21世纪高等学校计算机规划教材

21st Century University Planned Textbooks of Computer Science

面向对象技术及UML教程

Object-Oriented Technology and UML

李磊 王养廷 主编

杜启军 副主编

- 以Rational Rose为主要工具
- 以UML基础知识为主要内容
- 以实际应用UML为主要目的

高校系列

人民邮电出版社
POSTS & TELECOM PRESS

21世纪高等学校计算机规划教材
21st Century University Planned Textbooks of Computer Science

面向对象技术及UML教程

Object-Oriented Technology and UML

李磊 王养廷 主编
杜启军 副主编

高校系列

人民邮电出版社
北京

图书在版编目（CIP）数据

面向对象技术及UML教程 / 李磊，王养廷主编. --
北京 : 人民邮电出版社，2010.5
21世纪高等学校计算机规划教材
ISBN 978-7-115-22425-5

Ⅰ. ①面… Ⅱ. ①李… ②王… Ⅲ. ①面向对象语言
，UML－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2010)第042496号

内 容 提 要

本书主要包括 3 部分内容：面向对象编程的基本知识、UML 介绍和 UML 工具的介绍，其中详细讲解了 UML 的主要模型图的图符、含义和应用。主要内容包括用例图、顺序图和协作图、类图和对象图、包图、状态图和协作图、构件图和 UML 部署图。在讲述 UML 各种模型图时，不仅介绍图符的用法和含义，还着重介绍这些模型图的应用。另外，本书还以 Rational Rose 为例简要介绍了如何使用 UML 工具进行 UML 主要模型图的绘制，以及如何利用 Rational Rose 进行模型到代码、代码到模型的双向工程。

本书在内容组织和安排上强调实用性，书中介绍了面向对象的概念、面向对象实现技术以及相关的软件开发过程，最后给出一个实例详细介绍如何在实际项目中应用 UML 进行面向对象分析和设计。

本书内容浅显易懂，适合作为高等院校相关专业的 UML 教材，也可以作为计算机行业从业人员学习 UML 的参考书。

21 世纪高等学校计算机规划教材

面向对象技术及 UML 教程

◆ 主　　编　李　磊　王养廷
副 主 编　杜启军
责任编辑　刘　博

◆ 人民邮电出版社出版发行　　北京市崇文区夕照寺街 14 号
邮编　100061　　电子函件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京华正印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：12.25
字数：312 千字　　2010 年 5 月第 1 版
印数：1－3 000 册　　2010 年 5 月北京第 1 次印刷

ISBN 978-7-115-22425-5

定价：22.00 元

读者服务热线：(010)67170985　印装质量热线：(010)67129223
反盗版热线：(010)67171154

出版者的话

现今社会对人才的基本要求之一就是应用计算机的能力。在高等学校，培养学生应用计算机的能力，主要是通过计算机课程的改革，即计算机教学分层、分类规划与实施；密切联系实际，恰当体现与各专业其他课程配合；教学必须以市场需求为导向，目的是培养高素质创新型人才。

人民邮电出版社经过对教学改革新形势充分的调查研究，依据目前比较成熟的教学大纲，组织国内优秀的有丰富教学经验的教师编写了一套体现教学改革最新形势的“21世纪高等学校计算机规划教材——高校系列”。在本套教材的出版过程中，我社多次召开教材研讨会，广泛听取了一线教师的意见，也邀请众多专家对大纲和书稿做了认真的审读与研讨。本套教材具有以下特点。

1. 覆盖面广，突出教改特色

本套教材主要面向普通高等学校（包括计算机专业和非计算机专业），是在经过大量充分调研的基础上开发的计算机系列教材，涉及计算机教育领域中的所有课程（包括专业核心骨干课程与选修课程），适应了目前经济、社会对计算机教育的新要求、新动向，尤其适合于各专业计算机教学改革的特点特色。

2. 注重整体性、系统性

针对各专业的特点，同一门课程规划了组织结构与内容不同的几本教材，以适应不同教学需求，即分别满足不同层次计算机专业与非计算机专业（如工、理、管、文等）的课程安排。同时本套教材注重整体性策划，在教材内容的选择上避免重叠与交叉，内容系统完整。学校可根据教学计划从中选择教材的各种组合，使其适合本校的教学特点。

3. 侧重培养应用能力

目前社会对人才的需要更侧重于其应用能力，包括须具备扎实的计算机基础理论、良好的综合素质和实践能力。本套教材注意通过实践教学与实例教学培养解决实际问题的能力和知识综合运用的能力。

4. 教学经验丰富的作者队伍

高等学校在计算机教学和教材改革上已经做了大量的工作，很多教师在计算机教育与科研方面积累了相当多的宝贵经验。本套教材均由有丰富教学经验的教师编写，并将这些宝贵经验渗透到教材中，使教材独具特色。

5. 配套资源完善

所有教材均配有PPT电子教案，部分教材配有实践教程、题库、教师手册、学习指南、习题解答、程序源代码、演示软件和素材等，以方便教与学。

我社致力于优秀教材的出版，恳切希望老师们在使用的过程中，将发现的问题及时反馈给我们，以便再版时修改。

前 言

统一建模语言（Unified Modeling Language，UML）汇集和融合了著名的面向对象技术专家 Grady Booch、James Rumbaugh 和 IvarJacobson 的研究成果。它是一种简洁的、统一的可视化建模语言。UML 是一种定义良好、易于表达、功能强大且适用于各种应用领域的建模语言，已被 OMG 采纳为标准，目前 UML 已成为面向对象技术领域内占主导地位的标准建模语言。

对于系统的使用者，开发人员可以用 UML 来表达系统的功能，让使用者了解系统能够提供的功能和价值。对于系统分析师，他们可以使用 UML 作为系统架构设计的工具。对于软件工程师，他们可以利用 UML 进行构件和类的设计，描述系统中对象之间的动态交互。对于系统的管理者，他们也可以使用 UML 来刻画系统中硬件或是软件构件的部署与配置情形。

本书以面向对象程序设计、UML 基础知识和实际应用为主，由浅入深地向读者介绍了面向对象编程的基础知识、UML 的基础知识以及如何在实际项目中应用 UML 技术，并且以 Rational Rose 为例介绍了如何使用 UML 工具。

本书共分为 13 章，第 1 章主要讲解了面向对象的基础知识；第 2 章主要介绍了 RUP（统一软件过程），RUP 是目前的软件过程中与 UML 结合最好的过程；第 3 章简要介绍了 UML 的基础内容，包括 UML 的概念、UML 的发展历史、UML 的主要内容，以及 UML 2.0 中的新内容；第 4 章主要介绍了 UML 的工具，并以 Rational Rose 工具为例，讲解了如何绘制 UML 的各种关系图，以及如何使用 Rose 的双向工程；第 5 章到第 11 章详细介绍了 UML 的用例图、顺序图、协作图、类图、对象图、包图、状态图、活动图、构件图和部署图，包括 10 种模型图的含义、使用的图符以及如何在实际应用中使用这 10 种模型图；第 12 章介绍了面向对象的实现技术，包括面向对象语言的介绍、面向对象类和接口的设计，以及面向对象的基本编程规范；第 13 章以一个具体的实例“打击游戏”为例，将该实例作为一个软件项目，参照软件公司的项目组织和开发方式，应用 UML 完成项目的需求分析、设计实现、集成和测试的过程，帮助读者学会如何使用 UML 进行实际软件项目的开发。

本书既可以作为高等院校计算机专业的学生学习 UML 或面向对象程序设计的基础教材，也可作为从事计算机行业的技术人员学习 UML 的参考书。

本书由李磊、王养廷和杜启军共同编写，其中第 4 章、第 5 章、第 6 章、第 7 章、第 8 章、第 9 章由李磊编写，第 1 章、第 2 章、第 12 章、第 13 章由王养廷编写，第 3 章、第 10 章、第 11 章由杜启军编写。

由于作者的水平有限，书中难免存在错误之处，恳请广大读者批评指正。如果读者有问题，请发送电子邮件到 cd_sywe@sohu.com，编者将及时给您答复。

编 者

2010 年 1 月

目 录

第 1 章 面向对象技术概述

面向对象（Object-Oriented）不仅是一些具体的软件开发技术与策略，而且是一整套关于如何看待软件系统与现实世界的关系，用什么观点来研究问题并进行求解，以及如何进行系统构造的软件方法学。本章先来了解一下面向对象技术的基本知识和基本概念，为后面介绍面向对象分析和设计建模语言（UML）做好铺垫。面向对象程序设计方法已经被大家广泛接受，成为当前最流行的程序设计方法。对于程序设计的初学者，经常会遇到的问题是：为什么要使用面向对象的程序设计方法？接下来的问题是：当遇到一个实际问题时，如何进行面向对象的分析、设计和实现？本章首先回答第一个问题，在以后的章节中，结合 UML 详细讲解第二个问题。

1.1 面向对象方法学

面向对象是一个常用的词，大家经常说起，也经常看到。那么什么是面向对象呢？这里从两个层次来介绍这个概念：第一个层次就是面向对象思想；第二个层次是面向对象程序设计语言。

严格意义上说，面向对象思想与程序设计无关，它是人们对自然世界的一种认识，把世界中的所有事物都看作是对象，每个对象既是独立的，同时这些对象又是相互联系的。把这种思想应用到软件开发上，开发人员可以把需要解决的问题看成是多个独立的，同时又相互联系的对象组成的一个系统，这样有助于开发人员更深入地理解问题本身。

现在绝大多数的程序设计语言和开发环境都声称自己是面向对象的，那么什么是面向对象程序设计语言呢？主要看这个语言定义的类型是否都支持对象的声明，以及这个语言对对象的封装、继承和多态的支持程度。不同的语言面向对象的程度不同，如 Java 语言比 C++ 语言的面向对象程度高。原因是 Java 语言对基本的数据类型进行了对象封装，这样使用 Java 语言开发的程序，所有的变量都可以是对象。

面向对象分析设计方法是指如何把现实世界的问题经过转换，可以用面向对象程序设计语言实现。也就是常说的面向对象分析和面向对象设计。

1.1.1 面向对象方法学概述

程序设计的目的是设计出可以使用的软件系统。为了设计出高质量的软件系统就需要研究程序设计中涉及的基本概念、描述工具和所采用的方法。至今为止，常见的有结构化程序

设计方法和面向对象程序设计方法。

早期的计算机软件由于规模小、复杂度低，主要由单个的程序员来完成软件的开发。随着高级语言的出现，人们希望计算机可以完成更复杂的工作，这就要求编写规模更大、复杂度更高的软件。随着软件复杂程度的提高，软件的开发和维护费用也不断地在提高，人们需要更加有效的软件开发方法，因此提出了结构化程序设计方法的概念，出现了一批支持结构化方法的程序设计语言，如 Pascal、C、Ada 等。结构化程序设计方法主要特点是：自顶向下、逐步求精；模块化；语句结构化。简单地说，利用结构化程序设计方法实现程序设计需要经过两个基本过程：分解和组装。

随着软件规模的进一步增加，结构化的程序设计方法的弱点也就暴露出来了，采用结构化程序设计方法分析问题主要是从计算机实现的角度来考虑问题，面对复杂问题时，不利于理清问题本身。因此对问题的抽象程度比较低，程序的重用性和扩展性比较低。在这种情况下面向对象程序设计方法应运而生。

面向对象程序设计方法包括面向对象分析、面向对象设计和使用面向对象程序设计语言的实现，还可以包括面向对象的测试。采用面向对象方法进行分析时，所分析的问题都使用对象来描述，对象之间通过消息进行联系。对象经过抽象变成一个一个的类。下面通过一个例子来说明面向对象程序设计与结构化程序设计的区别。

例如有一个文本文件 Test.txt，现在要求把文本文件中所有的小写字母变成大写字母，结果还保存在 Text.txt 文件中。

首先采用结构化程序设计方法进行分析和设计。结构化程序设计需要先分析程序要完成的主要工作，接下来分析程序需要的数据结构。

程序要完成的工作有：

- 把文件 Test.txt 中的内容读入内存。
- 对文件中的内容进行检查，如果发现小写字母进行转换。
- 把转换后的结果保存到文件中。

程序中用到的数据结构分析如下。

为了能够方便进行检查和处理，需要开辟一个缓冲区来存放从文件中读入的数据。根据每次从文件中读入数据大小的不同，需要开设的缓冲区大小也不相同。如果每次从文件中读入一个字符进行处理和保存，这时候只需要一个存储单元；如果每次从文件中读入一行，这时就需要一个大一点的缓冲区了。

通过上面的分析可以看出，这个软件包括 3 个部分：读入数据，处理数据，保存数据。再加上一个总控模块。程序的模块结构图如图 1.1 所示。

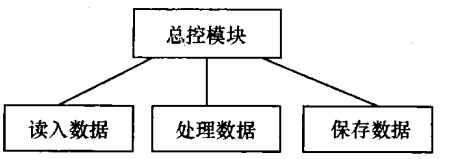

图 1.1　模块结构图

如果采用面向对象分析方法来分析这个例子，则首先会找出一个一个的对象，然后再找出这些对象之间的联系。

首先找出系统中那些可能的类，从问题描述中可以看出，这个例子是要处理一个文本文件；一般的面向对象语言都会提供一个文件类，用于基本的文件操作。这样我们就找到了两个类：

- TextFile：要处理的文本文件类。
- File：系统提供的标准文件处理类（不同的系统名字不同）。

接下来看一下这两个类的关系，第一个类 TextFile 继承了第二个类 File，并对第一个

类进行了扩充，支持文本文件的操作。两个类的类间关系如图 1.2 所示。

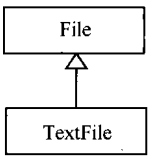

图 1.2　类间关系图

从上面的例子可以看出，面向对象程序设计方法从分析问题的角度和分析设计的方法上与结构化程序设计都是不一样的。如何进行面向对象的分析和设计，本书将在后面章节中进行详细介绍。

1.1.2　面向对象方法学的基本特征

面向对象程序设计的基本特征包括了封装、继承和多态，对象之间是通过消息相互作用的。下面对这些特征进行介绍。

1. 抽象

抽象是解决实际问题经常采用的策略，也是人类认识世界的本能方式。所谓抽象是指从许多事物中，舍弃个别的、非本质的属性，抽取出共同的、本质的属性的过程，它是形成概念的必要手段。

交通图就是应用抽象的一个很好范例。在人们驾车外出旅行时，需要一张交通路线图。在这张图上，有道路、河流、山脉、旅游景区、快餐店、加油站等各种标志，它们都是对实际景观抽象的结果。这些标志只能说明某个地理位置有一条道路、一条河流、一座加油站等，而并没有反映出某一座特定的加油站的特征。实际上，每一个加油站在其建筑、占用面积、人员管理等诸多方面都有所不同，但所有这些加油站都是用来为汽车加油的，这是所有加油站的共同特征。对于所有的出行人来说只要知道有一个能够加油的地方就可以了，并不需要知道这个加油站的人员、规模等信息。

数据抽象是一种更高级别的抽象方法。它将现实世界中存在的事物作为抽象单元，其抽象内容既包括事物的属性特征，也包括行为特征。数据抽象是面向对象程序设计所采用的核心方法，通过数据抽象得到现实世界的一个一个的事物，这些事物就是一个一个的对象。

例如，现实世界中一个学校有许多学生，在抽象过程中每个学生可以用学号、姓名、性别、年龄、家庭住址等信息进行描述；同时每个学生还可以有注册学籍、选修课程等行为。通过这些从每个学生身上抽象出的信息和行为，可以任意描述出该学校每一个现实世界中的学生。

2. 封装

封装是指将现实世界中某个事物的属性与行为聚集在一个逻辑单元内部的机制。封装指将对象属性和操作结合在一起，构成一个独立的对象。它的内部信息是隐藏的，不允许外界直接存取对象的属性，而只能通过指定的接口与对象联系。

实际上，封装并不是新的概念，在日常生活中，处处可以看到封装机制的应用。例如，一台日常的电视机由许多电器元件组成，每一个电器元件都有一定的性能指标，完成一定的功能。但是在使用电视机时，用户不需要了解这些电器元件的功能，只需要操作电视机前面面板上提供的按钮来实现换台、调节音量等功能。这就是封装，用户只需要操作电视机提供的按钮，而不需要了解内部每个电器元件的工作原理与工作过程。

在面向对象的程序设计中，封装是指将对象的属性和行为分别用数据结构和方法描述，并将它们绑定在一起形成一个可供访问的基本逻辑单元。用户对数据结构的访问只能通过提供的方法实施。例如，一个学生的基本信息可能有学号、姓名、性别、出生年月、家庭住址等，对这些属性的操作行为主要应该包括：获取这些属性的当前值，将这些属性设定为某个

给定的值等。现在，将描述这些属性的数据结构和为了对它们实施各种操作而设计的方法封装在一个对象中，并将其中的数据结构隐藏起来，不允许外界直接访问，而将其中的方法作为外界访问该对象属性的用户接口对外开放。这样一来，其他对象只能通过这些方法对该对象实施各项操作。显而易见，封装是实现数据隐藏的有效手段，是一种很好的管理数据与操作行为的机制，它可以保证数据结构的安全性，提高应用系统的可维护性和可移植性。

3. 消息传递

消息是指对象之间在交互中所传递的通信信息。简单地说，消息是一个对象要求另一个对象实施某项操作的请求。在一条消息中，需要包含消息的接收者和要求接收者执行哪项操作的请求，但并没有说明应该怎样做。具体的操作过程由接收者自行决定，这样可以很好地保证系统的封装性。

消息传递是对象之间相互联系的唯一途径。发送者发送消息，接收者接收该消息，并通过调用相应的方法响应该消息。

这个过程被不断地重复，使得整个应用程序在人的有效控制下运转，最终得到相应的结果。可以说，消息是驱动面向对象程序运转的源泉。

4. 继承

继承是类之间的一种常见关系。这种关系为共享数据和操作提供了一种良好的机制。通过继承，一个类的定义可以基于另外一个已经存在的类，分别将它们称为“子类”和“父类”，“父类”又称为“基类”。子类可以继承父类的全部内容，并在此基础上，对父类表述的内容加以扩展或覆盖。

根据继承关系的特性，可以将继承分为下面两种主要形式。

- 直接继承和间接继承。如果类 C 的定义直接派生于类 B，则称 C 直接继承于 B，且 B 是 C 的直接父类。如果类 C 直接继承于 B，类 B 直接继承于类 A，则称 C 间接继承于 A，A 为 C 的间接父类。间接继承体现了继承关系的可传递性。
- 单继承和多继承。如果一个类只有一个直接父类，则该继承关系被称为单继承；如果一个类有多于一个以上的父类，则该继承关系被称为多继承。Java 语言只支持单继承，不支持多继承。

继承机制是实现程序代码重用的基石，是提高软件系统的可扩展性和可维护性的主要途径。所谓继承是指一个类的定义可以基于另外一个已经存在的类，即子类基于父类，从而实现父类代码的重用。

在定义子类时，子类从父类继承了父类所有的属性和方法，并根据自己的需要添加新的属性和方法。例如已有一个父类：

类名：Person 主要属性：name、age、sex 主要方法：born()、died()

定义一个新的类 Student 是类 Person 的子类，并定义了新的属性和方法如下：

类名：Student 主要属性：major、department 主要方法：register()

这时新定义的子类中有 5 个属性：name、age、sex、major、department；3 个方法：born()、died()、register()。

父类与子类相比，涵盖了更加共性的内容，更具有一般性；而子类所添加的内容更具有个性，是一般性之外的特殊内容。因此这种类的继承关系充分地反映了类之间的“一般—特殊”关系。类的继承具有传递性，即子类还可以再派生子类，最终形成一个类层次结构。位于上层的父类概念更加抽象，位于下层的子类概念更加具体。所以说，从下往上看，是逐步抽象的过程；从上往下看，是逐步细化的过程。实际上，解决问题的过程就是不断抽象和细化的过程。

父类可以派生子类，子类又可以派生他自己的子类，如此下去就形成了一棵或多棵类树。一般的面向对象语言开发环境都会提供一个以层次安排的类组成的类库，这些类可以完成程序设计语言中最常用和最基本的功能，方便开发者使用，同时开发者也可以直接从类库中的类派生自己需要的类。

5. 多态

当对象收到消息时要予以响应。不同的类对象收到同一个消息可以产生完全不同的响应效果，这种现象叫做多态。利用多态机制，用户可以发送一个通用的消息，而实现的细节由接收对象自行决定，这样，同一个消息可能会导致调用不同的方法。

实际上，多态概念的应用相当广泛。例如，在定义一个父类“几何图形”时，为其定义了一个绘图操作。当“几何图形”的子类“正方形”和“三角形”都继承了“几何图形”类的绘图操作时，该操作根据不同的类对象，将执行不同的操作，在“正方形”类对象调用绘图操作时，该操作将绘制一个正方形。而当“三角形”类对象调用绘图操作时，该操作将绘制一个三角形。这样当系统请求绘制一个几何图形时，同样的“绘图”操作消息因为接收消息的对象不同，将执行不同的操作。

在面向对象程序设计中，多态性依托于继承性。利用类的继承机制可以形成一个类的层次结构，把具有通用功能的消息放在较高层次，而具体的实现放在较低层次，在这些较低层次上生成的对象能够对通用消息作出不同的响应。多态性是面向对象程序设计的精髓之一，它可以增加应用程序的可扩展性、自然性和可维护性。

上面阐述了面向对象的几个主要的特性：抽象性、封装性、消息传递、继承性和多态性。深入理解这几个特性是掌握面向对象程序设计方法的关键。

类的继承性使得在定义新类时，开发人员可以充分利用已存在的类，即将新类作为子类，已存在的类作为父类，形成子类继承父类的结构，从而实现父类代码的重用，提高软件开发的效率。而且还可以在子类与父类之间利用多态性增加系统的灵活性、理解性和扩展性。

对于同一个消息，不同类的对象可以作出不同反应的现象被称为多态性。多态性是指不同类的对象调用同一个方法，却执行不同的代码段的现象。在程序设计阶段，它指的是一个给定类型的变量可以引用不同类型的对象，并且能够自动地调用变量所引用的对象类型的特定成员方法，这就使得针对某个成员方法的调用，将根据应用这个调用的对象类型得到不同的操作行为。

例如，在 Java 语言程序中，实现这种处理机制的方法是利用指向父类对象的引用可以指向其子类对象的特征，使用该引用调用成员方法，并根据该父类引用所指的当前对象类型，确定调用哪个成员方法。由于直到程序执行时才会得知该父类引用所指对象的类型，因此选择执行哪一个成员方法，只有在程序执行时才能够动态的确定，而在程序编译时不能确定，

这种连接机制称为动态联编。

可以看出，若要实现多态性，需要具备下面两个条件。

第一，多态性作用于子类，它是依赖于类层次结构中的一项新功能。在面向对象语言中，需要提供一个指向父类对象的引用或指针，用来指向它的任何子类对象的能力，这是实现多态性的先决条件。因此需要先定义一个指向父类的引用或指针，然后根据需要在程序的运行过程中让它引用或指向其子类的对象，并根据当前引用或指向的对象类型调用相应的成员方法。

第二，若得到多态性的操作，相应的方法必须同时包含在父类和子类中，且对应的方法的定义完全一样，子类中该方法的访问属性不能严于父类中该方法的访问属性。

多态机制是建立在继承机制的基础上的，它是面向对象程序设计语言的精华，恰当使用多态特征可以设计出灵活性高、使用范围广的程序。

1.2 面向对象的类和对象

面向对象最基本的两个概念就是类和对象，前面介绍了对象可以从 3 个层面来理解，同样类这个概念也可以从两个层面上来理解，一个是面向对象方法，一个是面向对象程序设计语言。面向对象方法中的类可以指一类事物，而对象既可以指一类事物也可以指某个具体的事物。在这个层面上这两个概念有一定的区别，但有时也可以通用。在面向对象程序设计语言层面上来说，可以把类看成是一个数据类型，如整型这样的数据类型，而对象则是该数据类型的一个变量。

对象是现实世界中事物的抽象，用来描述现实世界中的每一个事物，如现实世界中的一个人、一位教师和一名学生都是一个一个的对象。在面向对象分析中，开发人员根据需要开发的系统，了解现实世界中该问题领域中的每个实际的事物，这些事物被描述成一个个对象。例如，要开发一个学生成绩管理系统，来管理一个学校的学生成绩。这样开发人员需要先了解实际的学生成绩如何管理，学生成绩管理中涉及学生、教师、成绩和课程等领域的事物，这些事物就是一个一个的对象。

在系统实现时选择具体的程序设计语言或开发环境，每个对象映射成一组状态和多个操作方法。这些状态描述了每一个对象的具体特征，其他对象可以请求这个对象操作方法来改变对象的状态。

类是指一类事物的集合。它是对现实世界一类事物抽象的结果，其主要包含对一类事物属性及作用在这些属性上的行为的描述。在编写程序时，首先要对描述各类事物的类进行定义。在实际的程序设计中，可以简单地认为类是一种类型。在面向对象设计时主要对类的外部接口进行设计，在类的实现时完成一个类的属性和方法的描述。这个类的对象可以通过接口与其他对象进行通信，其他对象只能通过接口来访问该对象，从而实现类的信息封装。

在编程实现时，程序是由有限个对象构成的，对象是程序操作的基本单位。所谓程序运行，就是对象之间不断地发送消息及响应消息的过程。因此，定义类之后，需要通过对类实例化来构造对象。在 Java 语言中，对象属于引用型变量，需要经历声明、创建、初始化、使用和清除几个阶段。

1.3　面向对象程序设计语言

面向对象程序设计方法需要能够描述面向对象的程序设计语言支持，否则这种设计方法只能是纸上谈兵。目前，随着面向对象程序设计方法的日趋成熟，支持面向对象的语言也逐渐丰富起来。下面简要介绍面向对象程序设计语言的特征以及几种有代表性的面向对象的程序设计语言。

所谓面向对象的程序设计语言（Object-Oriented Programming Language，OOPL）是指提供描述面向对象方法所涉及的类、对象、继承和多态等基本概念的程序设计语言，具体地讲，它应该支持面向对象的主要特性。至今为止，出现过很多种面向对象的程序设计语言，但比较知名且具有代表性的面向对象程序设计语言主要有以下几种：Simula67、Smalltalk、Eiffel、C++、Java。下面就这几种语言的特点作简要介绍。

1. Simula67 语言

Simula 语言于 1967 年发布，因此又被称为 Simula67。Simula67 的前身是 Simula1，最初的设计目的是用于模拟离散事件。随着不断地改进和完善，发展到 Simula67 时已经演变为一种通用的程序设计语言，而模拟离散事件只是它的一个应用领域。Simula 的基础是 ALGOL60（在 20 世纪 60 年代非常流行的结构化程序设计语言）。它沿用了 ALGOL60 的数据结构和控制结构，并引入了对象、类和继承等概念。

Simula 语言主要特点如下。

- 具有主程序的概念。它同 ALGOL60 一样，可执行程序由一个主程序和若干个分程序构成，可以部分地支持类的分别编译。
- 支持嵌套定义，特别是允许类被嵌套定义。
- Simula 中的类不同于数据类型，它是一组对象模板。程序可以直接存取对象的数据结构，因此 Simula 的对象并没有完全实现数据抽象。
- 引入了虚拟子程序的概念。所谓虚拟子程序是指在定义子程序时不指明参数，而在子类中定义，这样可以使得每个子类定义的参数格式不同，从而提高程序设计的灵活性。
- 支持部分的多态性。在 Simula 程序中，如果说明 a 是 classA 类的对象引用，则可以用 a 指向 classA 类或 classA 类的子类对象。如果 b 是 classA 类的子类 classB 的对象引用，则赋值 a:=b 是合法的；但 b:=a 只有在 a 实际指向 classB 类或 classB 类的子类对象时才合法。前者将在编译时检查，后者则在运行时检查。
- Simula 除了虚拟子程序应用动态联编外，一般都采用静态编译。虽然 Simula 也提供了一些用于强制性动态联编的命令，但都存在一些弊病。

2. Smalltalk 语言

Smalltalk 是第一个真正的面向对象的程序设计语言。1972 年由美国 Xerox 公司研究中心（PARC）所属的一个软件概念小组（Software Concepts Group）经过数年的努力，在 Flex 系统的基础上研制成功。后经不断地构思、试验和改进，陆续推出若干个版本，其中在 1981 年推出的 Smalltalk80 最具有影响力，成为面向对象程序设计语言发展史上的里程碑。

Smalltalk 有 5 个核心概念：对象、类、实例、消息和方法。对象是面向对象系统的唯一元素。在它的内部，包含了一些用来描述实体属性状态的私有变量和实现各种操作的方法。

类描述了一组性质相似的对象，类的每个对象被称为该类的一个实例。消息是发送者要求接收者为之实施某项操作的命令请求，接收者通过调用相应的方法响应消息。方法描述了操作的实现细节。

继承性是 Smalltalk 的特色。所谓继承是指子类可以调用父类的全部属性和操作，整个系统的数据是通过子类机制组成树形结构。这种机制为信息共享提供了有效的技术支持。Smalltalk 的基本语法结构是表达式。表达式是一个字符序列，所描述的对象被称为表达式的值。

在 Smalltalk 中，建立程序就是根据类创建对象，执行程序就是不断地向对象发送消息。Smalltalk 的主要特点是：信息表示与信息处理高度一致；属于弱类型语言；具有比较完善的抽象机制；语言融合于环境之中。

3. Eiffel 语言

Eiffel 是继 Smalltalk80 之后的另一种纯面向对象的程序设计语言。它是由 OOP 领域中的著名专家 B.Meyer 等人于 1985 年在美国交互软件公司(Interactive Software Engineering Inc) 设计的。它支持对象、类、方法、实例、消息、继承和动态联编等面向对象的基本概念，特别是支持多继承。就这点而言，Eiffel 是众多面向对象的程序设计语言的先驱。Eiffel 的主要特点是支持全面的静态类型化，拥有大量的软件开发工具。

静态类型化使得 Eiffel 能够保证在运行时发往对象的消息都可以被对象识别，动态联编则可以保证一条消息所请求执行的方法被动态的确定。

Eiffel 在许多方面克服了 Smalltalk80 存在的缺陷，借鉴了混合式面向对象的程序设计语言所采用的策略，又发展了独具特色的机制，因此，在面向对象程序设计领域中有较高的地位，也颇为引人注目。在 20 世纪 90 年代初期，用 Eiffel 开发的产品数目曾经一度仅次于 C++，名列第二。

4. C++语言

C++是一种十分流行的面向对象的程序设计语言。C++语言最先由 AT&T 公司 Bell 实验室计算机科学研究中心的 B.Stroustrup 在 20 世纪 80 年代初设计并实现。它以 C 语言为基础，增加了对数据抽象的支持，并具有面向对象的特征。

C++是对 C 语言的扩充，扩充的绝大部分内容借鉴于其他著名程序设计语言中的精华特性。例如，从 Simula67 中吸取了类，从 ALGOL60 中吸取了引用和在分程序中声明变量，综合了 Ada 的类属、抽象类和异常处理等。

一方面，C++保持了 C 语言的紧凑、灵活、高效和易于移植的优点，它对数据抽象的支持主要体现在类的机制，对面向对象的编程风范主要体现在虚拟函数。由于 C++既有数据抽象和面向对象的能力，又比其他面向对象语言的运行性能好。另一方面，加之 C 语言普及率非常高，从 C 语言过渡至 C++较为平滑，C++与 C 语言的兼容性好，使得大批 C 语言程序可以方便地在 C++环境下重用，因此 C++受到了广大软件开发人员的青睐，很快成为面向对象的主流语言。

5. Java 语言

为了保证 C 语言与 C++环境的极大兼容性，维护用户使用 C 语言的习惯，C++语言实际上既可以作为支持结构化程序设计的语言，又可以作为支持面向对象程序设计的语言使用，而 Java 语言则是一种完全的面向对象的程序设计语言。它由 SUN MicroSystem 公司于 1995 年 5 月正式发布，其简捷易学、面向对象、适用于网络分布环境、解释执行、支持多线程、