普通高等教育软件工程类“十三五”规划教材

Java平台

项目设计与开发

解绍词　金　霜　杜伟奇／主编

科学出版社

普通高等教育软件工程类“十三五”规划教材

Java 平台项目设计与开发

解绍词　金　霜　杜伟奇　主编

科学出版社
北京

内 容 简 介

本书内容共14章，分为两部分。其中，第1~9章为第一部分，基于Java平台标准版，主要面向个人PC桌面应用程序开发，介绍了Java语言基本语法、面向对象特性、多线程程序设计、输入输出和异常处理、集合与泛型、图形用户界面、网络通信编程、数据库编程等内容；第10~14章为第二部分，基于Java平台企业版，主要面向复杂的企业级应用，介绍了JSP、Servlet、JavaBean、项目实训等内容。

本书可作为高等院校计算机、软件工程等专业的本科生专业课程、实训课程教材，也可作为相关培训教材，适合Java初学者和进阶者阅读。

图书在版编目(CIP)数据

Java平台项目设计与开发 / 解绍词，金霜，杜伟奇主编. — 北京：科学出版社，2019.5

ISBN 978-7-03-061054-6

Ⅰ. ①J… Ⅱ. ①解… ②金… ③杜… Ⅲ. ①JAVA语言-程序设计 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字 (2019) 第073238号

责任编辑：李小锐 / 责任校对：彭 映

责任印制：罗 科 / 封面设计：墨创文化

科学出版社出版

北京东黄城根北街16号

邮政编码：100717

http://www.sciencep.com

成都锦瑞印刷有限责任公司印刷

科学出版社发行 各地新华书店经销

*

2019年5月第 一 版 开本：787×1092 1/16

2019年5月第一次印刷 印张：16 3/4

字数：381 000

定价：56.00元

（如有印装质量问题，我社负责调换）

前　言

随着计算机及软件开发技术的发展，Java 作为面向对象、跨平台的编程语言，自 1996 年正式发布以来，现已成为 IT 领域里最受欢迎的主流编程语言之一。面向对象的 Java 语言具备“一次编程，处处运行”的能力，使其成为软件服务提供商和系统集成商用以支持多种操作系统和硬件平台的首选解决方案。

Java 语言由于具有语法简单、面向对象、应用领域广等特点，因此，它非常适合作为普通高等院校程序设计课程，尤其是面向对象程序设计课程的教学内容。本书内容针对 Java 2 平台标准版(Java2 Platform Standard Edition，Java SE)与 Java 2 平台企业版(Java2 Platform Enterprise Edition，Java EE)两个版本，采用循序渐进、由浅入深、概念与例程相结合的撰写方式，对结构的安排、例程的选择进行了充分考虑，以确保难度适中，更贴近于实用。

在学习本书之前，读者应具备基本的计算机操作基础，但不必具备编程基础。掌握一门语言最好的方式就是实践。本书的着眼点是将基础的理论知识讲解和实践应用相结合，使读者在理解面向对象的思想上，快速掌握 Java 编程技术。

本书作者由具有多年丰富教学经验和实践经验的一线教师组成。本书在总结作者多年教学和科研成果基础上，参考国内外 Java 先进程序设计思想及案例的基础上完成的，适合 Java 初学者和进阶者阅读。

目前市场上有关 Java 面向对象程序设计的图书很多，但本书有以下独到之处：

(1) 内容组织合理。针对常用的 Java SE 与 Java EE 两个平台版本进行内容组织，强调知识的系统性、连贯性、实用性。基本概念、编程方法由易到难逐层展开，内容表达环环相扣，读者易学易用。

(2) 强调面向对象的分析思路和设计思想。通过生动的实例阐明封装、继承、多态等相关概念，以典型的例子再现封装、继承、多态等的应用。

(3) 详略得当。由于 Java 面向对象开发涉及知识面比较广，本书仅对关键部分进行详细介绍，其他部分点到为止，以使读者的注意力能够集中到关键环节。

(4) 问题定义清晰，解题思路明确。对于比较复杂的案例，对其分析、设计过程及出现的问题都作了全面的介绍，把编程理论和编程实践完美地结合在一起。

(5) 详尽的实现步骤。本书配有大量的实现过程截图和实现步骤说明，以便读者可以通过阅读自行完成项目开发。

本书共 14 章，内容包括：绪论，类与对象，继承与多态，多线程程序设计，输入输出和异常处理，集合与泛型，图形用户界面，网络通信编程，数据库编程，Web 应用程

序开发基本知识，JSP 基础知识，Servlet，JavaBean 和 JSP 项目实训。其中，1～10 章由解绍词、杜伟奇编写，11～14 章由金霜编写，解绍词负责全书统稿。

由于时间仓促加上作者水平有限，书中实例虽然经过了多次测试，但难免出现不足之处，恳请广大读者给予批评指正。

目　录

第1章 绪 论

1.1 面向对象程序设计

1.1.1 面向对象程序设计思想的诞生

随着软件复杂度的提高，以及互联网的迅猛发展，原先面向过程的软件开发方式已经越来越无法满足软件开发的需要，面向对象的软件开发模式应运而生。作为应对软件危机的最佳对策，目前面向对象(object oriented，OO)技术已经引起人们的普遍关注。许多编程语言都推出了面向对象的新版本，一些软件开发合同甚至指明必须使用基于OO的技术和语言。下面简要列出OO技术的发展历程。

(1)诸如“对象”和“对象的属性”这样的概念，可以追溯到20世纪50年代初，首先出现于早期关于人工智能的著作中。直至1966年，具有当时更高级抽象机制的Simula语言的开发代表了OO技术的实际发展。

(2)Simula语言提供了比子程序更高一级的抽象和封装，并且为仿真一个实际问题，引入了数据抽象和类的概念。20世纪70年代一些科学家吸取了Simula类的概念，开发出了Smalltalk语言。

(3)几乎在同时，“面向对象”这一术语被正式确定。在Smalltalk中一切都是对象，即某个类的实例。最初的Smalltalk世界中，对象与名词联系紧密。

(4)Smalltalk语言还影响了20世纪80年代早期和中期的很多面向对象语言，如Objective-C(1986年)、C++(1986年)、Flavors(1986年)、Self(1987年)、Eiffel(1987年)。同时，面向对象的应用领域也被进一步拓宽，对象不再仅仅与名词相联系，还涉及事件和过程。

(5)随着互联网的迅猛发展，Sun公司于1995年推出了纯面向对象的Java语言。自此之后，OO技术在开发中越来越占主导地位。

1.1.2 面向对象的开发方法

目前，面向对象开发方法的研究的成熟度越来越高，国际上已有不少面向对象的产品出现。面向对象的开发方法有Booch方法、Coad方法和OMT方法等。

1. Booch方法

Booch方法最先描述了面向对象软件开发方法的基础问题，指出面向对象开发是一种在本质上区别于传统的功能分解的设计方法。面向对象的软件分解跟人对客观事务的理解更加接近，而功能分解只通过问题空间的转换获得。

2. Coad 方法

Coad 方法是 1989 年由 Coad 和 Yourdon 提出的面向对象开发方法。该方法的主要优点是通过多年来大系统开发的经验与面向对象概念的有机结合，在对象、结构、属性和操作的认定方面，提出了一套系统的原则。该方法完成了从需求角度进一步进行类和类层次结构的认定。尽管 Coad 方法没有引入类和类层次结构的术语，但事实上已经在分类结构、属性、操作、消息关联等概念中体现了类和类层次结构的特征。

3. OMT 方法

1991 年，由 James Rumbaugh 在《面向对象的建模与设计》一书中提出了 OMT 方法，该方法是一种新兴的面向对象的开发方法，对真实世界的对象建模可以说是开发工作的基础所在，然后围绕这些对象使用分析模型来进行独立于语言的设计。面向对象的建模和设计促进了对需求的理解，有利于开发更清晰、更容易维护的软件系统。该方法为大多数应用领域的软件开发提供了一种实际的、高效的保证，是努力寻求一种问题求解的实际方法。

4. 统一建模语言

软件工程领域在 1995～1997 年取得了前所未有的进展，其成果超过了软件工程领域过去 15 年的成就总和，其中最重要的成果之一就是统一建模语言(unified modeling language，UML)的出现。UML 将是面向对象技术领域内占主导地位的标准建模语言。

UML 不仅统一了 Booch 方法、OMT 方法、OOSE 方法的表示方法，而且对其作了进一步的发展，最终统一为大众接受的标准建模语言。UML 是一种定义良好、易于表达、功能强大且普遍适用的建模语言。它融入软件工程领域的新思想、新方法和新技术。它的作用域不仅仅限于支持面向对象的分析与设计，还支持从需求分析开始的软件开发全过程。

1.1.3 面向对象程序设计的三大特征

学习程序设计语言的关键在于学习其编程思想。这一点，从 Bruce Eckel 的 *Thinking in Java* 一书中可以体会到。那面向对象语言的编程思想体现在哪儿呢?怎样真正地“Thinking in Java”，而不仅仅是“Programming in Java”呢?起点就在于其三大特征——封装、继承、多态。

有人认为采用封装、继承、多态语法写出一个 Java 程序，就完全掌握了面向对象语言。恐怕这只能说仅仅学会了用 Java 语法写程序而已。事实上，封装、继承、多态是一种设计理念、一种程序艺术，与程序语言毫无关系。如果真正透彻地理解了这三大特征的真谛，即使不用 C++、Java、Objective-C 等面向对象的语言，用 C 语言也能写出面向对象的程序。在面对一个项目时，有经验的开发者可以立刻在脑海里构思出怎样从软件角度设计它，如何将类与类之间关联起来，怎样用封装、继承、多态等机制勾画出程序的基本架构，以及优化程序架构。至于一些细节问题，如采用什么语言，C++还是 Java，从语法上怎样来实现，这些都不是关键，毕竟语言只不过是程序设计思想的一种表现形式而已。

当然，要达到这种集设计与编程为一体的境界，还需要脚踏实地、稳扎稳打地从编程中逐步提高，在实践中成长。

本书将把这三大特征的设计理念作为一条主线，一步一步贯穿到后面的“类与对象”“继承与多态”章节中，以实例的方式逐步、详细地讲解这三大特征在程序设计中是如何体现的，该怎样去运用它们。下面简单介绍这三大特征。

1. 封装

封装，即将对象的数据和基于数据的操作封装成一个独立性很强的模块。封装是一种信息隐蔽技术，使得用户只能见到对象的外特性，而不能见到对象的内特性。封装的目的就是将对象的使用者和所有者分开，使用者不必知道对象的内部细节，只需通过对象所有者提供的通道来访问对象。

通俗地解释就是，把对象不需要对外提供的数据隐藏起来，对外形成一道屏障。数据如何隐藏、隐藏在哪儿?答案是隐藏在类里，基于数据的操作也隐藏在类里。这样一来细节都能隐藏起来。那如果外界想对封装的数据进行访问该怎么办呢？这时就需要借助于对象定义的公共接口，这种公共接口就如同与外界沟通的一个桥梁。以下举例说明封装的概念。如果你是“学生”这个类的一个对象，具有姓名、性别、学号、英语成绩、C 语言成绩、高数成绩等数据，一般来说，这些都是你的私有信息，别人无权知道。如果班长要统计全班成绩并排名又如何操作呢?这时就需要得到你的允许，即把成绩公开。可以定义一个公有的方法，如 getGrade()，通过这个方法访问你的某些私有数据，班长就可以得到他需要的成绩，当然，他不需要的信息仍然封装在类里，没有必要公开。

又如另一个对象封装的例子，图 1-1 中椭圆形表示数据(属性)、矩形表示操作(方法)，公司里的一个部门就是一个对象，三个对象有其封装好的数据及对数据的操作(方法)。以财务部为例，财务数据属于隐蔽信息，如果销售部门想看财务数据，一般是不允许的，但经上级批准，也可以通过公有的接口方法“管理财务”操作来实现。

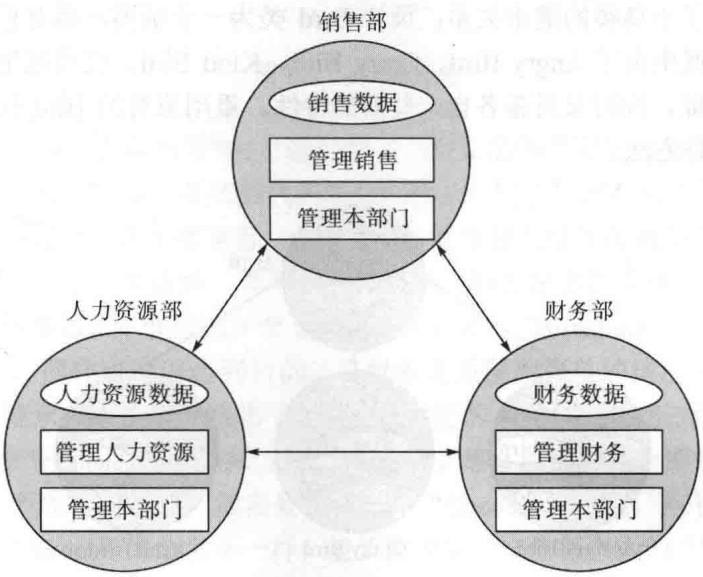

图 1-1　对象的封装示意图

2. 继承

继承，也就是在现有类的基础上创建新类，并在其中添加新的属性和功能。现有类与新类之间是一种一般与特殊的关系，现有类具有该类及其新类的共同特征，而新类还具有一些特殊特征。

类的封装引出继承的概念。日常生活中的类通常可以再派生出新类，如人类可分为白种人、黄种人、黑种人子类；交通工具类可分为轿车、卡车、公共汽车等子类。可以说这就是面向对象最大的优点，而代码重用的主要方式就是继承(后续章节还会提到代码重用的其他方法)。继承是在封装的基础上实现的，前面提到最好把一个封装好的类放在一个.java文件里，这么做的目的是出于方便类的重用，可以直接在这个类的基础上派生出子类。该子类可以使用现有类的所有功能，并在无须重新编写现有类的情况下对这些功能进行扩展。试想要设计一个关于狗的卡通(cartoon)程序，首先想到的是定义一个 CartoonDog 类，原先定义的 Dog 类已经具有了普通狗的特征，就没有必要再从头写代码，可以直接继承 Dog 类，在此基础上添加一些卡通的特性，生成一个新的 CartoonDog 类，既省时又省力。

Java 语言中通过继承创建的新类称为“子类”或“派生类”，被继承的类称为“基类”、“父类”或“超类”。继承的过程就是从一般到特殊的过程。在继承过程中子类继承了父类的特性，包括方法和变量；子类也可修改继承的方法或增加新的方法，使之更适应特定要求。继承使代码可以重用，使得数据、方法的大量重复定义在一定程度上得以避免，使系统的可重用性得到了保证，促进了系统的可扩充性，同时也使程序结构清晰、易于维护，提高了编程效率。

下面以《愤怒的小鸟》游戏为例。假如你是此游戏的开发者，当游戏版本不断更新，如 Angry Bird、Space Angry Bird、Crazy Bird……如果游戏的每一个版本都从头开始开发，重复劳动、开发时间加长、效率低下等问题都不可避免，而采用面向对象的设计就是一种比较好的解决方案。

图 1-2 展现了小鸟类的继承关系，设计 Bird 类为一个基类，具有普通鸟的特征，在 Bird 类的基础上派生出了 Angry Bird、Crazy Bird、Kind Bird。这些派生出的子类继承了 Bird 类的所有功能，同时又具备各自一些新的特性。重用原有的 Bird 代码，不失为一种提高开发效率的好方法。

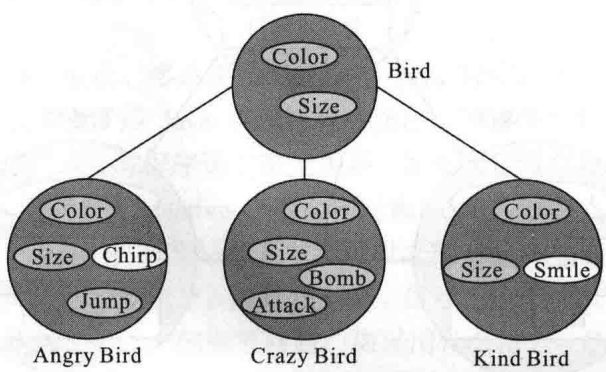

图 1-2　类的继承示意图

3. 多态

多态，即在一个程序中同名的不同方法可以共存，子类的对象可以响应同名的方法，但具体的实现方法不同，完成的功能也不同。

从字面上理解“多态”稍有难度。事实上，“多态”源于现实生活。如生活中常提到“打”这个动词，它可以组词为“打捞”“打扫”“打架”等，这就是“多态”，同一个“打”的行为有不同的反映。多态就是同一个方法可以用来处理多种不同的情形。在程序中，“多态”可以理解为父类与子类都有的一个同名的方法，针对继承关系下不同对象可以采取不同的实现方式。

再以《愤怒的小鸟》游戏为例，如图 1-3 所示，功能“Shoot”是三个类都具有的，以射向隐藏好的小鸟。Angry Bird 和 Crazy Bird 的对象，如红色鸟、小蓝鸟、白鸟，同样都具有“Shoot”功能。然而它们发射后执行的方式与效果是截然不同的，小蓝鸟弹出后分离出攻击力更强的三只小鸟，白鸟弹出后会像炸弹一样爆炸，这就是程序中的多态。需要注意的是，多态是以继承为基础的，只有继承关系下的类才具有多态特性。

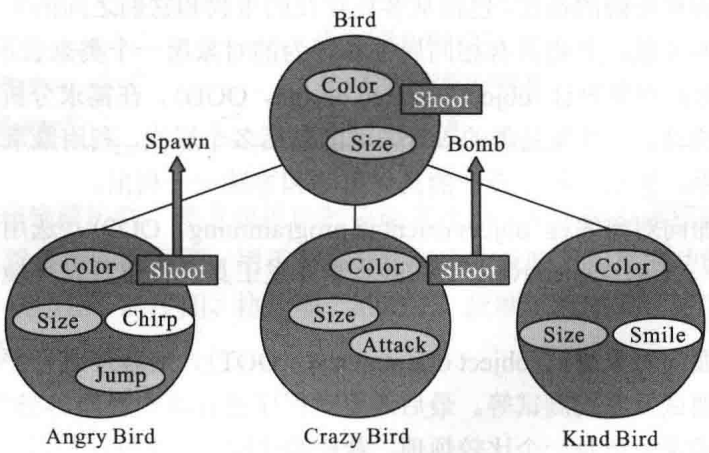

图 1-3　多态示意图

那么，多态到底有什么作用呢?由上述可知，一些实现细节可通过封装隐藏起来，使得代码模块化；继承可以扩展已存在的代码模块(类)。它们的目的都是为了代码重用。继承虽然已经扩展了功能，但还不够丰富，多态的引入就是要在继承的基础上进一步实现变异的可能性，增强程序的可扩展性。简单地说，就是一种方法多种实现。

那么，如果不用多态，简单地把 3 个 Shoot 重新命名为“Birdshoot”“Angrybirdshoot”“Crazybirdshoot”，同样也可以达到目的。在继承关系层次简单的情况下多态的优越性是无法体现的。在继承关系复杂的情形下，假设《愤怒的小鸟》接二连三地派生出不同的版本，如 PC 版、季节版、手机版、PC 版 2、季节版 2 等，如果同一个游戏开发小组的成员每人承担其中一个版本的工作，都需要实现一个“Shoot”的功能，组长负责最后的版本整合。试想如果“Shoot”的命名不一样，组长就得牢牢记住每个人的“Shoot”方法名，如“Birdshoot”“Angrybirdshoot”“Crazybirdshoot”“Angrybirdshoot2”“Crazybirdshoot3”……

以便在他的整合程序里调用。方法名少时还好处理，如果数量较多时，不仅容易记错名字，而且将极大地提高工作难度。所以，他最大的希望就是大家在开发前定好规矩，都统一命名为“Shoot”，然后把“Shoot”的定义与具体实现分离开来。无论组员怎样去实现，如何扩展程序，与他都没有关系，他只管关心“Shoot”这个公共接口，只需记住“Shoot”这个词，无须考虑实现的细节，执行时就能够自动调用所有子类的“Shoot”功能。同时，组员们只管写自己的程序。这时，多态就是最好的解决方案。可见，多态使面向对象语言具有了灵活性、扩展性、代码共享的特征，把继承的优势发挥得淋漓尽致。

1.1.4 面向对象的程序设计

面向对象的软件开发是一项巨大的工程，也是一门专业学科，仅仅依靠学习语法知识就立即进行代码编写是无法达到开发要求的。在实际软件开发过程中，面向对象的软件开发需要经历一个从需求分析、设计、编程、测试到维护的生命周期，需将面向对象的思想渗透软件开发的各个方面。以下简要介绍面向对象软件开发的几个基本流程。

第一阶段：面向对象需求分析(object oriented analysis，OOA)，需要系统分析员对用户的需求做出分析和明确的描述。包括从客观存在的事物和它们之间的关系归纳出有关的类以及它们之间的关系，并将具有相同属性和行为的对象用一个类来表示。

第二阶段：面向对象设计(object oriented design，OOD)，在需求分析的基础上，对每一部分进行具体的设计。首先是类的设计，可能包括多个层次，利用继承和组合等机制设计出类的层次关系；然后将程序设计的具体思路和方法一一列出。

第三阶段：面向对象编程(object oriented programming，OOP)，选用适当的面向对象编程语言(如 C++、C#、Objective-C 或 Java)和开发工具，设置开发环境进行代码的编写工作。

第四阶段：面向对象测试(object oriented test，OOT)，对程序进行严格的测试，包括单元测试、集成测试及系统测试等。最后还要对程序进行维护管理。

面向对象的需求分析是一个比较烦琐、漫长的过程，主要包括与用户沟通交流、收集信息、整理需求等过程。相关的软件工程课程会有详细的讲解。

本节主要讲述如何根据面向对象需求分析结果，将类与类之间的联系找出来，用 UML 设计出类的层次结构。这部分设计是编程的基础，只有理解类设计的主导思想，设计出清晰、合理、扩展性强的类结构，才能完成好面向对象程序的编写。在实际软件开发中，开发人员在编写代码之前，进行面向对象设计工作是必不可少的步骤。

1. 类的建模

通常使用统一建模语言(UML)来设计类的结构，它不是编程语言而是为计算机程序建模的一种图形化“语言”。所谓“建模”就是勾画出工程的蓝图，就像盖房子需要首先设计出房子的模型一样，软件工程“建模”就是在考虑实际的代码细节之前，用 UML 图示将程序结构在很高的层次上表示出来。UML 除了能帮助进行程序的结构设计外，对程序具体工作流程的理解也是非常有帮助的。因为对于大型的程序，仅仅看源代码就想弄清各部分之间的联系还是有一定难度的，而 UML 提供了一种直观的方法让我们了解程序概