程序计数器的值都可以用线性的顺序找到合适的异常处理块，以处理在该程序计数器值下发生的异常情况。

④如果找不到匹配的 catch 子句，那么当前方法得到一个“未截获异常”的结果并返回到当前方法的调用者，好像异常刚刚在其调用者中发生一样。如果在调用者中仍然没有找到相应的异常处理块，那么这种错误传播将继续下去。如果错误被传播到最顶层，那么系统将调用一个默认的异常处理块。

在操作数区中，机器指令只从操作数栈中取操作数，对它们进行操作，并把结果返回到栈中。选择栈结构的原因：在只有少量寄存器或非通用寄存器的机器上，也能够高效地模拟 Java 虚拟机的行为。操作数栈是 32 位的，它用于给方法传递参数，并从方法接收结果，也用于支持操作的参数，并保存操作的结果。例如，iadd 指令将两个整数相加。相加的两个整数应该是操作数栈顶的两个字。这两个字是由先前的指令压进堆栈的。这两个整数将从堆栈弹出、相加，并把结果压回到操作数栈中。

每个基本数据类型都有专门的指令对它们进行必需的操作。每个操作数在栈中需要一个存储位置，除了 long 和 double 类型，它们需要两个位置。操作数只能被适用于其类型的操作符所操作。例如，压入两个 int 类型的数，如果把它们当作一个 long 类型的数，则这是非法的。

4 方法区

方法区与传统语言中的编译后代码或 UNIX 进程中的正文段类似。它保存方法代码（编译后的 Java 代码）和符号表。每个类文件包含一个 Java 类或一个 Java 界面编译后的代码。可以说，类文件是 Java 语言的执行代码文件。为了保证类文件的平台无关性，Java 虚拟机规范中对类文件的格式做了详细说明。其具体细节请参考 Oracle 公司的 Java 虚拟机规范。

本章小结

本章主要讲述 Java 的诞生与发展历程，以及 Java 的基础知识，是学习 Java 的入门章节。通过本章的学习，读者应掌握 Java 语言的规范与特点、结构与编写方法，为后续深入学习 Java 语言打下基础。