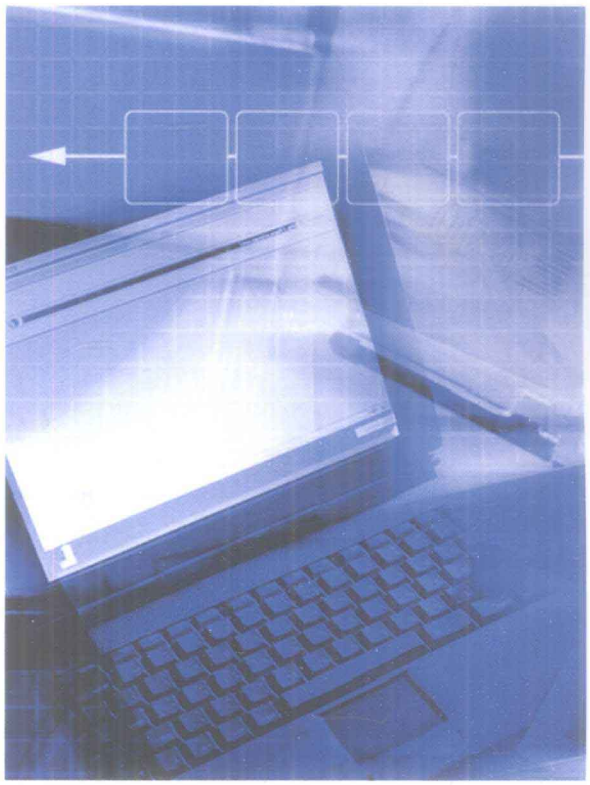
高等学校计算机应用规划教材

UML面向对象分析与设计教程

- UML建模语言概述
- Rational Rose建模工具
- 类图和对象图
- 包图和用例图
- 序列图和活动图
- 协作图和状态图
- 构件图和部署图
- Rational统一过程
- UML建模应用



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面向对象分析与设计教程

胡荷芬　高　斐　编著

清华大学出版社

北　京

内 容 简 介

UML 是当前比较流行的一种建模语言，它可以用于创建各种类型的项目需求、设计乃至上线文档，特别适合采用面向对象的思维方式进行软件建模。规范化、可视化的软件建模已成为当今软件技术的主流之一。

本书依据统一建模语言 UML 与面向对象编程语言，结合实际案例，深入全面地探讨了软件建模的概念、规范和方法。全书共 13 章，前 3 章介绍了面向对象、UML 建模语言和 Rational Rose 建模工具的一些基本理论和概念。第 4~12 章详尽地介绍了 UML 中类图、对象图、用例图、包图、序列图、协作图、活动图、状态图、构件图和部署图的概念及其在实际中的使用。最后，通过一个综合案例对使用 Rational Rose 进行 UML 建模的全过程进行了深入剖析。此外，各章后面配有适量的练习题和上机题，以加深读者的理解和提高。

本书最大的特点是将理论和实际紧密地结合，实例丰富、图文并茂，讲解详尽、实践性强。

本书可以作为高等院校计算机软件工程专业 UML 和面向对象技术的教材，也可以作为广大软件开发人员和系统架构分析设计人员自学 UML 的参考书。

图书在版编目(CIP)数据

UML 面向对象分析与设计教程/胡荷芬，高斐 编著. —北京：清华大学出版社，2012.5

(高等学校计算机应用规划教材)

ISBN 978-7-302-28541-0

Ⅰ. ①U…　Ⅱ. ①胡… ②高…　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(2012)第 066479 号

责任编辑：王　定　胡花蕾
封面设计：牛艳敏
版式设计：孔祥丰
责任校对：蔡　娟
责任印制：何　芊

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　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
课 件 下 载：http://www.tup.com.cn,010-62794504

印 刷 者：北京四季青印刷厂
装 订 者：三河市李旗庄少明印装厂
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张**：18.25　　**字　　数**：421 千字
版　　次：2012 年 5 月第 1 版　　**印　　次**：2012 年 5 月第 1 次印刷
印　　数：1～5000
定　　价：30.00 元

产品编号：042354-01

前　言

UML 是当前比较流行的一种建模语言，它可以用于创建各种类型的项目需求、设计乃至上线文档，特别适合采用面向对象的思维方式进行软件建模。随着 UML 建模语言的不断发展，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。Rational Rose 是目前最为业界瞩目的可视化软件开发工具，通过它可以便捷高效地完成 UML 的建模工作。

本书是一本关于 UML 的教材，书中包含了 UML 的基础知识、基本元素和使用方法，在讲述 UML 的使用过程中，是结合 Rational Rose 讲述的，从中大家能感受到使用 Rational Rose 开发 UML 的便捷性和高效性。在讲述 UML 的同时，结合了大量的实战案例，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。相信不同的读者都会在本书的学习过程中得到不同的收获。

本书内容

本书共分 13 章，其中：

第 1 章　基于面向对象的 UML：主要介绍了面向对象的基本知识、建立系统模型的方法和 UML 统一建模语言的基本内容。学习本章的目的是使读者对 UML 有初步的认识。

第 2 章　UML 语言综述：对 UML 语言进行了系统的介绍，包括常用的 UML 元素、UML 的通用机制、UML 的扩展机制等必须了解的知识点。

第 3 章　UML 工具——Rational Rose：对 UML 的主流开发工具——Rational Rose 进行了介绍，包括其安装、使用、四种视图模型和生成代码的方法。本章为后面绘制 UML 图打好了基础。

第 4 章　类图和对象图：介绍了类图和对象图的基本概念及其在 UML 中的表示。通过讲解使用 Rose 创建类图的方式，使读者在学完本章后，能够根据类图和对象图描绘出系统的静态结构。

第 5 章　包图：对包图中的基本概念和它们的使用方法逐一进行了详细介绍。通过本章的学习，能够熟练使用包图描述系统的组织结构。

第 6 章　用例图：介绍了用例图的概念和作用，讲解了用例图的重要组成元素和如何通过 Rational Rose 创建用例图。

第 7 章　序列图：针对 UML 中交互图之一的序列图进行了介绍，包括它的基本概念、组成、在项目中的相关概念和使用 Rose 创建序列图的方法。

第 8 章　活动图：介绍了活动图的基本概念、组成元素及使用 Rose 创建活动图的方法。

第 9 章　协作图：协作图是交互视图的另外一种表现方式。通过本章的学习，可以掌握协作图的基本概念和使用方法，进而能够熟练使用协作图描述系统中对象之间的交互。

第 10 章　状态图：介绍了构成状态图的元素、状态图的组成。通过本章的学习，能够从整体上理解状态图并掌握状态图的画法。

第 11 章　构件图和部署图：对 UML 中描述系统的物理实现和物理运行情况的构件图和部署图进行了详细的讲解。在学完本章后，能够根据构件图和部署图的基本概念，创建图中的各种模型元素，描绘出系统的物理结构，并能将前面介绍过的其他图结合起来，完成对整个系统的建模。

第 12 章　Rational 统一过程：介绍了统一过程的概念、结构、配置和实现 Rational 统一过程的方法。

第 13 章　网上购物商店：给出了一个网上购物商店的综合实例，将前面各章介绍的 UML 的各种图形和模型元素综合起来，实现对一个完整系统的设计和建模过程。整个过程按照软件设计的一般流程进行，以便对实际项目的系统建模有一个直观的认识。

本书特点

理论实际，紧密结合。本书在介绍理论知识的同时，每一章都给出了大量案例讲解，力求让读者在理解基础知识后，就能学以致用，快速上手。每章都配有相应的课后习题和上机练习，方便读者课后的实践练习。

配套源码，免费课件。为了方便读者自学，本书不仅在每章后面附有精心设计的课后习题和上机练习，还配套有免费的课件、实例源码等，这样就能使读者牢固而扎实地掌握各种基本知识点。

图文并茂，步骤详细。在具体介绍 Rational Rose 软件功能的时候，本书提供了详细的图例，详尽地说明了每一步功能的实现，让读者一眼就能明了整个功能的使用方法和绘制步骤。每一个步骤都以通俗易懂的语言进行讲述，读者只需要按照步骤操作，就可以轻松完成软件的建模。

面向读者

本书既可以作为高等院校计算机软件工程专业学生学习 UML 和面向对象的技术教材，也可作为广大软件开发人员和系统架构分析设计人员自学 UML 的参考书。

鸣　谢

本书由胡荷芬、高斐主持编写，此外，胡书敏、林丹、李辉、田芳、王建国、赵海峰、刘勇、徐超、周建军、徐兵、黄飞、林海、马建华、孙明、高峰、郑勇、刘建、李彬、彭丽、许小荣等同志在整理材料方面给予了编者很大的帮助，在此，对他们表示衷心的感谢。

由于作者水平有限，书中不足之处在所难免，欢迎广大读者、同仁批评指正。

编　者

2012 年 3 月

目　录

第1章　基于面向对象的UML

UML 是一种在多种面向对象建模方法的基础上发展的通用可视化建模语言，它拥有一整套完整而成熟的建模技术，被广泛地运用于各种不同的领域。借助于基于面向对象的UML 可以帮助软件工程的开发人员更好地理解业务流程，建立更可靠、更完善的系统模型，从而方便我们对各种软件工程进行正确的描述和交流。

1.1　面向对象是 UML 的基础

UML 统一建模语言的出现正是由于面向对象建模思想发展的产物，它是软件工程领域公认的面向对象的建模语言。可以毫不夸张地说，没有面向对象，就没有 UML。它们的关系是如此的密不可分。

1.1.1　什么是面向对象

从 20 世纪 60 年代提出面向对象的概念到现在，面向对象已经发展成为一种比较成熟的编程思想，并且逐步成为软件开发领域的主流技术。面向对象程序设计(Object-Oriented Programming，OOP)立足于创建软件代码的重复使用，具备更好地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。

1. 什么是对象

对象(Object)是面向对象(Object-Oriented，OO)系统的基本构造块，是一些相关的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，如在房间里面的桌子、椅子、电脑等。我们都可以认为是对象。根据《韦氏大词典》(Merriam-Webster's Collegiate Dictionary)，对象包含了以下两种释义：

(1) 某种可为人感知的事物。

(2) 思维、感觉或动作所能作用的物质或精神体。

第一种释义“某种可为人感知的事物”所指的是我们可以看到和感知到的物理对象，并且它占据一定事物的空间。这样说可能比较抽象，下面以“仓库管理系统”为例，解释一下“某种可为人感知的事物”的具体含义。先想一下在仓库管理这个概念中应该有哪些物理对象：

- 到仓库来领取或外借物料的员工
- 负责仓库的仓库管理人员
- 管理仓库信息的电脑
- 领取或外借仓库中的物料
- 存放物料的货架
- 仓库本身这一建筑物

以上列举的其实并没有涵盖“仓库管理系统”中所有的对象，因为其他一些对象对仓库管理系统而言并不是必须的。

第二种释义“思维、感觉或动作所能作用的物质或精神体”，也就是指“概念性对象”。以仓库管理系统为例，可以列举出：

- 领取或外借仓库物料的员工所在部门
- 员工的工号
- 仓库中存放的物料编号

这些对象是我们不能看到、听到的，但是在描述抽象模型和物理对象时，仍然起着很重要的作用。

在软件工程设计中的对象和上面词典中对象的含义又有所不同。软件工程中的对象，是指一种将状态和行为有机结合起来形成的软件构造模型，它可以用来描述或代表现实世界中的一个对象。也可以这样说，软件对象其实就是现实世界对象的一种模型，它有自己的状态和行为。

可以利用一个或者多个变量来标识软件对象的状态。变量是指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，而方法则是与对象相关联的函数(子程序)。

2. 面向对象与面向过程的区别

在面向对象程序设计(OOP)方法之前，结构化程序设计占据主要的地位。结构化程序设计是一种自上而下的设计方法，通常使用一个主函数来概括出整个程序需要做的事，而主函数是由一系列子函数所组成的。对于主函数中的每一个子函数，又都可以被分解为更小的函数。结构化程序设计思想就是把大的程序分解成具有层次结构的若干个模块，每个模块再分解为下一层模块，如此自顶向下，逐步细分，把复杂的大模块分解为许多功能单一的小模块。结构化程序设计特征就是以函数为中心，也就是以功能为中心来描述系统，用函数来作为划分程序的基本单位，数据在过程式设计中往往处于从属的位置。结构化程序设计的优点是易于理解和掌握，这种模块化、结构化、自顶向下、逐步求精的设计原则与大多数人的思维和解决问题的方式比较接近。

但是，对于比较复杂的问题，或在开发中需求变化比较多的时候，结构化程序设计往往显得力不从心。事实上，开发一个系统的过程往往也是一个对系统不断了解和学习的过程，而结构化程序设计是自上而下的，这要求设计者在一开始就要对需要解决的问题有一定的了解。在问题比较复杂的时候，要做到这一点会比较困难。当开发中需求发生变化的

时候，以前对问题的理解也许会变得不再适用。

结构化程序设计的方法把密切相关、相互依赖的数据和对数据的操作相互分离，这种实质上的依赖与形式上的分离也使得大型程序的编写比较困难，难于调试和修改。在由很多人进行协同开发的项目组中，程序员之间很难读懂对方的代码，代码的重用变得十分困难。由于现代应用程序的规模越来越大，对代码的可重用性和易维护性要求也越来越高，面向对象技术对这些提供了很好的支持。

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，最终被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。

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。更重要的是，可以利用不断成熟的各种框架，如.NET 的.NET Framework 等，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，以及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实现，这些具体的实现又常常变化。这样的结果，就是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，由细节实现依赖于核心逻辑才对。而面向对象程序设计由类的定义和类的使用两部分组成，主程序中定义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通过面向对象的发送消息机制来实现。对象接收到消息后，启动消息处理函数完成相应的操作。

以“仓库管理系统”为例，使用结构化程序设计方法的时候，首先要在主函数中确定仓库管理要做哪些事情，分别使用函数将这些事情进行表示，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。

而在使用面向对象技术来实现的仓库管理系统中，以领取仓库物品的员工为例。首先要了解该员工的主要属性，如工号、所属部门等；其次要确定该员工要做什么操作，如领

取或外借物料、办理手续等。并且把这些当成一个整体进行对待，形成一个类，即员工类。使用这个类，可以创建不同的员工实例，也就是创建许多具体的员工模型，每个员工拥有不同的工号，一些员工在不同的部门，都可以在仓库中领取材料或物品。员工类中的数据和操作都是给应用程序进行共享的，可以在员工类的基础上派生出普通员工类、部门经理类、总经理等，这样可以实现代码的重用。

3. 对象与类的确定

面向对象技术认为客观世界是由各种各样的对象组成的，每个对象都有自己的数据和操作，不同对象之间的相互联系和作用构成了各种系统。在面向对象程序设计中，系统被描绘成一系列完全自治、封装的对象组成，对象和对象之间是通过对象暴露在外的接口进行调用的。对象是组成系统的基本单元，是一个组织形式的含有信息的实体。而类是创建对象的模板，在整体上可以代表一组对象。例如，创建人这个类，它就代表人这个概念。现在有一个名字叫李平的人，就表示李平是“人”这个类的一个实体对象，并且包含了“姓名是李平”这样一个信息。可以使用这个类来表达李平、章飞等具体的对象。这样，设计类而不是设计对象就可以避免重复编码，类只需要编码一次，就可以被实例化属于这个类的无数个对象。

对象是由状态(Attribute)和行为(Behavior)构成的。实际上，状态、属性(Property)、数据(Data)等这些在各种书中提到的概念，都是用于描述一个对象的数据元素，这些概念具体到各种语言便有不同的叫法。对象的状态值用来定义对象的状态。例如，当判断员工是否可以领取仓库中物料的时候，可以通过员工的工号(不妨称为第一个状态)和员工目前领取物料的数量(可以称为第二个状态)来进行判断。行为、操作(Operation)以及方法(Method)这些在各种书中提到的概念，是用于描述用以访问对象的数据或修改/维护数据值的方法，如在描述领取或外借物料员工的行为时，“告诉仓库管理员工号”和“选择要领取或外借的物料”等。

对象只有在具有状态和行为的情况下才有意义，“状态”用来描述对象的静态特征，“行为”用来描述对象的动态特征。对象是包含客观事物特征的抽象实体，封装了状态和行为。在程序设计领域，可以用“对象=数据+数据的操作”来进行表达。

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“员工”的类被用于描述能够到仓库中领取或外借物料的员工对象。

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)。
- 对象所要执行的操作，也就是类的对象要被调用执行哪些操作，以及执行这些操作时对象要执行哪些操作，如数据库操作等。

类是对象集合的抽象，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。类用来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。

一个对象又被称做类的一个实例，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。例如，声明了一个“员工”类，可以在这个基础上创建“一个姓名叫李平的员工”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。确定一个类通常包含以下方面：

- 确定系统的范围，如仓库管理系统，需要确定一下和仓库管理相关的内容。
- 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。例如，仓库管理系统中，有一个某部门名叫李平的员工，某部门名叫章飞的人是和李平类似的，都是普通员工。
- 将对象抽象成为一个类，按照上面的类的定义，确定类的数据和操作。

在面向对象程序设计中，类和对象的确定是软件开发非常重要的一步，类和对象的确定直接影响软件设计的优劣。如果划分得当，对于软件的维护与扩充以及软件的重用性都非常重要。

4. 消息和事件

当使用某一个系统的时候，单击一个按钮，通常会显示相应的信息。以仓库管理系统为例，单击“仓库管理系统”界面某一个按钮的时候，会显示出当前仓库的信息。那么，当前的程序是如何运行的呢？

- “仓库管理系统”界面的某一个按钮发送鼠标单击事件给相应的对象一个消息。
- 对象接收到消息后有所反应，它提供了仓库的相关信息给界面。
- 界面将仓库的相关信息显示出来，完成任务。

在这个过程中，首先要触发一个事件，然后，发送消息，那么什么是消息呢？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 5 部分组成：消息的发送对象、消息的接收对象、消息传递方式、消息内容(参数)、消息的返回。传入消息内容的目的有两个：一个是让接受请求的对象获取执行任务的相关信息，另一个是行为指令。

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。但在现代高级语言中通过一些其他的技术可以在类中进行事件的加入。最熟悉的事件就是，用鼠标左键单击对象时发生的事件 Click 和界面被加载到内存中时发生的 Load 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，不必预先确定消

息产生的顺序。

1.1.2　面向对象的基本特征

面向对象技术强调的是在软件开发过程中，对于客观世界或问题域中的事物的认知，采用人类在认知客观世界的过程中普遍运用的思维方法，直观、自然地描述有关事物。

抽象、封装、继承、多态是面向对象程序的基本特征。正是这些特征使得程序的安全性、可靠性、可重用性和易维护性得以保证。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. 抽象

现代人每天都要获得大量信息，例如电子邮件、新闻信息等，但是我们的大脑懂得如何去简化所接收的信息，从中提取出重要的部分，让信息细节通过抽象(Abstraction)过程进行管理。通过抽象可以做到以下几点：

(1) 将需要的事物进行简化。通过抽象能够识别和关注当前状况或物体的主要特征，淘汰掉所有非本质信息。也就是说，通过抽象可以忽略事物中与当前目标无关的非本质特征，强调与当前目标有关的特征。同时抽象与真实世界的不同，特征是根据使用对象的不同或者说是使用者的类型不同进行确定的。

以仓库管理系统为例。从员工的角度来看，关注的是仓库管理系统能否领取或借到需要的物料。对于仓库管理人员，关注的往往是能否更好地管理好仓库和物品等。

(2) 将事物特征进行概括。如果能够从一个抽象模型中剔出足够多的细节，那它将变得非常通用，能够适用于多种情况或场合。这样的通用抽象常常相当有用。例如，以员工为例，抽象出“工号”、“姓名”、“所属部门”、“职位”等来描绘员工，能够描绘出员工的一般功能，但是要具体到员工在企业里的工作地点等，描述力就差了一些。

抽象模型越简单，展示的特点越少，它就越通用，也越具有普适性。抽象越复杂，越具有限制性，用于描述的情况也就越少。

(3) 将抽象模型组织为层次结构。能够通过抽象按照一定的标准将信息系统地进行分类处理，依此来应付系统的复杂性。这一过程被称为分类(Classification)。

例如在科学领域，将园林树木分为林木类、花木类、果木类和叶木类。当按照各种不同的分科规则，还可以将花木类进行分类，将其继续区分为牡丹、海棠等，直到能够构造出一个自顶向下渐趋复杂的抽象层次结构为止。例如，可将紫叶李和石楠划分为叶木类，将竹类划分为林木类。图 1-1 所示是园林树木抽象层次的一种结构。

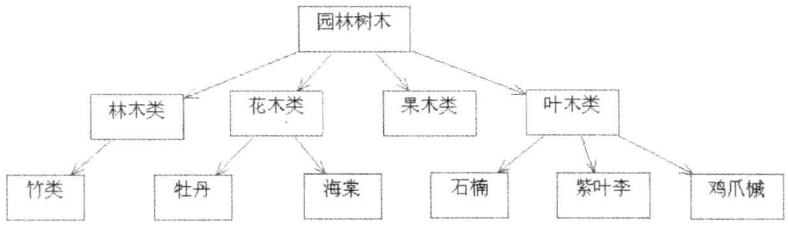


图 1-1　园林树木抽象层次结构

然而，在抽象过程中，由于规则的不严格性，会导致抽象面临着挑战。例如，鲸鱼可以划分在哺乳类中，但是也有标准让其划分在鱼类中。合适的规则集合对于抽象非常重要。

(4) 将软件重用得以保证。抽象强调实体的本质、内在的属性。在认知新东西的时候，通常会搜索以前创建和掌握的抽象模型，用来更好地抽象。当认知到飞机后，抽取出飞机的抽象模型，当去认知直升飞机的时候通常会自动联想到飞机。我们将这种进行特性对比，并且找到可供重用的近似抽象的过程，称为模式匹配和重用。在软件系统开发过程中，模式匹配和重用也是面向对象软件开发的重要技术之一，它避免了每做一个项目必须重新开始的麻烦。如果能够充分地利用抽象的过程，在项目实施中将获得极大的生产力。良好的抽象，再怎么着也不会过时。

抽象忽略了事物中与当前目标无关的非本质特性，强调与当前事物相关的特性，并将事物正确地归类，得出事物的抽象模型，并且为对象的重用提供了保障。我们是通过类来实现对象状态和行为的抽象。

2. 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。比如说一台电脑，关注的通常是这台电脑能够干什么，而不用关注其芯片是怎样做出来的。

封装的结果使对象以外的部分不能随意去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，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：

(1) 避免对封装数据的未授权访问。当对象为维护一些信息，这些信息比较重要，不能够随便向外界传递时，只需要将这些信息属性设置为私有即可。

(2) 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。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。

(3) 当类的私有方法必须修改时，限制了在整个应用程序内的影响。当对象采用一个