HZ BOOKS
华章教育

高等院校软件工程专业规划教材

UML系统分析与设计

薛均晓 李占波 主 编
李庆宾 韩 英 副主编
李俊锋 张朝阳 参 编

UML System Analysis and Design

机械工业出版社
China Machine Press

高等院校软件工程专业规划教材

UML系统分析与设计

薛均晓 李占波 主 编
李庆宾 韩 英 副主编
李俊锋 张朝阳 参 编

UML System Analysis and Design

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

UML 系统分析与设计 / 薛均晓，李占波主编 . —北京：机械工业出版社，2014.9
（高等院校软件工程专业规划教材）

ISBN 978-7-111-47669-6

I. U…　II. ① 薛…　② 李…　III. 面向对象语言 - 程序设计 - 高等学校 - 教材　IV. TP312

中国版本图书馆 CIP 数据核字（2014）第 187507 号

本书根据软件工程专业对 UML 系统分析与设计的教学要求编写而成。全书共分为面向对象系统分析与设计引论、统一建模语言 UML 与建模工具 Rational Rose、需求分析与用例模型、系统静态分析与静态模型、系统动态分析与交互模型、系统动态分析与行为模型、系统设计与实现模型、软件工程引论与统一软件过程 RUP 以及综合案例等 9 章。书中对 UML 和面向对象系统分析与设计的基本概念、理论和方法进行了严谨、系统的阐述，每一章均配备了案例和不同难度的习题，注重理论联系实践，可作为高等学校软件工程专业课程的教材，也可以作为软件开发人员学习软件建模技术，进行软件系统分析与设计，以及从事工程开发的参考用书。

出版发行：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：李　艺　　责任校对：殷　虹
印　　刷：三河市宏图印务有限公司　　版　　次：2014 年 9 月第 1 版第 1 次印刷
开　　本：185mm × 260mm　1/16　　印　　张：15
书　　号：ISBN 978-7-111-47669-6　　定　　价：35.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换
客服热线：（010）88378991　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzjsj@hzbook.com

前　言

自20世纪40年代问世以来，计算机在人类社会的各个领域得到了广泛的应用。随着性能的提高和应用范围的迅速扩大，计算机软件系统的规模越来越大，复杂程度越来越高，软件可靠性问题也越来越突出。为了解决长期以来计算机软件开发的低效率问题，面向对象分析与设计成为现代软件企业广为采用的一项有效技术。

统一建模语言（Unified Modeling Language，UML）是一种定义良好、易于表达、功能强大且普遍适用的建模语言。它融入了软件工程领域的新思想、新方法和新技术。它的作用域不只局限于支持面向对象的分析与设计，还支持从需求分析开始的软件开发的全过程。UML综合了在大型、复杂系统的建模领域得到认可的优秀的软件工程方法。至少在近10年内，UML是面向对象技术领域内占主导地位的标准建模语言。

UML的重要内容包含下列五类模型：

1）用例模型：从用户角度描述系统功能，并指出各功能的操作者。

2）静态模型：描述系统的静态结构，包括类图和对象图。

3）交互模型：描述对象间的交互关系，包括序列图和协作图。

4）行为模型：描述系统的动态模型和组成对象间的交互关系，包括状态图和活动图。

5）实现模型：描述系统代码部件的物理结构及系统中软硬件的物理体系结构，包括包图、构件图和部署图。

Rational Rose是基于UML的可视化建模工具，使用它可以进行项目需求分析、结构规划和生成框架代码。

本书将以“实用性”和“应用性”为基本理念，坚持“理论扎实为主”和“实践操作为重”的原则，在介绍系统分析与设计以及UML统一建模语言基础理论的基础上，通过完整的建模案例讲解如何在工程实践中使用面向对象的思想和UML建模方法。

在内容安排上，本书主要包含面向对象思想、UML通用知识点、Rational Rose的安装和操作、使用Rose设计UML图（用例图、类图、对象图、序列图、协作图、状态图、活动图、包图、构件图、部署图）以及统一软件过程RUP等。

全书理论与实践并重，遵循从简单到复杂、从浅显到深入的思路，将理论分析与UML系统建模的实际应用相结合，重在应用，让读者快速掌握UML系统建模的方法和技巧。本书对于相关知识点都给出了相应的应用案例，使读者能够了解现实项目中UML的具体应用。同时，每一章的后面都提供了针对性的习题，使读者通过强化实训练习，达到巩固并加深理解所学知识的目的。

全书分为9章，由薛均晓、李占波主编，李占波教授对本书的定位和总体规划提出了指导性建议。编写分工如下：薛均晓编写了第1章、第2章和第9章，李庆宾（郑州航空工业管理学院）编写了第3章和第4章，李俊锋编写了第5章，韩英编写了第6章，张朝阳编写了第7章和第8章，最后由薛均晓负责全书的通稿工作。

本书在编写过程中得到了出版社和编者所在学校的帮助和大力支持，在此表示最诚挚的感谢。由于时间仓促且水平有限，书中难免有疏漏和欠妥之处，恳请专家和广大读者批评指正。

编者

2014年7月

教学建议

教学章节	教学要求	课　时
第 1 章 面向对象系统分析与设计引论	• 掌握面向对象领域中的基本概念、三大要素 • 掌握面向对象系统分析与设计的方法	2（理论）
第 2 章 统一建模语言 UML 与建模工具 Rational Rose	• 掌握 UML 的含义和特点 • 掌握 UML 的模型元素 • 掌握 UML 各种图形的基本概念及其作用	2（理论）
	• 掌握 Rational Rose 的安装方法 • 掌握 Rational Rose 的主界面组成及其基本操作 • 掌握 Rational Rose 的四种视图模型及其生成代码的方法	2（理论）+2（实验）
第 3 章 需求分析与用例模型	• 理解需求分析的含义 • 掌握用例图的基本概念 • 掌握用例图的建模方法 • 理解并掌握分析参与者的要点和寻找用例的方法	2（理论）+2（实验）
第 4 章 系统静态分析与静态模型	• 理解静态分析的含义与必要性 • 掌握类图的基本概念，理解抽象类和接口 • 掌握类图的建模方法	2（理论）+2（实验）
	• 掌握对象图的基本概念 • 掌握对象图与类图的区别和联系	2（理论）
第 5 章 系统动态分析与交互模型	• 理解动态分析的含义与必要性 • 理解动态建模的任务 • 理解交互的含义 • 掌握序列图的基本概念和建模方法	2（理论）+2（实验）
	• 掌握协作图的基本概念和建模方法 • 掌握序列图与协作图的区别和联系以及二者之间的转换方法	2（理论）+2（实验）
第 6 章 系统动态分析与行为模型	• 理解行为图的含义 • 掌握状态图的基本概念和建模方法	2（理论）+2（实验）
	• 掌握活动图的基本概念和建模方法 • 理解并掌握活动图与流程图的区别和联系 • 理解四种动态分析模型各自的特点以及使用场合	2（理论）+2（实验）
第 7 章 系统设计与实现模型	• 理解系统体系结构的概念 • 掌握包图的基本概念和建模方法	4（理论）+2（实验）
	• 掌握构件图的概念 • 掌握构件图的构造和建模方法	
	• 掌握部署图的概念 • 掌握部署图的构造和建模方法	

（续）

教学章节	教学要求	课　时
第 8 章 软件工程引论与统一软件过程 RUP	• 理解软件工程的含义 • 掌握统一过程的概念、结构、配置和 Rational 统一过程的方法	4（理论）
第 9 章 综合实例——银行核心业务系统	• 理解银行核心业务需求及业务流程 • 掌握软件产品文档的格式和标准	4（理论）
总课时	第 1 ~ 9 章建议课时	32(理论)+16(实验)
	综合实训建议课时	16（实验）

说明：

1）建议课堂教学在多媒体机房内完成，实现“讲－练”结合。

2）建议理论讲解和学生上机实践同步进行，使学生边学边练，在练习中理解课堂所讲授的面向对象分析方法。

3）建议安排一次学期综合实训，让同学们选择一个感兴趣的项目进行分析和设计，并使用 UML 进行建模，完成产品的文档书写。

目　录

第 1 章　面向对象分析与设计引论

自 20 世纪 40 年代问世以来，计算机在人类社会的各个领域得到了广泛的应用。20 世纪 60 年代以前，计算机刚刚投入实际使用，软件设计往往只是为了一个特定的应用而在指定的计算机上设计和编制，软件的规模比较小，文档资料通常也不存在，很少使用系统化的开发方法，设计软件往往等同于编制程序，基本上是个人设计、个人使用、个人操作、自给自足的私人化的软件生产方式。20 世纪 60 年代中期，随着计算机性能的提高，计算机的应用范围迅速扩大，软件开发的需求也急剧增长。软件系统的规模越来越大，复杂程度越来越高，软件可靠性问题也越来越突出。原来的个人设计、个人使用的方式不再满足企业和市场要求，改变软件生产方式，提高软件生产率变得尤为迫切。

为了解决长期以来计算机软件开发的低效率问题，计算机业界提出了软件工程的思想和方法。面向对象技术是一种系统开发方法，是软件工程学的一个重要分支。在面向对象编程中，数据被封装（或绑定）到使用它们的函数中，形成一个整体称为对象，对象之间通过消息相互联系。面向对象建模与设计是使用现实世界的概念模型来思考问题的一种方法。对于理解问题、与应用领域专家交流、企业级应用建模、编写文档、设计程序和数据库来说，面向对象模型都非常有用。

1.1　软件系统概述

1.1.1　软件的概念和特点

计算机是一个复杂的设备，它通常包含的要素有：计算机硬件、计算机软件、人员、数据库、文档和过程。其中，硬件是提供计算能力的电子设备；软件是程序、数据和相关文档的集合，用于实现所需要的逻辑方法、过程或控制；人员是硬件和软件的用户和操作者；数据库是通过软件访问的大型的、有组织的信息集合；文档是描述系统使用方法的手册、表格、图形及其他描述性信息；过程是一系列步骤，它们定义了每个系统元素的特定使用方法或系统驻留的过程性语境。

计算机软件是指由系统软件、支撑软件和应用软件组成的软件系统。系统软件用于管理计算机的资源和控制程序的运行，主要功能是调度、监控和维护计算机系统，管理计算机系统中各种独立的硬件，使得它们可以协调工作。系统软件使得计算机使用者和其他软件将计算机当作一个整体而不需要顾及底层每个硬件是如何工作的（如 Windows、Linux、DOS、UNIX 等操作系统都属于系统软件）。支撑软件包括语言处理系统、编译程序，以及数据库管理系统等。应用软件是用户可以使用的，由各种程序设计语言编制的应用程序的集合。应用软件是为满足用户不同领域、不同问题的应用需求而提供的，它可以拓宽计算机系统的应用领域，为人们的日常生活、娱乐、工作和学习提供各种帮助（如 Word、Excel、QQ 等都属于应用软件）。

计算机软件是用户与硬件之间的接口界面。用户主要是通过软件与计算机进行交流。如

果把计算机比喻为一个人，那么硬件就表示人的身躯，而软件则表示人的思想、灵魂。一台没有安装任何软件的计算机我们把它称为“裸机”。

软件是计算机系统设计的重要依据。为了方便用户，为了使计算机系统具有较高的总体效用，在设计系统时，必须通盘考虑软件与硬件的结合，以及用户的要求和软件的要求。软件的正确含义应该是：①运行时，能够提供所要求功能和性能的指令或计算机程序集合；②程序能够满意地处理信息的数据结构；③描述程序功能需求以及程序如何操作和使用所要求的文档。

软件具有与硬件不同的特点：①表现形式不同。硬件有形，看得见，摸得着。而软件无形，看不见，摸不着。软件大多存在人们的头脑里或纸面上，它的正确与否，是好是坏，一直要到程序在机器上运行才能知道。这就给设计、生产和管理带来许多困难。②生产方式不同。硬件是设备制造，而软件是设计开发的，是人的智力的高度发挥，不是传统意义上的硬生产制造。虽然软件开发和硬件制造存在某些相似点，但二者有根本的不同：两者均可通过优秀的设计获得高品质产品，然而硬件在制造阶段可能会引入质量问题，这在软件设计开发中并不存在（或者易于纠正）；二者都依赖人，但是人员和工作成果之间的对应关系是完全不同的；它们都需要构建产品，但是构建方法不同。软件产品成本主要在于开发设计，因此不能像管理制造项目那样管理软件开发项目。③要求不同。硬件产品允许有误差，而软件产品却不允许有误差。④维护不同。硬件是要用旧用坏的，在理论上，软件是不会用旧用坏的，但在实际上，软件也会变旧变坏。因为在软件的整个生存期中，一直处于改变维护状态。

1.1.2 软件的本质

现在的软件技术具有产品和产品交付载体的双重作用。作为一个产品，它显示了由计算机硬件体现的计算能力，更广泛地说，显示的是由一个可被本地硬件设备访问的计算机网络体现的计算潜力。无论是驻留在移动电话还是在大型计算机中，软件都扮演着信息转换的角色：产生、管理、获取、修改、显示或者传输各种不同的信息，简单如几个比特的传递或复杂如从多个独立的数据源获取的多媒体演示。而作为产品交付载体，软件提供了计算机控制（操作系统）、信息通信（网络）以及应用程序开发和控制（软件工具和环境）的基础平台。

软件提供了我们这个时代最重要的产品——信息。它会转换个人数据（例如个人财务交易），使信息在一定范围内发挥更大的作用；它通过管理商业信息提升竞争力；它为世界范围的信息网络提供通路（比如因特网），并对各类格式的信息提供不同的查询方式。

在最近半个世纪里，计算机软件的作用发生了很大的变化。硬件性能的极大提高、计算机结构的巨大变化、内存和存储容量的大幅度增加、还有种类繁多的输入和输出方法都促使计算机系统的结构变得更加复杂，功能更加强大。如果系统开发成功，复杂的结构和功能可以产生惊人的效果，但是同时复杂性也给系统开发人员带来巨大的挑战。

现在，一个庞大的软件产业已经成为工业经济中的主导因素。早期的独立程序员也已经被专业的软件开发团队所代替，团队中的不同专业技术人员可分别关注复杂应用系统中的某一个技术部分。然而同过去独立程序员一样，在开发现代计算机系统时，软件开发人员依然面临同样的问题：

1）为什么软件需要如此长的开发时间？

2）为什么开发成本居高不下？

3）为什么在将软件交付顾客使用之前，我们无法找到所有的错误？

4）为什么维护已有的程序要花费高昂的时间和人力代价？

5）为什么软件开发和维护的过程仍旧难以度量？

种种问题显示了业界对软件以及软件开发方式的关注，这种关注促使了业界对软件工程实践方法的采纳。

1.1.3 软件工程

很多人都熟悉计算机编程的思想。他们了解，要使计算机执行某一复杂的任务，而这一任务运行着一项业务，就不得不编写一系列指令来明确规范计算机应该完成的事情。这些指令由诸如 C++、Java 或 C# 之类的编程语言编写，形成了我们所熟知的计算机软件。

然而，很少有人知道程序员在开始编写代码之前事先需要完成的工作。程序员不能简单地制定业务操作的规则，或是猜测需要输入系统中的数据类型，这些数据类型会被存储并且稍后会被访问，以屏幕显示或报告的形式提供给用户。

要构建能够适应 21 世纪挑战的软件产品，就必须认识到以下几个简单的事实：

软件已经深入到我们生活的各个方面，其后果是，对软件应用所提供的特性和功能感兴趣的人们显著增多。当要开发一个新的应用领域或嵌入式系统时，一定会听到很多不同的声音。很多时候，每个人对发布的软件应该具备什么样的软件特性和功能似乎都有着不同的想法。因此，在制定软件解决方案前，必须尽力理解问题。

年复一年，个人、企业和政府的信息技术需求日臻复杂。过去一个人可以构建的计算机程序，现在需要由一个庞大的团队来共同实现。曾经运行在一个可预测、自包含、特定计算环境下的复杂软件，现在可以嵌入到消费类电子产品、医疗设备、武器系统等各种环境上执行。这些基于计算机的系统或产品的复杂性，要求对所有系统元素之间的交互非常谨慎。因此，设计成为软件开发的关键活动。

个人、企业、政府在进行日常运作管理以及战略战术决策时越来越依靠软件。软件失效会给个人和企业带来诸多不便，甚至是灾难性的危害。因此，软件必须保证高质量。随着特定应用感知价值的提升，其用户群和软件寿命也会增加。随着用户群和使用时间的增加，其适应性和可扩展性需求也会同时增加。因此，软件需具备可维护性。

由这些简单事实可以得出一个结论：各种形式、各个应用领域的软件都需要工程化。软件工程是研究如何以系统性的、规范化的、可定量的过程化方法开发和维护软件，以及如何把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来的学科。

软件工程的基础是过程（Process）。软件过程将各个技术层次结合在一起，使得合理、及时地开发计算机软件成为可能。过程定义了一个框架，构建该框架是有效实施软件工程技术必不可少的。软件过程构成了软件项目管理控制的基础，建立了工作环境以便于应用技术方法、提交工作产品（模型、文档、数据、报告、表格等）、建立里程碑、保证质量及正确管理变更。

软件过程是构建工作产品时所执行的一系列活动、动作和任务的集合。活动（Activity）主要实现宽泛的目标（如与利益相关者进行沟通），与应用领域、项目大小、结果复杂性或者实施软件工程的重要程度没有直接关系。动作（Action）（如体系结构设计）包含了主要工作产品（如体系结构设计模型）生产过程中的一系列任务。任务（Task）关注小而明确的目标，能够产生实际产品（如构建一个单元测试）。

在软件工程领域，过程不是对如何构建计算机软件的严格的规定，而是一种可适应性调

整的方法，以便于工作人员（软件团队）可以挑选适合的工作动作和任务集合。其目标通常是及时、高质量地交付软件，以满足软件项目资助方和最终用户的需求。

过程框架（Process Framework）定义了若干个框架活动（Framework Activity），为实现完整的软件工程过程建立了基础。这些活动可广泛应用于所有软件开发项目，无论项目的规模和复杂性如何。此外，过程框架还包含一些适用于整个软件过程的普适性活动（Umbrella Activity）。一个通用的软件工程过程框架通常包含以下 5 个活动。

1）沟通：在技术工作开始之前，和客户（及其他利益相关者）的沟通与协作是极其重要的；其目的是理解利益相关者的项目目标，并收集需求以定义软件特性和功能。

2）策划：如果有地图，任何复杂的旅程都可以变得简单。软件项目好比是一个复杂的旅程，策划活动就是创建一个“地图”，以指导团队的项目旅程，这个地图称为软件项目计划，它定义和描述了软件工程工作，包括需要执行的技术任务、可能的风险、资源需求、工作产品和工作进度计划。

3）建模：无论你是庭园设计家、桥梁建造者、航空工程师、木匠还是建筑师，你每天的工作都离不开模型。你会画一张草图来辅助理解整个项目大的构想——体系结构、不同的构件如何结合，以及其他特征。如果需要，可以把草图不断细化，以便更好地理解问题并找到解决方案。软件工程师也是如此，利用模型来更好地理解软件需求，并完成符合这些需求的软件设计。

4）构建：它包括编码（手写的或者自动生成的）和测试以发现编码中的错误。

5）部署：软件（全部或者部分增量）交付到用户，用户对其进行测评并给出反馈意见。

上述五个通用框架活动既适用于简单小程序的开发，也可用于大型网络应用程序的建造以及基于计算机的大型复杂系统工程。不同的应用案例中，软件过程的细节可能差别很大，但是框架活动都是一致的。

对许多项目来说，随着项目的开展，框架活动可以迭代应用。也就是说，在项目的多次迭代过程中，沟通、策划、建模、构建、部署等活动不断重复。每次项目迭代都会产生一个软件增量（Software Increment），每个软件增量实现部分的软件特性和功能。随着每一次增量的产生，软件逐渐完善。

1.2 面向对象的含义

1.2.1 什么是面向对象

在现实世界中，一个复杂的事物往往是由许多部分组成的。例如，一辆汽车是由发动机、底盘、车身和车轮等部件组成的。当人们生产汽车时，分别设计和制造发动机、底盘、车身和车轮等，最后把它们组装在一起。组装时，各部分之间有一定联系，以便协同工作。

面向对象系统开发的思路和人们在现实世界中处理问题的思路是相似的，即基于现实世界来设计与开发软件系统的方式。面向对象技术以对象为基础，使用对象抽象现实世界中的事物，以消息来驱动对象执行处理。和面向过程的系统开发不同，面对对象技术不需要一开始就使用一个主函数来概括整个系统的功能，而是从问题域的各个事物入手，逐步构建出整个系统。

在程序结构上，人们常常用下面的公式来表述面向过程的结构化程序：

面向过程程序 = 算法 + 数据结构

算法决定了程序的流程以及函数间的调用关系，也就是函数之间的相互依赖关系。算法和数据结构二者相互独立，分开设计。在实际问题中，有时数据是全局的，很容易超出权限范围修改数据，这意味着对数据的访问是不能控制的，也是不能预测的，如多个函数访问相同的全局数据，因为不能控制访问数据的权限，程序的测试和调试就变得非常困难。另外，面向过程的程序中主函数依赖于子函数，子函数又依赖于更小的子函数。这种自顶向下的模式，使得程序的核心逻辑依赖于外延的细节，一个小小的改动，有可能带来一系列连锁反应，引发依赖关系的一系列变动，这也是过程化程序设计不能很好处理需求变化、代码重用性差的原因。在实践中，人们慢慢意识到算法和数据是密不可分的，通过使用对象将数据和函数封装（或绑定）在一起，程序中的操作通过对象之间的消息传递机制实现，就可以解决上述问题。因此，就形成了面向对象的程序设计：

面向对象程序 =（对象 + 对象 + … ）+ 消息

面向对象程序设计的任务包括两个方面：一是决定把哪些数据和函数封装在一起，形成对象；二是考虑怎样向对象传递消息，以完成所需任务。各个对象的操作完成了，系统的整体任务也就完成了。

1.2.2 对象

对象（Object）是面向对象的基本构造单元，是一些变量和方法的集合，用于模拟现实世界中的一些事物模型，如一台计算机、一个人、一本书等。当然也可以模拟一些虚拟的东西，比如一个学号、一个编号、一个院系等。

事实上，对象是对问题域中某些事物的抽象。显然，任何事物都具有两方面特征：一是该事物的静态特征，如某个人的姓名、年龄、联系方式等，这种静态特征通常称为属性；二是该事物的动态特征，如兴趣爱好、学习、上课、体育锻炼等，这种动态特征称为操作。因此，面向对象技术中任何一个对象都应当具有两个基本要素，即属性和操作。一个对象往往是由一组属性和一组操作构成的。

1.2.3 类

为了表示一组事物的本质，人们往往采用抽象方法将众多事物归纳、划分成一些类。例如，我们常用的名词“人”，就是一种抽象表示。因为现实世界只有具体的人，如王安、李晓、张明等。把所有国籍为中国的人归纳为一个整体，称为“中国人”，也是一种抽象。抽象的过程是将有关事物的共性进行归纳、集中的过程。依据抽象的原则进行分类，即忽略事物的非本质特征，只注意那些与当前目标有关的本质特征，从而找出事物的共性，把具有共同性质的事物划分为一类，所得出的抽象概念称为类。

在面向对象的方法中，类的定义如下：

类是具有相同属性和操作的一组对象的集合，它为属于该类的全部对象提供了统一的抽象描述。

事实上，类与对象的关系如同模具和铸件之间的关系。类是对象的抽象定义，或者说是对象的模板；对象是类的实例，或者说是类的具体表现形式。

1.2.4 消息

在面向对象系统中，要实现对象之间的通信和任务传递，采用的方法是消息传递。由于

在面向对象系统中，各个对象各司其职、相互独立，要使得对象不是孤立存在的，就需要通过消息传递来使其之间发生相互作用。通过对象之间互发消息、响应消息、协同工作，进而实现系统的各种服务功能。

消息通常由消息的发送对象、消息的接收对象、消息传递方式、消息内容（参数）、消息的返回五部分内容组成。消息只告诉接收对象需要完成什么操作，而不告诉接收对象如何完成操作。消息接收者接收它能识别的消息，并独立决定采用什么方法完成所需的操作。

1.2.5 封装

封装是指把一个对象的部分属性和功能对外界屏蔽，也就是说从外界是看不到，甚至是不可知的。例如，计算机里面有各种电子元件和电路板，但这些在外面是看不到的，在计算机的外面仅保留用户需要用到的各种按键，即计算机的外部接口。人们使用计算机的时候也不必了解计算机内部的结构和工作原理，只需要知道按键的作用，通过各个外部的按键让计算机执行相应的操作即可。

封装是一种防止相互干扰的方式。所谓封装，包含两层含义：一是将有关的数据和操作封装在一个对象中，形成一个整体，各个对象之间相互对立，互不侵扰。二是将对象中某些属性和操作设置为私有，对外界隐蔽，同时保留少量接口，以便与外界联系，接收外界的消息。这样做既有利于数据安全，防止无关人员修改数据，又可以大大降低人们操作对象的复杂度。使用对象的人完全可以不必知道对象的内部细节，只需了解其外部功能即可自如操作对象。

1.2.6 继承

继承是指子类可以自动拥有其父类的全部属性和操作。继承可以指定类从父类中获取特性，同时添加自己的独特特性。例如，已经建立了一个“员工”类，又想另外建立一个“财务人员”类和“销售人员”类，而后两个类与“员工”类的内容基本相同，只是在“员工”类的基础上增加一些属性和操作，显然不必重新设计一个新类，只需在“员工”类的基础上添加部分新内容。继承简化了对现实世界的描述工作，大大提高了软件的复用性。

1.2.7 多态

同一条消息被不同的对象接收到时可能产生完全不同的行为，这就是多态性。多态性支持“同一接口，多种方法”的面向对象原则，使高层代码只写一次而在低层可以多次复用。

实际上，在现实生活中可以看到许多多态性的例子。如学校发布一条消息：8月25日新学期开学。不同的对象接收到该条消息后会做出不同的反应：学生要准备好开学上课的必需物品；教师要备好课；教室管理人员要打扫干净教室，准备好教学设施和仪器；宿舍管理人员要整理好宿舍等。显然，这就是多态性。可以设想，如果没有多态性，那么学校就要分别给学生、教师、教室管理人员和宿舍管理人员等许多不同对象分别发开学通知，分别告知需要做的具体工作，这是一件非常复杂的事情。有了多态性，学校在发消息时，不必一一考虑各种类型人员的特点，而不断发送各种消息，只需要发送一条消息，各种类型人员就可以根据学校事先安排的工作机制有条不紊地工作。

从编程角度来看，多态提升了代码的可扩展性。编程人员利用多态性，可以在少量修改

甚至不修改原有代码的基础上，轻松加入新的功能，使代码更加健壮，易于维护。

1.3 面向对象的有效性

1.3.1 面向过程方法的困难

面向过程方法认为客观世界是由一个个相互关联的小系统组成的，各个小系统依据严密的逻辑组成的，环环相扣，井然有序。面向过程方法还认为每个小系统都有着明确的开始和明确的结束，开始和结束之间有着严谨的因果关系。只要将这个小系统中的每一个步骤和影响这个小系统走向的所有因素都分析出来，就能完全定义这个系统的行为。所以如果要分析问题，并用计算机来解决问题，首要的工作是将过程描绘出来，把因果关系都定义出来；再通过结构化的设计方法，将过程进行细化，形成可以控制的、范围较小的部分。通常，面向过程的分析方法是找到过程的起点，然后顺藤摸瓜，分析每一个部分，直至达到过程的终点。这个过程中的每一部分都是过程链上不可分割的一环。事实上，面向过程方法是一种“自顶向下，逐步细分”的解决问题的方法。

在面向过程方法中，计算机解决问题的过程中每一步都会产生、修改或读取一部分数据。每一个环节完成后，数据将顺着过程链传递到下一部分。当需要的最终结果在数据中被反映出来，即达到预期状态的时候，过程就结束了。显然，数据对于问题的解决至关重要。为了更好地管理数据，不至于让系统运行紊乱，人们通过定义主键、外键等手段将数据之间的关系描绘出来，结构化地组织它们。然而随着需求越来越复杂，系统越来越庞大，功能点越来越多，一份数据经常被多个过程共享，这些过程对同一份数据的创建和读取要求越来越趋于复杂和多样，经常出现互相矛盾的数据需求，因此分析和设计也变得越来越困难。同时，这种步步分析的过程分析方法要求过程中的每一步都是预设好的，有着严谨的因果关系。然而，客观世界从来都不是一成不变的，尤其到了信息化时代，外部世界无时无刻不在发生着变化，系统所依赖的因果关系变得越来越脆弱。显然，客观世界的复杂性和频繁变革已经不是面向过程可以轻易应付的了。

1.3.2 面向对象方法的有效性

不同于面向过程方法，面向对象程序设计是一种自下而上的程序设计方法，往往从问题的一部分着手，一点一点地构建出整个程序。面向对象设计以数据为中心，类作为表现数据的工具，成为划分程序的基本单位。面向对象是把构成问题的事物分解成各个对象，建立对象的目的不是为了完成一个步骤，而是为了描述某个事物在整个解决问题的步骤中的行为。

使用计算机解决问题首先要对现实世界进行抽象描述，现实世界中的情况非常复杂，是由独立的事物组成，每一个事物按照最简单的独立方式对应事件，但是在众多大型事物即刻交互的时候，很难进行预测。这种任务使用面向过程方法难以编程。因为，使用面向过程方法基于如下假设：程序结构控制执行流程。因此对于使用过程程序处理上述任务，必须有独立的程序来测试或响应为数众多的变化条件。该问题的解决方案是按照问题自身的相似方式来构建程序，作为独立的软件事物集合，每一个元素都是待抽象的现实世界系统的一个对象。这样就解决了应用域模型和软件域模型之间的冲突，而将劣势转换为优势。

另一方面，图形用户界面（GUI）在 20 世纪 80 年代和 90 年代的快速普及对现代开发方法带来了特殊的困难。GUI 引入了之前将仿真编程引入主流商业应用中遇到的问题。原因在