全国计算机等级考试二级教程

——Java语言程序设计

（2017年版）

教育部考试中心

高等教育出版社

全国计算机等级考试二级教程

——Java 语言程序设计（2017 年版）

Quanguo Jisuanji Dengji Kaoshi Erji Jiaocheng
——Java Yuyan Chengxu Sheji

教育部考试中心

主编　柳西玲
参编　许　斌　郎　波　金铁鹰

高等教育出版社·北京

内容提要

本书按照《全国计算机等级考试二级Java语言程序设计考试大纲（2013年版）》的要求编写，内容包括：Java语言概论，基本数据类型，运算符与表达式，流程控制，Java的继承、多态、高级类特性和数组，异常和断言，输入输出及文件操作，线程，编写图形用户界面，Applet程序设计，集合与泛型，Java编程风格，应用开发工具与安装使用，等等。

本书是教育部考试中心指定教材，是考生参加全国计算机等级考试二级Java语言程序设计的必备参考书，也可作为学习Java编程的参考书。

图书在版编目（CIP）数据

全国计算机等级考试二级教程：2017年版. Java语言程序设计 / 教育部考试中心编. --北京：高等教育出版社，2016.10

ISBN 978-7-04-046555-6

Ⅰ.①全… Ⅱ.①教… Ⅲ.①电子计算机-水平考试-教材②JAVA语言-程序设计-水平考试-教材 Ⅳ.①TP3

中国版本图书馆CIP数据核字（2016）第235617号

策划编辑 何新权　　责任编辑 何新权　　封面设计 王 琰　　版式设计 杜微言

责任校对 刘春萍　　责任印制 耿 轩

出版发行 高等教育出版社
社　　址 北京市西城区德外大街4号
邮政编码 100120
印　　刷 北京市大天乐投资管理有限公司
开　　本 787mm×1092mm 1/16
印　　张 20.5
字　　数 500千字
购书热线 010-58581118
咨询电话 400-810-0598

网　　址 http://www.hep.edu.cn
　　http://www.hep.com.cn
网上订购 http://www.hepmall.com.cn
　　http://www.hepmall.com
　　http://www.hepmall.cn
版　　次 2016年10月第1版
印　　次 2016年10月第1次印刷
定　　价 43.00元

本书如有缺页、倒页、脱页等质量问题，请到所购图书销售部门联系调换

物 料 号 46555-00

积极发展全国计算机等级考试 为培养计算机应用专门人才、促进信息 产业发展作出贡献

（序）

中国科协副主席　中国系统仿真学会理事长
第五届全国计算机等级考试委员会主任委员
赵沁平

当今，人类正在步入一个以智力资源的占有和配置，知识生产、分配和使用为最重要因素的知识经济时代，也就是小平同志提出的“科学技术是第一生产力”的时代。世界各国的竞争已成为以经济为基础、以科技（特别是高科技）为先导的综合国力的竞争。在高科技中，信息科学技术是知识高度密集、学科高度综合、具有科学与技术融合特征的学科。它直接渗透到经济、文化和社会的各个领域，迅速改变着人们的工作、生活和社会的结构，是当代发展知识经济的支柱之一。

在信息科学技术中，计算机硬件及通信设施是载体，计算机软件是核心。软件是人类知识的固化，是知识经济的基本表征，软件已成为信息时代的新型“物理设施”。人类抽象的经验、知识正逐步由软件予以精确地体现。在信息时代，软件是信息化的核心，国民经济和国防建设、社会发展、人民生活都离不开软件，软件无处不在。软件产业是增长快速的朝阳产业，是具有高附加值、高投入高产出、无污染、低能耗的绿色产业。软件产业的发展将推动知识经济的进程，促进从注重量的增长向注重质的提高方向发展。软件产业是关系到国家经济安全和文化安全，体现国家综合实力，决定21世纪国际竞争地位的战略性产业。

为了适应知识经济发展的需要，大力促进信息产业的发展，需要在全民中普及计算机的基本知识，培养一批又一批能熟练运用计算机和软件技术的各行各业的应用型人才。

1994年，国家教委（现教育部）推出了全国计算机等级考试，这是一种专门评价应试人员对计算机软硬件实际掌握能力的考试。它不限制报考人员的学历和年龄，从而为培养各行业计算机应用人才开辟了一条广阔的道路。

1994年是推出全国计算机等级考试的第一年，当年参加考试的有1万余人，2012年报考人数已达549万人。截至2013年年底，全国计算机等级考试共开考38次，考生人数累计达5 422万人，有2 067万人获得了各级计算机等级证书。

事实说明，鼓励社会各阶层人士通过各种途径掌握计算机应用技术，并通过等级考试对他们的能力予以科学、公正、权威性的认证，是一种比较好的、有效的计算机应用人才培养途径，符合我国的具体国情。等级考试同时也为用人部门录用和考核人员提供了一种测评手段。从有关公司对等级考试所作的社会抽样调查结果看，不论是管理人员还是应试人员，对该项考试的内容和形式都给予了充分肯定。

计算机技术日新月异。全国计算机等级考试大纲顺应技术发展和社会需求的变化,从2010年开始对新版考试大纲进行调研和修订,在考试体系、考试内容、考试形式等方面都做了较大调整,希望等级考试更能反映当前计算机技术的应用实际,使培养计算机应用人才的工作更健康地向前发展。

全国计算机等级考试取得了良好的效果,这有赖于各有关单位专家在等级考试的大纲编写、试题设计、阅卷评分及效果分析等多项工作中付出的大量心血和辛勤劳动,他们为这项工作的开展作出了重要的贡献。我们在此向他们表示衷心的感谢!

我们相信,在21世纪知识经济和加快发展信息产业的形势下,在教育部考试中心的精心组织领导下,在全国各有关专家的大力配合下,全国计算机等级考试一定会以“激励引导成才,科学评价用才,服务社会选材”为目标,服务考生和社会,为我国培养计算机应用专门人才的事业作出更大的贡献。

前　言

为了促进我国计算机知识的普及，提高全社会的计算机应用水平，适应国民经济信息化的需要，国家教委考试中心（现教育部考试中心）于1994年推出了全国计算机等级考试，该项考试为社会提供了一个统一、公正、科学的考核标准，深受社会各界的欢迎。

为适应考试需要，教育部考试中心最新制订了《全国计算机等级考试二级Java语言程序设计考试大纲（2013年版）》，并对教材进行了修订。本书以JDK 1.6版本为基础，介绍Java语言编程基础知识以及应用开发技术，内容包括：Java体系结构、基本数据类型、流程控制语句、类、数组和字符串操作、输入/输出及文件操作、图形用户界面编写、线程、Applet程序设计、集合与泛型以及应用开发工具和安装使用等。书中的实例均通过编译并可直接在JDK 1.6上运行，各章后均配有习题及参考答案，供考生练习和使用，是考生复习备考的必备教材。

本书由教育部考试中心组织编写并审定。第1、2、6、11章由柳西玲编写，第3、4、9章由许斌编写，第8、10、12章由郎波编写，第5、7章由金铁鹰编写。全书由柳西玲统稿。

由于编写时间仓促，书中难免存在不妥和错误，望读者给予指正并提出宝贵意见，以便修订时改进。

编　者

目　录

第 1 章　Java 语言概论 …… 1
1.1　Java 语言简介 …… 1
1.1.1　Java 语言的由来 …… 1
1.1.2　Java 语言的目标 …… 1
1.1.3　Java 语言实现机制 …… 2
1.2　Java 语言面向对象编程 …… 3
1.2.1　面向对象编程的基本概念 …… 3
1.2.2　类与包 …… 4
1.2.3　对象创建、初始化、使用和删除 …… 11
1.2.4　Java 源程序结构 …… 17
1.2.5　Java 程序编写及运行的过程 …… 18
习题 …… 22
第 2 章　基本数据类型 …… 23
2.1　概述 …… 23
2.1.1　标识符 …… 23
2.1.2　关键字 …… 24
2.1.3　常量 …… 24
2.1.4　变量 …… 25
2.2　基本数据类型 …… 25
2.2.1　整型数据 …… 26
2.2.2　浮点型数据 …… 27
2.2.3　布尔型数据 …… 27
2.2.4　字符型数据 …… 27
2.2.5　各类数据之间的转换 …… 27
2.3　引用数据类型 …… 29
2.3.1　引用赋值 …… 29
2.3.2　方法参数传递 …… 29
2.3.3　this 与 super 的引用 …… 30
2.4　Java 类库中对基本数据类型的对象包装器(wrapper)类 …… 31
习题 …… 32
第 3 章　运算符和表达式 …… 34
3.1　概述 …… 34
3.1.1　运算符 …… 34
3.1.2　表达式 …… 35
3.2　算术运算符和算术表达式 …… 35
3.2.1　一元算术运算符 …… 35
3.2.2　二元算术运算符 …… 36
3.2.3　算术运算符的优先级 …… 39
3.3　关系运算符和关系表达式 …… 39
3.4　布尔逻辑运算符和布尔逻辑表达式 …… 41
3.5　位运算符和位运算表达式 …… 44
3.5.1　位逻辑运算符 …… 44
3.5.2　移位运算符 …… 45
3.5.3　位运算符的优先级 …… 46
3.6　赋值运算符和赋值表达式 …… 47
3.6.1　赋值运算符 …… 47
3.6.2　扩展赋值运算符 …… 47
3.7　条件运算符与条件表达式 …… 48
3.8　运算符的优先级和复杂表达式 …… 48
3.9　表达式语句 …… 50
习题 …… 50
第 4 章　流程控制 …… 53
4.1　概述 …… 53
4.2　分支(选择)语句 …… 53
4.2.1　条件语句 …… 53
4.2.2　多分支语句 …… 56
4.3　循环语句 …… 58
4.3.1　while 循环 …… 58
4.3.2　do-while 循环 …… 59
4.3.3　for 循环 …… 60
4.4　跳转语句 …… 61
4.4.1　break 语句 …… 61
4.4.2　continue 语句 …… 63
4.4.3　return 语句 …… 63
4.5　循环语句与分支语句的嵌套 …… 64
4.6　递归 …… 68

习题 …… 69
第5章 Java的继承、多态、高级类特性和数组 …… 72
5.1 概述 …… 72
5.1.1 Object类 …… 72
5.1.2 Class类 …… 74
5.1.3 String类 …… 76
5.2 覆盖方法 …… 80
5.3 重载方法 …… 82
5.4 高级类特性 …… 84
5.4.1 static关键字 …… 84
5.4.2 final关键字 …… 85
5.4.3 枚举类型 …… 85
5.4.4 抽象类 …… 87
5.4.5 接口 …… 89
5.5 内部类 …… 91
5.6 数组 …… 92
5.6.1 一维数组的创建、初始化和引用 … 92
5.6.2 多维数组的创建、复制和调整大小 …… 94
习题 …… 97
第6章 异常和断言 …… 99
6.1 概述 …… 99
6.2 异常处理类型 …… 101
6.2.1 捕获异常 …… 101
6.2.2 声明抛出异常 …… 103
6.2.3 自定义异常 …… 104
6.3 异常处理编程的提醒 …… 104
6.3.1 try和catch语句 …… 105
6.3.2 finally语句 …… 107
6.3.3 异常处理的原则 …… 108
6.4 断言 …… 110
6.4.1 断言语法 …… 111
6.4.2 断言的使用 …… 112
6.4.3 什么情况下不要使用断言 …… 113
习题 …… 114
第7章 输入输出及文件操作 …… 117
7.1 概述 …… 117
7.1.1 计算机中数据的I/O方向 …… 117
7.1.2 Java中的标准输入/输出类 …… 118
7.1.3 Java中包含的输入输出流类 …… 118
7.2 文件 …… 123
7.2.1 创建文件 …… 123
7.2.2 File类提供的方法 …… 123
7.2.3 随机文件流 …… 126
7.2.4 压缩文件流 …… 129
7.3 字节流 …… 133
7.3.1 字节输入流 …… 133
7.3.2 字节输出流 …… 133
7.3.3 内存的读写 …… 134
7.4 字符流 …… 134
7.4.1 字符输入流 …… 135
7.4.2 字符输出流 …… 135
7.5 对象流 …… 138
7.6 过滤流 …… 140
7.7 管道流 …… 141
7.8 不同流的速度比较 …… 142
7.8.1 内存映射文件 …… 142
7.8.2 文件通道 …… 142
7.8.3 CRC32类 …… 142
7.9 输入输出流和正则表达式 …… 145
7.9.1 Pattern类 …… 145
7.9.2 Matcher类 …… 145
7.10 Java I/O流的其他应用 …… 147
习题 …… 148
第8章 线程 …… 150
8.1 概述 …… 150
8.1.1 什么是线程 …… 150
8.1.2 Java中的线程模型 …… 151
8.2 线程的创建 …… 151
8.3 线程的调度与控制 …… 154
8.3.1 线程优先级与线程调度策略 …… 154
8.3.2 线程的基本控制 …… 155
8.4 线程同步 …… 158
8.4.1 多线程并发操作中的问题 …… 158
8.4.2 对象的锁及其操作 …… 160
8.4.3 死锁的防治 …… 163
8.4.4 线程间的交互:wait()和

notify() …… 163
8.4.5 不建议使用的一些方法 …… 167
8.5 线程状态与生命周期 …… 167
8.6 线程相关的其他类与方法 …… 169
8.6.1 支持线程的类 …… 169
8.6.2 线程组 …… 169
8.6.3 Thread 类的其他方法 …… 170
习题 …… 170
第 9 章 编写图形用户界面 …… 173
9.1 概述 …… 173
9.2 用 AWT 编写图形用户界面 …… 173
9.2.1 java.awt 包 …… 173
9.2.2 组件、容器和布局管理器 …… 173
9.2.3 常用容器 …… 175
9.2.4 LayoutManager（布局管理器） …… 177
9.3 AWT 事件处理模型 …… 184
9.3.1 事件类 …… 186
9.3.2 事件监听器 …… 187
9.3.3 AWT 事件及其相应的监听器接口 …… 189
9.3.4 事件适配器 …… 192
9.4 AWT 组件库 …… 194
9.4.1 基本组件的应用 …… 194
9.4.2 组件与监听器的对应关系 …… 202
9.5 用 Swing 编写图形用户界面 …… 203
9.5.1 Swing 概述 …… 203
9.5.2 Swing 的类层次结构 …… 205
9.5.3 Swing 的特性 …… 206
9.6 Swing 组件和容器 …… 209
9.6.1 组件的分类 …… 209
9.6.2 使用 Swing 的基本规则 …… 210
9.6.3 各种容器面板和组件 …… 210
9.6.4 布局管理器 …… 226
9.7 Swing 的事件处理机制 …… 227
习题 …… 228
第 10 章 Applet 程序设计 …… 233
10.1 Applet 的基本概念 …… 233
10.1.1 Applet 的生命周期 …… 234
10.1.2 Applet 的类层次结构 …… 235
10.1.3 Applet 类 API 概述 …… 236
10.2 Applet 的编写 …… 239
10.2.1 Applet 编写的步骤 …… 239
10.2.2 用户 Applet 类的创建 …… 239
10.2.3 在 HTML 页中包含 Applet …… 240
10.3 Applet 中的图形化用户界面 GUI …… 244
10.3.1 基于 AWT 组件的 Applet 用户界面 …… 244
10.3.2 基于 Swing 的 Applet 用户界面 …… 246
10.3.3 Applet 中的事件处理 …… 249
10.4 Applet 的多媒体支持 …… 250
10.4.1 显示图像 …… 250
10.4.2 动画制作 …… 251
10.4.3 播放声音 …… 253
10.5 Applet 与工作环境的通信 …… 254
10.5.1 同页面 Applet 之间的通信 …… 255
10.5.2 Applet 与浏览器之间的通信 …… 255
10.5.3 Applet 的网络通信 …… 256
习题 …… 257
第 11 章 集合与泛型 …… 259
11.1 概述 …… 259
11.2 集合框架 …… 259
11.2.1 核心接口 …… 259
11.2.2 接口的实现类 …… 261
11.2.3 接口的运算算法 …… 262
11.2.4 接口的方法 …… 262
11.3 简单集合类 …… 267
11.3.1 Vector …… 268
11.3.2 对象数组 …… 269
11.3.3 HashTable …… 271
11.4 泛型 …… 273
11.4.1 泛型的由来 …… 273
11.4.2 泛型的特点 …… 277
11.4.3 泛型定义 …… 278
11.4.4 泛型在使用中的规则、限制和注意事项 …… 279
习题 …… 279

第 12 章　Java SDK 6.0 的下载和操作 …… 282
12.1　Java SDK 6.0 的下载与安装 … 282
12.1.1　Java SDK 6.0 的下载 …… 282
12.1.2　Java SDK 6.0 的安装 …… 282
12.2　Java SDK 6.0 的操作命令 …… 286
12.3　Java 编程规范 …… 287
12.3.1　Java 编程规范的作用与意义 …… 287
12.3.2　Java 命名约定 …… 288
12.3.3　Java 注释规则 …… 289
12.3.4　Java 源文件结构规则 …… 290
12.3.5　Java 源代码排版规则 …… 291
12.3.6　编程建议 …… 292
习题 …… 293
附录 1　考试指导 …… 294
附录 2　全国计算机等级考试二级 Java 语言程序设计考试大纲（2013 年版） …… 303
附录 3　全国计算机等级考试二级 Java 语言程序设计样题及参考答案 …… 305
附录 4　习题参考答案 …… 313

第1章 Java语言概论

1995年5月,Sun公司在"Sun World 95"大会上发布了Java,新一代的网络计算机语言由此诞生,1996年发布了开发工具和程序库JDK 1.0,直至1998年发布JDK 1.2,实现了全球信息网络平台。1999年,Sun公司将Java 2平台分为J2SE、J2EE和J2ME三大块,形成了Java技术的基本架构,一直保持至今。这样的基本架构具有很强的开放性,使Java技术能够适应全球不同硬件平台和技术飞速发展所带来的各种变化需求。其中,Java的API(Application Programming Interface)标准成为IT技术的一次重大革命。Java允许三类API:核心API、扩充API、特殊API。2005年,Sun公司将Java 2平台产品改称为Java SE、Java EE和Java ME。从2000年以后,几乎每年对核心Java SE进行改进,至今,已发布了Java SE 6.2版本。Java为Web提供了简便而功能强大的API接口,为Web提供了动态内容的交互页面技术,使Web网页翻开了新的一页,使互联网从通信为主的功能扩展到网络应用系统服务。这种突破性的技术,促使企业的IT系统实现了一次革命性的飞跃,并显示了Java语言的巨大生命力。

1.1 Java语言简介

1.1.1 Java语言的由来

1991年,Sun公司的成员Jame Gosling、Bill Joe等,为电视、控制烤面包机等家用电器开发了一个分布式系统软件Oak(一种橡树名字),它是Java语言的前身。当时,Oak并没有引起人们的注意,直到1994年后期,随着互联网和3W的飞速发展,他们用Java编写了HotJava浏览器,得到Sun公司首席执行官Scott McNealy的支持,并进一步地研制和开发。因为促销和法律的原因,1995年,Oak更名为Java。Java的出现给软件业带来了巨大的冲击,它的独立于平台和安全可靠非常适合网络要求,很快被工业界认可。许多大公司如IBM、Microsoft、DEC等购买了Java的使用权,并被PC Magazine评为1995年十大优秀科技产品。从此,开始了Java应用的新篇章。目前已有15亿台手机内置了Java技术。Java已无处不在,从太空探测、企业电子商务到多层级的网络游戏都已应用Java实现。在现实的人类生活中,将离不开Java技术。

1.1.2 Java语言的目标

创建Java时,其主要目标是:面向对象、简单化、解释型与平台无关、多线程、安全高效、动态性。

1. 面向对象

类思维方法去实现编程,使软件开发人性化、形象化。

2. 简单化

Java的简单化首先是本身系统的精练，占内存少，在很普通的计算机上就能运行。其次，它避免了许多其他编程语言的缺点，如取消影响程序代码健壮性的指针运算和编程者对内存管理，使编程很容易入门。

3. 解释型、与平台无关

Java的虚拟机(Java Virtual Machine，简称JVM)制定了字节码设计规范，保证了软件体系结构中立，为软件移植建立了良好的基础。与其他解释执行的语言不同，Java设计的字节码很容易地直接转换成对应于特定CPU的机器码，不仅使软件开发周期缩短并且使软件执行时得到较高的性能。

4. 多线程

多线程机制使应用软件能并行执行同步机制，保证了对共享数据的正确操作。编程者分别用不同的线程完成特定的行为，而不需要用全局的事件循环机制，有利于网络上实时交互的生动描述和实现。

5. 安全高效

Java不允许编程用指针和对内存释放，从根本上避免了非法内存操作。在编译时，对代码进行类型和语法检查；在执行时，校验字节码、代码段格式和规则检查、访问权限和类型转换合法性检查、操作数堆栈的上溢或下溢检查、方法参数合法性检查；在平台安装时，检查配置设定资源域的访问等。

6. 动态性

为适应Web应用的快速变化需求，允许程序在运行中下载代码段去动态改变程序的状态。在类库中可自由加入新的方法和实例变量。通过接口支持多重继承，使类的继承更具有扩展性。

1.1.3 Java语言实现机制

Java语言为实现其目标，使用了Java虚拟机(JVM)、垃圾回收机制和Java运行环境(JRE)。

1. JVM

Java语言的执行模式是编译加解释。编写好的Java源程序首先由编译器转换为标准字节代码，然后由JVM去解释执行。Java虚拟机规范对JVM的定义为：在计算机中用软件模拟实现的一种虚拟机。JVM运行的代码存储在.class文件中，每个文件包含最多一个public类的代码。JVM的代码格式由简洁、高效的字节码构成。JVM用字节代码程序与各操作系统和硬件分开，保证Java程序独立于平台，而JVM本身可以用软件或硬件实现。每个Java解释器，不管是Java开发工具还是浏览器，都有一个JVM的具体实现来完成下面3项任务：

- 加载代码：由类加载器完成。
- 校验代码：由字节码校验器完成。
- 执行代码：由解释器完成。

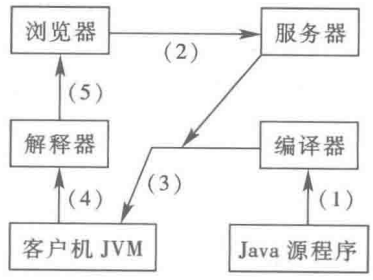

图1-1 Java程序的下载和执行

Java程序的下载和执行步骤如图1-1所示：

(1) 源程序经编译器得字节代码；

(2) 浏览器与服务器连接，要求下载字节码文件；

(3) 服务器将字节代码文件下载到客户机；

(4) 客户机上的 JVM 执行;

(5) 在浏览器上显示并交互。

2. 垃圾回收机制

在 Java 语言中,所有代码都封装在类中,需要时创建类的实例(对象)来处理,这种动态的实例存储在内存堆中。Java 有一个系统级的线程,对内存的使用进行自动跟踪。该线程能在 JVM 空闲时,对不用的内存进行自动回收。它消除了释放内存的必要,也避免了内存的泄漏,将编程者从琐碎繁忙的内存管理中解脱出来。

3. Java 运行环境(JRE)

任何程序运行都需要一定的软件和硬件环境,这称为平台。Java 语言的运行平台包括 Java 应用程序接口(API)和 JVM,如图 1-2 所示。

Java Application 或 Applet	
Java 基本 API	Java 基本扩展 API
Java 基本类	Java 标准扩展类
JVM	
移植接口	

图 1-2 Java 运行平台

Java 有 3 种平台:Java SE、Java ME 和 Java EE。它们都立足于核心开发工具包 JDK 的各种版本。本书以 JDK1.6.2 版本为基础,集成开发环境以 NetBeans 6.0 为基础。

1.2 Java 语言面向对象编程

Java 重要的特性之一是它是完全面向对象的编程语言,其核心是用人类解决问题的抽象方法对复杂的客观问题进行分析、组织和解答。对于程序设计而言,其难点当然是寻找尽可能能正确描述问题的抽象。面向对象的编程语言是利用类和对象将问题的数据和操作封装起来,并用标准接口与外界交互,使代表客观世界实体的各种类在程序中能独立和继承。其特性是要求程序具有封装性、继承性、多态性。

1.2.1 面向对象编程的基本概念

学习 Java 语言首先应了解面向对象的一些基本概念:抽象、封装、继承和多态。

1. 抽象

抽象是人类处理复杂事务的提升。**面向对象编程的思路与传统程序设计不同,它强调按照与人类思维方法中的抽象、分类、继承、组合、封装等原则去模拟现实世界的物理存在,**将客观世界事物都抽象定义为各种对象的组合。人们管理抽象的一个有效方法是用层次分类。允许根据物理意义将复杂的系统分解为更多更易处理的部件。这种分层抽象方法也经常被用于程序设计。例如,人们对日常生活中的手机、汽车、电视机等许多设备的使用,都忽略其具体的工作细节,而将它们作为由许多部件组成的一个整体加以利用。面向对象技术也利用这种思路将现实世界的问题抽象成许多对象的类来表示。通过对问题的抽象、分类、封装和组合来更有效地处理问题,也更能清晰地描述客观事物。

面向对象的概念是 Java 的核心。对编程者来讲,重要的是要理解现实系统怎么去抽象转化为软件系统。对任何软件而言,都不可避免地要经历概念抽象的提炼、成长、衰老这样一个生命周期,而面向对象的设计方法,以类为基本单元,封装成包,使软件在生命周期的每一个阶段都有足够的应变能力,去适应千变万化的大千世界。在编程阶段,通过抽象找出各种类,再对各种类

之间的消息进行收集和处理,把问题分解成许多标准接口的构件。当问题有变化时,就能很从容自信地解除或更换现实软件的某些构件代码来适应变化。抽象导致面向对象编程的方法与数据封装,形成不同层次上的构件,使软件构件化,实现了不同层次上的重用。

2. 封装

封装是面向对象技术中隐蔽信息的一种机制。它在程序中将对象的状态和行为封装成为一个完整的、结构高度集中的整体;对外有明确的功能、接口单一通用、可在各种环境下独立运行的单元。其目的是将对象的具体实现细节隐蔽,只通过一个公用接口和消息与其他对象通信。封装使对象的使用者和设计者分开,源代码可独立编写和维护,既保证不受外界干扰,也有利于代码重用。这种封装好的单元是软件标准构件化的基础,可提高软件生产效率、降低成本和缩短开发周期,实现软件工业化。在Java语言中,**"类"是程序封装的最小单位。**

3. 继承

继承是描述两个类之间的关系。继承允许一个新类(称为子类)包含另一个已有类(称为父类)的状态和行为。这样,从最一般的类开始,用子类去逐步特殊化,可派生出一系列的子类,使父类和子类的关系层次化,可降低程序的复杂度,并提供了类之间描述共性的方法,减少了类的重复说明。**子类的派生过程称为类的继承。继承是抽象分层管理的实现机制。**

在面向对象的继承概念中,有单继承和多继承两种。单继承是指任何子类都只能从一个父类派生,而多继承是指一个子类可由多个父类派生。因此,单继承的类层次是树状结构,多重继承的类层次是网状结构。在Java中,除Object类外,任何类都有父类。继承仅支持单继承,多重继承要通过接口实现,这大大降低了继承的复杂度,给编程带来方便。

Java中,创建了一个类后可用关键字extends去创建它的子类。

4. 多态

多态是允许一个类中有多个同名方法,但方法的具体实现却不同的机制。为提高程序的抽象和简洁,许多对象的操作方法名相同,但方法的具体实现细节却不同。如一种排序方法,可以用在不同的数据类型上,对数值类型和字符类型,它们的排序程序就不相同。这种在同一个程序中同名的方法可用不同代码实现的特性称为多态性。可利用子类对父类的方法覆盖来具体实现。多态可以使类的对象响应同名消息(方法)去完成某种特定功能,而具体实现代码却不相同。例如,同样是复制和粘贴操作,在文本处理和图形处理中具体代码是完全不同的。

1.2.2 类与包

类是对具有相同特性对象的封装组合,是Java程序的基本单位。编程的过程就是编写类的过程。

1. 类的声明

一个类在使用之前必须先声明。类声明的语法为:

【类修饰符】**class 类名**【extends 父类名】【implements 接口名列表】{类体}

其中:【】内是可选项,{ }内是类体。

类修饰符可以有多个,说明使用该类的权限,包括:

- public:访问控制符指明该类为公共类,可被其他类访问或引用其成员变量和成员方法。注意:Java语言规定包含main()方法的类必须是公共类。
- abstract:抽象类,指明不能实例化的类。为更贴近人的思维和客观世界,定义它常常为改

变它的子类而设置。

- final:终结类,指明该类不能有子类。

如该类没有修饰符,说明该类具有默认访问权限,即只允许与该类相同包中的类可以访问和调用,其他包中的类不能访问和使用。

class 关键字后是由编程人员定义的一个合法标识符的类名。

extends 关键字用于声明类是从某个父类派生,则该父类名应写在 extends 之后。

implements 关键字用于声明类要实现某些接口,而这些接口名应写在 implements 之后。

类体中声明了该类的所有变量和方法,称为**成员变量**和**成员方法**。下面是类声明的举例。

例 1.1 定义一个日期 Date 类,声明为:

```
public class Date{
    //成员变量
    private int day;
    private int month;
    private int year;
    //成员方法
    public void setDate(d,m,y){
        day = d;
        month = m;
        year = y;
    }
    public void showDate(){
        system. out. println(day+":"+month+":"+year);
    }
}
```

2. 成员变量

类体中的成员变量定义了类的特性,其声明语法为:

【变量修饰符】 类型 变量名【=初值】【,变量名【=初值】】,…;

变量修饰符可以有多个,说明使用该变量的权限和规则。变量修饰符包括:

- public:允许所有的类来访问该变量。
- protected:仅允许相同包的类和该类的子类来访问。
- private:仅允许本类访问的内部变量。
- static:表示该变量是**类变量**,是该类全部对象共享的变量,它独立于该类的任何对象,可直接进行访问。对一些类的对象具有相同的属性值时,可利用 static 来声明,如例 1.2 中的 PI。没有被 static 修饰的变量是**实例变量**(也称**对象变量**),只能通过对象来访问它。

成员变量的类型可以是 Java 中的任意数据类型。具体到某个成员变量其类型应该是唯一的。成员变量名是 Java 的合法标识符。一次可定义多个变量,每个变量可设置自己的初值。在例 1.1 中 day、month、year 就是实例变量的定义。

访问类变量值的语法与实例变量类似,其格式为:

对象名.类变量名

访问 Date 的实例变量值的语法为:

对象名. 实例变量名

例 1.2 TestMemberVariable. java //对类变量和实例变量的访问实例

```
class Circle{
    static double PI = 3.14159265;
    int radius;
}
public class TestMemberVariable{
    public static void main(String args[]){
        Circle c = new Circle();              //Circle()是 Circle 类的构造方法
        c.radius = 10;                        //对实例变量赋初值
        System.out.println(Circle.PI);        //访问类变量
        System.out.println(c.radius);         //访问实例变量
        System.out.println(c.PI);             //访问类变量
    }
}
```

运行结果如下：

```
3.14159265
10
3.14159265
```

注意：成员变量区别于方法中声明的局部变量，成员变量的使用范围（作用域）在整个类，而局部变量的作用域只在方法内部。一旦从方法中返回，局部变量就消失。成员变量只存在于类中，不能出现在方法中。

3. 成员方法

类体中的成员方法定义了类的操作，其声明语法为：

【方法修饰符】 **返回值的类型 方法名**（参数表）{方法体}

方法修饰符包括：

- public、protected、private：表示访问权限，与变量修饰符意义相同。
- static：表示是**类方法**，可直接进行访问和调用。不用 static 修饰的方法称为**实例方法**。
- abstract：表示是抽象方法。没有定义方法体。static 方法和 final 方法都不允许是 abstract 方法。
- native：表示是其他语言代码的方法，用它与 Java 代码集成。
- synchronized：表示多个并发线程对共享数据访问的控制。
- final：表示是终结方法，不允许子类方法覆盖。

返回值的类型除 void 关键字说明该方法没有返回值外，编程者必须在方法声明中指定返回变量的数据类型，即在方法体中必须有 return 语句来指定返回值类型。

方法名是合法的 Java 标识符。()内的参数表是调用方法时所需传递消息的列表。

Java 允许参数列表为空。方法体是对方法的具体实现，由局部变量和 Java 语句代码组成。其中的局部变量只在该方法内有效，当方法返回时，局部变量就不存在了。通常局部变量由小写字母组成。

在 Java 程序中要调用方法需要根据方法是否有返回值和调用方法本身所在位置的情况而

定,原则是:

(1) 方法是否有返回值的情况。

- 如有返回值,将调用当做一个数值处理(类型必须与返回值相同)。

如
```
double x = avg(r,s,t);
System.out.println(avg(16.4,57.8,44.8)); //直接作为一个数值用
```

- 如没有返回值,可直接调用。

如
```
public void setDate(d,m,y){
    day = d;
    month = m;
    year = y;
}
Date birthdate = setDate(01,07,1990);
```

(2) 调用方法所在位置可能就在本类中,也可能在其他类或超类中。

- 任何情况可用实例变量名.方法名来调用。
- 如在类方法中,可直接调用本身的类方法,但不能直接调用实例方法。
- 如在实例方法中,可直接调用本身类的类方法或实例方法。
- 在实例方法中能用 this.方法名或 super.方法名调用。

例 1.3 TestCircumference.java //实例变量调用实例方法的举例

```
class Circle{
    static double PI = 3.14159265;
    int radius;
    public double circumference(){                    //实例方法
        return (2 * PI * radius);
    }
}
public class TestCircumference{
    public static void main(String args[]){
        Circle c1 = new Circle();
        c1.redius = 20;
        double circumf1 = c1.circumference();         //实例变量调用实例方法
        System.out.println("圆的周长=" + circumf1);
    }
}
```

运行结果如下:

```
圆的周长=125.663706
```

例 1.4 TestCircumference.java //调用类方法示例

```
class Circumference{
    static double PI = 3.14159265;
    public static double circumf(int radius){         //类方法
        return (2 * PI * radius);
    }
```