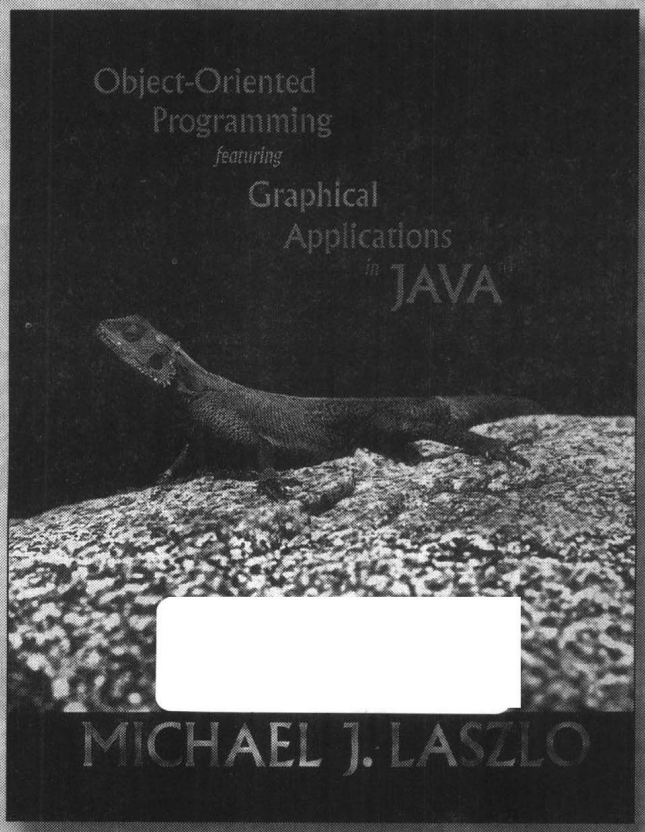
计 算 机 科 学 丛 书

面向对象程序设计
——图形应用实例

(美) Michael J. Laszlo
Nova 东南大学 著 杨秀梅 何玉洁 牟永敏 周冬梅 等译



Object-Oriented Programming Featuring Graphical Applications in Java

机械工业出版社
China Machine Press

面向对象程序设计（OOP）的思想和方法在现代软件设计中越来越重要。本书使读者站在软件工程的高度，理解和掌握面向对象程序设计技术并能应用它解决实际问题。书中以大量的Java程序（大多数是二维计算机图形程序）为实例阐明了面向对象程序设计中的重要概念和设计方法。开篇先阐述了OOP中的对象模型、过程抽象和数据抽象，接着介绍了继承和组合，最后讨论了设计模式和应用程序框架。本书还使用了统一建模语言UML来描述一些设计概念，使读者站在更高的分析与设计层次来认识和理解所需解决的问题。本书还附有大量的练习，针对每节的内容提出问题，让读者进一步巩固所学的理论和方法。

本书可作为计算机专业本科生的教学参考，对涉及OOP的广大软件开发设计者而言也是不错的指导。

本书版权登记号：图字：01-2002-1836

图书在版编目（CIP）数据

面向对象程序设计——图形应用实例 /（美）拉斯洛（Laszlo, M. J.）著；杨秀梅等译. -北京：机械工业出版社，2002.7

（计算机科学丛书）

书名原文：Object-Oriented Programming Featuring Graphical Applications in Java

ISBN 7-111-10143-X

Ⅰ. 面… Ⅱ. ① 拉… ②杨… Ⅲ. JAVA语言－程序设计 Ⅳ. TP312

中国版本图书馆CIP数据核字（2002）第021162号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：杨海玲

北京第二外国语学院印刷厂印刷 · 新华书店北京发行所发行

2002年7月第1版第1次印刷

787mm × 1092mm 1/16 · 21.5印张

印数：0 001-4 000册

定价：35.00元

出版者的话

文艺复兴以降，源远流长的科学精神和逐步形成的学术规范，使西方国家在自然科学的各个领域取得了垄断性的优势；也正是这样的传统，使美国在信息技术发展的六十多年间名家辈出、独领风骚。在商业化的进程中，美国的产业界与教育界越来越紧密地结合，计算机学科中的许多泰山北斗同时身处科研和教学的最前线，由此而产生的经典科学著作，不仅擘划了研究范畴，还揭橥了学术的源变，既遵循学术规范，又自有学者个性，其价值并不会因年月的流而减退。

近年，在全球信息化大潮的推动下，我国的计算机产业发展迅猛，对专业人才的需求日益迫切。这对计算机教育界和出版界都既是机遇，也是挑战；而专业教材的建设在教育战略上显得举足轻重。在我国信息技术发展时间较短、从业人员较少的现状下，美国等发达国家在其计算机科学发展的几十年间积淀的经典教材仍有许多值得借鉴之处。因此，引进一批国外优秀计算机教材将对我国计算机教育事业的发展起积极的推动作用，也是与世界接轨、建设真正的世界一流大学的必由之路。

机械工业出版社华章图文信息有限公司较早意识到“出版要为教育服务”。自1998年始，华章公司就将工作重点放在了遴选、移译国外优秀教材上。经过几年的不懈努力，我们与Prentice Hall，Addison-Wesley，McGraw-Hill，Morgan Kaufmann等世界著名出版公司建立了良好的合作关系，从它们现有的数百种教材中甄选出Tanenbaum，Stroustrup，Kernighan，Jim Gray等大师名家的一批经典作品，以“计算机科学丛书”为总称出版，供读者学习、研究及庋藏。大理石纹理的封面，也正体现了这套丛书的品位和格调。

“计算机科学丛书”的出版工作得到了国内外学者的鼎力襄助，国内的专家不仅提供了中肯的选题指导，还不辞劳苦地担任了翻译和审校的工作；而原书的作者也相当关注其作品在中国的传播，有的还专诚为其书的中译本作序。迄今，“计算机科学丛书”已经出版了近百个品种，这些书籍在读者中树立了良好的口碑，并被许多高校采用为正式教材和参考书籍，为进一步推广与发展打下了坚实的基础。

随着学科建设的初步完善和教材改革的逐渐深化，教育界对国外计算机教材的需求和应用都步入一个新的阶段。为此，华章公司将加大引进教材的力度，在“华章教育”的总规划之下出版三个系列的计算机教材：针对本科生的核心课程，剔抉外版菁华而成“国外经典教材”系列；对影印版的教材，则单独开辟出“经典原版书库”；定位在高级教程和专业参考的“计算机科学丛书”还将保持原来的风格，继续出版新的品种。为了保证这三套丛书的权威性，同时也为了更好地为学校和老师们服务，华章公司聘请了中国科学院、北京大学、清华大学、国防科技大学、复旦大学、上海交通大学、南京大学、浙江大学、中国科技大学、哈尔滨工业大学、西安交通大学、中国人民大学、北京航空航天大学、北京邮电大学、中山大学、解放军理工大学、郑州大学、湖北工学院、中国国家信息安全测评认证中心等国内重点大学和科研机构在计算机的各个领域的著名学者组成“专家指导委员会”，为我们提供选题意见和出版监督。

权威的作者、经典的教材、一流的译者、严格的审校、精细的编辑，这些因素使我们的图

书有了质量的保证，但我们的目标是尽善尽美，而反馈的意见正是我们达到这一终极目标的重要帮助。教材的出版只是我们的后续服务的起点。华章公司欢迎老师和读者对我们的工作提出建议或给予指正，我们的联系方法如下：

电子邮件：hzedu@hzbook.com
联系电话：（010）68995265
联系地址：北京市西城区百万庄南街1号
邮政编码：100037

专家指导委员会

（按姓氏笔画顺序）

尤晋元	王　珊	冯博琴	史忠植	史美林
石教英	吕　建	孙玉芳	吴世忠	吴时霖
张立昂	李伟琴	李师贤	李建中	杨冬青
邵维忠	陆丽娜	陆鑫达	陈向群	周伯生
周克定	周傲英	孟小峰	岳丽华	范　明
郑国梁	施伯乐	钟玉琢	唐世渭	袁崇义
高传善	梅　宏	程　旭	程时端	谢希仁
裘宗燕	戴　葵			

译 者 序

在现代软件开发中，理解面向对象程序设计(OOP)的思想和方法非常重要。虽然有很多程序设计语言都支持OOP，如Smalltalk、C++、Java等，但一个初学者往往是在学习Java时开始接触这种思想。Java语言对OOP思想进行了完整、清晰的表达描述，它是一种纯粹的面向对象语言。因此在本书中用Java语言把丰富但抽象的OOP思想和具体示例结合起来，用具体的示例来帮助大家更快、更直观地理解这种思想和方法。同时这些示例大都为二维图形程序，不仅能学到如何进行图形程序设计，而且在学习中会感到生动有趣。

本书由浅入深，从基本的面向对象概念到软件重用方法逐步进行了介绍。第1章到第3章阐述了OOP中的对象模型、过程抽象和数据抽象等基本概念；第4章和第5章介绍简单的类重用方法——继承和组合；最后第6章和第7章讨论设计重用方法——设计模式和应用程序框架。

尤其值得强调的是：本书使用了统一建模语言UML来描述其中的一些设计概念，这可以使我们站在更高的分析与设计层次来认识、理解解决问题的方法；同时，本书引入了设计模式，这将会使我们在工作中受益于前人的设计成果。

本书中还附有大量的练习，它们都是在每节的基础上提出问题，让大家更进一步理解所学理论和方法。

读者只需要有Java的基本知识就可以看懂本书，其他新的概念如UML、Java 2D等在使用前都会有详细介绍。

本书主要译者为：杨秀梅、何玉洁、牟永敏、周冬梅。参加本书翻译和其他工作的还有马爱华、李少波、朱丹、王帅、贺莹、丁凌云、成然、杨永刚、柏亮、邓涛、朱红松、刘泽深、杨雄高、迟玉强、刘惠华、崔仲凯、李龚、刘文娜、付立武、彭兰茜、李燕华、刘彬、李跃等，在此向他们表示感谢。

译 者

2002年1月

前　言

本书主要目的是用Java语言来探究面向对象程序设计(OOP)的基本思想。对象模型把知识和行为封装在对象中，是面向对象程序设计的基础。由于对象模型对处理程序复杂性很有效而且有越来越多的程序设计语言提供对它的支持，如Smalltalk、C++、Java等，因而近年来，这种程序设计模型占据越来越重要的地位。

面向对象的程序设计方法最早出现在Simula和Smalltalk等语言中，并且已经延续了几十年，但是一个新手往往是在学习Java时开始接触对象模型这种思想。Java语言为表达面向对象程序设计思想提供了清晰的表示法，因此在本书中将会用Java语言把丰富但抽象的面向对象程序设计思想和具体示例结合起来，而且大家可以以Java为工具编写、编译、运行书中的Java程序。此外，本书的大部分示例和练习都是和二维（2D）计算机图形相关的，有的还能产生有趣和令人惊讶的图形效果。这里之所以要使用2D图形示例，一是为了吸引读者，二是为了把新的设计思想和实际应用相结合。随着内容的深入，新的东西将不断被补充到2D图形示例的面向对象图形的类和接口中。

本书适合于已具有Java基础知识的读者，如果你还没有相关知识，请先阅读相关Java语言方面的书籍。尽管本书的目的不是讲授Java语言，但几乎所有的Java基本特性都在使用前进行解释。本书还依赖于两个附加的资源：一是Java 2中的Java 2D应用程序接口（Java 2D API），用于产生二维图形；二是用于表示系统设计的统一建模语言（UML）的一个小子集，其中主要是用类图来表示系统的静态结构，用顺序图来表示对象间交互。Java 2D API和UML的特征将在需要的时候进行介绍，你不需要事先熟悉。

内容组织

第1章介绍对象模型的基本概念：对象和类，消息传递和方法，以及软件重用的四个基本机制——组合、继承、设计模式、应用程序框架。在其他重要的程序设计模型方面，要介绍对象模型。

第2章讨论了过程抽象，而过程被看作具体实现被隐藏的操作。本章中还讨论了Java 的异常处理机制和基于过程抽象的两个标准程序设计技术：过程分解（即过程是由其他操作定义的）和递归（即过程是由过程实现的操作自身定义的）。之所以在本书的开头就讨论过程的概念，是因为过程在对象模型中起着很重要的作用。

第3章讨论了数据抽象。数据抽象是将数据值的具体内部结构隐藏，而把它看作一组相关操作和一个使用这些操作的协议。在对象模型中，数据值被视为对象。本章中还讨论了封装（相关软件元件组合在一起的技术）和信息隐藏。最后介绍了Java中的2D计算机图形，并用Java语言开发了一个计算机图形应用程序模板。

第4章讨论了软件重用的主要机制——组合。应用组合方法，可以将一个新类定义为由其他类组合成的类，这些类称为新类的组件。组合类的每个实例都包含了自己的组件。本章最后给出了一个交互的计算机图形应用程序的模板，读者可以利用它编写用户实时交互程序。

第5章讨论了继承，继承是指新类可以获得已有类（称为父类）的属性和行为，这样称新类是已有类的子类。在本章中讨论了三种基本的继承方法：扩展继承（子类是在继承父类的属性或行为的基础上增加新的属性或行为）；特征继承（子类重新定义了一个或多个它所继承的行为）；说明继承（子类实现了父类定义但没有实现的行为）。在本章中还介绍了如何使用继承来创建相关联的一组数据类型，同时介绍了多态性。

第6章讨论了设计模式。设计模式是描述了解决设计过程中反复出现的问题的有效方法，主要包括解决问题的一组软件元件和如何组合这些软件元件。在众多的设计模式中，本章只讨论三种：迭代器模式、模板方法模式、组合模式。迭代器模式提供访问一个聚集的各个元素的方法，而又隐藏其内部结构。模板方法模式定义一个由具体步骤和抽象步骤组成的算法，使得它的子类通过实现抽象步骤就可以使算法“新生”。组合模式用于将对象组合成基本组件和组合体的层次结构，使客户对基本组件和组合体能进行统一处理。本章应用这三种设计模式设计了多个图形应用程序，如有限点集三角形剖分，建立构造区域几何图形（Constructive Area Geometry, CAG）树（由布尔集合运算并、交和差把2D图形组合成的二叉树），以及构造情景图（scene graphic）（由基本组件和复合图形构成的层次图）。（尽管这样简单描述会使你觉得这些图形程序例子深奥而复杂，但配合上图形，你会发现它们很容易理解。）最后还介绍了其他的设计模式以及设计模式分类的标准方案。

第7章主要讲述了应用程序框架。应用程序框架主要是为了简化在某个特定领域的应用程序的设计。程序员只要依据应用程序框架的协定扩展和实现应用程序框架中提供的类和接口就可以很容易地开发出自己的应用程序。在Java中，应用程序框架是由AWT（Abstract Window Toolkit）、Swing和Java的事件模型组成的，这三部分组合起来构成创建图形用户接口（GUI）应用程序的框架。本章最后一部分是开发基于GUI的绘制和编辑各种图形的程序。

如何使用本书

练习是本书的重要部分，并且是紧随着讲解的内容出现的。有些练习要求大家实现刚刚介绍过的类，有些要求大家使用新介绍的概念和类来设计程序。总之本书中的大部分资料是不断累积的，后面章节中可能用到前面定义的类，很多类和接口被不断加入到面向图形的程序包中。有时，为了理解使用新的概念，会在后面部分修改前面定义过的类。

补充资料

本书中的练习包括从简单的、很容易回答的问题到独立的程序设计，以及非常复杂的程序设计项目。特别重要的练习或是介绍书中后面需用材料的练习都标明为重点，对这些练习，如果你不想解决它们，至少应该了解其中一部分重要内容。

本书中所有的Java程序（按章排列）和图形包都可以到http://www.aw.com/cssupport地址下载，该站点中还可以找到有关如何下载和安装这些文件的帮助资料。书中所有的图形的PowerPoint幻灯片文档也在该站点中。书中练习的答案面向教师提供，需要的教师可以和Addison Wesley Longman的销售代表处联系。

致谢

Nova东南大学计算机信息和科学研究生院（SCIS）为这本书的写作提供了机会和良好的

环境。特别要感谢SCSI的院长Edward Lieblein和我的同事，尤其是MaxineCohen、Sumitra Mukherjee和Junping Sun，他们给了我很大鼓励和支持。

很荣幸有机会在SCSI从事与本书内容相关的教学工作，如面向对象程序设计、程序设计语言理论和计算机图形学。本书大部分材料都是在教学工作中积累起来的。我还要感谢那些提出问题和见解的学生，虽然至今我叫不出他们许多人的名字。最后要感谢数学与计算机科学研究所（Institute for Mathematics and Computer Science，IMACS)的同事和学生。

我很感激下列的审阅者，他们花费大量的时间和精力撰写审阅意见，我们最后的手稿采用了其中的许多建议，他们是D. Robert Adams (大峡谷科罗拉多州立大学), Manuel E. Bermudez (佛罗里达大学), James R. Connolly (奇科加利福尼亚州立大学), Frank Coyle (南查珀尔希尔北卡罗来纳大学), John R. Glover (休斯顿大学), Chung Lee (加利福尼亚州立大学波蒙纳分校), Ronald McCarty (宾夕法尼亚伊利市贝伦德学院), Jong-Min Park (圣迭戈加利福尼亚州立大学), Shih-Ho Wang (戴维斯加利福尼亚大学) 和Marvin V. Zelkowtiz (马里兰大学)。

我还要感谢Addison Wesley的本书编辑Maite Suarez-Rivas女士的真诚支持。同时对Jarrod Gibbons (市场协调员), Gina Hagen (设计经理), Katherine Harutunian (项目编辑), Michaël Hirsch (营销经理), Marilyn Lloyd (项目经理) 和Patty Mahtani (助理管理编辑) 表示深深的感谢。

最后，我要感谢我的家人：我的妻子Elisa和我们的孩子Arianna Hannah和David Joshua，还有我的父母Maurice 和Phyllis。他们的爱、鼓励和耐心是我完成这本书的动力。

目　录

第1章 对象模型

众所周知，计算机处理的是二进制的0和1。为了底层处理，如取指令、分析指令、加法运算和存储数据，计算机的确是用位进行操作的。但实际上，我们眼中的计算机是和二进制的0和1相去甚远的，我们可以利用它实现想要的任何功能：在屏幕上画图、在窗口中输入文本、用鼠标点击选择一个对象，甚至使用强大的图形程序进入一个虚拟的三维世界。我们通过抽象的层次设计脱离与计算机底层处理的联系，以增强我们思考和表达的能力，而隐藏计算机工作的内部细节。当今的软件将计算机转变成定义自己的运算规则而且几乎与计算机内部处理没有明显联系的虚拟机。

抽象的概念不仅在使用计算机时用，而且在编写计算机程序时也使用。我们使用大多数计算机语言提供的高级特性进行程序设计，例如迭代、递归、条件表达式和过程调用，而不使用计算机本身的语言（机器语言）。这样程序员就可以避开计算机底层逻辑的细节和繁杂，而致力于应用抽象方法思考如何恰当地解决面对的问题。如果没有这样的抽象，程序员的效率将被限制在机器层次的位运算上，我们今天就不可能享受这么多令人难以置信的应用软件带来的方便和快乐。

什么是抽象？*抽象是事物的一个理想化的模型，这个模型体现了事物的本质特征，而忽略了那些无关紧要的部分*。抽象可以帮助我们处理事物的复杂性。通过抽象，我们可以更容易通过事物的本质特征来使用或理解事物，而避开事物的不重要部分的影响。我们利用抽象来处理复杂性。举例来说，在开车时，驾驶员将汽车作为具有如下要素的和对自己非常重要的一种抽象：加速用的油门，减速用的刹车、控制方向用的方向盘、播放音乐的收音机等，而其他的设备如给油设备、轮胎、打火拴和轴承不是驾驶员的抽象的内容，因为他不会直接操作这些设备。驾驶员不必关心不同汽车的不同的内部结构，无需接受驾驶不同汽车的培训，就可以轻易驾驶各种各样的汽车。

另一个例子，Java中的对象是一个抽象。对象是用来表示现实世界中的真实事物的（其中的真实是非常广泛的），但对象不可能也没有必要将真实事物的方方面面都描述和表示，对象只需表达真实事物在整个系统中那些本质的特性。如在一个学生注册系统中，每一个学生都用一个对象表示，系统可能需要学生对象的姓名或地址，修改学生对象的电话号码，或通知它已经通过某一门课程的选修申请。实际上，学生对象仅能表示它所代表的真正学生的一小部分，这样的对象不能吃比萨、跳舞、读书，而只能产生学生注册系统所需要的行为。

面向对象程序设计语言像Java等都提供了创造和使用不同的抽象的特性。这些语言特性支持一系列的基本原则，它们组合起来就形成*面向对象程序设计模型* (object-oriented programming model)，亦即通常所说的*对象模型* (object model)。在这一章我们主要介绍的是对象模型。1.1节中主要介绍对象模型的基本概念。其他的程序设计模型，如强制程序设计模式和函数程序设计模型，支持其他类型的抽象并由其他的程序设计语言所实现。1.2节中将对象模型和Java语言与其他几种重要的程序设计模型结合起来介绍。

1.1 对象模型概念

Java程序将一组可以按预定的方式相互通信的对象组合在一起，完成一个预定的功能。没有一个单独的对象能满足需要，只有它们相互正确地联合才能完成系统的功能要求，每一个对象可提供其他对象要求的特定服务。当一个对象要求某个服务时，它会发一个请求（称为消息）给可提供服务的另一个对象，接受消息的对象通过执行它的操作来作出响应并常常要发送另外的消息给其他的对象。这样消息不断在对象组成的网上来回传送，这就是对象模型下完成计算的方式。我们想象中的Java程序运行时实际的对象也是这样工作的。

那么对象是从哪来的呢？每个对象的行为是如何描述的呢？对象之间又是如何协调工作的呢？所有这些都是由Java程序来完成的。Java程序是由一组类和接口的定义组成。每个对象都属于某个类。每个对象的行为都由它所属的类和所实现的接口决定。通过对象的行为，它可以创建新的对象、取消已有的对象；同样通过对象的行为，它可以执行一定的操作，并发送消息使其他对象也完成一定的操作。

本节余下部分将详细讨论对象模型中的基本元素——对象、消息、类、方法和各种各样的协作和重用，并概述这些基本元素是如何组合在一起的。

1.1.1 对象

对象（object）是一个可以接收和响应消息的软件元素。对象表示真实的事物，真实的事物的含义非常广泛。它可以是可触摸的东西，如飞机、苹果，也可以是抽象概念，如颜色和图形轮廓；它还可以是主动的、可以启动或控制过程的事物，如定时器、传感器或人，或是像电梯和字典一样只被动响应服务请求的事物；对象还可以是实现的制品，如链表中的节点，或是用户交互的按钮或菜单。正确地确立对象是程序设计中的一个重要问题，并常常和所涉及的问题有关。一般来说，在程序和程序实现过程中，那些在问题域中扮演着一定角色的事物都会被确立为对象。

对象的行为（behavior）是指如何响应它所接收的消息。事实上，对象能接收它能理解的对应于不同消息的预定义行为 (defined behavior)。每一个消息都会由预定义的行为响应；一个对象包含一组预定义的行为，因此对象能理解的消息种类是由对象的类决定的。

对象接收到一个消息时，它的响应是由预定义的行为和对象的当前状态（state）共同决定的。一个对象可以有任意个实例域（instance field）或简称域（field）。对象的状态和存储在它的域中的值相对应。对象响应消息时，它的状态也随之变化，称为状态转变（state transition），这也就意味着它的域值的改变。对象的状态记录它从创建开始到现在的一切过程。换句话说，对象的状态捕获和它的行为相关的历史事件。

除了行为和状态，对象还有一个标识（identity）用来区别它和其他对象。标识不依赖于对象的状态，也不随对象状态的变化而变化。这就如同虽然你的状态变化了（年龄的增长，吃爆米花，明白了一个道理），但你并不会变成另一个人。如果同一个类中的两个对象恰巧状态相同，那么它们仍然是不同的对象。

到目前为止，我们分别定义了对象的三个特性：预定义行为、状态和标识。对象如何响应接受的消息是由预定义行为和它的状态共同决定的。对象要接收消息，那它就要和其他对象相区别——它必须要有自己的标识。

为了用Java说明这些概念，这里举例一个例子，我们将用一个类表示笛卡儿平面中的点(见图1-1)。类Point有两个域，分别表示点的*x*和*y*的坐标。语句
将完成下列操作：

```
Point p = new Point(3, 2);
```

- 创建一个Point类的新对象。
- 初始化点的*x*坐标值为3，*y*坐标值为2。
- 声明一个新的引用变量p。
- 指定变量p为对新的Point对象的引用。

虽然简略，但上述语句却完成了创建对象、初始化它的状态并确定了对它的引用。接着就可以通过引用变量p或这个引用的副本发送消息给这个新Point对象。

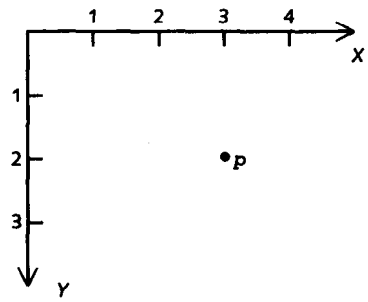


图1-1 平面和点*p*=(3, 2)

1.1.2 消息

对象间通过相互发送消息（message）来进行交互。发送消息的对象称为发送者（sender）或客户（client），而接受消息的对象称为接收者（receiver）或服务器（server）。这里用客户－服务器这个术语来表达，因为发送者请求另一个对象的服务时被视为客户，提供服务的接收者被视作服务器。为了完成所提供的服务，一个对象通常还会依靠另外一些对象的服务，也就是说对象有时扮演服务器的角色，有时却扮演客户的角色。通常对象扮演的角色是和消息相关的：相对于收到的消息，对象为响应服务的服务器；但为了完成请求服务，它又发送消息给其他对象，此时它又变成客户。

一个消息由下列三个元素组成：

- 消息名，又称选择器。
- 零个或多个参数列表，为接收对象提供数据信息。
- 指示接收对象的引用。

参数可能是对象引用或是基本数据类型(如整型)值。注意，尽管发送者可以把自己作为参数传递，但消息中并不包括对发送者的引用。

你或许对Java消息的语法结构很熟悉。假设p是表示平面上一个点对象，语句：

```
p.setCoordinates(8, 9);
```

是一个消息，它由选择器setCoordinates、两个参数8和9和接收对象p组成。消息响应的结果是p把它的位置变为（8，9）。

是哪个对象发送p.setCoordinates(8,9)这个消息的呢？在这个简单的例子里不能确定。实际上，这个消息作为一条语句出现在某个对象的预定义行为（方法）体内，在执行这个方法的过程中，这个对象发送的消息。

通常一个对象还会发送消息给它自己。这种情况下，消息的接收者也就是发送者，消息的接收者可以从消息中省略掉。这样点p就会用语句

```
setY(4);
```

改变它的y坐标值为4。这个消息含有消息名setY和参数4，接收对象也就是发送者是隐含的。因为关键字this总是用来表示消息的发送者（它的代码正在执行中），所以上面的消息可以写成

```
this.setY(4);
```

使得接收者是显式的。

关键字this还可以作为参数，使对象可以发送一个标识自身为发送者的消息。例如，如果p和q都是Point对象，p可以发送消息

```
q.setCoordinates(this);
```

告诉q修改它的坐标值使它们和p的值相等。接收者q期望setCoordinates这一消息的参数是一个Point对象，事实上this就是一个Point对象，是用关键字this标识的发送者自己，即点P。

当对象收到一个消息时，它就按照预定义的行为完成响应，这些行为是由接收对象的*方法*（method）定义的，关于方法我们将稍后讨论。执行预定义行为只可能会产生三种结果：

- 返回一个值给消息发送者。
- 接收者状态的改变。
- 作为参数传给接收者的对象的状态变化。

所产生的这些结果是由特定的行为决定的，可能是一种、两种或者三种同时产生。有时接收者状态发生变化是由于它的一个或多个对象的属性的状态发生变化而引起的，如果这些属性正好被其他对象共享，那么不仅发送者和接收者，其他对象也会受到这种行为的影响。行为的结果有时很难理解，而且经常会带来意想不到的或不正确的行为的产生。

1.1.3 对象接口

对象可响应的消息是由对象接口决定的。对象的接口是以一组操作的形式出现的，每一个操作都对应于在响应某个消息时对象所完成的预定义行为。对象的接口是由它的类型（type）决定的，而对象的类型又是由它所属于的类决定的。

客户通过对象的接口来理解对象支持的各种行为。例如，我们可以利用下面类似Java语言的语法来表达Point对象接口：

```
public class Point {
  // constructs a new point at position (x,y)
  public Point(int x, int y)

  // constructs a new point that is a copy of point p
  public Point(Point p)
```

```
    // constructs a new point at (0,0)
    public Point()

    // returns the x coordinate of this point
    public int getX()

    // changes the x coordinate of this point to newX
    public void setX(int newX)

    // returns the y coordinate of this point
    public int getY()

    // changes the y coordinate of this point to newY
    public void setY(int newY)

    // changes the position of this point to (newX,newY)
    public void setCoordinates(int newX, int newY)

    // changes the position of this point
    // to (p.getX(),p.getY())
    public void setCoordinates(Point p)

    // translates this point by dx along x and dy along y
    public void moveBy(int dx, int dy)

    // returns a string-descriptor for this point: "(x,y)"
    public String toString()
}
```

使用对象操作是要遵守一定的规则，这些规则称为对象的协议（protocol），协议描述如何使用对象的每一个操作。客户要想和对象进行正确交互，就必须遵守对象的接口和协议。例如，当一个Point对象接收到带有一个参数的setCoordinates消息时，这个参数必须是Point对象；下面的消息就违反了对象协议：

```
p.setCoordinates(null);
```

这里讲的Point对象的接口相对比较简单，不过在本书后面，我们将看到有很多有趣的接口和协议的对象。

到目前为止，我们所描述的接口是指对象的公有接口（public interface），这里的公有强调的是任何类型的对象都可以使用它的操作，也就是说公有接口定义了一组可被任何对象发送的消息。除了公有接口外，对象还可以提供限制型的操作，供某些类型的客户使用。只供特定类型客户使用的操作称为限制型接口（restricted interface）。私有接口（private interface）是限制性最强的接口类型，而公有接口的限制最弱的。私有接口是公有接口的超集，它包括所有公有接口的操作，而且它还加入了一些仅对同一类的对象可用的私有操作。

Java提供另外两种限制型接口：保护型接口（protect interface）和包接口（package interface）。从公有接口到保护型接口，再到包接口，最后到私有接口，接口的访问限制越来越强，而作用会变得越来越大。接口访问权限的设置可以使一个授权的客户操纵一个对象而另一个未授权的客户则不能。

1.1.4 方法和过程

过程（procedure）是实现某一操作的一段代码。过程定义了一个可计算的程序，此程