高等学校计算机应用规划教材

# UML系统建模基础教程

- 面向对象思想
- UML通用知识点
- Rational统一过程
- Rational Rose操作
- 使用Rose设计UML
- 用例图、序列图
- 类图与对象图
- 协作图、活动图
- 包图、构件图
- 部署图、状态图
- UML系统建模实例

胡荷芬 张帆 高斐 编著

清华大学出版社

高等学校计算机应用规划教材

# UML 系统建模基础教程

胡荷芬　张帆　高斐　编著

清华大学出版社

北　京

# 内 容 简 介

本书详细介绍了 UML 系统建模的思想和具体方法，内容包括面向对象思想、UML 通用知识点、Rational 统一过程、Rational Rose 的安装和操作、使用 Rose 设计 UML、用例图、类图、对象图、序列图、协作图、活动图、包图、构件图、部署图和状态图，最后以典型案例详解 UML 各种技术的综合应用。

本书采用理论结合案例的方法进行讲解，理论讲述清晰，技术讲解细致，案例丰富，在讲述 UML 案例时，结合了 Rational Rose 这个使用比较广泛的 UML 开发工具。每章的最后还提供了习题，供读者更好地了解和掌握 UML 技术。

本书可作为高等学校计算机及相关专业课程的教材，也适合 UML 初学者和网站开发人员参考使用。



**图书在版编目(CIP)数据**

UML 系统建模基础教程/胡荷芬，张帆，高斐　编著. —北京：清华大学出版社，2010.5
(高等学校计算机应用规划教材)

ISBN 978-7-302-22519-5

Ⅰ. U… Ⅱ. ①胡… ②张… ③高… Ⅲ. 面向对象语言，UML—程序设计—高等学校—教材 Ⅳ. TP312

中国版本图书馆 CIP 数据核字(2010)第 068299 号

**责任编辑**：刘金喜　鲍　芳
**装帧设计**：康　博
**责任校对**：胡雁翎
**责任印制**：杨　艳
**出版发行**：清华大学出版社　　**地　　址**：北京清华大学学研大厦 A 座
http://www.tup.com.cn　　**邮　　编**：100084
**社　　总　　机**：010-62770175　　**邮　　购**：010-62786544
**投稿与读者服务**：010-62776969，c-service@tup.tsinghua.edu.cn
**质　量　反　馈**：010-62772015，zhiliang@tup.tsinghua.edu.cn
**印 装 者**：北京鑫海金澳胶印有限公司
**经　　销**：全国新华书店
**开　　本**：185×260　**印　张**：19　**字　数**：439 千字
**版　　次**：2010 年 5 月第 1 版　　**印　　次**：2010 年 5 月第1 次印刷
**印　　数**：1～5000
**定　　价**：28.00 元

产品编号：022158-01

# 前　言

UML 是当前比较流行的一种建模语言，这种语言可以用于创建各种类型的项目需求、设计乃至上线文档。Rational Rose 是目前最为业界瞩目的可视化软件开发工具，通过 Rose 能用一种统一的方式设计各种项目的 UML 图。

UML 的设计动机是，让开发者用清晰和统一的方式完成项目的前期需求和设计文档，而这些需求和设计文档能够让项目的开发变得更加便捷和清晰。随着 UML 建模语言的深入，它已经获得了广泛的认同，目前已经成为主流的项目需求和分析建模语言。

本书选择 Rational Rose 作为开发 UML 的工具，不仅是因为它提供了绘制所有 UML 图的功能，而且还因为它能够有效地实现“建模到生成代码”的效果。

本书是一本关于 UML 的教材，书中包含了 UML 的基础知识、UML 的基本元素以及 UML 的使用方法，在讲述 UML 的使用过程中，是结合 Rational Rose 讲述的，从中大家能感受到 Rational Rose 开发 UML 的便捷性和高效性。同时，在讲述 UML 的元素时，结合了大量实战的案例，并且为了提高大家的学习效率，在每个章节后面还提供了一定数量的习题。

本书分为 15 章。书中各章的排布遵循从简单到复杂，从浅到深的思路，由于是基于实际项目的，所以这本书能让读者更快地掌握 UML 的基本元素和建模技巧，也能让读者学会通过 Rational Rose 开发 UML 的方法，是 UML 初学者必备的书籍。

**本书内容**

第 1 章：面向对象思想，介绍了面向对象思想的三大要素、面向对象与项目设计和用面向对象思想建立系统模型的方法。

第 2 章：UML 通用知识点概述，介绍了常用的 UML 元素、UML 的通用机制和 UML 的扩展机制。

第 3 章：Rational 统一过程，介绍了统一过程的概念、结构、配置和 Rational 统一过程的方法。

第 4 章：Rational Rose 的安装和操作，介绍了 Rational Rose 的安装和操作方法以及 Rational Rose 的操作技巧。

第 5 章：使用 Rose 设计 UML，介绍了 Rational Rose 的四种视图模型和 Rational Rose 生成代码的方式。

第 6 章：UML 统一建模语言，介绍了用例图的概念和构成要素、用例的重要元素、用例之间的各种重要关系和用 Rose 创建用例图的步骤。

第 7 章：类图和对象图，介绍了类图和对象图的基本概念，然后介绍了使用 Rose 创建类图的方式，随后介绍了对象图以及用 Rose 创建对象图的具体案例。

第 8 章：序列图，介绍了序列图的基本概念、序列图的组成、序列图在项目中的相关

概念、使用 Rose 创建序列图的方式以及使用 Rose 在实际项目中创建序列图的具体案例。

第 9 章：协作图，介绍了协作图的基本概念、组成协作图的元素、使用 Rose 创建协作图的方式以及使用 Rose 在实际项目中创建协作图的具体案例。

第 10 章：活动图，介绍了活动图的基本概念、组成活动图的元素、使用 Rose 创建活动图的方式以及使用 Rose 在实际项目中创建活动图的具体案例。

第 11 章：包图，介绍了包图的基本概念、使用 Rose 创建包图的方式以及使用 Rose 在实际项目中创建包图的具体案例。

第 12 章：构件图和部署图，介绍了构件图与部署图的基本概念、使用 Rose 创建构件图和部署图的方式以及使用 Rose 在实际项目中创建构件图和部署图的具体案例。

第 13 章：状态图，介绍了构成状态图的元素、状态图的组成、使用 Rose 创建状态图的方式以及使用 Rose 在实际项目中创建状态图的具体案例。

第 14 章和第 15 章，在这两章里，从需求分析讲起，分别通过网上选课系统、银行系统，介绍了创建系统用例图模型的方式、创建系统静态模型的方式、创建系统动态模型的方式和创建系统部署模型的方式。

**本书特点**

从入门到精通：本书遵循由浅入深、循序渐进的方式，按照知识点的梯度逐渐深入，这样编写的目的是让大家能快速地学习和掌握 UML 技术。

基于实战案例教学：本书的 UML 相关知识点都配套了实际的案例，能让读者了解到现实项目中 UML 的具体应用。

面向 Rational Rose：目前有很多种 UML 的开发工具，但 Rational Rose 在业内使用比较广泛，通过学习本书，能让读者了解到 Rational Rose 的常规用法。

习题配套：为了让读者快速掌握 UML 技术，每一章的后面提供了相关的填空题、选择题和上机题。

**学时安排**

本课程总学时为 36 学时，各章学时分配见下表（供参考）：

**学时分配建议表**

| 课 程 内 容 | 学 时 数 | | |
|---|---|---|---|
| | 合　计 | 讲　授 | 实　验 |
| 第 1 章 面向对象设计 | 1 | 1 | |
| 第 2 章 UML 通用知识点概述 | 2 | 2 | |
| 第 3 章 Rational 统一过程 | 2 | 2 | |
| 第 4 章 Rational Rose 的安装和操作 | 3 | 2 | 1 |
| 第 5 章 使用 Rose 设计 UML | 2 | 2 | |
| 第 6 章 用例图 | 3 | 2 | 1 |
| 第 7 章 类图与对象图 | 3 | 2 | 1 |

(续表)

| 课程内容 | 学时数 | | |
|---|---|---|---|
| | 合　计 | 讲　授 | 实　验 |
| 第 8 章 序列图 | 3 | 2 | 1 |
| 第 9 章 协作图 | 2 | 1 | 1 |
| 第 10 章 活动图 | 3 | 2 | 1 |
| 第 11 章 包图 | 2 | 1 | 1 |
| 第 12 章 构件图和部署图 | 3 | 2 | 1 |
| 第 13 章 状态图 | 3 | 2 | 1 |
| 第 14 章 网上选课系统 | 2 | 1 | 1 |
| 第 15 章 银行系统 | 2 | 1 | 1 |
| 合计 | 36 | 25 | 11 |

本书不仅可以作为大学计算机及相关专业的 UML 课程教材，也适合自学者及网站开发人员参考使用。

本书由胡荷芬、张帆、高斐、王坚宁主持编写，此外，郁伟、张彬、林美、李辉、田芳、王建国、赵海、刘峰、徐凤、周挺、山云峰、黄裕丹等同志也参与了本书的编写和最终的整理工作，在此，编者对他们表示衷心的感谢。

在本书的编写过程中，借鉴了许多现行教材的宝贵经验，在此，谨向这些作者表示诚挚的感谢。由于时间仓促，加之编者水平有限，书中难免有错误或不足之处。敬请广大读者批评指正。

编　者

# 目　录

# 第1章　面向对象设计

面向对象技术现在已经逐渐取代了传统的技术，成为当今计算机软件工程学中的主要开发技术，随着面向对象技术的不断发展，越来越多的软件开发人员加入到了它的阵营之中。面向对象技术之所以会为广大的软件开发人员青睐，是由于它作为一种先进的设计和构造软件的技术，使计算机以更符合人的思维方式去解决一系列的编程问题。使用面向对象技术编写的程序极大地提高了代码复用程度和可扩展性，使得编程效率也得到了极大的提高，同时减少了软件维护的代价。

面向对象技术发展的重大成果之一就是出现了统一建模语言(UML)。UML 是面向对象技术领域内占主导地位的标准建模语言，它统一了过去相互独立的数十种面向对象的建模语言共同存在的局面，通过统一语义和符号表示，系统地对软件工程进行描述、构造，形成了一个统一的、公共的、具有广泛适用性的建模语言。

## 1.1　面向对象思想的基本概念

面向对象和过去的软件开发技术完全不同，是一种全新的软件开发技术。面向对象的概念从问世到现在，它已经发展成为一种相对成熟的编程思想，并且逐步成为软件开发领域的主流技术。面向对象的程序设计(Object-Oriented Programming，OOP)旨在创建软件重用代码，具备更好的模拟现实世界环境的能力，这使它被公认为是自上而下编程的最佳选择。它通过给程序中加入扩展语句，把函数“封装”进编程所必需的“对象”中。面向对象的编程语言使得复杂的工作条理清晰、编写容易。说它是一场革命，不是对对象本身而言，而是对它们处理工作的能力而言。

### 1.1.1　什么叫面向对象

面向对象技术是一种以对象为基础，以事件或消息来驱动对象执行处理的程序设计技术。从程序设计方法上来讲，它是一种自下而上的程序设计方法，它不像面向过程程序设计那样一开始就需要使用一个主函数来概括出整个程序，面向对象程序设计往往从问题的一部分着手，一点一点地构建出整个程序。面向对象设计是以数据为中心，使用类作为表现数据的工具，类是划分程序的基本单位。而函数在面向对象设计中成了类的接口。以数据为中心而不是以功能为中心来描述系统，相对来讲，更能使程序具有稳定性。它将数据和对数据的操作封装到一起，作为一个整体进行处理，并且采用数据抽象和信息隐藏技术，最终将其抽象成一种新的数据类型——类

类与类之间的联系以及类的重用出现了类的继承、多态等特性。类的集成度越高，越

适合大型应用程序的开发。另外，面向对象程序的控制流程运行时是由事件进行驱动的，而不再由预定的顺序进行执行。事件驱动程序的执行围绕消息的产生与处理，靠消息的循环机制来实现。更加重要的是，我们可以利用不断成熟的各种框架，比如.NET 的.NET Framework 等，在实际的编程过程中，使用这些框架能够迅速地将程序构建起来。面向对象的程序设计方法还能够使程序的结构清晰简单，能够大大提高代码的重用性，有效地减少程序的维护量，提高软件的开发效率。

在结构上，面向对象程序设计和结构化程序设计也有很大的不同。结构化程序设计首先应该确定的是程序的流程怎样走，函数间的调用关系怎样，也就是函数间的依赖关系是什么。一个主函数依赖于其子函数，这些子函数又依赖于更小的子函数，而在程序中，越小的函数处理的往往是细节的实现，这些具体的实现又常常变化。这样的结果，就使程序的核心逻辑依赖于外延的细节，程序中本来应该是比较稳定的核心逻辑，也因为依赖于易变化的部分，而变得不稳定起来，一个细节上的小小改动，也有可能在依赖关系上引发一系列变动。可以说这种依赖关系也是过程式设计不能很好处理变化的原因之一，而一个合理的依赖关系，应该是倒过来的，由细节的实现依赖于核心逻辑才对。而面向对象程序设计是由类的定义和类的使用两部分组成的，主程序中定义数个对象并规定它们之间消息传递的方式，程序中的一切操作都是通过面向对象的发送消息机制来实现的。对象接收到消息后，启动消息处理函数完成相应的操作。

这里以常见的学生管理系统为例，我们使用结构化程序设计方法的时候，首先在主函数中确定学生管理要做哪些事情，分别使用函数将这些事情表示出来，使用一个分支选择程序进行选择，然后再将这些函数进行细化实现，确定调用的流程等。而使用面向对象技术来实现学生管理系统，对于该系统中的学生，则先要定义学生的主要属性，比如说学号、院系等，要对学生做什么操作，比如说查询学生信息、修改学生信息等。并且把这些当成一个整体进行对待，形成一个类，即学生类。使用这个类，我们可以创建不同的学生实例，也就是创建许多具体的学生模型，每个学生拥有不同的学号，一些学生会在不同的院系。学生类中的数据和操作都是给应用程序共享的，我们可以在学生类的基础上派生出中文系学生类、计算机系学生类、金融系学生类等，这样就可以实现代码的重用。

### 1.1.2　对象

对象(Object)是面向对象(Object-Oriented，简称 OO)系统的基本构造块，是一些相关变量和方法的软件集。对象经常用于建立现实世界中我们身边的一些对象的模型。对象是理解面向对象技术的关键。

我们可以看看现实生活中的对象，比如我们使用的手机、电脑和打印机等。我们可以认为万物皆是对象。根据《韦氏大词典》(*Merriam-Webster's Collegiate Dictionary*)的词典释义，对象有如下两点释义。

- 某种可为人感知的事物；
- 思维、感觉或动作所能作用的物质或精神体。

该释义的第一部分“某种可为人感知的事物”所指的便是我们熟悉的“对象”，它是

可以看到和感知到的“东西”，而且可以占据一定事物的空间。这个释义或许让我们感觉这是在上政治课，接下来让我们以学生管理系统为例，解释一下这个释义的第一部分。我们想象一下学生管理系统中围绕学生管理这个概念中应该有哪些物理对象。

- 被管理的信息所属的对象学生；
- 对学生信息进行管理的管理员；
- 对学生信息有权进行查询的校方人员；
- 管理信息的电脑，以及需要在电脑中存储的学生信息。

或许以上所列举的还不够。我们也可能在这个地方找到了很多的对象，但是它们并不都是所要创建的学生管理系统所必需的，不过我们不用担心这个，在后面的章节使用用例进行需求分析的时候，我们会进行详细的讲解。

释义的第二部分是“思维、感觉或动作所能作用的物质或精神体”，也就是我们所说的“概念性对象”，以学生管理系统为例，可以列举出如下一些。

- 学生所在的院系；
- 学生的学号；
- 学生的班级；
- 学生的成绩。

在这里也可以列举出不少这样的概念性对象。这些对象是人们不能看到的、听到的，但是在描述抽象模型时和物理对象时，仍然起着很重要的作用。

软件对象可以这样定义：所谓软件对象，是一种将状态和行为有机地结合起来而形成的软件构造模型，它可以用来描述现实世界中的一个对象。

也就是说软件对象实际上就是现实世界对象的模型，它有状态和行为。一个软件对象可以利用一个或者多个变量来标识它的状态。变量是由用户标识符来命名的数据项。软件对象可以利用它的方法来执行它的行为。方法是与对象相关联的函数(子程序)。

我们可以利用软件对象来代表现实世界中的对象。例如，用一个飞行试驾程序来代表现实世界中正在飞行的飞机，或者用机床数控程序来代表现实世界中运行的机床。同样也可以使用软件对象来表示抽象的概念，比如，点击按钮事件就是一个用在 GUI 窗口系统的公共对象，它可以代表用户点击程序界面中确定按钮的动作。

### 1.1.3　类

类(Class)是具有相同属性和操作的一组对象的组合，也就是说，抽象模型中的“类”描述了一组相似对象的共同特征，为属于该类的全部对象提供了统一的抽象描述。例如，名为“学生”的类被用于描述为被学生管理系统管理的学生对象。

类的定义要包含以下要素。

- 定义该类对象的数据结构(属性的名称和类型)；
- 类的对象在系统中所需要执行的各种操作，比如对数据库的操作。

类是对象集合的再抽象，类与对象的关系如同一个模具和使用这个模具浇注出来的铸件一样，类是创建软件对象的模板——一种模型。类给出了属于该类的全部对象的抽象定

义，而对象是符合这种定义的一个实体。类的用途有如下两点。

- 在内存中开辟一个数据区，存储新对象的属性。
- 把一系列行为和对象关联起来。

一个对象又被称作类的一个实例，也称为实体化(Instantiation)。术语“实体化(Instantiation)”是指对象在类声明的基础上创建的过程。比如说，我们声明了一个“学生”类，我们可以在这个基础上创建“一个姓名叫李刚的学生”这个对象。

类的确定和划分没有一个统一的标准和方法，基本上依赖于设计人员的经验、技巧以及对实际项目中问题的把握。通常的标准是“寻求共性、抓住特性”，即在一个大的系统环境中，寻求事物的共性，将具有共性的事物用一个类进行表述，在用具体的程序实现时，具体到某一个对象，要抓住对象的特性。确定一个类的步骤通常包含以下方面。

(1) 确定系统的范围，如学生管理系统，需要确定一下与学生管理相关的内容。

(2) 在系统范围内寻找对象，该对象通常具有一个和多个类似的事物。比如，在学生管理中，某院系有一个名叫李刚的学生，而另一个院系名叫王芳的人是和李刚类似的，都是学生。

(3) 将对象抽象成为一个类，按照上面类的定义，确定类的数据和操作。

在面向对象程序设计中，类和对象的确定非常重要，是软件开发的第一步，软件开发中类和对象的确定直接影响到软件的质量。如果划分得当，对于软件的维护与扩充以及体现软件的重用性方面，都非常重要。

### 1.1.4　消息与事件

当使用某一个系统的时候，我们点击鼠标的左键，通常会显示相应的信息。以学生管理系统为例，通过鼠标点击“学生管理系统”界面某菜单的时候，会显示出当前的操作人所需要的信息。那么当前的程序是如何运行的呢？

(1) “学生管理系统”界面的某一个菜单项发送鼠标点击事件给相应的对象一个消息。

(2) 对象接收到消息后有所反应，把操作者需要的信息显示在界面。

(3) 界面将相关信息显示出来，完成任务。

可以看得出，在这个过程中，我们首先要触发一个事件，然后发送消息，那么消息是什么呢？所谓消息(Message)，是指描述事件发生的信息，是对象间相互联系和相互作用的方式。一个消息主要由五部分组成：消息的发送对象、消息的接收对象、消息的传递方式、消息内容(参数)、消息的返回。传入消息内容的目的有两个，一个是让接收请求的对象获取执行任务的相关信息，另一个是行为指令。

那么什么是事件呢？所谓事件，通常是指一种由系统预先定义而由用户或系统发出的动作。事件作用于对象，对象识别事件并作出相应的反应。与对象的方法集可以无限扩展不同，事件的集合通常是固定的，用户不能随便定义新的事件。但是现代高级语言中可以通过一些其他技术在类中加入事件。我们通常所熟悉的一些事件，比如 Click，鼠标左键单击对象时发生的事件；Load，当界面被加载到内存中时发生的事件等。

对象通过对外提供的方法在系统中发挥自己的作用，当系统中的其他对象请求这个对

象执行某个方法时，就向该对象发送一个消息，对象响应这个请求，完成指定的操作。程序的执行取决于事件发生的顺序，由顺序产生的消息来驱动程序的执行，而不必预先确定消息产生的顺序。

## 1.2　面向对象的三大要素

封装、继承、多态是面向对象程序的三大特征，这些特征保证了程序的安全性、可靠性、可重用性和易维护性。随着技术的发展，把这些思想用于硬件、数据库、人工智能技术、分布式计算、网络、操作系统等领域，越来越显示出其优越性。

### 1.2.1　封装

封装(Encapsulation)就是把对象的状态和行为绑到一起的机制，使对象形成一个独立的整体，并且尽可能地隐藏对象的内部细节。封装有两个含义：一是把对象的全部状态和行为结合在一起，形成一个不可分割的整体。对象的私有属性只能够由对象的行为来修改和读取。二是尽可能隐蔽对象的内部细节，与外界的联系只能够通过外部接口来实现。

封装的信息屏蔽作用反映了事物的相对独立性，我们可以只关心它对外所提供的接口，即能够提供什么样的服务，而不用去关注其内部的细节问题。比如说使用手机，我们关注的通常是这个手机能实现什么功能，而不太会去关心这个手机是怎么一步步制造出来的。

封装的结果使对象以外的部分不能随意更改对象的内部属性或状态，如果需要更改对象内部的属性或状态，则需要通过公共访问控制器来进行。通过公共访问控制器来限制对象的私有属性，有以下好处。

- 避免对封装数据的未授权访问。
- 当对象为维护一些信息，并且这些信息比较重要，不能够随便向外界传递，这个时候，只需要将这些信息属性设置为私有的即可。
- 帮助保护数据的完整性。
- 当对象的属性设置为公共访问的时候，代码可以不经过对象所属类希望遵循的业务流程而去修改对象的值，对象很容易失去对其数据的控制。我们可以通过访问控制器来修改私有属性的值，并且在赋值或取值的时候检查属性值的正确与否。
- 当类的私有方法必须修改时，限制了对整个应用程序内的影响。

当对象采用一个公共的属性去暴露的时候，我们知道，甚至修改一下这个公共属性的名称，程序都需要修改这个公共属性被调用的地方。但是，通过私有的方式就能够缩小其影响的范围，将程序的影响范围缩小到一个类中。

比如说我们房子就是一个类的实例，室内的装饰和摆设只能被室内的居住者欣赏和使用，如果没有四面墙的遮挡，室内的所有活动在外人面前将一览无遗。由于有了封装，房屋内的所有摆设都可以随意改变且不影响他人，然而，如果没有门窗，即使它的空间再宽

阔，也没有实用的价值。房屋的门窗，就是封装对象暴露在外的属性和方法，专供人进出，以及空气流通和带来阳光。

但是在实际项目中，如果一味地强调封装，对象的任何属性都不允许外部直接读取，反而会增加许多无意义的操作，为编程增加负担。为避免这一点，在语言的具体使用过程中，应该根据需要和具体情况，来决定对象属性的可见性。

### 1.2.2　继承

对于客观事物的认知，既应当看到其共性，也应该看到其特性。如果只考虑事物的共性，不考虑事物的特性，就不能反映出客观世界中事物之间的层次关系，从而不能完整地、正确地对客观世界进行抽象的描述。如果说运用抽象的原则就是舍弃对象的特性，提取其共性，从而得到适合一个对象集的类的话，那么在这个类的基础上，再重新考虑抽象过程中被舍弃的那一部分对象的特性，则可以形成一个新的类，这个类具有前一个类的全部特征，是前一个类的子集，从而形成一种层次结构，即继承结构。以动物为例，可以分为哺乳动物、爬行动物、两栖动物和鸟类等，我们通过抽象的方式实现一个动物类以后，可以通过继承的方式分别实现哺乳动物、爬行动物、两栖动物、鸟类等类，并且这些类包含动物的特性，图 1-1 展示了这样一个继承的结构。

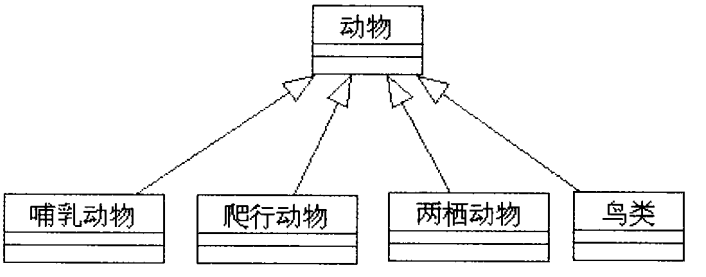


图 1-1　动物类继承结构示例

继承(Inheritance)是一种连接类与类之间的层次模型。继承是指特殊类的对象拥有其一般类的属性和行为。继承意味着“自动地拥有”，即在特殊类中不必重新对已经在一般类中定义过的属性和行为进行定义，而是自动地、隐含地拥有其一般类的属性和行为。继承对类的重用性提供了一种明确表述共性的方法。即一个特殊类既有自己定义的属性和行为，又有继承下来的属性和行为。尽管继承下来的属性和行为在特殊类中是隐式的，但无论在概念上还是在实际效果上，都是这个类的属性和行为。继承是传递的，当这个特殊类被它更下层的特殊类继承的时候，它继承来的和自己定义的属性和行为又被下一层的特殊类继承下去。我们有时把一般类称为基类，把特殊类称为派生类。

继承在面向对象软件开发过程中，有其强有力和独特的一面，通过继承可以实现以下几点。

- 使派生类能够比不使用继承直接进行描述的类更加简洁。派生类只需要描述那些与基类不相同的地方、特殊的地方，且把这些添加到类中然后继承就可以了。不使用

继承而去直接描述，需要将基类的属性和行为全部进行描述一遍。

- 能够重用和扩展现有类库资源。当我们使用已经封装好的类库的时候，如果需要对某个类进行扩展，通过继承的方式很容易实现，而不需要再重新编写，并且扩展一个类的时候并不需要其源代码。
- 使软件易于维护和修改。当要修改或增加某一属性或行为时，只需要在相应的类中进行改动，而它派生的所有类全都自动地、隐含地做了相应的修改。

在软件开发过程中，继承性实现了软件模块的可重用性、独立性，缩短了开发的周期，提高了软件的开发效率，同时使软件易于维护和修改。继承是对客观世界的直接反映，通过类的继承，能够实现对问题的深入抽象的描述，也反映出人类认知问题的发展过程。

### 1.2.3　多态

多态是指两个或多个属于不同类的对象对于同一个消息或方法调用所做出不同响应的能力。面向对象设计也借鉴了客观世界的多态性，体现在不同的对象可以根据相同的消息产生各自不同的动作。例如，我们在“动物”基类中定义了“进食”这个行为，派生类“猫”和“狗”都继承了动物类的进食行为，但其进食的事物却不一定相同，猫喜欢吃鱼，而狗喜欢啃骨头。这样一个进食的消息发出以后，猫类和狗类的对象接收到这个消息后各自执行不同的进食行为。如图 1-2 所示就是多态性的表现。

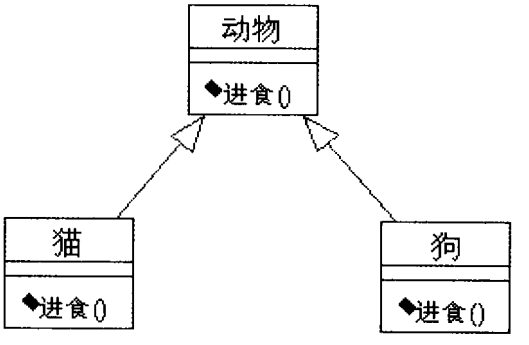


图 1-2　图形多态性示例

具体到面向对象程序设计来讲，多态性(Polymorphism)是指在两个或多个属于不同类的对象中，同一函数名对应多个具有相似功能的不同函数，可以使用相同的调用方式来调用这些具有不同功能的同名函数。

继承性和多态性的结合可以生成一系列虽类似但独一无二的对象。由于继承性，这些对象共享许多相似的特征；由于多态性，针对相同的消息，不同的对象可以有独特的表现方式，实现个性化的设计。

上述面向对象技术的几个特征的运用，对提高软件的开发效率起着非常重要的作用，通过编写可重用代码，编写可维护代码，修改代码模块，共享代码等方法可以充分发挥其优势。