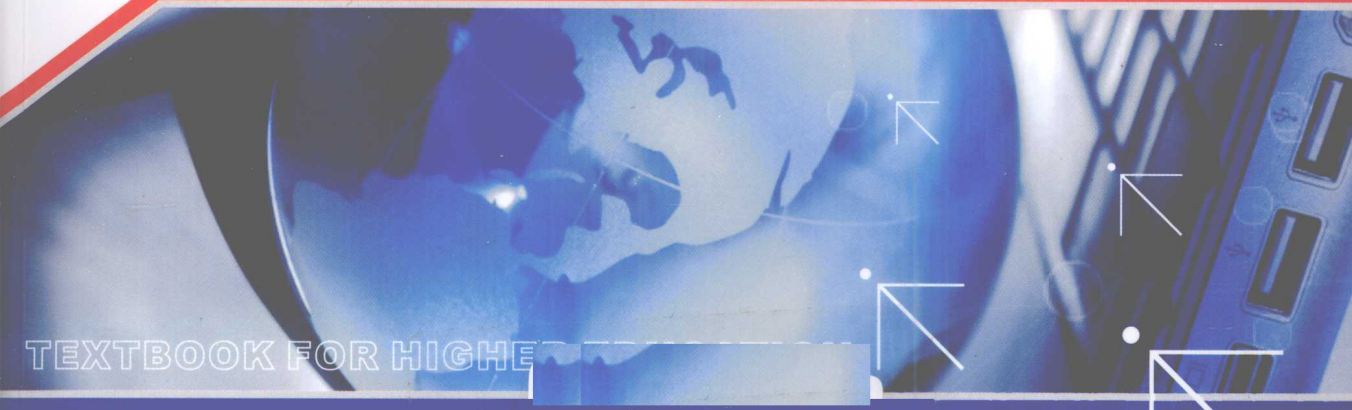
软件系统开发指导教程系列丛书

面向对象编程基础

——Java 语言描述

马春燕　张　涛　编



西北工業大學出版社
NORTHWESTERN POLYTECHNICAL UNIVERSITY PRESS

软件系统开发指导教程系列丛书
教育部国家高等学校特色专业建设项目
软件工程——软件系统开发专业方向建设项目 支持

面向对象编程基础
——Java语言描述

马春燕 张 涛 编

西北工业大学出版社

【内容简介】 本书为软件工程专业《软件系统开发指导教程系列丛书》之一。首先，本书介绍了面向对象基本概念和特点，以及根据需求说明设计类图的方法，重点围绕面向对象程序的封装性、继承性、多态性和关联关系等特性，阐述应用 Java 语言的面向对象编程实现技术。其次，本书还详细介绍了抽象类与接口、设计模式等面向对象设计的高级主题，以及 I/O 编程、GUI 编程等高级 Java 语言编程技术。

本书通过大量具体示例及贯穿全文的综合应用案例来阐述理论知识，具有较强的工程性和应用性。

本书可作为高等院校软件工程教育核心教材，也可作为计算机专业及相关专业的课程教材，以及软件开发人员参考用书。

图书在版编目(CIP)数据

面向对象编程基础：Java 语言描述/马春燕，张涛编. —西安：西北工业大学出版社，2010.6
（软件系统开发指导教程系列丛书）
ISBN 978-7-5612-2830-2

Ⅰ.①面… Ⅱ.①马…②张… Ⅲ.①JAVA 语言—程序设计 Ⅳ.①TP312

中国版本图书馆 CIP 数据核字(2010)第 124480 号

出版发行：西北工业大学出版社
通信地址：西安市友谊西路 127 号 邮编：710072
电　　话：(029)88493844　88491757
网　　址：www.nwpup.com
印 刷 者：陕西向阳印务有限公司
开　　本：787 mm×960 mm　1/16
印　　张：19.75
字　　数：419 千字
版　　次：2010 年 6 月第 1 版　2010 年 6 月第 1 次印刷
定　　价：33.00 元

《软件系统开发指导教程系列丛书》编委会

出版说明

2001年12月，教育部批准成立了35所国家示范性软件学院，旨在为国家软件产业的发展培养多层次、实用型、高水平、具有国际竞争力的专业人才，以适应社会对软件高端人才的需要。各软件学院在这样的大环境下，纷纷挖掘自身的优势，采用各种先进的教学模式，注重教育与教学改革，已形成了各具特色的软件工程教育体系。广大教师也在这样的教育教学中不断对传统软件工程教学进行总结，并汲取国外先进教学中的精髓，取长补短，积累了丰富的教学实践经验。

有鉴于此，我们组织策划了《软件系统开发指导教程系列丛书》。本系列丛书共10册，包括《计算机编程导论》《计算机系统导论》《面向对象编程基础——Java语言描述》《交互式用户界面设计与测试》《数据库系统——设计与应用》《数据结构与算法》《系统级编程》《网络与分布式计算》《软件规范测试与维护》《软件项目组织与管理》。本系列丛书的出版意欲将广大教师在培养国际化、应用型软件工程化人才的教育教学中积累的经验进行推广与传播。特别是将这种教学理念在一些外语基础薄弱，还不能适应双语教学的学生中推广。

本系列丛书的分册主编均是各软件学院活跃在教学第一线的教师。他们都具有多年的教学经验、深厚的专业功底和丰富的软件开发实践经验，因而保证了这套丛书理论与实践兼备，教与学互动，特色鲜明。

为确保本系列丛书的质量，我们邀请了软件学院的一些专家、教授，成立了《软件系统开发指导教程系列丛书》的编委会。他们当中有全国知名的计算机专家、科学院院士，也有在软件工程教育中有着丰富工作经验的教授、博导，也不乏具有出国留学经历的教师，他们对国际与国内该领域的技术状况、应用环境都十分了解。这些均有利于从总体上把握本系列丛书内容向着适应国内需要并与国际接轨的方向发展。

我们将以高度的社会责任感，投入满腔的工作热忱，精益求精，为广大读者提供高质量的精品图书。

西北工业大学出版社

2009年3月

前　言

为了满足我国软件产业发展需要，培养高层次软件人才，国家教育部先后在全国设立了35所国家示范性软件学院。笔者所在的软件学院以培养国际化、工程型、复合型软件人才为目标，积极引进了国外先进的课程体系。在多年的教学实践过程中，笔者积极探索对国外课程体系的吸收和改进，并编写了本书。

本书的主要指导思想是：

(1)培养学生用面向对象的思想进行软件设计的能力；

(2)培养学生应用面向对象机制进行规范化编程的能力；

(3)培养学生熟练运用Java语言和专业工具进行面向对象软件开发的能力。

在国内属于软件工程这一领域的图书中，要么是单纯介绍面向对象设计理论的书籍，要么只是Java语言的参考书，而本书则将面向对象设计与编程进行了完美结合。虽然，Java语言是辅助面向对象设计的编程语言，但是通过本书对它清晰的讲解，能使读者更加理解Java语言编程的精髓。

本书针对面向对象软件设计和编程，讲解了UML技术、面向对象设计模式、Java面向对象编程技术、Java GUI编程等软件设计和开发实用技术。在讲解面向对象基础理论和Java语言基础知识的同时，提供了丰富的应用示例，详细讲解知识的应用方法，强化和培养学生对知识的灵活应用能力和软件开发能力；书中还讲解了Eclipse开发工具、Javadoc技术和Java编码规范，强调对学生规范化软件开发和职业素养的培养，并且以企业实际项目进行案例分析，重视学生职业实战能力和工程素质的培养。

本书适合作为全国示范性软件学院软件工程专业及计算机相关学科本科教材，参考教学时数为40学时，配套实验课时72学时。若课内实验机时安排不足，可安排课内机时和课外机时各36学时。

为了便于读者使用，笔者为本书制作了电子课件。需要的读者可登录西北工业大学出版社网站下载。

本书第1～10章由马春燕编写，第11～12章由张涛编写。朱怡安教授、蒋立源教授、吴广茂教授、王丽芳教授等对本书的编写提出了不少宝贵意见和建议，笔者在此向他们表示衷心的感谢。

书中难免有不妥之处，敬请读者批评指正。

编　者

2010年3月

目　录

第 1 章　面向对象基础

1.1　对　　象

在学习软件对象的概念之前，可以先来看看现实世界中对象的概念。在现实世界中的任何有属性的单个实体或概念，都可看做是对象。对象可以是有形的，例如，学生张三、顾客李四、王教授和 402 教室等；对象也可以是无形的（即概念对象），例如，一个银行账户、一个客户订单、学生选修的课程、学生获得的学位等。

一般现实世界中的对象都拥有属性，由属性来描述对象的静态特征，例如，学生张三具有属性：姓名、学号、成绩等；顾客李四具有属性：姓名、账户等；银行账户具有属性：用户名、余额等；订单具有属性：货品名、单价、数量等。

在现实世界中，往往要对对象实体的属性进行操作。例如，打印学生张三的姓名、学号和成绩，查询顾客李四的账户余额，打印订单的价格等。对对象属性的操作，描述了对象的动态特征。

在用面向对象技术搭建软件的过程中，可以将现实世界描述的问题域中的对象抽象为具体的软件对象，通过一系列软件对象以及它们之间的互操作，来完成用户要求的功能。如图 1.1 所示，一个软件对象是一个软件结构，它封装了一组属性和对属性进行的一组操作，它是对用户需求中描述的对象实体的一个抽象。因此，软件对象，其实就是现实世界中对象模型的自然延伸。这里将软件对象简称为对象，本书以后章节中所指的对象，都是指软件对象。

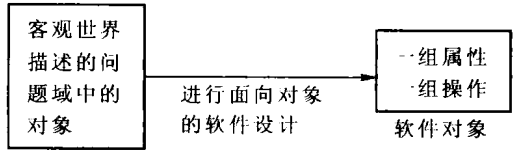


图 1.1　对象的概念

1.2　面 向 对 象

面向对象（Object Orientation）是一种软件开发方法，它包括利用对象进行抽象和封装的类、通过消息进行的通信、对象的生命周期、类层次结构和多态技术[1]等。对象是核心概念，对

象之间通过消息进行通信来完成相应的功能。它是现实世界中实体或概念的软件模型。如图 1.2所示为对象之间的联系。

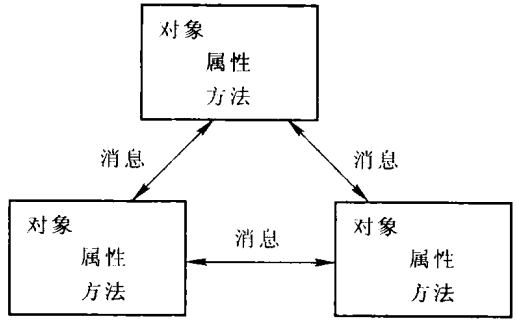


图 1.2　面向对象技术

通过面向对象的技术，可以较容易地实现一个现实世界中问题域的抽象模型。现以银行业务员为顾客提供存款和取款的服务操作来说明。为了实现顾客存款和取款的操作，银行业务员需要知道顾客的用户名、密码和账户余额等顾客账户详细信息，以便确定顾客是否有支取一定金额的权利。

当使用面向对象技术建立软件模型时，对于问题域中的顾客和账户，都可以成为建模对象，因为它们都是具有属性的实体或概念。顾客的属性包括用户名、身份证号和密码等；账户的属性包括卡号、账户余额等。从需求抽取的对象模型如图 1.3 所示，顾客对象和账户对象通过消息进行通信，银行业务员可以很方便地通过顾客对象访问其账户信息，以便完成顾客存款和取款等操作。

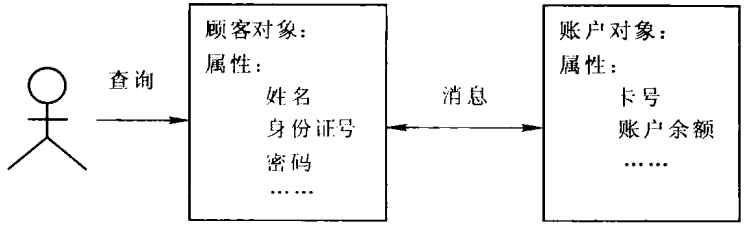


图 1.3　访问顾客账户信息的面向对象技术

从上例可以看出，进行面向对象的软件设计，就是将所要解决的现实世界的问题抽象为软件对象，对象和对象之间通过消息进行通信来完成一定的功能。

总之，面向对象技术是以对象为中心、以消息为驱动的软件建模技术，它将需求域中出现的实体或概念以及它们之间的关系抽象为对象及消息，对象之间通过消息进行通信，来完成相应的操作。

1.2.1　类与对象

几乎所有现实世界中的东西，都可以在软件中建模为对象。例如，可以将一台电视机，建模为一个对象；在更抽象的环境中，可以将一个二维“点”，建模为一个对象；或者更一般地，需求中描述的所有拥有属性的实体或概念，都可以建模为一个对象。每个对象都有一组和它相关联的属性（又称为数据或状态），如电视机的属性包括型号、频道和指示开关的状态等；二维“点”的属性包括 x 坐标和 y 坐标等。如图 1.4 所示给出了三个二维“点”对象。

对象除了拥有属性之外，还拥有建立在属性之上的方法（又称行为、操作或服务），提供操作属性的方法是对象的职责。电视机需要提供访问其型号的方法以及修改其状态的方法；二维“点”需要提供访问其 x 坐标值和 y 坐标值的方法。当然，对象提供哪些方法，要依据具体的系统需求来确定。通常对象都提供了访问和修改其属性的方法。当一个对象被使用时，主要关心的是它提供了哪些操作。

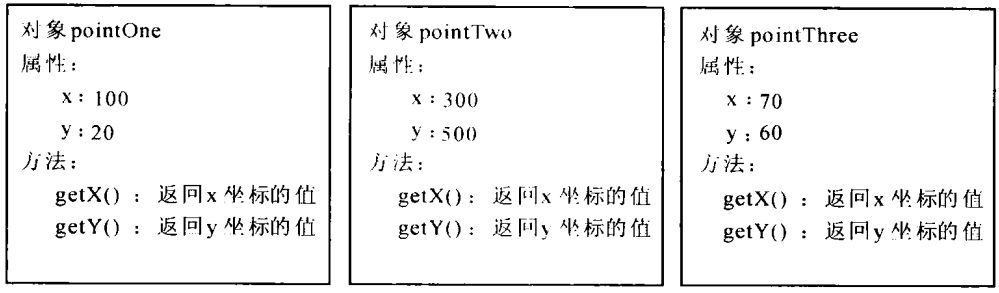


图 1.4　二维点对象举例

对象包含信息（即描述对象的属性）和用于处理对象的方法。任何对象都可包含其他对象，称为子对象，这些子对象又可包含其他子对象，直到问题域中最基本的对象被揭示出来。例如，一辆汽车可被看成一个对象，它包含发动机等许多组件。发动机又可以被看成一个子对象，它也可能包含其他子对象。至于对象要细化到哪一级，则取决于现实世界中系统的需求。

由于在现实世界中，任何实体都可归属于某类事物，任何对象都是某一类事物的实例，因而可以将所有二维“点”对象的共性抽取出来，形成类 Point。类 Point 是对所有二维“点”对象特征的描述或定义，所有的二维“点”都有 x 坐标和 y 坐标的属性，以及建立在该属性之上的操作 getX()和 getY()，这里假设 x，y 的数据类型为浮点型（float）。类 Point 如图 1.5 所示。

在软件中，类就是一个创建对象的空模板（即属性没有具体的值），它定义了通用于一个特定种类的所有对象的属性和方法；对象是类的实例，对象提供了类模板中属性的值，即给类中的属性赋予确定的取值，便得到该类的一个对象。如图 1.4 所示的对象 pointOne，pointTwo 和 pointThree 都是类 Point 的实例。

在面向对象系统中，每个对象都属于一个类。属于某个特定类的对象称为该类的实例。

因此，常常把对象和实例当做同义词(本书后续内容将对对象和实例不加区分)，即实例是从某类创建的一个对象。

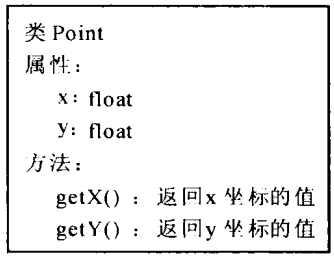


图 1.5　类 Point 图示

从程序设计的角度看，类是面向对象程序设计中最基本的程序单元。类实际上定义的是一种数据类型，这种数据类型就是对象类型。上述类 Point 就是对象类型，可以使用类名称 Point 来声明对象变量，即

Point　pointOne;

Point　pointTwo;

Point　pointThree。

当程序在内存中运行时，对象被创建并存在。在某一时刻，一个类可能只有一个对象存在，也可以有任意多个对象存在。当编写程序时考虑的是类，但程序运行时处理的是分配了内存空间的对象。

通过上面的例子，阐释了如何去认识软件建模中的类和对象。下面给出对象和类的概念[1]。

对象(Object)。对象是面向对象的基本单位。对象是一个拥有属性、行为和标示符的实体。对象是类的实例，对象的属性和行为在类定义中定义。

类(Class)。类是一组对象的描述，这一组对象有共同的属性和行为。

在概念上，类与非面向对象程序设计语言中的抽象数据类型比较相似，但是，由于类同时包括数据结构和行为，因而更为全面。类的定义描述了这个类的所有对象的属性，也描述了实现该类对象的行为——类的方法。

基于上述类和对象的例子，可以帮助理解类和对象的概念及其区别。

1.2.2　属性

在面向对象的思想中，通常用属性(Attribute)(或者称之为状态和数据)来描述对象的特征，在具体的应用环境中，属性有其确切的对应值。例如 1.2.1 节中提到的，用来表示二维“点”pointOne，pointTwo 和 pointThree 特征的 x 坐标和 y 坐标就是属性。

对象的属性，用于保持对象的状态信息，可以简单到是一个布尔型变量，记录“开”或“关”

状态；也可以是一个复杂的数据类型变量——对象类型。例如，如图 1.6 所示的类 Triangle，它包含五个属性，前三个属性 pointOne，pointTwo 和 pointThree 是对象类型（即 Point 类型的），后两个属性 perimeter 和 area 是浮点型（即简单的基本数据类型）。

在面向对象的编程语言里，这一组从属于某类对象的属性，是用变量来表示的，这些变量称为类的成员变量。

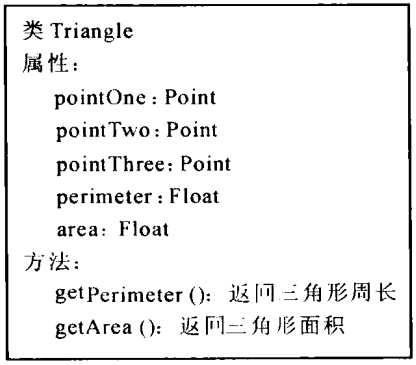


图 1.6　类 Triangle 图示

1.2.3　方法/操作

对象内部所包含的属性，仅仅只是对象的一部分，同时对象还需要提供一些对这些属性进行操作的方法。方法是操作的实现，用来实现对象的行为。对象的方法可以用来改变对象的属性，或者用来接收来自其他对象的信息以及向其他对象发送消息，因而这些方法通常作为类的一部分进行定义。

通常，更多关注的是一个对象能够提供的方法。例如，一个二维“点”Point 对象提供了访问其二维坐标的方法。对于使用 Point 对象的 Triangle 对象而言，关心的是 Point 对象提供了哪些方法，然后调用相应的方法实现计算三角形周长（getPerimeter）和计算三角形面积（getArea）的功能。更一般地来说，当使用面向对象的思想编程时，只关心一个对象提供了哪些方法，如果知道了一个对象提供的具体方法，就可以调用该对象的一些方法完成功能，以满足应用需求。所以，方法是一个对象允许其他对象与之交互的方式。

最后需要指出的是，对象方法是建立在对象属性之上的操作的实现。方法有多种类型，它包括给属性赋值的方法、获取属性值的方法，以及以某种方式处理属性并返回一个计算结果的方法等。在一些描述面向对象开发技术的著作中，也将方法（Method）称为行为（Behavior）、操作（Operation）、服务（Service）、函数（Function）等[2]。但是在面向对象的统一建模语言的一些著作中，通常行为、操作和方法是有区别的。行为是外界可见的对象活动，它包括对象如何通过改变内部状态，或向其他对象返回状态信息来响应消息[1]。操作是类的特征，用来定义

如何激活对象的行为。服务和操作只是名称上的区别。

1.2.4 消息(Message)机制

为了能完成任务,对象需要与其他对象进行互操作。互操作可能发生在同一个类的不同对象之间,或是不同类的对象之间。通过发送消息给其他对象,实现对象之间的互操作(在 Java 中,这是通过方法调用来完成的)。例如,当一个用户按下鼠标键,选择了屏幕上对话框里的一个命令按钮时,一条消息就被发给了对话框对象,通知它命令按钮被按下了。

对象之间通过发送消息进行交互,以消息激活已公布的方法,来改变对象的状态或请求该对象完成一个动作。在对象的操作中,当一个消息发送给某个对象时,消息包含接收对象去执行某种操作的信息。发送一条消息至少要包括说明接收消息的对象名、发送给该对象的消息名(即对象名、方法名);一般还要对参数加以说明,参数可以是认识该消息的对象所知道的变量名,或者是所有对象都知道的全局变量名。例如,对象 pointOne,提供一个方法 getX( ),可以发送消息 pointOne. getX( )来获取对象 pointOne 的 x 坐标。

当从面向对象的角度来思考问题时,会说一个对象向另一个对象传递了一个消息。Java 程序中的消息,实际上是对类的一些方法的调用,方法通过返回值来响应消息。虽然可以说是在调用类的方法,但最好是从消息传递的角度来思考,表达成 A 类传递了一个消息给 B 类。从具体的程序设计角度来讲,发送消息是通过调用某个类的方法实现的;收到消息是通过其他对象调用本对象的类的方法实现的。

1.3 面向对象程序的特点

面向对象程序的主要特点是封装性(Encapsulation)、继承性(Inherit)和多态性(Polymorphism)。本书在后续内容中将结合 Java 编程语言对面向对象程序的这三个特点进行分析。

1.3.1 封装性

在结构化程序设计中,例如 C 程序设计,函数能够不受限制地访问全局性数据,如图 1.7 所示。这样会导致函数和数据之间缺乏联系,大量的函数对全局变量的访问,导致程序结构不清晰,难以维护和修改。如果在程序投入使用后需要修改数据结构,通常会在整个应用中引起显著的连锁效应,例如,Y2k 危机,即千禧危机或千年问题。20 世纪 60 年代,由于计算机内存非常宝贵,所以编程人员一直使用月月/日日/年年或日日/月月/年年的方式存储和显示年份,对于公元 2000 年的 1 月 1 日,系统无法识别 00/01/01 是代表 1900 年的 1 月 1 日,还是 2000 年的 1 月 1 日,所有的软件都可能因为日期的混淆而产生资料流失、程序紊乱等问题,如此造成的损失无法估计。

面向对象程序设计将属性及对属性操作的方法封装在一起,形成一个相互依存、不可分离

的整体——对象。对系统的其他部分来说，属性和操作的内部实现被隐藏起来了，这就是面向对象的封装性。

对象是支持封装的手段，是封装的基本单位。面向对象的思想始于封装这个基本概念，即现实世界可以被描绘成一系列完全自治、封装的对象，这些对象通过一个受保护的公共方法访问其他对象。

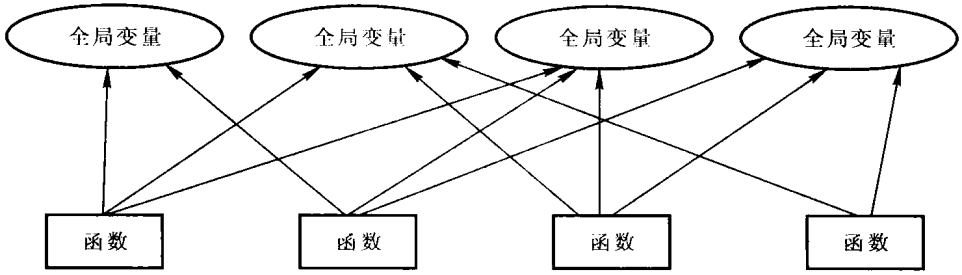


图 1.7　结构化程序设计

Java 语言(以下简称 Java)的封装性较强，因为 Java 没有游离于类之外的全程变量和全局函数。在 Java 中，绝大部分成员是对象，只有简单的数字类型、字符类型和布尔类型除外。而对于这些简单的基本数据类型，Java 也提供了相应的对象类型以便与其他对象交互操作。

在面向对象的思想中，每个类越独立越好。每个类都尽量不要对它的任何内部属性提供直接的访问。例如，在 Java 程序设计中，一般将属性的访问权限设为私有的，即只有对象内部的方法可以访问该对象的私有属性。类应该向外界提供能实现其职责的最少数目的方法，而且，向外界提供的公共方法应该尽量少地受到类内部设计变化的影响，即将所有类的封装最大化。

所以，封装保证了对象内部的数据信息细节被隐藏起来，对象以外的部分不能随意访问和修改对象的内部数据(属性)，每个对象的状态只能通过定义良好的公共方法才能改变，从而有效地避免了外部错误对它的影响，不会产生连锁反应，使软件错误能够局部化，大大减少了查错和排错的难度。

1.3.2　继承性

继承性是面向对象的特征之二，它是基于现实生活中的语义进行说明的，表现了“是一个”(is_a)的关系。如果两个类有继承关系，一个类就自动继承另一个类的所有数据和操作。被继承的类称为基类、父类或超类，继承了父类或超类的所有数据和操作的类称为子类。

在面向对象的技术中，继承是子类自动地共享基类(或父类)中定义的数据和方法的机制。继承性使得用户在开发新的应用系统时不必完全从零开始，可以继承原有的相似系统的功能，或者从类库中选取需要的类，再派生出新的类以实现所需要的功能；继承性还使得相似的对象可以共享程序代码和数据结构，从而大大减少了程序中的冗余信息，并支持程序的重用和保持

接口的一致性。采用继承的方式来组织设计系统中的类，具有如下特点：

(1)一类对象拥有另外一类对象的所有属性与方法。

(2)便于代码的重用。

(3)使得程序结构清晰，降低编码和软件维护的工作量。

(4)使得开发人员能够集中精力于他们要解决的问题。

1.3.3 多态性

在面向对象的软件技术中，多态性是继承关系的特点，是指子类对象可以当做基类对象使用，同样的消息既可以发送给基类对象也可以发送给子类对象。在类继承关系的不同层次中，基类和子类可以共享(公用)一个行为(方法)的名字，然而不同层次中的每个类却各自按自己的需要来实现这个行为。当对象接收到发送给它的消息时，根据该对象所属的类动态地选用在该类中定义的实现算法。

多态性的概念[1]阐述如下：

多态性使得对任何对象自动调用其恰当的方法成为可能。多态现象总是和继承以及从通用基类得到子类一起发生。它是通过将对象与恰当的方法进行动态绑定来实现的。

面向对象的多态性有以下优势：

(1)使程序员所写的大部分程序代码都只操作基类的变量，不需要知道和子类对象类型息息相关的信息，只要处理整个族系的共同表达方式即可。

(2)可以使程序员轻易扩增新类，而大部分程序代码都不会被影响，并且使所撰写的程序便于阅读和维护。

(3)具备多态性质的程序，不仅能够在原本的项目开发过程中逐渐成长，也能较容易地增加新功能以扩充项目规模。

第 2 章　UML 类图及其设计

面向对象是一种思维方式，需要用一种语言表达和交流。UML（Unified Modeling Language，统一建模语言）就是表达面向对象需求分析、设计的首选标准建模语言。UML 是对面向对象开发系统的产品进行说明、可视化和编制文档的手段，它是软件界第一个统一的标准建模语言。要在团队中开展面向对象软件的设计和开发，掌握 UML 语言进行可视化建模是必不可少的。

自 1997 年 OMG 采纳 UML 作为基于面向对象技术的标准建模语言以来，经过多年的发展，UML 已发展到 UML 2.0 版本[3]。

UML 提供了不同视图描述系统的静态结构和动态行为。类图主要用在面向对象软件开发的分析和设计阶段，是面向对象系统的建模中最常见的图，用来描述系统的静态设计视图。它也是构建其他动态设计视图（如状态图、序列图和协作图等）的基础。本书主要讲解 UML 类图的语法和从需求规格说明构建 UML 类图的指导性经验及案例分析，然后围绕 UML 类图的编程实现，阐述 Java 的面向对象编程机制。

2.1　UML 类图

类图主要用在面向对象软件开发的分析和设计阶段，描述系统的静态结构。类图图示了所构建系统的所有实体、实体的内部结构以及实体之间的关系，即类图中包含从用户的客观世界模型中抽象出来的类、类的内部结构和类与类之间的关系。它是构建其他模型图的基础，没有类图，就没有状态图、协作图等其他 UML 模型图，也就无法表示系统的动态行为。

2.1.1　类

在类图中，类用长方形表示。长方形分成上、中、下三个区域，上面的区域内表示类的名字，类名字的第一个字母一般大写，中间区域内表示类的属性列表，最下面的区域表示类的操作列表，如图 2.1 所示。

在类的属性列表中，每个属性信息占据一行，其格式如下：

访问权限控制 属性名：属性类型

例如，如图 2.2 所示的类 CatalogItem 中的属性 code 的格式为

－code：String

在类的操作列表中，每个操作的信息也单独占据一行，其格式如下：