GENERAL EDUCATION

高等学校通识教育系列教材

Java语言程序设计

（第2版）

于静 主编

杨娜 张虹 顾鸿虹 副主编

清华大学出版社

GENERAL EDUCATION

高等学校通识教育系列教材

Java语言程序设计

（第2版）

于静 主编

杨娜 张虹 顾鸿虹 副主编

清华大学出版社

北京

内 容 简 介

本书在讲解 Java 程序设计语言的基础语法之前，使用现实生活中的实例让读者从感性上体会和理解面向对象思想的理念，较早地融入面向对象的世界。本书的主要内容包括面向对象程序设计基础、Java 程序设基础、程序的流程控制、类的特性、接口与多态、异常处理、对象的管理、I/O 操作、多线程编程、网络编程、Eclipse 开发环境的使用、图形用户界面编程——Swing 技术以及数据库编程。

本书适合作为普通高等院校应用型本科计算机相关专业的入门教材，也可供其他初学者或软件开发人员参考使用。

图书在版编目(CIP)数据

Java 语言程序设计/于静主编. —2 版. —北京：清华大学出版社，2018
(高等学校通识教育系列教材)
ISBN 978-7-302-48852-1

Ⅰ. ①J…　Ⅱ. ①于…　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(2017)第 286788 号

责任编辑：刘向威　张爱华
封面设计：文　静
责任校对：李建庄
责任印制：沈　露

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　**邮　　编**：100084
社 总 机：010-62770175　　**邮　　购**：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
课件下载：http://www.tup.com.cn,010-62795954
印 装 者：北京鑫海金澳胶印有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：16.75　　**字　　数**：409 千字
版　　次：2009 年 8 月第 1 版　2018 年 2 月第 2 版　　**印　　次**：2018 年 2 月第1次印刷
印　　数：1～2000
定　　价：45.00 元

产品编号：076553-01

前言

作为一名程序开发的初学者，首先应该掌握一种先进科学的程序设计思想。面向对象是一种为众多程序员所使用的程序设计方法，其思想是按照现实世界的本来面貌来理解世界，直接通过对象及其相互关系来反映世界，这样建立起来的系统才能符合现实世界的本来面目。面向对象程序设计思想对于程序员来说极为重要，它是一种思维方式，直接决定了开发的软件质量。Java 语言是 Sun 公司于 1995 年正式推出的一种程序设计语言，在众多面向对象程序设计语言中脱颖而出，它具有高性能、跨平台性、可移植性、健壮性、安全性等优良特性，以其独特的魅力在软件开发领域占据绝对霸主的地位。

本书站在思想的高度上，让读者首先从感性上体会和理解面向对象思想的理念，较早地融入面向对象的世界，然后再利用 Java 语言逐个讲解关于面向对象的知识点，使读者不仅能够学习到 Java 语言本身，还能最终掌握面向对象程序设计思想的精髓。

本书面向的读者对象主要包括普通高等院校应用型本科计算机相关专业的学生。综合学生特点和当前软件行业的就业需求，编者总结实际教学经验与体会，设计各章节内容及教学重点。书中实例蕴含的思想大部分来自实际的工程项目经验，具有一定的实用性和参考价值。书中的“注意”和“技巧”两个环节作为理论知识的补充，进一步增强了本书的实践性。

本书主要讲解面向对象程序设计基础、Java 程序设计基础、程序的流程控制、类的特性、接口与多态、异常处理、对象的管理、I/O 操作、多线程编程、网络编程、Eclipse 开发环境的使用、图形用户界面编程——Swing 技术以及数据库编程等内容。本书章节安排合理，内容循序渐进。

全书注重提高读者运用 Java 语言和面向对象思想解决实际问题的实践能力。在第 1 版的基础上，本版将 JDK 的版本更新到 8.0，介绍了必要的 JDK8.0 新特性。

全书分 13 章。

第 1 章介绍 Java 的发展历史及 Java 语言的特性、面向对象程序设计中的基本概念及意义，讲解 JDK 的安装及配置方法，并使用 Java 语言编写第一个 HelloWorld 命令行程序。

第 2 章讲解 Java 语言的基础及程序结构，介绍 Java 程序的组织形式，Java 中类和对象的概念、修饰符、对象的创建与引用，Java 支持的数据类型和操作符。

第 3 章学习 Java 程序中的流程控制，包括选择结构、循环结构以及与程序转移有关的其他流程控制语句，强调 Java 程序的流程控制与 C 语言程序设计的流程控制之间的不同。

第 4 章介绍 Java 面向对象技术的一大特性——继承，以及由继承机制派生出的抽象类等概念，同时介绍在编程中经常使用的内部类等。

第 5 章讨论深受软件设计人员青睐的接口技术与多态，学习接口的定义、应用，及其与抽象类的区别。此外，讨论如何利用多态机制实现向上转型，以及如何增强系统的可扩

展性。

第 6 章学习异常的概念、Java 异常处理机制,包括 try-catch、多重 catch、try-catch-finally 等几种异常流程控制的使用,throws、throw 关键字的应用,以及如何使用自定义异常类。

第 7 章讲解在 Java 程序开发中经常使用的一种数据结构——集合类,介绍对象数组与普通数组的区别,重点学习 ArrayList、LinkedList、Vector 类的使用,此外还包括 HashSet、HashMap 类的使用以及泛型的概念。

第 8 章学习 Java 的 I/O 技术,如何完成对各种外部设备的 I/O 操作,重点掌握对文件的操作,另外还讨论了 I/O 流与通道的概念。

第 9 章讨论 Java 的多线程编程,包括线程的概念,如何在程序中实现多线程,线程同步的作用等。

第 10 章讲解 Java 的网络编程,介绍网络编程基础知识,理解端口和套接字,掌握使用 InetAddress、ServerSocket 等类编写 TCP、UDP 程序。

第 11 章介绍 Eclipse 集成开发环境,讨论如何安装 Eclipse 及其插件、创建 Java 项目、修改项目属性、创建包和类以及如何运行程序,包括对存在缺陷的程序如何利用 Eclipse 进行调试。

第 12 章介绍 Swing 技术,在 Eclipse 环境下如何安装 Visual Editor 插件实现一个简单的图形用户界面,接着重点讲解 Swing 中容器的布局概念及各种组件的使用。

第 13 章介绍数据库编程,讨论 JDBC 技术的使用,最终使读者能够采用驱动程序访问数据库,结合 Swing 技术开发简单的桌面应用程序。

本书第 1 章由柯瑜编写,第 2～5 章由杨娜编写,第 6～7 章、第 13 章由于静编写,第 8～9 章由顾鸿虹编写,第 10～12 章由张虹编写。全书由于静负责内容结构设计和统稿工作。

由于编者水平所限书中难免有疏漏之处,恳望读者批评指正。

编　者

2017 年 12 月

目 录

第1章 Java和面向对象的程序设计

本章要点

- 了解Java的发展历史。
- 掌握Java的三大特性。
- 熟练掌握面向对象与类的概念。
- 熟练掌握JDK的安装及配置方法，能够编写简单Java程序。

本章首先向读者介绍Java的发展历史及Java语言的特性，再介绍面向对象程序设计的基本概念及意义，最后讲解JDK的安装及配置方法，并使用Java语言编写第一个HelloWorld命令行程序。

1.1 关于Java

什么是Java？很多人认为Java是一种编程语言。的确，从程序设计者的角度来看，Java有自己的语法规则和运行机制，是一种简单易用的新型编程语言。但是，随着软件工程的不断发展、互联网应用的迅速膨胀，现在的Java已经更像是一种标准或一种平台。从Java ME(Java Micro Edition)在PDA、智能手机等嵌入式设备的广泛应用到Java EE(Java Enterprise Edition)架构在企业解决方案中的备受推崇，可以看出Java已经悄悄地融入人们日常生活的各个角落。

1.1.1 Java的出现

Java在刚刚出现时并不顺利，Sun公司在1991年制订了一个名为Green的研究计划，准备开发一种语言应用于智能化家用电器中。这就要求该语言能顺利地在不同指令系统的处理器上执行，不仅在嵌入式设备领域非常重要，在不同的计算机领域也有重要意义。于是Sun公司的James Gosling等人开发了名为Oak的语言，即Java的前身。但是，智能化家用电器对于当时来说似乎太超前了，Oak语言的发展速度并不是那么理想。好在后来互联网技术飞速发展，互联网的广泛应用促成了Java的诞生，Java跨平台运行的设计初衷正好符合了当时互联网的发展要求。从此，Java借助互联网的东风蓬勃发展起来。2009年，甲骨文公司收购了Sun公司。2014年，甲骨文公司发布了Java 8。

1.1.2 Java的特性

1. 平台无关性

Java之所以能够实现跨平台运行是因为Java采用了一种虚拟机作为中间层来屏蔽平

台差异。Java 的源程序经过编译会产生类文件,类文件结构不同于传统的编译器生成的二进制文件,必须经过虚拟机的解释才能运行,因此 Java 被认为是一种解释型语言。虚拟机作为中间层负责把编译好的类文件转换为不同平台的指令,这样,虚拟机就屏蔽了各种平台的差异性。Java 程序的运行机制可以用图 1.1 表示。

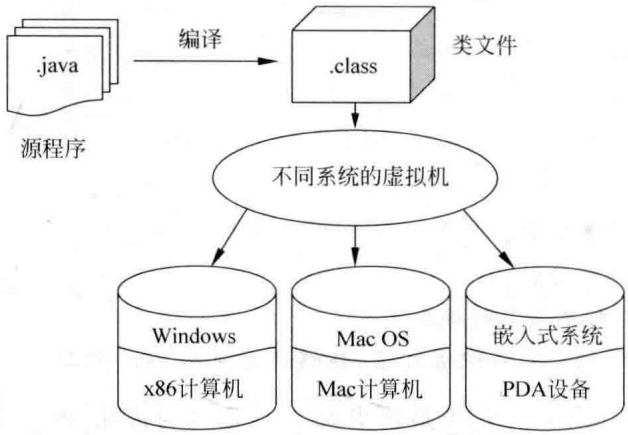

图 1.1 Java 程序的运行机制

注意:对于 Java 的跨平台性,读者要有一个清楚的认识。在程序设计中要特别注意,Java 程序不能做到绝对的与平台无关。例如,Java 支持 JNI(Java Native Interface),允许 Java 和其他语言编写的代码进行交互,这一点就可能会使 Java 程序与平台产生一定的联系。还有,如果编写的 Java 程序调用外部程序,也可能会因为外部程序的平台相关性造成 Java 程序无法跨平台运行。因此不能依靠 Java 的跨平台性解决所有问题,更换平台后也需要对 Java 程序进行充分的测试。

2. 面向对象

Java 支持面向对象编程(object oriented programming)的特性:封装(encapsulation)、继承(inheritance)和多态(polymorphism)。通过这三种特性,程序开发者可以把自己的代码以一种更符合人类理解的方式进行组织:把实现一种功能的系统抽象成一种类,把相关的数据和所支持的数据操作封装到类中,留出用户可以操作的"界面"。不仅如此,通过继承机制可以对已经封装好的类进行扩展,在不改动原有代码的基础上增加更多功能。这种机制也被称作类的派生,由父类派生出子类,子类继承父类的特性,并可增加更多特性。而多态机制则可以一种父类对象的形式来操作不同的子类对象,系统会自动判断该对象属于哪一个子类并执行操作。

Java 的多态主要是建立在后期绑定(late binding)或叫动态绑定(dynamic binding)基础之上的。所谓后期绑定是指绑定动作在执行期才根据对象类型而进行。这样,Java 先天就具备了对对象类型的自动判断能力。而以前的面向对象编程语言(如 C++)是不支持动态绑定的,绑定动作发生在编译期而不是在执行期,所以 C++要实现多态的效果必须要使用虚函数。此外,Java 还抛弃和修改了 C++的很多不符合面向对象程序设计的特性,例如 Java 不使用指针、不使用 goto 语句、以接口的形式实现多重继承等。由此可见,Java 是一种比 C++更符合面向对象标准的程序设计语言。

3. 支持多线程

多线程技术主要是指多个程序流程同时执行,完成不同任务。多线程技术在很多应用

方面尤其是设计图形用户界面的程序方面发挥着重要作用。以前的编程语言，无论是C还是C++都要使用特殊的代码实现多线程，而Java是内部支持多线程的，即在虚拟机中就支持多线程操作，在这一点上也反映了Java的先进性。

1.1.3 Java与C、C++语言的比较

首先，在面向对象的特性方面，Java具有优势。C语言仅仅支持结构体，仅仅能完成数据的封装，无法进行方法的封装，继承和多态更是不被C语言支持，这样C语言先天就不具备面向对象的特点。而C++语言虽然支持封装和继承，但多态仍然是建立在编译时绑定的基础上，必须要通过虚函数实现，不仅如此，C++语言的指针操作允许使用goto语句，容易产生歧义的多重继承等特性也使C++语言不能严格符合面向对象的思想。

其次，在内存管理方面，C、C++语言需要设计者对内存的使用和管理有详细的规划，这尤其体现在内存回收方面。C、C++语言分配的内存空间必须由用户在不使用的时候进行回收。如果程序较为复杂就容易产生内存回收不充分的问题：使用过的内存空间没有回收就会形成内存漏洞，这段内存将无法再被其他程序使用。而Java在其虚拟机内部实现了垃圾内存自动回收功能，能够自动判断哪些内存不再被用户使用，在内存不足或用户调用时会自动回收这些内存空间。这样就让用户在系统设计时从繁杂的内存管理工作中解脱出来，从而更合理地设计和优化系统。

此外，对于不同的C、C++编译器，各种数据类型所占空间有可能不一样，这种差异会影响程序的可移植性。因为设计者在重新编译源程序时必须要考虑新的编译器对某种数据类型所占用空间的规定，必要时需要修改源代码以配合新编译器，从而增加了程序设计者的负担。而各种数据类型所占内存空间在Java中是固定的。

Java虽然相对于C、C++语言来说有了重大改进，但Java也有自身的缺点。第一，Java是一种解释型语言，编译后的Java类文件并不能直接被操作系统执行，需要虚拟机的解释，因此其效率和C、C++语言相比还是有差距的。第二，Java编写的软件需要操作系统中安装有虚拟机，所以Java程序的发布相对烦琐一些。但“瑕不掩瑜”，Java仍然可以说是程序设计领域的重要革命。

1.2 面向对象的程序设计

面向对象是一种更贴近现实生活的程序设计思想，使用一种简单直观的思路来进行程序设计。在面向对象程序设计中，开发者把要解决的问题描述为符合客观现实的系统。这个系统不仅包括自己的属性状态，也包括自己的支持动作。这样的系统更贴近于人们接触的真实世界，让开发者以更符合真实世界的逻辑来思考和解决问题，而不是以更符合计算机处理的逻辑来思考和解决问题。

1.2.1 面向对象与类的概念

1. 封装、类以及界面

面向对象程序设计第一个要解决的问题是对现实系统的描述。对某一具体问题进行概括，总结出同类问题所具有的基本属性以及操作这些属性的方法，这就是面向对象程序设计

中的重要概念——抽象。当通过抽象描述出一个系统后,以某种数据结构组织这些属性并规定相应的方法操作这些属性,使之形成一个独立的系统。外界与该系统的交互必须通过规定的“界面”来进行,从而保证了该系统内部属性的独立性。这就满足了面向对象程序设计的第一大特点——封装。封装好一个系统后,这个系统就是面向对象程序设计中的“类”。类中包含的属性和方法,称为该类的成员。类与外界通过“界面”进行交互,这种“界面”一般为某种方法,而不是让某种属性直接与外界发生关系。当规定了某种方法为外界可见时,该方法就可以被认为是该类的“界面”。

注意:在规定某个类的“界面”时,常会把某方法规定为外部可见,即该类的“界面”。虽然也可以把某种属性规定为外部可见,但这是一种不好的设计习惯。由于属性表示了某种状态,当属性被多个外界用户直接修改时,往往不能确定这个属性的变化是否符合逻辑。这与 C 语言中的全局变量的弊端一样。正如 C 语言中建议尽量减少全局变量的使用,面向对象程序设计中也建议尽量避免把属性规定为外部可见。

举一个简单的例子,例如在程序中需要描述一部汽车,可以把汽车中的基本属性抽象出来:发动机、刹车装置、车轮等,抽象出来的方法包括加速、刹车等。当然,一个真实的汽车所具有的属性和方法不仅仅这么简单,对汽车这个概念的描述细化到什么程度,完全取决于实现程序的需要,描述得越细,所支持的功能就越多,但类的设计也就越复杂。当抽象好汽车这个系统后就形成了汽车类,汽车类给用户的“界面”是“加速”和“刹车”,即规定的外部可见方法。当用户想“加速”时,应该使用“加速”方法,“加速”方法按照自己实现的逻辑运作,例如启动发动机,发动机控制车轮转动。试想一下,如果规定车轮为“界面”,用户想“加速”时,直接操纵车轮加速而跳过发动机的启动过程,这就使该汽车处于一个车轮转动而发动机尚未启动的状态,这是不符合逻辑的。所以应该避免将车轮作为“界面”,同时规定好“加速”的程序,封装到“加速”方法中,留给用户使用。

2. 对象的概念

在面向对象程序设计中,还要弄清楚对象的概念。类与对象的关系可以总结为:类是对象的定义,对象由类来生成。当抽象封装好一个类以后,实际上完成的是制作一个“模板”的工作,如果想要应用这个类,多数情况下要把该类具体化为某一个对象,这个过程就好像利用“模板”做出一个模型的过程,这个过程也叫类的实例化过程。例如上个例子中设计好的汽车类,若想使用汽车类,必须用该汽车类实例化一辆具体的汽车,否则不能说让汽车加速,因为不知道这个汽车具体是指哪一辆汽车。因此可以这样认为,类是指对某一个系统的描述,对象是指符合该描述的具体系统。或者说,类是概念模型,对象是现实模型。

3. 继承和多态

除了封装以外,继承和多态是面向对象程序设计中的另外两大特点。继承和多态提高了面向对象程序设计的代码重用性、灵活性和可维护性。

所谓继承,是指当描述一个新类时,可以在已经描述好的类基础之上进行扩展描述,而不用重新定义已经描述的属性和方法。这时,已经描述好的类称为父类,扩展描述的新类称为子类。还以刚才的汽车为例,已经描述好了汽车应该包含发动机、刹车装置和车轮属性以及“加速”“减速”方法,如果想定义货车这个类时,可以利用已有的汽车类作为父类,在此基础之上再定义后车厢的属性以及控制后车厢翻斗的方法,而父类中已有的发动机、“加速”等方法和属性会被货车类继承。这也符合对客观世界的认识——货车属于汽车中的一种,是

汽车的子集，如图 1.2 所示。

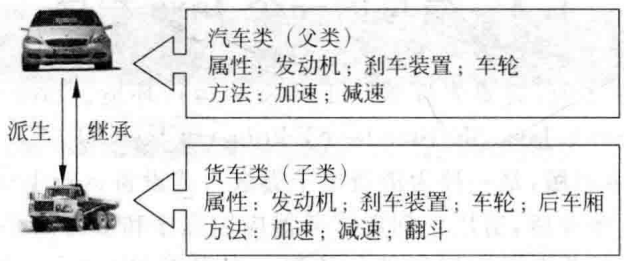

图 1.2　面向对象中的继承

多态是面向对象的第三大特点，多态往往需要和继承综合考虑。多态实现了把不同的子类对象都当作父类类型看待，屏蔽了不同子类对象之间的差异。例如，以汽车类为父类，实现两个子类：货车类和轿车类。由继承关系可知，货车和轿车作为汽车的子类，继承了父类的“加速”方法。但是实际上货车和轿车的加速程序是不同的，货车在加速前需要检查后车厢的状态，如果后车厢没有归位的话是不应该加速的，而轿车则没有这方面的要求。这样，“加速”方法应该在子类中根据不同的车型重新定义。定义以后，发布一条这样的命令：“A 汽车加速”。注意，这里发布的命令是“汽车”加速，并没有指定该汽车是货车还是轿车。多态在这里就发挥了作用，多态机制会自动判断 A 汽车的类型，如果 A 汽车是货车则执行货车的加速方法，若是轿车则执行轿车的加速方法。这样，程序员在开发程序时就无须关心子类的方法实现，而采用父类中统一的方法操纵子类。

多态机制除了和继承机制综合应用以外，也可和“接口”技术综合应用，应用方法相同。有关继承、接口和多态的概念，在后面的章节中会详细介绍。

1.2.2　面向对象程序设计的意义

传统的面向过程程序设计思想认为：程序由算法和数据结构组成。这种思路把数据与对数据的操作相分离，数据可以在整个程序的不同算法中随意流动，可见，如果程序过于庞大复杂，开发者就很难对数据的状态有一个全方位的把握。而面向对象程序设计则把程序理解为对象的集合。数据和算法被封装在对象中，对象仅仅留出可以被外部对象调用的“界面”。对象之间通过这些“界面”进行访问和通信，共同组成一个庞大的系统。由于每个对象分工明确，分别掌管属于自己数据的控制权，所以即使系统非常复杂，数据的状态也比较容易把握。

此外，面向对象程序设计中的继承可以让开发者以更符合客观现实的逻辑实现代码重用，利用已经实现的类来构造子类。子类不仅可以继承父类中已有的属性和方法，更可以增加新的属性和方法来满足自己的需要。不仅如此，当父类中有不符合子类特殊逻辑的方法时，子类可以重新定义该方法而不用修改原来的父类，这样不仅保证了父类的稳定性和通用性，也给子类留出了可定制的空间。

最后，面向对象程序设计中的多态机制为开发者管理同一父类型的不同子类型的对象提供了方便，屏蔽了子类间的差异，以父类的“界面”统一管理不同子类。

面向对象的程序设计是目前程序设计方法的主流思想，用户在学习 Java 语言程序设计的过程中应该把面向对象的思想渗透其中，体会其特点，学会其应用。

1.3 编写第一个 Java 程序

在编写 Java 程序之前,需要先部署一下开发和运行环境。Java 语言分为三个版本:Java ME(Micro Edition)、Java SE(Standard Edition)和 Java EE(Enterprise Edition)。其中 Java ME 为 Java 微型版,是一种为消费电子类嵌入式设备运行 Java 程序设计的解决方案。Java EE 为 Java 企业版,适用于创建服务器应用程序和服务的平台架构。Java SE 是 Java 标准版,用于普通桌面级应用软件的编程。本书以 Windows 平台下,使用 Java SE JDK 8.0 来部署 Java 程序的开发运行环境。

1.3.1 安装 Java SE 的 JDK

1. JDK 与 JRE

JDK(Java Development Kit)是 Java 语言的开发工具包,工具包里除了包含 Java 语言编译器、调试器以及演示程序以外,一般还会包含 Java 程序的运行环境:JRE(Java Runtime Environment)。JRE 是某一平台运行 Java 程序的软件环境,包括虚拟机和核心类库等。如果仅仅部署 Java 程序的运行环境,可直接下载安装 JRE,不必安装 JDK。但如果想要编写 Java 程序,则应该安装 JDK,以实现 Java 程序的编译。

2. 安装 Java SE JDK

JDK 可在 Sun 公司网站 http://www.oracle.com/technetwork/java/javase/downloads/index.html 免费下载。进入页面,选择正确的操作系统和语言版本后即可下载 JDK 的安装版(本书使用 jdk-8u131-windows-x64.exe)。

运行 JDK 安装程序启动安装向导,单击“下一步”按钮进入定制安装界面,如图 1.3 所示。在图 1.3 中,选择“开发工具”,设置安装位置,再单击“下一步”按钮安装所选组件,如图 1.4 所示。

图 1.3 安装 Java SE JDK