华章程序员书库

Pearson

HZ BOOKS 华章IT

面向对象的思考过程

（原书第4版）

The Object-Oriented Thought Process

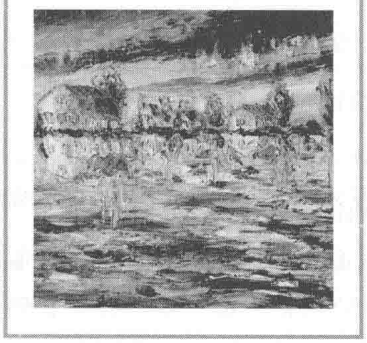
Fourth Edition

[美] 马特·魏斯费尔德（Matt Weisfeld） 著

黄博文 译

机械工业出版社
China Machine Press

华章程序员书库



The Object-Oriented Thought Process
Fourth Edition

面向对象的思考过程
（原书第4版）

[美] 马特·魏斯费尔德（Matt Weisfeld） 著

黄博文 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

面向对象的思考过程（原书第 4 版）/（美）马特·魏斯费尔德（Matt Weisfeld）著；黄博文译．—北京：机械工业出版社，2016.11
（华章程序员书库）
书名原文：The Object-Oriented Thought Process, Fourth Edition

ISBN 978-7-111-55308-3

I. 面… II. ①马… ②黄… III. 面向对象语言－程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2016）第 262345 号

本书版权登记号：图字：01-2013-5977

面向对象的思考过程（原书第 4 版）

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：陈佳媛　　责任校对：殷　虹
印　　刷：北京诚信伟业印刷有限公司　　版　　次：2016 年 11 月第 1 版第 1 次印刷
开　　本：186mm×240mm　1/16　　印　　张：15.5
书　　号：ISBN 978-7-111-55308-3　　定　　价：69.00 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）88379426　88361066　　投稿热线：（010）88379604
购书热线：（010）68326294　88379649　68995259　　读者信箱：hzit@hzbook.com

The Translator's Words 译 者 序

很多IT从业人员进入这个行业都是从学习一门编程语言开始的。对于编程，我们往往过于关注语言的语法细节，反而忽略了其背后的设计理念。面向对象的思考过程就是一个非常优秀的设计理念。它可以独立于语言存在。如果你熟练掌握了面向对象的思考过程，那么就可以轻松地在不同的面向对象的语言之间切换。

本书透彻地阐述了面向对象这一概念。作者Matt在书中反复强调学习面向对象的思考过程优于学习任何编程语言或工具。事实上，他也是这么做的。Matt阐述了面向对象的三要素：继承、封装、多态，并且自己加上了第四个要素：组合。关于组合，Matt不惜篇幅做了大量的讲解，并且列举了很多通俗易懂的例子，这也是本书的一大特色。

Matt也纠正了人们的一些普遍误解，比如面向对象的范式与面向过程的范式并不是完全对立的关系。而且在应用面向对象的设计和开发时，Matt也讲解了不少如何与遗留系统集成的技巧。同时，Matt也简要介绍了UML这个建模利器。为了不混淆重点，他把介绍UML的章节放置在很靠后的位置。因为他明白，先了解面向对象的各项概念是最重要的。

我虽然拥有多年的从业经验，但是再看本书时仍然有不少收获。其实自从我接触了函数式编程，就渐渐成为函数式编程的拥趸。我会时不时地“鼓吹”函数式编程范式的好处，顺便“贬低”一下面向对象编程。但同时我也有个疑问，既然函数式编程这么好，为什么这几年的发展只能算是波澜不惊，而没有掀起大风浪呢？读了本书之后，我似乎找到了答案。首先面向对象的思考过程更加符合大家对世界的直观感受，毕竟不是每个人都是数学家。函数式编程可以简化很多问题，但它并不能简化所有问题。其次是面向对象的编程范式和函数式编程的范式并不是完全对立的，正如作者讲过，面向过程的编程范式和面向对象的编程范式也不是完全对立的。比如目前流行的一些语言（Scala、Go等）都具备函数式的特点，也兼具面向对象的特点（只不过它们的面向对象的机制与传统的方式有所不同）。所以无论你喜欢哪种编程范式，了解彼此的不同之处是至关重要的。而本书则是了解面向对象范式的优秀

书籍。

本书已经更新到了第 4 版。从本书长达 10 多年的跨度来看，面向对象范式经久不衰。Matt 也适时地在新版中加入了一些新的主题，比如可移植数据、分布式系统、Web 服务等。Matt 不仅阐述了这些技术，还讲述了它们的前世今生。这样可以帮助读者更加充分地了解技术的演化之路。

无论你是否有面向对象编程的经验，本书都适合你作为面向对象思考的旅程开端。最后，希望本书能给大家带来超凡的阅读体验。

About the Author 作者简介

Matt Weisfeld 居住于美国俄亥俄州的克利夫兰市。他既是大学教授、软件开发工程师，也是作家。他在信息技术领域拥有 20 年的经验，之后进入大学任教。他先后当过软件开发工程师、企业家以及兼职教授。Matt 拥有计算机科学硕士学位以及工商管理硕士学位。Matt 除了本书之外，Matt 还撰写了其他两本关于软件开发的书籍，并在杂志和期刊上发表了多篇文章。这些杂志和期刊包括《 developer.com 》《 Dr.Dobb's 》《 C/C++ Users 》《 Software Development 》《 Java Report 》和国际期刊《 Project Management 》等。

前　言 Preface

本书内容概要

正如书名所述，本书讲述了面向对象思考的过程。选择一本书的主题和书名是个很重要的决定，但如果主题概念性很强，决定就没那么容易了。大部分书籍都只涉及了编程及面向对象设计的某个方面。一些主流的书阐述了诸如面向对象分析、面向对象设计、面向对象编程、设计模式、面向对象的数据结构（XML）、统一建模语言（UML）、面向对象 Web 开发、面向对象移动开发、进阶面向对象编程语言等主题，当然也包括了其他与面向对象编程相关的主题。

然而，许多人仔细研究这些书后，都未曾注意到这些主题都建立在同一个基础之上，即如何以面向对象的方式进行思考。从学生到软件开发专业人员，往往虽然阅读了这些书，但没有花费充分的时间和精力来真正理解代码背后的设计理念。

我认为仅学习一种特定的开发方法、一种编程语言或者一组设计工具并不能说明学会了面向对象这一概念。简单来说，以面向对象方式编程就是一种思考方式。本书就讨论这种面向对象的思考过程。

把面向对象的思考过程从语言、开发实践以及工具中剥离出来并不是一个简单的任务。在学习面向对象这一理念时，往往要求先深入学习一门编程语言。例如，很多年以前，大量的 C 语言程序员在没有直接接触面向对象概念之前，就开始通过 C++ 语言来了解面向对象。其他软件专家第一次接触面向对象则是在演示文稿中使用 UML 创建对象模型。他们也没有直接学习面向对象的概念。即使到现在，互联网作为商业平台的几十年后，编程书籍以及专业的培训材料并没有先介绍面向对象这一概念。

学习面向对象的概念与学习使用面向对象语言进行编程有着巨大差异，理解这点很重要。我在编写本书第 1 版前就意识到了这点。当我阅读 Craig Larman 的文章《What the UML

Is-and Isn't》时，他指出：

但是，在软件开发工程和 UML 绘图语言领域，读写 UML 标记的能力有时候好像等同于面向对象的分析和设计能力。事实当然并非如此，后者比前者更加重要。因此我推荐先学习面向对象分析和设计的相关教学资料，它优先于学习使用 UML 标记的相关工具。

因此，尽管学习一门建模语言是非常重要的步骤，但先学习面向对象的技能更加重要。如果未完全理解面向对象概念前就学习 UML，这就像还未了解任何与电路相关的知识就开始学习电路图一样。

学习编程语言也有相同的问题。如前所述，很多 C 语言程序员还未直接了解任何面向对象的概念，就想通过使用 C++ 语言来达到面向对象的境界。在面试中经常会出现这样的情况，很多自诩 C++ 程序员的开发人员只是会使用 C++ 编译器的 C 程序员。甚至现在，诸如 C#.NET、VB.NET、Objective-C 以及 Java 等语言已经相当普及了，工作面试中的一些关键问题可以迅速暴露出这些程序员缺乏面向对象的思想。

Visual Basic 的早期版本并不是面向对象的。C 语言也不是面向对象的。而 C++ 在设计时就向后兼容 C 语言。因此，使用 C++ 编译器编写只含 C 语言语法的程序，而放弃使用 C++ 的面向对象功能是完全可能的。Objective-C 是标准 ANSI C 语言的一个扩展。更糟糕的是，程序员可能使用勉强够用的面向对象功能把程序写成了既不是面向对象的也不是面向过程的四不像产品。

因此，在学习使用面向对象的开发环境之前，先学习基本的面向对象概念至关重要。与其直接学习一门编程语言（比如 Objective-C、VB .NET、C++、C# .NET 或 Java）或建模语言（如 UML），还不如把时间花在学习面向对象的思考过程上。

我在使用 C 语言编程很多年后，于 20 世纪 80 年代后期开始参加 Smalltalk 语言的学习课程。当时我所在的公司认为公司的软件开发人员应该学习这个极具前途的技术。老师授课时说面向对象的范式是全新的思维方式（事实上它从 20 世纪 60 年代就已经萌芽了）。他接着说，虽然几乎我们所有人都是很优秀的程序员，但还有 10% ~ 20% 的人从来没有按照面向对象的方式做事。如果该说法确实正确，那么很可能是因为很多优秀的程序员从没有花时间进行编程范式的转变，没有深入学习面向对象概念。

第 4 版中的新增内容

正如在前言中经常提及的一样，第 1 版中我的愿景仅仅围绕概念本身，而不是具体的新兴技术。尽管我在第 2 版、第 3 版以及第 4 版仍然坚持该目标，但也引入了几个章节讲述关于应用程序的主题，这些主题与面向对象概念契合度很高。第 1 ~ 10 章涵盖了基本的面向

对象概念，第 11 ~ 15 章将这些概念应用到了一些常用的面向对象技术中。例如，第 1 ~ 10 章提供了面向对象的基础课程（比如封装、多态、继承等），而第 11 ~ 15 章则介绍了一些实际应用。

第 4 版相对于之前的版本拓展了很多主题。以下列出了改进及更新的主题：

- 移动设备开发，包括手机应用、移动应用以及混合开发等。
- iOS 环境下的 Objective-C 代码示例。
- 使用 XML 及 JSON 实现可读性强的数据交换。
- 使用 CSS、XSLT 等技术实现数据渲染与转换。
- Web 服务，包括简单对象访问协议（SOAP）、RESTful Web 服务等。
- 客户端 / 服务器端技术以及封送对象。
- 持久化数据和序列化对象。
- 很多章节都扩充了代码示例，在 Pearson 的网站可以在线查看用 Java、C# .NET、VB .NET 及 Objective-C 编写的代码示例。

目标读者

本书介绍了面向对象的基本概念，并使用代码示例进一步解释概念。最难平衡的一点在于保证代码既具有表现力又能充分阐述概念。本书的目标是让读者即使不用编译器也能理解面向对象的概念和技术。当然，如果你想使用编译器，我们也提供了代码供执行和探索。

本书目标读者包括业务经理、设计师、开发工程师、程序员、项目经理，以及其他想了解面向对象的人士。阅读本书会为你建立一个稳固的基础，有助于阅读其他更与面向对象概念相关的高端书籍。

在这些更高端的书中，我最喜欢的一本是《Java 面向对象设计》（Object-Oriented Design in Java），作者为 Stephen Gilbert 及 Bill McCarty。我很喜欢这本书的讲述方式，并且将其作为我教授的面向对象概念课程的教材。我在本书中多次引用了《Java 面向对象设计》，所以我推荐你看完本书后可以继续阅读《Java 面向对象设计》。

我认为很优秀的其他书还包括《Effective C++》，作者为 Scott Meyers；《Classical and Object-Oriented Software Engineering》，作者为 Stephen R. Schach；《Thinking in C++》，作者为 Bruce Eckel；《UML Distilled》，作者为 Martin Fowler；以及《Java Design》，作者为 Peter Coad 和 Mark Mayfield。

当我在技术社区以及大学中教授编程和 Web 开发的入门课程时，我很快发现大多数程序员容易学会语言语法，但很难掌握该语言面向对象的本质。

本书讲述方式

显然，我坚定地认为熟悉面向对象的思考过程优先于学习编程语言或建模语言。本书有相应的代码示例及 UML 图，但阅读本书并不需要掌握一门具体的编程语言或 UML。我已经说过了要先学习概念本身，那为什么本书有如此多的 Java、C# .NET、VB .NET 以及 Objective-C 代码和 UML 图？首先，它们可以很好地展示面向对象概念。其次，它们对于面向对象的使用方式是极其重要的，有助于介绍和展示如何使用面向对象。关键在于不要过分关注 Java、C# .NET、VB .NET 及 Objective-C 或 UML，而应当使用它们来帮助理解面向对象的深层理念。

注意，在理解类及其自身的属性和方法时，我非常喜欢使用 UML 类图实现可视化。事实上，本书中只用到了 UML 组件中的类图。我认为 UML 类图提供了一种强大的方式来模拟对象模型的概念本质。我依旧使用对象模型作为教学工具来展示类设计以及类之间的关系。

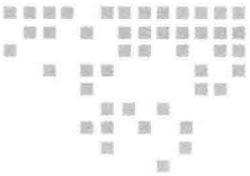
本书中的代码示例阐述了诸如循环和函数之类的概念。然而，理解代码本身并不是理解面向对象概念的先决条件。如果你想了解语言本身的更多细节，那么有一本讲解语言语法的书放在手边则非常有用。

我不能过分强调本书不讲述 Java、C# .NET、VB .NET、Objective-C 或 UML。你可以自学这些知识。我希望本书能激发你对其他面向对象概念的兴趣，比如面向对象分析、面向对象设计以及面向对象编程。

本书中的源码

本书中提到的示例代码可以在华章网站（www.hzbook.com）下载。

目　录 Contents



第 1 章 Chapter 1

面向对象的概念简介

很多程序员其实并不知道在 20 世纪 60 年代就已出现面向对象的软件开发方式。尽管受欢迎的面向对象的编程语言（例如 Smalltalk 和 C++）已被广泛使用，但直到 90 年代中后期面向对象范式才开始快速发展。

面向对象方法论的兴起恰逢互联网作为商业及娱乐平台之际。总之，对象借助网络能良好协作。后来显然互联网存活下来，而面向对象的技术已经在开发新的基于 Web 的技术中占据了重要位置。

本章标题是“面向对象的概念简介”。标题中关键词是“概念”而非“技术”。在软件行业，技术变迁非常快，而概念则是逐步演进。我使用单词“演进”是因为尽管它们保持相对稳定，但也在变化。这正是需要关注这些概念的原因。尽管它们相对稳定，但经常被重新反思，也会导致一些很有意思的讨论。

从 90 年代中后期的最原始的浏览器到如今移动、手机、Web 应用占据主导地位，通过这些多样的行业技术的发展很容易追溯过去 20 年间的演化。甚至如今我们正在探索混合软件，而新的开发技术就在下个拐角。在整个旅程中，每一步都存在面向对象的概念。这也是为什么本章主题如此重要，因为从 20 年前到现在仍在使用这些概念。

1.1 基本概念

本书主要目标是让你学会思考如何将面向对象概念应用于面向对象的系统设计中。历史上定义面向对象的语言拥有以下特点：封装（encapsulation）、继承（inheritance）和多态（polymorphism）。因此，如果设计一门语言时没有完全实现以上特性，那么通常我们认为该

语言不是完全面向对象的。即使实现了这三点，我也往往会加入组合特性。因此，我认为面向对象的概念如下：

- 封装
- 继承
- 多态
- 组合

本书接下来会详述这些特性。

从本书第 1 版开始，我一直在努力解决的问题是这些概念如何直接与当前的设计实践关联起来，因为设计实践始终在变化当中。例如，在面向对象设计中使用继承总是饱受争议。继承是否实际上破坏了封装？（稍后的章节会讨论这一主题。）甚至现在，很多开发人员都试图尽量避免使用继承。

我的方式是始终围绕概念来学习。无论你是否使用继承，你至少需要理解什么是继承，这会保证你的设计决策是有据可依的。正如在介绍中提及的那样，本书目标读者是希望学习基本的面向对象概念的总体介绍的人。请记住这一点，本章中我会展现基本的面向对象概念，希望读者能在做重要的设计决策前打下一个坚实的基础。本章涵盖了这些概念的基本知识，随后的章节中的主题也会讨论这些概念的细节。

1.2 对象及遗留系统

在面向对象成为主流之前，开发人员面临的问题一直就是如何将新的面向对象技术与现有的系统集成起来。面向对象与结构化（或命令式）编程之间拥有明显的界限，而结构化编程当时则是主流的开发范式。而我始终反对面向对象和结构化编程是不兼容的这一观点，因为我认为面向对象与结构化编程绝不是互斥的。它们是互补的，因为对象可以与结构化代码很好地集成。直到现在我还经常听到这样的问题：你是一个面向过程的程序员还是面向对象的程序员？我会毫不犹豫地说：“我都是。”

同样，面向对象的代码并不意味着完全替代结构式代码。很多非面向对象的遗留系统（即现存的较老的系统）仍在正常工作，所以为什么要冒着潜在的风险来替换它们？大多数情况下，你不应当替换它们，至少不应当使用这样的理由。使用非面向对象的代码编程并不存在本质上的错误。然而，新一代开发者则会倾向考虑使用面向对象技术（有些情况下也只能这样做）。

过去的 20 年间，面向对象的开发领域一直在平稳显著地增长，全球社会对网络（比如互联网及移动基础设施）的依赖有力地帮助了面向对象技术飞速发展，甚至成为业界主流。浏览器与移动应用之间的海量交易开辟了全新的市场。在这个市场中很多软件都是全新的，几乎未受到遗留系统的影响。即使需要与遗留系统打交道，也可以使用包装对象将遗留系统包装起来。