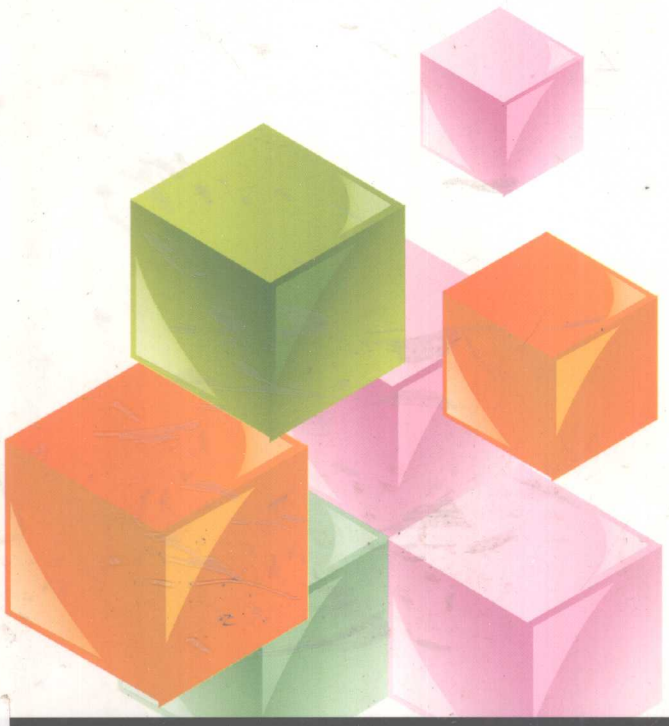
开发专家
之 Java语言程序设计篇

Java 面向对象程序设计教程

主编 黎 明 杨先凤

- Java导论
- Java语言基础
- Swing组件
- Applet应用程序
- 多线程程序设计
- 数据库编程
- 网络编程



电子科技大学出版社

Java 面向对象程序设计教程

主　编　黎　明　杨先凤

副主编　陈汶滨　王兵　朱晓梅　孙瑜

电子科技大学出版社

图书在版编目（CIP）数据

Java 面向对象程序设计教程 / 主编 黎明，杨先凤.
成都：电子科技大学出版社，2005.9

ISBN 7-81094-921-7

I. J ... II. ① 黎... ② 杨... III. JAVA 语言—程序
设计 IV.TP312

中国版本图书馆 CIP 数据核字（2005）第 093916 号

Java 面向对象程序设计教程

主 编 黎 明 杨先凤

出 版	电子科技大学出版社（成都市建设北路二段四号，邮编：610054）
责任编辑	罗 雅
发 行	电子科技大学出版社
经 销	新华书店
印 刷	四川文瑞印务有限责任公司
开 本	787mm×1092mm 1/16 印张 21.125 字数 541 千字
版 次	2005 年 9 月第一版
印 次	2005 年 9 月第一次印刷
书 号	ISBN 7—81094—921—7/TP・480
定 价	31.50 元

前　言

随着网络技术的迅速发展和应用普及，Java 语言以其面向对象、平台无关性、多线程、安全性、健壮性等特征，逐步成为当前流行的网络编程语言之一，也是 21 世纪开发网络应用程序的受欢迎的优良工具。

本书将面向对象的基本理论与 Java 语言程序设计相结合，培养读者运用面向对象的思维方法去分析问题和解决问题，指导读者顺应网络时代对人才的新需求，在较短的时间内学会利用先进的 Java 工具开发软件产品。作者根据多年的教学及实践经验，对本书的内容取舍、组织编排和程序实例进行了精心设计。本书在写作风格上突出实用性，将基础知识与程序实例相结合，例子翔实丰富，具有较强的实用性和指导性。

本书内容丰富，共分为 14 章。主要内容包括：Java 语言基础、Java 程序控制结构、数组、类、对象、继承、多态、接口、包、事件处理、图形用户界面、Java Applet 程序设计、异常处理、多线程、输入与输出流、数据库编程、网络编程。

书中所有程序实例都经过上机调试并获得通过，每章附有习题。通过例题和习题帮助读者理解 Java 语言的基本概念和编程技巧。希望读者在学习本书的过程中，尽可能将书中的大量实例及每章习题进行上机练习。

本书适合作为高校相关专业师生教学用书，同时可以作为 Java 开发人员的参考用书。

本书第 1、3 章由西南石油学院黎明编写，第 2、5、6、8、11、13、14 章由杨先凤编写，第 4、7 章由陈汶滨编写，第 9 章由王兵编写，第 10 章由朱晓梅编写，第 12 章由孙瑜编写。黎明同志做了全书的统稿工作，杨先凤设计了全书结构，并对全书进行了认真和反复的修改。李建教授在百忙之中抽出宝贵的时间认真细致地审阅了全书，并提出了许多宝贵意见，编者深表感谢。在本书的编写过程中，得到了西南石油学院的领导、计算机科学学院的领导、软件工程教研室的全体同事、电子科技大学出版社、电子科技大学廖建明老师等的大力支持和帮助，在此一并表示衷心的感谢。

尽管作者在写作过程中投入了大量的时间和精力，但由于水平有限，书中错误难免，恳请广大读者批评指正。

编　者

2005 年 7 月

目　录

第 1 章　Java 导论

本章导读

本章将介绍面向对象编程的概念、Java 语言的发展历程、Java 语言的特点、Java 的开发运行环境、Java 语言对软件开发技术的影响及未来前景，两种类型的 Java 程序以及 Java 程序的编辑、编译与运行过程。

1.1　面向对象程序设计入门

如果以前从来没有使用面向对象语言，需要在开始编写 Java 代码之前先理解这些概念：什么是对象、什么是类及对象和类的关系怎样。

1.1.1　面向对象技术的提出

面向对象的编程思想由来已久，但真正意义上的纯面向对象编程语言目前只有 Java。本节将结合几种高级语言对面向对象程序设计思想进行简要介绍。

我们知道，所有的计算机程序均由两类元素组成：代码和数据。如何实现这两类元素的有效结合而形成可运行的程序，是多年来程序设计人员所探索的问题。最初，程序的构筑一般围绕“正在发生什么”而编写代码，这种方法被称为面向过程的编程。使用这种方法编写的程序都具有线性执行的特点。面向过程的编程模型可认为是代码作用于数据，像 Pascal、C 这样的过程式语言采用此模型是相当成功的。然而，使用面向过程的方法对小程序的编写可能是比较有效的，但当程序变得非常大且更为复杂时，就会出现种种问题，直至失去对代码的有效控制。由此对软件工程中的编程方法问题提出了新的要求。

为了管理不断增加的复杂性，另外一种编程方式被提了出来，即面向对象的编程(OOP，Object-Oriented Programming)。这种编程方式围绕“谁将受到影响”进行，即以代码的相关数据为核心点进行程序编写。面向对象的编程着眼于它的数据（即对象）和为此数据严格定义的接口来组织程序，程序实际上是用数据控制对代码的访问。这种方式的最大特点是代码与其相关数据被分离开来进行处理，有利于程序规模的扩大，而程序的可维护性得到增强。

1.1.2 面向对象的编程思想

前面提到的面向过程程序，它遵循面向过程的问题求解方法，其中心思想是用计算机能够理解的逻辑来描述和表达待解决的问题及其具体的解决流程。数据结构和算法是面向过程问题求解的核心所在。而面向对象技术则代表了一种全新的程序设计思路，其观察、表述、处理问题的方法，与传统的面向过程的编程方法不同。面向对象的程序设计和问题求解力求符合人们日常自然的思维习惯，尽量分解、降低问题的难度和复杂性，从而提高整个求解过程的可监测性、可控制性和可维护性，以此达到以较小代价和较高效率获得较满意效果之目的。

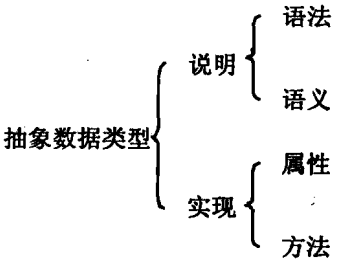


图 1-1 抽象数据类型

面向对象编程一个实质性的要素是抽象。

抽象表示使一个对象/类有别于其他对象/类的重要特性。抽象就是抽象出事物的本质特性而暂时不考虑它们的细节。过程抽象和数据抽象是常用的两种主要抽象手段。

一个抽象数据类型（Abstract Data Type，ADT）可以分解成 4 个部分，如图 1-1 所示。语法和语义构成抽象数据类型 ADT 的说明，让使用者了解这个抽象数据类型的特征。属性和方法构成 ADT 的实现，展示这个 ADT 是怎样做成的。

1.1.3 面向对象的概念

1. 对象

简单定义可以是："展示一些定义好行为的、有形的实体。"

以黑板为例：有明确边界、有形的、可见的实体；具有预先定义好的用途；可以讲课用，练习粉笔字等。但，对象也包括：机构、组织、单位等没有物理上的边界，却具有一个概念上的边界，是客观存在的，也称为对象。

根据面向对象方法的倡导者 Grady Booch 的理论，对象具有下列特性：

（1）具有一种状态。通过一系列属性和它们的值来表示；

（2）可以展示一种行为。行为是指在一定的期间内属性的改变；

（3）具有唯一的标识。每一个对象都有唯一的标识。即便是孪生兄弟或姐妹也不例外。

因此，可以认为对象是具有某些特殊属性（数据）和行为方式（方法）的实体。可以是有生命的个体，比如一个人或一只老虎。也可以是无生命的个体，比如一辆汽车或一台计算机。也可以是一个抽象的概念，如天气的变化或鼠标所产生的事件。

对象有两个特征：属性（Property）和行为（Behavior）。如：一个人的属性有：姓名、性别、年龄、身高、体重等，行为有：唱歌、踢球、骑车、学习等。

2. 消息

单一对象的存在并没有多大的作用，只有多个对象相互作用才会完成复杂的行为。对

象和对象之间是通过传递消息来完成相互通信的。

一个消息由三方面内容组成：

（1）消息的接收者，即消息的目标对象；

（2）接收对象采用的方法；

（3）执行方法所需用的参数（Parameters）。

例如：一辆自行车放在车棚中并没有什么用，当我们人骑上它，并加速时才体现它的作用，其中接收者（自行车），采用的方法（加速），所需的参数（上升到更高的档位）。

3. 类

具有共同属性和行为的一系列对象。世界充满了对象，有意识地识别周围的对象并将它们分类是一件有意义的事情。

面向对象方法将系统看作是现实世界对象的集合。对象代表一个实体，该实体有一个标识并展示一定数量的属性和行为。对一系列类似对象的共同的属性和行为的描述构成了类。一个系统可以包含任意数量的对象和类。面向对象的系统可以很容易地升级以反映系统的变化和扩展。

现在，软件开发中使用面向对象方法来代替过程方法是历史发展的必然趋势。面向对象方法提高了软件系统的稳定性，因为它包含规模和复杂度可变的问题领域。

4. 类与对象的关系

面向对象方法将系统看作是现实世界对象的集合。

面向对象系统是以类为基础的。一系列拥有共同属性和行为的对象可以归为一类。属性代表类的特性。行为代表可以由类来完成的操作。如：考虑一个叫“交通工具”的类。该类的属性有交通工具的构造、颜色、发动机能量（来源）等。类的行为有：启动、行驶、加速、停止。

对象是类的一个实例，它提供了类的属性和行为。比如：本田摩托车是交通工具类的一个对象。类与对象的关系如图 1-2 所示：

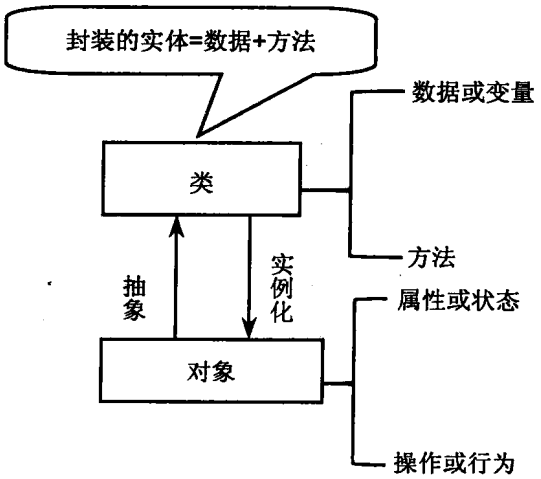


图 1-2　类与对象的关系图

1.1.4 面向对象编程的基本原则

一般说来，面向对象的系统至少需具备三大特性：封装性、继承性、多态性。将封装、继承、多态（包括重载）等面向对象方法应用于程序的开发工具和开发过程中，不仅可以加快开发的速度，还可极大地增强程序的可维护性和可扩展性，提高代码重用率。因此，在面向对象编程过程中需要遵循这三项原则。下面对它们分别作简要介绍：

1. 封装性

封装（Encapsulation）是将代码及其处理的数据绑定在一起的一种编程机制，该机制保证了程序和数据都不受外部干扰且不被误用。一个对象的基本要素包括属性和作用在属性上的操作（方法或事件）。对象的使用实现了数据抽象，它将一组数据和对这组数据的操作结合成一个内在的整体，不允许外界对这组数据任意进行访问，这里就用到了封装的原理。封装的目的是为了实现数据隐藏和数据保护，为对象提供一个对外操作的接口，外界只能从对象所提供的操作接口来认识和操作该对象。

2. 继承性

继承（Inheritance）是一个对象获得另一个对象的属性的过程。继承很重要，因为它支持了层级分类的思想。众所周知，大多数事物均可按层级（即从上到下、从高到低）分类管理。显然，如果不使用层级的概念，在进行描述时，我们就不得不分别定义每个事物的所有属性。使用了继承，一个对象就只需定义使它在所属类中独一无二的属性即可，因为它可以从它的父类那里继承其他所有的通用属性。所以，完全可以这样说，正是继承机制使一个对象成为一个更具通用性的类的一个特定实例成为可能。

继承是现实生活中一个非常容易理解的概念。在面向对象的程序设计方法中，引入继承机制的目的在于：其一，避免可公用代码的重复开发，减少数据冗余；其二，增强数据的一致性，尽量降低模块间的耦合程度。

3. 多态性

多态（Polymorphism）来自于希腊语，表示“多种形态”，即允许一个接口被多个同类动作所使用的特征，具体使用哪个动作与应用场合有关。所谓多态性就是当不同的对象收到相同的消息时，产生不同动作的特性。这里所说的消息可以理解为方法或事件。通俗地讲，多态性就是使用一个名称来定义不同的方法，这些方法执行类似的但又不同的操作，即以相同的接口来访问功能不同的函数，从而实现“一个接口，多种方法”。在同一个类中可有许多同名的方法，但其参数数量与数据类型不同，而且操作过程与返回值也可能会不同。称为多态或同名异式，解决了其他语言中不能重名的问题。

4. 多态性、封装性与继承性的相互作用

如果使用得当，在由多态性、封装性和继承性共同组成的编程环境中可以写出比面向过程模型环境更健壮、扩展性更好的程序。精心设计的类层级结构是实现代码可重用性的基础；封装可以使你在不破坏依赖于类公共接口的代码基础上对程序进行升级迁移；而多态性则有助于编写清晰、易懂、易读、易修改的程序。