HZ BOOKS 华章IT

精选Java实用设计模式，展示Java语言中更加智能化的编码实例

既涵盖面向对象编程、函数式编程和响应式编程模式及使用方法，又介绍从MVC架构向微服务和无服务器架构转变的发展趋势，以及Java新版本的特性及其实践

Java 核心技术系列

[PACKT] PUBLISHING

Java 设计模式及实践

Design Patterns and Best Practices in Java

[印度] 卡马尔米特·辛格（Kamalmeet Singh）
[荷兰] 艾德里安·伊恩库列斯库（Adrian Ianculescu） 著
[罗马尼亚] 路西安-保罗·托尔耶（Lucian-Paul Torje）
张小坤 黄凯 贺涛 译

机械工业出版社
China Machine Press

Java
核心技术
系列

Java
设计模式及实践

Design Patterns and Best Practices in Java

[印度] 卡马尔米特·辛格（Kamalmeet Singh）
[荷兰] 艾德里安·伊恩库列斯库（Adrian Ianculescu） 著
[罗马尼亚] 路西安-保罗·托尔耶（Lucian-Paul Torje）
张小坤 黄凯 贺涛 译

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

Java 设计模式及实践 /（印）卡马尔米特·辛格（Kamalmeet Singh）等著；张小坤，黄凯，贺涛译 . —北京：机械工业出版社，2019.6
（Java 核心技术系列）
书名原文：Design Patterns and Best Practices in Java

ISBN 978-7-111-62943-6

I. J… II. ①卡… ②张… ③黄… ④贺… III. JAVA 语言－程序设计 IV. TP312.8

中国版本图书馆 CIP 数据核字（2019）第 109640 号

本书版权登记号：图字 01-2018-6836

Java 设计模式及实践

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：李忠明　　责任校对：殷　虹
印　　刷：北京瑞德印刷有限公司　　版　　次：2019 年 7 月第 1 版第 1 次印刷
开　　本：186mm×240mm　1/16　　印　　张：13.75
书　　号：ISBN 978-7-111-62943-6　　定　　价：79.00 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　　读者信箱：hzit@hzbook.com

The Translator's Words 译 者 序

设计模式（design pattern）是前辈的经验积累，是软件开发人员解决软件开发过程中的一般问题的通用方案，能够帮助提高代码的可重用性，增强系统的可维护性，快速地解决开发过程中常见的诸多难题。从四人组（即 Erich Gamma、Richard Helm、Ralph Johnson、John Vlissides）的经典著作《设计模式》[⊖]，到市面上各式各样的讲述设计模式的出版物，相信读者已经读过不少。本书的特点是理论与实践相结合，在讲述理论知识的基础上，提供了大量的设计模式实现源码，为你提供良好的 Java 实践。本书的另一大特色是详细阐述了 Java 语言的最新版本所引入的特性，并针对其在经典设计模式中的应用进行了探索。

本书可分为四部分。第一部分（第 1 章）主要介绍了面向对象编程的基本概念和设计模式的基本原则；第二部分（第 2 ～ 4 章）分别介绍了创建型、行为型、结构型三类设计模式，阐述了各种设计模式的概念、实现方式以及适用范围；第三部分（第 5 ～ 7 章）主要介绍了函数式编程及响应式编程的基本概念和应用实现；第四部分（第 8 和 9 章）主要介绍了常见的应用架构及其发展趋势，以及 Java 新版本所引入的功能特性。

参加本书翻译工作的有张小坤、黄凯、贺涛，全书由黄凯审校。在翻译过程中，感谢刘锋先生对一些专业词汇提供了规范性的解释。

由于书中概念术语繁多，且有许多概念和术语目前尚无标准的中文译法，加之译者水平有限，译文中难免有不当之处，恳请读者批评指正。

译者

2019 年 2 月

⊖ 该书已由机械工业出版社引进出版。——编辑注

前　言 Preface

借助设计模式，开发者可以改进代码库，提高代码可重用性，并使技术架构更加健壮。随着编程语言的不断发展，新的语言特性在得到广泛应用之前往往需要大量时间去理解。本书旨在降低接受最新趋势的难度，为开发人员提供良好的实例。

本书的目标读者

本书适用于每一位有意愿编写高质量代码的 Java 开发人员。本书讲述了很多开发者在编码时经常疏忽的最佳实践。书中涵盖了许多设计模式，这些设计模式经开发团队实践和测试过，是用来解决特定问题的最佳方案。

本书内容

第 1 章介绍了 Java 语言不同的编程范式。

第 2 章介绍了多种设计模式中的创建型模式，讲述了多种类型的创建型设计模式。

第 3 章介绍了行为型设计模式，主要解析了多种用来管理代码和对象行为的设计模式。

第 4 章介绍了结构型设计模式，详细解析了用于管理对象结构的设计模式。

第 5 章向读者介绍了函数式编程及与之相关的设计模式。

第 6 章通过实例介绍了响应式编程及其 Java 实现。

第 7 章进一步探索了响应式编程的核心内容及与之相关的设计模式。

第 8 章从 MVC 架构到微服务和无服务器应用，探索了近年来开发者尝试和测试过的多种架构模式。

第 9 章介绍了 Java 的历史、最佳实践和最新版 Java 中的更新，并在最后表达了作者对 Java 未来的期待。

如何充分利用本书

拥有 Java 开发经验者将能从本书中获益良多，推荐读者在阅读过程中探索并充分实践各章中提供的示例代码。

下载示例代码及彩色图像

本书的示例代码及所有截图和图表，可以从 http://www.packtpub.com 通过个人账号下载，也可以访问华章图书官网 http://www.hzbook.com，通过注册并登录个人账号下载。

关于作者 *About the Authors*

Kamalmeet Singh 在 15 岁时第一次尝试了编程并立刻爱上了它。他在获得信息技术学士学位之后加入了一家创业公司，在那里进一步提升了对 Java 编程的热爱之情。IT 行业 13 年的工作经验，以及在不同的公司、国家和领域的沉淀，使他成长为一名王牌开发人员和技术架构师。他使用的技术包括云计算、机器学习、增强现实、无服务器应用程序、微服务等，但他的最爱仍然是 Java。

我要感谢我的妻子 Gundeep，她总是鼓励我接受新的挑战，把最好的留给我。

Adrian Ianculescu 是一名拥有 20 年编程经验的软件开发人员，其中 12 年使用 Java，从 C ++ 开始，然后使用 C #，最后自然地转向 Java。Adrian 在 2 ～ 40 人的团队中工作，他意识到开发软件不仅仅是编写代码，而对以不同的方法和框架设计软件和架构产生了兴趣。在公司工作一段时间之后，他开始转变为自由职业者和企业家，以追随他童年时代的梦想——制作游戏。

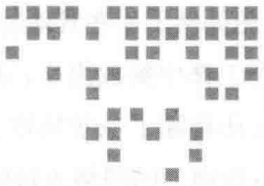
Lucian-Paul Torje 是一名有抱负的软件工匠，在软件行业工作了近 15 年，几乎对所有与技术有关的事情都感兴趣，这就是他涉猎广泛的原因，包括从 MS-DOS TSR 到微服务，从 Atmel 微控制器到 Android、iOS 和 Chromebook，从 C / C ++ 到 Java，从 Oracle 到 MongoDB。每当有人需要使用创新的方法解决问题时，他都热衷于尝试！

About the Reviewer 关于评审者

Aristides Villarreal Bravo 是 Java 开发人员、NetBeans 梦之队的成员以及 Java 用户组的领导者。Aristides 住在巴拿马，他组织并参加了与 Java、JavaEE、NetBeans、NetBeans 平台、免费软件和移动设备相关的各种会议和研讨会。他还是 jmoordb 的作者，主要写关于 Java、NetBeans 和 Web 开发的教程和博客。

Aristides 参与了几个关于 NetBeans、NetBeans DZone 和 JavaHispano 等主题的网站访谈，他是 NetBeans 插件的开发人员。

目　录 *Contents*



第 1 章 Chapter 1

从面向对象到函数式编程

本章介绍如何使用设计模式来写具有健壮性、可维护性、可扩展性的代码，以及 Java 的最新特性。为此，我们需要讨论以下问题：

- 什么是编程范式
- 命令式范式
- 声明式和函数式范式
- 面向对象范式
- 统一建模语言（UML）综述
- 面向对象原则

1.1 Java 简介

1995 年，一个新的编程语言发布了，它从广为人知的 C++ 语言以及鲜为人知的 Smalltalk 语言继承而来。这个新的编程语言就是 Java，它尝试着去改善之前大部分语言的局限性。比如，令 Java 广为流行的一个重要特性是：编写一次就能随处使用。这个特性意味着，你能够在一台 Windows 机器上开发代码，但是可以让代码运行在 Linux 上，或者说其他的系统上，你所需要的只是 JVM（Java Virtual Machine，Java 虚

拟机）。Java 还提供了一些其他特性，比如：垃圾回收器，让开发人员从繁杂的申请内存和释放内存工作中解放出来；JIT（Just In Time，即时编译技术），让 Java 更加智能和快速。Java 还移除了一些特性，比如指针，这样会让 Java 更加安全。上面提到的所有特性以及后续增加的网络支持特性使 Java 成为开发人员的一个普遍选择。自 Java 诞生以来，隔几年就有一种新语言诞生和消失，而 Java 11 已经成功发布并被公众接受，这正说明了 Java 语言的成功。

1.2 Java 编程范式

什么是编程范式，自从有软件开发开始，开发人员尝试了不同的方式来设计编程语言。对于不同的编程语言，我们都有一系列的概念、原则和规定。这些概念、原则和规定就被称为编程范式。从理论上来说，我们希望编程语言只遵从一个编程范式。但是实际上，一个语言往往拥有多个编程范式。

在接下来的几节里，我们会重点介绍 Java 语言所基于的编程范式，包括命令式、面向对象、声明式和函数式编程，以及用来描述这些编程范式的主要概念。

1.2.1 命令式编程

命令式编程是这样一种编程范式：用语句更改程序的状态。这个概念出现在运算的开始，并且与计算机的内部结构紧密相连。程序是处理单元上运行的一组指令，它以命令的方式改变状态（状态即存储器中的变量）。“命令”这个名称，顾名思义，指令的执行即是程序的运行。

今天大多数流行的编程语言或多或少都基于命令式编程发展而来。命令式语言最典型的示例就是 C 语言。

命令式编程示例

为了更好地理解命令式编程范式的概念，让我们举一个例子：你计划在某城镇与一个朋友会面，但他不知道如何到达那里。我们来试着以“命令式”的方式向他解释如何实现目标：

1）在中央火车站乘坐 1 号电车；

2）在第三站下车；

3）向右走，朝第六大道行进，直到到达第三个路口。

1.2.2　面向对象编程

面向对象编程经常与命令式编程联系在一起，在实践当中，两者是可以共存的。Java 就是这种协作的生动证明。

接下来，我们将简要介绍面向对象的基本概念，代码都会以 Java 语言实现。

1. 对象和类

对象是面向对象编程（OOP）语言的主要元素，它包括状态和行为。

如果我们将类视为模板，则对象是模板的实现。例如，如果“人类”是一个定义了人类所拥有的属性和行为的类，那么你我都是这个“人类”类的对象，因为我们已经满足了作为“人类”所有的要求。或者，我们把“汽车”视为一个类，那么一辆特定的本田思域汽车就是“汽车”类的一个对象。这辆本田思域汽车可以满足汽车类所具备的所有属性和行为，比如有引擎、方向盘、车灯等，还能前进、倒退等。我们可以看到面向对象范式如何与现实世界相关联。几乎现实世界中的所有东西都可以从类和对象的角度来考虑，因此 OOP 能够毫不费力地流行起来。

面向对象基于四个基本原则：

- ❑ 封装
- ❑ 抽象
- ❑ 继承
- ❑ 多态

2. 封装

封装主要是指属性和行为的绑定。封装的思路是将对象的属性和行为保存在一个地方，以便于维护和扩展。封装还提供了一种隐藏用户所不需要的细节的机制。在 Java 当中，我们可以为方法和属性提供访问说明符来管理类使用者的可见内容以及隐藏内容。

封装是面向对象语言的基本原则之一。封装有助于不同模块的分离，使得开发人员可以或多或少地独立开发和维护解耦模块。在内部更改解耦模块 / 类 / 代码而不影响其外部暴露行为的技术称为代码重构。

3. 抽象

抽象与封装密切相关，并且在某种程度上它与封装重叠。简而言之，抽象提供了一种机制，这种机制使得对象可以公开它所做的事，而隐藏它是如何做到这些事的。

我们拿现实世界中的“汽车”作为例子来说明抽象。为了驾驶一辆汽车，我们并不需要知道汽车引擎盖下是什么样的，我们只需要知道它给我们暴露的数据和行为。数据显示在汽车的仪表盘上，行为就是我们可以用控制设备来驾驶汽车。

4. 继承

继承是指对象或类基于另一个对象或类的能力。有一个父类或者基类，它为实体提供顶级行为。每一个满足“父类的属性和方法是子类的一部分”条件的子类实体或者子类都可以从父类中继承，并根据需要添加其他行为。

让我们来看一个现实世界的例子。如果我们将 Vehicle 视为父类，我们知道 Vehicle 类可以具有某些属性和行为。例如，Vehicle 类有一个引擎、好几个门等等，并且它拥有移动这个行为。现在满足这些条件的所有实体（例如 Car、Truck、Bike 等），都可以从 Vehicle 类继承并添加给定的属性和行为。换句话说，我们可以说 Car 是一种 Vehicle。

让我们来看代码如何实现。我们首先创建一个名为 Vehicle 的基类，此类拥有一个构造函数，这个函数能够接受一个 String（字符串）类型的参数（车辆名称）：

```
public class Vehicle
{
  private Stringname;
  public Vehicle(Stringname)
  {
    this.name=name;
  }
}
```

现在我们创建一个拥有构造函数的 Car 类。Car 类继承自 Vehicle 类，因此它继承并可以访问在基类中声明为 protected（保护）或 public（公共）的所有成员和方法：