深入学习的参考书籍

[1] 结城浩. 图解设计模式 [M]. 杨文轩，译. 北京：人民邮电出版社，2016.

☆☆☆

该书使用 Java 和 UML，以易于理解的方式介绍了 GoF 的 23 种设计模式。特别推荐给设计模式初学者阅读。

[2] Erich Gamma，Ralph Johnson，Richard Helm，John Vlissides. 设计模式：可复用面向对象软件的基础 [M]. 刘建中，译. 北京：机械工业出版社，2007.

☆☆

这是 GoF 设计模式的经典之作。该书编写于 Java 和 UML 出现之前，类图采用的是 UML 的前身 OMT，实例代码也是使用 C++ 和 Smalltalk 编写的。但是，如果下定决心要好好学习设计模式，该书可以说是必读的一本。

[3] Martin Fower. 企业应用架构模式 [M]. 王怀民，周斌，译. 北京：机械工业出版社，2010.

☆☆

该书将从画面查找、更新数据库中的信息的企业系统中的典型设计结构总结为模式，建议使用 JavaEE 和 .NET 开发应用程序系统的工程师务必掌握书中的内容。

[4] 木村聡. Java フレームワーク開発入門 [M]. 东京：SoftBank Creative，2010.

☆☆

该书使用丰富的 Java 代码，细致地介绍了理解框架结构所必需的基础知识，包括反射、设计模式、AOP 和 DI 等主题。这是一本为掌握了 Java 基本语法的技术人员编写的进阶书。

本章的关键词

集合论、职责分配

第7章 化为通用的归纳整理法的面向对象

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

“面向对象”这一概念据说是由 Smalltalk 的开发者艾伦·凯提出的。在 20 世纪 70 年代，有人在目睹凯的研究成果后受到极大冲击。这个人在计算机历史上很有名，请问他是谁?

A. 美国微软公司的创始人比尔·盖茨

B. 美国苹果公司的创始人史蒂夫·乔布斯

C. Java 的创始人詹姆斯·高斯林

D. Linux 的创始人林纳斯·托瓦兹

B. 美国苹果公司的创始人史蒂夫・乔布斯

解 析

如第 1 章所述，据说“面向对象”这一概念是由艾伦・凯提出的。在 20 世纪 70 年代，凯任职于美国施乐公司的帕洛阿尔托研究中心，他提出了理想的个人电脑“Dynabook”的设想，还主导开发了基于 GUI 的工作站和面向对象编程语言 Smalltalk。凯在 2003 年被授予图灵奖。

据说美国苹果公司创始人史蒂夫・乔布斯在看到运行 Smalltalk 的工作站原型后，受到了很大冲击。后来，基于鼠标和 GUI 的用户界面被引入苹果公司的 Lisa 和 Macintosh 中，之后还被引入到 Windows 计算机中，不断继承发展。

本章重点

至此，我们介绍了 OOP 及其扩展技术，本章将换个话题，介绍面向对象的另一面——“通用的归纳整理法”。

面向对象在编程领域取得了成效，为了让这种成效扩展到整个系统，它又被应用到了上流工程，最终成为“对事物进行分类和整理的基本结构”。这样一来，面向对象的应用范围得到了很大扩展，但另一方面，这也引起了“一切都是对象”综合征。

通过阅读本章，我们将知道面向对象的下流工程和上流工程应该是不同的。第 8 章之后会介绍面向对象的各种应用技术，但为了防止“只见树木不见森林”，请大家先记住面向对象具有两个方面，即下流工程的“编程技术”和上流工程的“通用的归纳整理法”。

7.1 软件不会直接表示现实世界

我们先来介绍一下第 2 章最后提到的现实世界和软件的沟壑。

冒昧地请大家思考一下下面这个简单的问题。

软件直接表示现实世界吗？

突然问这个问题，大家可能难以理解，我们换成具体的问题。

医院的系统直接表示现实的医院吗？
银行的系统直接表示现实的银行吗？

这样就容易思考了。现实世界中的医院有门诊楼和病房，配备有检查器材和住院设施等，还有医生、护士和办公人员等在里面工作，生病或受伤的患者会在里面接受检查、取药、支付费用。

另外，现实世界中的银行有总行、分行等大楼，有银行职员在里面工作。银行职员又细分为分行行长、存款主管和贷款主管等。顾客到银行办理存款、贷款和取款等业务。在银行，每天都要处理现金、票据或者支票

等有价证券及诸多文件，当然还会使用 ATM 等计算机（图 7-1）。

图 7–1　现实世界中的医院和银行

怎么样？大家觉得软件可以“直接”表示这样的现实世界吗？

换句话说，如果医院系统的计算机中有患者，那么计算机中的医生和护士能对其进行检查和照顾吗？

另外，顾客和银行职员能在银行系统的计算机中处理金钱业务或交换文件吗？

当然是不可能的。

医院系统是用来协助医院工作人员的，能记录患者的病历、计算治疗费用等。而实际看病的是现实世界中的人，检查和照顾患者的也是现实世界中的人。

同理，银行系统是用来协助银行工作人员的，能记录金钱交易记录、制作各种文件等。顾客可以从 ATM 中取出实际的钱。然而，收到钱的是现实世界中的人，进行存款、贷款等交易的也是现实世界中的人。

当然，计算机不会“完全替换”现实世界中的工作和娱乐，它只是为了让人们变轻松而承担了现实世界中的一部分工作。计算机擅长的是处理大规模的计算、记录信息等固定工作和记忆工作。股票的自动交易系统和汽车导航等乍一看像是在执行一些高级的判断，其实它们所有的操作都只是按照程序员的预先设计进行的。

因此，管理计算机的软件也只是承担了现实世界中的一部分工作，而不能表示现实世界本身（图 7-2）。

> 计算机只是承担了现实世界中的一部分工作。
>
> 因此，管理计算机的软件也不能直接表示现实世界，而只是表示一部分工作。

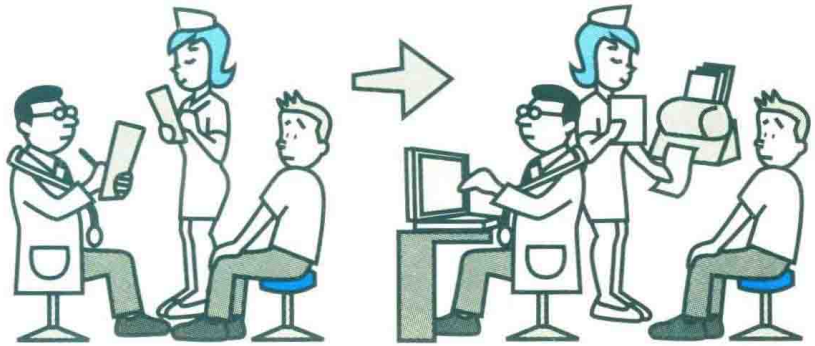


图 7-2　现实世界与将一部分工作交给计算机的现实世界

7.2　应用于集合论和职责分配

前面介绍过，计算机只是承担了现实世界中的一部分工作，因此，在开发软件时，我们并不是直接从编程开始的，在编程之前，还需要进行业务分析和需求定义等工作（关于业务分析和需求定义，我们将在第 9 章详细介绍）。在系统开发的早期阶段进行的这些工作通常称为**上流工程**。

面向对象是作为编程语言出现的，它在该领域发挥的提高生产率、质量和可重用性的效果受到了人们的认可。后来，该技术也被应用到上流工程，以提高系统开发整体的生产率和质量。

很多人认为，既然采用面向对象编程，那么上流工程中也需要采用与

面向对象相应的手法。不过，虽然类、多态和继承等 OOP 结构能够直接应用于程序设计，但并不一定适用于上流工程的业务分析和需求定义。正如第 2 章中讨论的那样，现实世界和 OOP 结构是似是而非的。

因此，当在上流工程中应用 OOP 结构时，需要灵活处理。

这里的“应用”是指仅使用可以使用的部分，换句话说，就是仅使用合适的部分。也可以说将“似是而非”中“非”的部分去掉，只使用“是”的部分。

通过只应用可以使用的部分，在上流工程中，面向对象会提供两种基本结构：一种是集合论；另一种是职责分配。下面我们将分别进行介绍。

首先是**集合论**。在 OOP 中，类用于汇总子程序和变量。实例是基于类定义在运行时分配的内存区域。这种在运行时由 1 个类创建很多个实例的结构与集合论中的集合和元素非常相似，因此，类和实例在上流工程中会被应用于集合论。

这样想来，类和实例的适用范围就变得非常广。“加藤和山田是实例，职工是类”“2 月 3 日下的书的订单是实例，订单是类”，等等，类和实例在这些情况中都可以应用。第 2 章介绍的不恰当的比喻示例“婴儿是类，我的儿子和女儿是实例”，在这里也完全没有问题。

另外，将类定义的共同部分汇总为其他类的继承结构也被应用在了全集和子集思想上。这样一来，继承就可以应用在“将顾客进一步分为个人顾客和法人顾客”“将人类分为男性和女性”等各种示例中（图 7-3）。

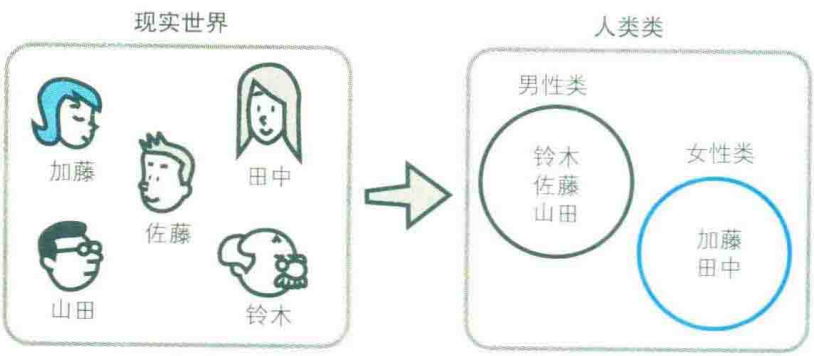


图 7-3　被应用于集合论的面向对象

然后是**职责分配**。在 OOP 中，消息传递（message passing）是一种通过指定实例来调用类中汇总的子程序（方法）的结构。这种结构在上流工程中被应用于表示“具有某种特定功能的事物按照固定的方法相互联系的情形”的职责分配模型中。

这样想来，消息传递的适用范围也变得非常广。在饭店里跟服务员点菜、向建筑公司 A 发活、让斑点狗抬爪子等都可以看作消息传递（图 7-4）。

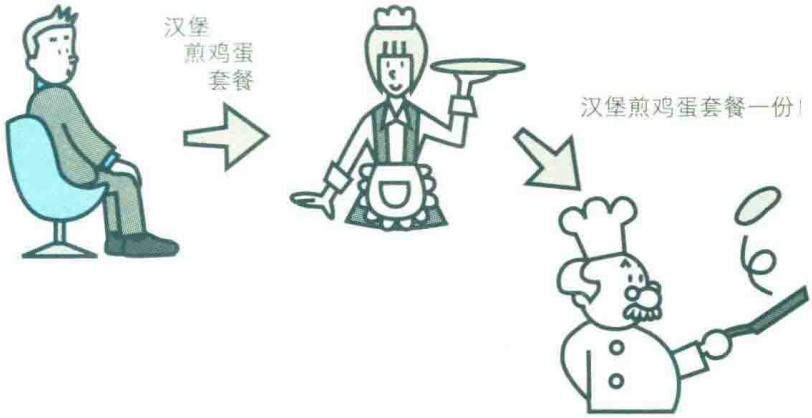


图 7–4　表示职责分配的消息传递

7.3　在上流工程中化为通用的归纳整理法

集合论和职责分配的概念非常强大。

集合论基本上可以适用于现实世界中所有的事物。应用集合论，世界上所有的物（object）都会成为采用类和实例进行整理的对象。名词可以看作类，专有名词可以看作实例……像这样扩展的话，面向对象就跨入了认识论和哲学领域。

另外，职责分配的思想也很强大，比如可以应用于社会组织、“×× 人员”“×× 部主任”等拥有某种职务的人、计算机等机器、软件的进程和线程等各种领域。

最终，面向对象将成为拥有对事物进行分类整理的结构和表示职责分配的结构的**通用的归纳整理法**。再加上作为编程技术的一面，面向对象的适用范围得到了大幅扩展。从业务分析到编程，什么都可以。这样一来，我们就可以将面向对象定位为支持当前整个软件开发过程的综合技术。

说点题外话，笔者认为，如果用一个词来形容面向对象成为归纳整理法这一现象，相比“发展”“扩展”等表示对象范围扩大的词语，“变化”“转变”等表示改变的词语更合适。这是因为编程技术和归纳整理法的含义是完全不同的。本章章名中使用“化为”一词，也就是因为此。

> 面向对象包含抽象的“归纳整理法”和具体的“编程技术”两方面。

7.4 两种含义引起混乱

虽然面向对象的适用范围变广了，但两种含义也会引起混乱。

类这个术语在归纳整理法中表示“现实世界中存在的事物”，而在编程中是指“汇总子程序和变量的结构”。实际上，它们是完全不同的。

因此，如果不明确区分这些含义，就容易引起误解，让人认为这是表示现实世界，并将其直接表示为软件的技术。关于这一点，如果大家注意到本章前面介绍的“计算机只是承担了现实世界中的一部分工作”，就应该会明白这是误解。

另外，“将现实世界直接表示为软件”的概念也很有吸引力。对于辛苦地进行软件开发的技术人员来说，该概念可能听起来像禅学答案：“不用考虑得太难，据实编写软件就可以了。”

实际上，类和实例、消息传递、继承等结构都容易让人联想到现实世界的情形。在设计类的职责分配进行调试时，大家有时会有错觉，感觉自己编写的软件构件就像有生命一样。即使是认为面向对象和现实世界不一样的人，为了形象地表达 OOP 结构，有时也会使用比喻。这些都导致“面

向对象是直接将现实世界表示为程序的技术”这种解释变得很普遍。

7.5 分为 OOP 的扩展和归纳整理法进行思考

到目前为止，我们介绍了面向对象的两个方面，即归纳整理法和编程技术。

现在，面向对象已经不再是单纯的编程技术，而是软件开发的综合技术，包含很多应用技术。关于这些应用技术，我们在后面会逐个介绍，这里先对各个技术是属于归纳整理法还是属于 OOP（编程技术）进行简单的汇总（表 7-1）。

表 7-1 将面向对象的应用技术分为 OOP 和归纳整理法

应用技术	分　类	章　节
类库 框架 组件	OOP 的扩展	第 6 章
设计模式	OOP 的扩展	第 6 章
UML	OOP 和归纳整理法	第 8 章
建模（业务分析、需求定义）	归纳整理法	第 9 章
面向对象设计	OOP 的扩展	第 10 章
开发流程	-（与 OOP 和归纳整理法都没有关系）	第 11 章

我们可以粗略地认为，上流工程是归纳整理法，设计之后的下流工程是编程技术。

不过，它们的共同目的都是编写出优秀的软件。我们在第 1 章中也介绍过，面向对象是用来编写优秀软件的综合技术，如今看来确实就是这样的。

7.6 为何化为了通用的归纳整理法

在本章最后，我们来稍微思考一下，作为编程语言结构提出来的面向对象为何会化为通用的归纳整理法。OOP 的类和集合论实际上是完全不同的结构，冷静想来，将它们放在一起，思维跳跃有点大。

我们先来说一下“类”这个名称。在被认为是最早的 OOP 的 Simula 67 中，该名称被用作汇总子程序和变量的结构的关键字。英文“class”有“种类”的意思，因此，将类应用于分类结构，并进一步发展为集合论也是顺理成章的了。

另外，“面向对象”一词本身就具有“面向物”“以物为中心”的含义，给人以“从以功能为中心到以物为中心”“万物都是对象”的印象。不过，其实“面向对象”一词在 Simula 67 出现时还没有，据说是之后提出 Smalltalk 的艾伦·凯命名的。Smalltalk 中使用继承结构来组织类库，最上位的类的名称是 `Object`[1]。基于该结构，Smalltalk 中编写的所有类都是 `Object` 类的子类，因此，人们很容易联想到“与现实世界一样，万物都是对象（object）”。这样一来，面向对象化为集合论也就水到渠成了。

当时，在 Simula 67 中，开发者将汇总子程序和变量的结构命名为“类”时也许并没有考虑这么深（Simula 67 的开发者是挪威人，母语并不是英语）。

虽然历史上没有“如果……就……”，但是如果当时没有命名为“类”，而是命名为不容易让人联想到集合论的“模块”等其他名称，那么软件开发技术的历史可能会完全不同。

① Java、.NET 和 Ruby 等诸多 OOP 都沿用了这种结构。

专栏

对象的另一面

语言在先，还是概念在先

本书的立场是：面向对象的中心是编程语言。也就是说，OOP 是结构化编程的发展形式，设计模式和 UML 等都是由此衍生出来的技术。人们将类、多态和继承等结构仿照现实世界进行讲解只是打个比方。

不过，原本被艾伦·凯命名为“面向对象”的也不是编程语言的结构，而是“独立性较高的对象互相发送异步消息的模型”这一概念。如果关注到这一点，那么面向对象的中心就是该概念，并将其适用于现实世界的模型化和编程两方面。

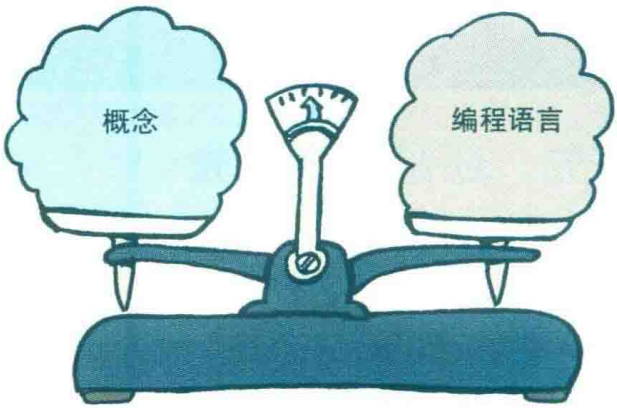
＊＊＊

笔者从 20 世纪 90 年代初期开始关注面向对象，当时的主流是以上述概念为中心的思想。当时，面向对象方法论相关的很多图书和研讨会中都会介绍说“有面向对象这样一个概念，它提供的各种技术能够让面向对象应用于软件开发”。与现在不同，当时几乎所有的编译器都是收费的，昂贵的 Smalltalk 开发环境等就像是高岭之花，只有企业或研究机构中的极少一部分人才能够接触到。由于面向对象在实际系统中的应用一直没有进展，所以许多学习面向对象的人并未实际进行编程，而只是阅读方法论的书，最终也只是在概念层面上掌握了面向对象。

直到 1995 年，免费的 Java 出现并广泛普及，这种状况才大有改观。人人都可以在自己的计算机上下载类、多态、继承结构以及将 `Object` 类作为祖先类的类库，并轻松地进行试验。如今，虽然可能还是有不少人认为面向对象的中心是上述概念，但不了解编程结构，只是从概念上来把握面向对象的人基本上没有了。

＊＊＊

笔者在很长一段时间内也认为概念是面向对象的中心，在通过业务分析和需求定义将面向对象的概念适用于现实世界，并试着使用 C++ 或 Java



来实现的过程中发现，这些概念并不像当初预期的那样可以很好地适用于现实世界的模型化和软件的设计。过分拘泥于适用概念，反而造成应用程序结构难以维护，这样的事例笔者曾经目睹了很多。

通过这些经历，笔者开始认为面向对象的中心是编程语言，编程技术和上流工程的归纳整理法是不同的。通过这样考虑，就能够看到全貌，之前对该技术抱有的模糊的期待感也会消失了。

一个形容新技术的有魅力的概念会激发人们的想象力，成为扩展该技术适用领域的原动力，但到了该技术真正普及的阶段，概念的面纱就会被摘掉，只留下实实在在的技术本身。

本章的关键词

UML、类图、时序图、通信图、用例图、活动图、状态机图

第8章 UML：查看无形软件的工具

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

制定并提出开发方法论的人称为方法学家。下面哪一项是国际标准化组织 OMG（Object Management Group，对象管理组织）在制定 UML 时经常提起的表示方法学家和恐怖分子区别的戏言？

A. 恐怖分子想靠智慧和力量改变世界，而方法学家却仅想靠力量改变世界

B. 恐怖分子要破坏世界，而方法学家要支配世界

C. 恐怖分子能谈判，而方法学家不能谈判

D. 恐怖分子是集团战斗，而方法学家只是一个人在战斗