本章的关键词

术语洪流、比喻滥用、“一切都是对象”综合征

第 1 章 面向对象：让软件开发变轻松的技术

热身问答

在阅读正文之前，请挑战一下下面的问题来热热身吧。

问题

下列哪一项是最早提出“面向对象”概念的艾伦·凯（Alan Kay）的表述？

A. 程序模块、图标和数据库等万物都可以表示为对象

B. 正如万物都在变化，编程技术也在变化

C. IT 领域的创新技术基本上都出现于 1960 年之前

D. 预测未来最好的方法就是创造它

D. 预测未来最好的方法就是创造它

解 析

面向对象的起源可以追溯到挪威的两名技术人员在 1967 年开发的 Simula 67 编程语言。之后，任职于美国施乐公司的艾伦·凯率领的团队开发了 Smalltalk，沿用了 Simula 67 语言的结构，确立了面向对象的概念。除此之外，凯在 IT 领域还做出了很多贡献，比如开发出图形用户界面（Graphical User Interface，GUI）、提出作为现代笔记本电脑原型的“DynaBook 设想”等。

“预测未来最好的方法就是创造它。”（The best way to predict the future is to invent it.）据说这句名言是公司高层追问研究内容的未来走向时，凯给出的回答。想必也只有提出了诸多创新性的技术概念的凯，才能讲出这样的名言吧。

本章重点

本章将介绍面向对象的基本思想，以及面向对象所面向的技术领域的全貌。

面向对象最初是作为一种编程语言提出的，后来人们将其不断扩展，并应用到各个领域，如今将其称为“软件开发的综合技术”可能更为合适。

遗憾的是，尽管这是一门非常优秀的技术，但很多从事软件开发的人好像都认为它非常难。本章我们将分析造成这种局面的 3 个主要因素，而消除这些因素正是正确理解面向对象的关键。

1.1 面向对象是软件开发的综合技术

我们先从一个简单的问题开始介绍。

“为什么要基于面向对象来开发软件?”

不管谁问这样的问题，笔者都会这样回答：

“为了轻松地开发软件。”

可能有的人听到“轻松”二字会感觉很意外。这是因为当提到面向对象时，不少人仍感觉“很难，难以对付”。

面向对象包含了各种技术，几乎涵盖了从 Java、Ruby 等编程语言到需求规格书和设计内容的图形表示、可重用的软件构件群、优秀设计的技术窍门、业务分析和需求定义的有效推进方法、顺利推进系统开发的开发方法等软件开发的所有领域。

不过，这些技术单独来看是完全不同的。如果要找出它们的共同点，大概就是它们都是软件开发相关的技术，都是用来顺利推进软件开发的。

因此，如果用一句话来概括面向对象，那就是“能够轻松地进行较难的软件开发的综合技术”。

1.2　以对象为中心编写软件的开发方法

面向对象的英文是 Object Oriented，直译为“以对象为中心”。

在面向对象普及之前，主流的开发方法是“面向功能”的，具体地说，就是把握目标系统整体的功能，将其按阶段进行细化，分解为更小的部分。如果采用面向功能的开发方法来编写软件，当规格发生改变或者增加功能时，修改范围就会变得很广，软件也很难重用。

面向对象技术的目的是使软件的维护和重用变得更容易，其基本思想是重点关注各个构件，提高构件的独立性，将构件组合起来，实现系统整体的功能。通过提高构件的独立性，当发生修改时，能够使影响范围最小，在其他系统中也可以重用。

1.3　从编程语言演化为综合技术

面向对象最初是以编程语言的身份出现的，在之后的 40 多年里，经过不断发展，逐渐被应用到了开发的各个领域。这里我们来简单回顾一下面向对象的全貌和发展过程，如图 1-1 所示。

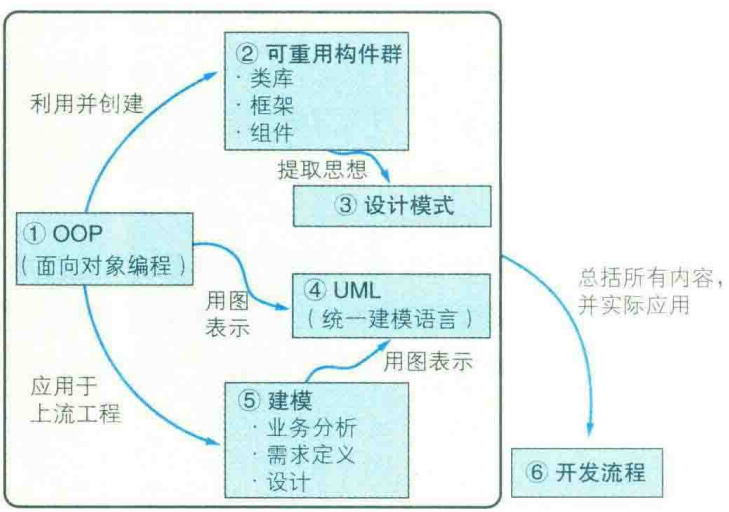

图 1-1　面向对象的全貌和发展过程

面向对象起源于 1967 年在挪威设计的 Simula 67 编程语言。该语言拥有类、多态和继承等以往的编程语言中没有的优良结构，被称为最早的**面向对象编程语言**（Object Oriented Programming language，OOP[1]）。后来，艾伦·凯率领的团队开发的 Smalltalk 沿用了该结构，确立了“面向对象”的概念。此后，具有相同结构的 C++、Objective-C、Java、C# 和 Ruby 等诸多编程语言相继被开发出来，并延续至今。

OOP 使得大规模软件的可重用构件群的创建成为可能，这些被称为**类库**或者**框架**。另外，创建可重用构件群时使用的固定的设计思想被提炼为**设计模式**。

另外，使用图形来表示利用 OOP 结构创建的软件结构的方法称为**统一建模语言**（Unified Modeling Language，UML）。在此基础上，还出现了将 OOP 思想应用于上流工程的**建模**，以及用于顺利推进系统开发的**开发流程**。

如今，面向对象已经成为一门覆盖软件开发整体的综合技术。虽然这些技术所涉及的领域和内容并不相同，但目的都是顺利推进软件开发。因此，通过以各种形式灵活运用前人的研究和技术窍门，有助于我们提高软件的质量和开发效率。

1.4 在混乱的状态下去理解，就会觉得很难

尽管面向对象是众多优秀技术的集大成，却经常被认为很难理解，难以对付。也有人认为不擅长抽象思考的人在学习面向对象时会感觉很难，要经过很多年才能掌握，等等。不管是多么方便的工具，如果很难理解其内涵，无法熟练使用，那就没有办法了。

不过，笔者认为，相较于技术本身的复杂性，面向对象让人感觉很难

① 严格来说，正确的表达方式是，将面向对象编程语言（Object Oriented Programming Language）称为 OOPL，使用 OOPL 进行编程的操作称为面向对象编程（OOP）。不过，由于本书的很多讲解中并未严格区分面向对象编程语言和面向对象编程，所以都表述为 OOP。

的更主要的原因在于混乱的现状。造成混乱的主要原因大致有三点：术语洪流、比喻滥用和“一切都是对象”综合征。下面我们就来分别介绍一下。

1.5　混乱之一：术语洪流

第一点是**术语洪流**。

想必很多人在最开始接触面向对象时都是像图 1-2 那样，被灌输大量陌生的术语吧。下面列举了一些具有代表性的术语，但其实人们被灌输的术语远不止这些。

> 继承、泛化、特化、超类、子类、接口、多重继承、属性、关联、集合、委托、重写、重载、访问控制、构造函数、包、异常、垃圾回收机制、框架、类库、组件、设计模式、用例、建模、UML、重构、敏捷开发流程、RUP、XP……

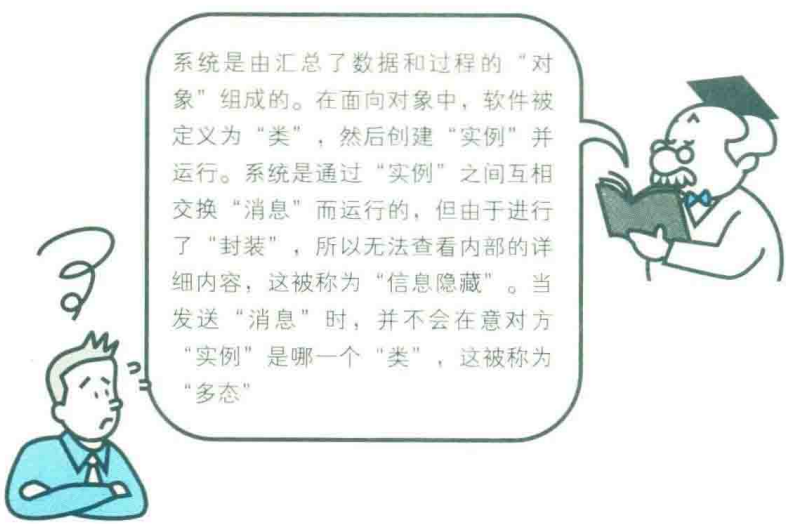

图 1-2　对大量术语感到混乱的开发者

简直就是术语大集合！在深入了解技术内容之前就被这么多术语吓退的人应该不在少数吧。

存在如此大量的术语的原因如下所示。

首先是面向对象所涉及的范围很广。如今面向对象几乎涵盖了软件开发的所有领域，在这些领域中，之前没有的新结构和新思想层出不穷。这些术语中也有不少英文词汇，可以说这是以欧美为中心迅速发展起来的技术的宿命。此外，还有一些词语被作为业界的宣传用语而被过度解释。像这样，术语多少本身源自该技术的广度，从一定意义上来说也是没有办法的事情。

而更本质的问题是基本术语的定义混乱，我们将在随后的第三点中对此进行讨论。

1.6 混乱之二：比喻滥用

第二点是**比喻滥用**。

这与其说是技术本身的问题，倒不如说是说明方法的问题。一般来说，比喻并不一定就是不好的。使用恰当的比喻能够直截了当、形象地说明内容；反之，如果比喻不恰当或者使用过度，就有可能造成混乱。尤其是在仅通过比喻来说明编程语言等的具体结构的情况下，由于每个人的理解各不相同，所以特别容易造成误解。

我们来举例说明一下。面向对象的基本结构有时会像下面这样说明（图 1-3）。

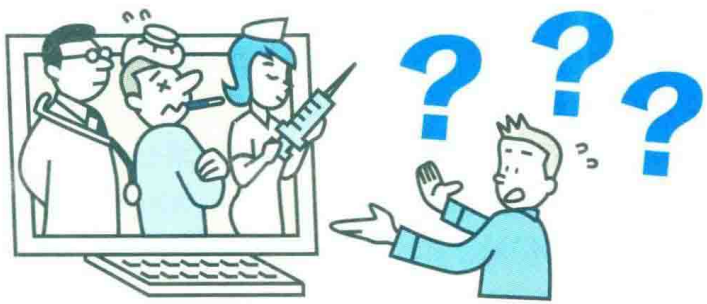

图 1-3 对使用比喻进行的说明感到混乱的开发者

“动物是超类，哺乳类和鱼类是子类。既产卵又用乳汁哺乳幼仔的鸭嘴兽也就相当于爬虫类和哺乳类的多重继承。”

“人具有‘出生年月日’的属性。如果给小王这样具体的一个人发出‘请告诉我你的年龄’的消息，就会得到‘28 岁’的回答。”

“正如医院里的医生、护士和药剂师互相联系、协同工作一样，对象也是通过在计算机中互相发送消息来进行工作的。”

从直观理解 OOP 的结构来说，这样的讲解也不一定就是不好的做法。不过，如果只强调比喻而不详细介绍编程语言的结构和目的，那么就只有比喻能给人留下深刻的印象，而且听众很有可能会根据自己的理解，对实际的结构产生错误的认识。特别是在不考虑系统化范围，即计算机的作业范围的情况下乱用比喻，再加上随后介绍的“一切都是对象”综合征的影响，读者就有可能会产生“当使用面向对象时，可以直接将现实世界表示为程序”的重大误解（关于该误解，我们将在第 2 章深入讨论）。

1.7　混乱之三：“一切都是对象”综合征

第三点是**“一切都是对象”综合征**。

面向对象的含义是“以对象为中心”。如果按照字面意思进行解释，那么现实世界的人、组织、事件、计算机系统的功能、系统管理的信息和程序的构成元素等一切事物都可以说是对象。这里，我们将这种极端的抽象称为“一切都是对象”综合征（图 1-4）。

这种极端看法大概源自“万物都是变化的”这一点。立足于这种观点是很难说明和理解编程语言的结构，以及现实世界中业务流程的整理方法等具体技术的。尽管如此，但就像“正如现实世界是由对象（物）组成的，在面向对象中，程序也是以对象为中心创建的”这句话所说的那样，我们经常会见到实际上不同的对象都被解释为“对象”这一个词语的情况。这样的解释的确会对直观理解面向对象的结构有所帮助，但反过来也有可能引起“只要使用面向对象，就可以直接将现实世界表示为程序”的误解。

图 1-4 “一切都是对象”综合征

不过，这种混乱并不只是说明方法的问题，也是面向对象本身的问题。这是因为面向对象的技术范围很广，比如对象和类等词语有时指编程语言的结构，但在其他情况下也指管理的信息、现实世界的人和物。

笔者认为，“一切都是对象”综合征才是造成面向对象混乱的最大原因（关于这一点，我们将在第 7 章中深入讨论）。

1.8 三种混乱增大了理解的难度

在很多情况下，以上为大家介绍的三点引发混乱的原因，即术语洪流、比喻滥用和“一切都是对象”综合征会同时出现。也就是说，在讲解面向对象时，讲解的人往往会一下子介绍很多陌生的术语，并以比喻为中心，只强调现实世界和程序结构的共同之处。然而，这样的讲解非但不能让人正确地理解该技术，反而会容易引起混乱和误解。

由于面向对象以范围很广、难度较大的软件开发为对象，所以在技术层面上原本就非常复杂，而以上几点混乱又进一步增大了理解该技术的难度，使其本质难以被看透。

1.9　因为不理解，所以才感觉神秘

不理解的事情有时会为我们带来意外的惊喜。

如果某项技术只是比较难，让人无法理解，那么放弃学习也没什么，而面向对象所涉及的编程语言、UML 以及大规模的可重用构件等技术都很基础，而且这些技术正在不断地渗透到软件开发的实际应用场景，所以现阶段很难完全避开面向对象。

“虽然将现实世界直接表示为程序这种解释我怎么都想不通，但是由于 Java 和 UML 使用起来比较方便，所以我正在使用。”“虽然其中的个别技术非常有用，但是整体上却很难理解。”……想必抱有这些想法的人不在少数吧。

面向对象有时会被比作魔法，虽然无法用道理解释清楚，但效果很好，或者被解释为一般人难以触及的理想的开发方法。这是将“不理解但有用的事物”神秘化，并推崇为“超越人类智慧的存在”的心理在起作用。

不过，面向对象的目的是驱动逻辑电路组成的计算机，是具有实践性的一门极为有用的技术，因此完全不存在无法解释的内容。

1.10　消除这三种混乱，就能看到面向对象的真面目

本书的目标是用清晰的逻辑向读者介绍面向对象所涉及的各项技术。

不过，本书并不会为大家详细讲解 Java 等编程语言的语法、UML 表示法等各项技术的详细内容，而是重点讲解这些技术是什么（What），以及为什么需要这些技术（Why）。

另外，为了使读者能够正确理解面向对象，笔者认为应该首先排除前文中介绍的引发混乱的三点原因，因此本书将采取如下方针。

- 将对术语和概念的介绍控制在最小限度
- 将最小限度地使用比喻进行讲解。如果使用比喻，则会明确其主旨
- 将编程的结构以及“以对象为中心”来把握事物的思想作为不同的内容分开介绍