高等学校应用型本科创新人才培养计划指定教材
高等学校计算机类专业“十三五”课改规划教材

Java设计模式

青岛英谷教育科技股份有限公司 编著

西安电子科技大学出版社
http://www.xduph.com

高等学校应用型本科创新人才培养计划指定教材

高等学校计算机类专业“十三五”课改规划教材

Java 设计模式

青岛英谷教育科技股份有限公司　编著

西安电子科技大学出版社

内 容 简 介

本书从最基本的设计原理及思想出发，深入讲解并剖析了 23 种常见的设计模式，每种模式都有相应的案例，这些案例通俗易懂，围绕模式的核心思想，便于读者进一步理解和学习设计模式。全书共分为 8 章，分别介绍了初识设计模式、六大原则、创建型模式、结构型模式、行为型模式、混合设计模式的应用以及设计模式对比。书中涉及 23 种设计模式，每种设计模式都从定义、应用以及实例三个方面进行详细介绍。

本书重点突出，偏重应用，结合实践进行讲解和剖析，读者能迅速理解并掌握程序设计的有关知识，全面提高 Java 程序设计能力。

本书适用面广，可作为本科计算机科学与技术、软件工程、网络工程、计算机软件、计算机信息管理、电子商务和经济管理等专业的程序设计课程的教材，也可作为相关领域程序设计人员的参考书。

图书在版编目(CIP)数据

Java 设计模式/青岛英谷教育科技股份有限公司编著. —西安：西安电子科技大学出版社，2016.1
高等学校计算机类专业“十三五”课改规划教材
ISBN 978-7-5606-3868-3

Ⅰ. ① J… Ⅱ. ① 青… Ⅲ. ① Java 语言—程序设计—高等学校—教材 Ⅳ. ① TP312

中国版本图书馆 CIP 数据核字(2015)第 247822 号

策　　划　毛红兵
责任编辑　毛红兵　何谦谦
出版发行　西安电子科技大学出版社(西安市太白南路 2 号)
电　　话　(029)88242885　88201467　　邮　　编　710071
网　　址　www.xduph.com　　电子邮箱　xdupfxb001@163.com
经　　销　新华书店
印刷单位　陕西华沐印刷科技有限责任公司
版　　次　2016 年 1 月第 1 版　　2016 年 1 月第 1 次印刷
开　　本　787 毫米×1092 毫米　1/16　印　张 15.5
字　　数　360 千字
印　　数　1～3000 册
定　　价　38.00 元

ISBN 978-7-5606-3868-3/TP

XDUP 4160001-1

高等学校计算机专业
“十三五”课改规划教材编委会

❖❖❖ 前　言 ❖❖❖

本科教育是我国高等教育的基础，而应用型本科教育是高等教育由精英教育向大众化教育转变的必然产物，是社会经济发展的要求，也是今后我国高等教育规模扩张的重点。应用型创新人才培养的重点在于训练学生将所学理论知识应用于解决实际问题，这主要依靠课程的优化设计以及教学内容和方法的更新。

另外，随着我国计算机技术的迅猛发展，社会对具备计算机基本能力的人才的需求急剧增加，“全面贴近企业需求，无缝打造专业实用人才”是目前高校计算机专业教育的革新方向。为了适应高等教育体制改革的新形势，积极探索适应新形势下人才培养的教学模式，我们组织编写了高等院校计算机及其相关专业的系列课改教材。

该系列教材面向高校软件专业应用型本科人才的培养，强调产学研结合，经过了充分的调研和论证，并参照多所高校一线专家的意见，教材内容具有系统性、实用性等特点，旨在使读者在系统掌握软件开发知识的同时，提高其解决问题的能力和综合应用能力。

该系列教材具有如下几个特色：

1. 以培养应用型人才为目标

本系列教材以培养应用型软件人才为目标，在原有体制教育的基础上对课程进行了改革，强化“应用型”技术的学习，使读者在经过系统、完整的学习后能够掌握如下技能：

- ✧ 掌握软件开发所需的理论知识和技术体系以及软件开发过程的规范体系；
- ✧ 能够熟练地进行设计和编码工作，并具备良好的自学能力；
- ✧ 具备一定的项目经验，能够进行代码调试、文档编写、软件测试等；
- ✧ 达到软件企业的用人标准，做到学校学习与企业需求能力的无缝对接。

2. 以新颖的教材架构来引导学习

本系列教材采用的教材架构打破了以知识为标准编写教材的传统方法，学习内容的选取遵循“二八原则”，即重点内容由企业中常用的 20%的技术组成。每章设有本章目标，以明确本章的学习重点和难点，章节内容结合示例代码，引导读者循序渐进地理解和掌握这些知识和技能，培养学生的逻辑思维能力，掌握软件开发的必备知识和技巧。

另外，本系列教材借鉴了软件开发中“低耦合，高内聚”的设计理念，组织结构上遵循软件开发中的 MVC 理念，即在保证最小教学集的前提下学生可以根据自身的实际情况对整个课程体系进行横向或纵向裁减。

3. 提供全面的教辅产品来辅助教学实施

为充分体现“实境耦合”的教学模式，方便教学实施，该系列教材配备了可配套使用

的项目实训教材和全套教辅产品。

- ✧ 实训教材：集多线于一面，以辅助教材的形式，提供适应当前课程(及先行课程)的综合项目，按照软件开发过程进行讲解、分析、设计、指导，注重工作过程的系统性，培养学生解决实际问题的能力，是实施“实境”教学的关键环节。
- ✧ 立体配套：为适应教学模式和教学方法的改革，本系列教材提供完备的教辅产品，主要包括教学指导、实验指导、电子课件、习题集、实践案例等内容，并配以相应的网络教学资源。教学实施方面，提供全方位的解决方案(课程体系解决方案、实训解决方案、教师培训解决方案和就业指导解决方案等)，以适应软件开发教学过程的特殊性。

本书由青岛英谷教育科技股份有限公司编写，参与本书编写工作的有王燕、宁维巍、朱仁成、宋国强、何莉娟、杨敬熹、田波、侯方超、刘江林、方惠、莫太民、邵作伟、王千等。本书在编写期间得到了各合作院校专家及一线教师的大力支持与协作，在此衷心感谢每一位老师与同事为本书出版所付出的努力。

由于作者水平有限，书中难免有不足之处，欢迎大家批评指正！读者在阅读过程中发现问题，可以通过邮箱(yinggu@121ugrow.com)发给我们，以帮助我们进一步完善。

本书编委会

2015 年 8 月

❖❖❖ 目　录 ❖❖❖

第1章　初识设计模式

本章目标

- 了解设计模式的概念
- 了解设计模式的历史
- 理解设计模式的要素
- 掌握设计模式的分类

1.1 设计模式概念

设计模式(Design Pattern)是一套被反复使用、多数人知晓、经过分类编目的优秀代码设计经验的总结。使用设计模式是为了提高代码的重用性，使代码更易理解并保证代码的可靠性。毫无疑问，设计模式的使用，于己、于他人、于系统都是有利的，设计模式使代码编制真正工程化，是软件工程的基石，使人们可以更加简单方便地复用成功的设计和体系结构，将已证实的技术表述成设计模式也会使新系统开发者更加容易理解其设计思路。在面向对象编程语言(例如 Java)中，设计模式提供了一套可以复用的面向对象技术。

Java 提供了丰富的 API，这似乎使编程变成了类似堆积木般的简单“拼凑”和“调用”，甚至有人提倡“蓝领程序员”，这些都是对现代编程技术的不了解所致。可复用面向对象软件系统一般划分为两大类：应用程序工具箱和框架(Framework)。我们平时开发的具体软件都是应用程序，Java 的基础类库属于工具箱，而框架是构成特定软件可复用设计的一组相互协作的类。框架通常定义了应用体系的整体结构类和对象的关系等设计参数，以便于具体应用实现者能集中精力于应用本身的特定细节。框架主要记录软件应用中共同的设计决策，强调设计复用，因此成熟的框架设计中必然要使用设计模式，熟悉这些设计模式，将有助于对框架结构的理解，从而能够迅速掌握框架的结构。例如：初次接触 EJB、Java EE 等框架，会觉得特别难学，难掌握，此时如果先掌握设计模式，则使我们具有剖析 EJB 或 Java EE 系统的能力。

Java 设计模式贯彻的原理是：面向接口编程，而不是面向实现。其目标原则是：降低耦合，增强灵活性。

1.2 设计模式简史

设计模式的研究起源于建筑工程设计大师 Christopher Alexander 的关于城市规划和建筑设计的著作。尽管他的著作是针对城市规划和建筑设计的，但是其观点实际上适用于所有工程设计领域，包括软件开发设计领域。Alexander 在其著作中指出，使用现在的设计方法所设计出的建筑物，不能满足所有工程设计的基本目的——改善人类的生活条件。Alexander 想要发明的建筑结构，是能使人类在舒适和生活质量上受惠的建筑结构。他得出的结论是，设计师必须不断努力，以创造出更加适合所有的住户、用户和他们的社区的结构，从而满足他们的需要。同样软件开发中的设计模式也需要不断地进行研究和创新，以更加适合软件工程的各个方面。

设计模式在软件行业中的应用可以追溯到 1987 年。Ward Cunningham 和 Kent Beck 在一起用 Smalltalk 做设计用户界面的工作，他们决定使用 Alexander 的理论发展出一个有五个模式的语言来指导 Smalltalk 的新手，因此他们写了一篇名为《Using Pattern Languages for Object-Oriented Programs(使用模式语言做面向对象的程序)》的论文。此后不久，James O. Coplien 开始搜集 C++ 语言的成例(成例也可认为是一种设计模式，更偏重于编码技巧)，这些 C++ 成例发表在 1991 年出版的《Advanced C++ Programming Styles and

Idioms(高级 C++ 编程风格和成例)》一书中。

1990 年到 1992 年，Erich Gamma、Richard Helm、Ralph Johnson 和 John Vlissides 四人(常被称为 Gang of Four、GoF 或“四人帮”，如图 1-1 所示)开始了搜集模式的工作。1993 年 8 月，Kent Beck 和 Grady Booch 主持了在科罗拉多的山区度假村召开的第一次关于模式的会议，模式研究的主要人物都参加了这次会议，包括 James O. Coplien，Doug Lea，Desmond D'Souze，Norm Kerth，Wolfgang Pree 等。

1995 年 GoF 的《Design Patterns: Elements of Reusable Object-Oriented Software(设计模式：可复用面向对象软件的基础)》出版。该书第一次将设计模式提升到理论高度，并将之规范化，同时书中提出了 23 种基本设计模式。此书发表之后，引起了设计模式的研究热潮，被确定为模式结构的数目也呈爆炸性增长。时至今日，在可复用面向对象软件的发展过程中，新的设计模式仍然不断出现。

图 1-1　GoF 的四个成员

1.3　设计模式要素

描述设计模式需要有一定的格式，可采用的格式有很多。例如：在 Alexander 的著作中使用的格式叫做 Alexander 格式，在 GoF 的著作中使用的格式叫做 GoF 格式。在 GoF 格式中，沿用了一些 Alexander 格式中的段落标题，这些标题常常叫做正则格式。尽管在不同的格式中，正则格式的细节有所不同，但设计模式应当包含以下几个要素：

1．模式名称(pattern name)

设计模式的名称简洁地描述了设计模式的问题、解决方案和效果。一个模式必须有一个有意义的、简短而准确的名字。好的模式名称便于设计人员之间进行思想交流、抽象讨论及研究设计。找到恰当的模式名也是设计模式编目工作的难点之一，命名一个新的模式，就可以进行模式讨论并在编写文档时使用它们。

2．问题(problem)

问题描述了应该在何时使用模式。它解释了设计问题和问题存在的前因后果，它可能

描述了特定的设计问题，如怎样用对象表示算法，也可能描述了导致不灵活设计的类或对象结构。有时候，问题部分会包括使用模式必须满足的一系列先决条件。

3．环境或初始环境(context 或 initial context)

环境说明模式的使用范围，也是模式应用之前的起始条件(也叫前提条件)。

4．解决方案(solution)

解决方案描述了设计的组成成分，以及它们之间的相互关系及各自的职责和协作方式。模式就像一个模板，可应用于多种不同场合，因此解决方案并不描述一个特定而具体的设计或实现，而是提供设计问题的抽象描述和说明怎样用一个具有一般意义的元素组合(类或对象组合)来解决这个问题。

5．效果(consequences)

效果描述了模式应用的效果和使用模式应权衡的问题。效果用来描述设计模式的利弊，它往往是衡量模式是否可用的重要因素，对于评价设计选择和理解使用模式的代价及好处具有重要意义。软件效果大多关注对时间和空间的衡量，表述了语言和实现问题。因为复用是面向对象设计的要素之一，所以模式效果包括其对系统的灵活性、扩展性和可移植性的影响，显式地列出这些效果对理解和评价各种模式具有很大的帮助。

6．举例(examples)

通常举例使用一个或多个示意性的应用来说明特定的真实环境，并说明模式如何应用到环境中、改变环境并且给出当模式结束时的末态环境。例子有助于使用者理解模式的使用方法和适用性，每一个例子均可以附带一个实现的样本，说明解答是如何给出来的。从熟知系统里取出来的、有视觉效果的或以比喻方式表达的例子，更易于使用者理解。

7．末态环境(resulting context)

末态环境是模式应用到系统之后的状态。末态环境是模式的末态条件和可能有的副作用。它包括模式带来的好结果和坏结果，以及新状态中含有的其他问题和可能涉及的其他有关系的模式。描述末态环境可以帮助比较末态环境与起始环境的区别和联系。

8．推理(rationale)

推理解释模式的步骤、规则，以及此模式作为一个整体是如何以特定的方式解决模式的。推理让使用者知道模式是如何工作的，为什么可以工作，以及使用此模式的优点是什么。模式的解答描述模式外部可见的结构和行为，而推理则给出模式在系统表层以下的深层结构和关键机制。

9．其他有关模式(related pattern)

其他有关模式描述在现有的系统中某种模式与其他模式的静态和动态的关系。相关模式的初始环境和末态环境通常是相容的，这些模式有可能是本模式的前任模式，即应用了这些模式可以给出本模式的初始环境；也有可能是本模式的继任模式，即本模式的应用给出这些模式的初始环境；还有可能是本模式的替换模式，即相同问题给出不同解答；也有可能是本模式的相互依赖的模式，可以(或必须)和本模式同时使用。

10. 已知应用(known uses)

已知应用是指在已有的系统中出现或应用的模式例子，它有助于证明此模式确实是对一个重复发生的问题的可行性解答。已知应用经常作为教学用的教材模式案例。

任何对模式的讲解或者论述必须依据一定的格式，正如 GoF 格式一样，所有的模式都应当包括模式的所有要素。

1.4 设计模式分类

常用的设计模式可以概括为 23 种，按照特点可以将其分为三大类型：创建型、结构型、行为型。

1.4.1 创建型

创建型模式是用来创建对象的模式，抽象了实例化的过程，帮助一个系统独立于其关联对象的创建、组合和表示方式。所有的创建型模式都有两个主要功能：

- ✧ 将系统所使用的具体类的信息封装起来。
- ✧ 隐藏类的实例是如何被创建和组织的。外界对于这些对象只知道它们共同的接口，而不清楚其具体的实现细节。

正因为以上两点，创建型模式在创建什么(what)、由谁(who)来创建，以及何时(when)创建这些方面，都为软件设计者提供了尽可能大的灵活性。

例如，假定在一个游戏开发场景中，会用到一个现代风格房屋的对象，则其 Java 代码如下：

```
Room room = new ModernRoom();
```

现在现代风格房屋的对象已经有了，但如果这时房屋的风格变化了，需要古典风格的房屋，则应创建一个古典风格的房屋，其 Java 代码如下：

```
Room room = new ClassicalRoom();
```

如果在程序中有多处用到了这样的创建逻辑，仅仅是因为房屋的风格变化了，就需要修改程序中所有这样的语句。这时我们可以使用创建型模式封装对象创建的逻辑，将对象的创建放在一个工厂方法中，代码如下：

```
RoomFactory factory = new ModernRoomFactory();
Room modernRoom = factory.create();
```

当房屋的风格发生变化时，只需要将 factory 重新构造为 ClassicalRoomFactory 的实例即可：

```
RoomFactory factory = new ClassicalRoomFactory();
Room classicalRoom = factory.create();
```

factory 对象在整个应用中可以只构造一次，因此上述改动量很小，而其他用到 room 的地方仍然不变，这就是为什么需要创建型模式。创建型模式的作用可以概括为如下两点：

- ✧ 封装创建逻辑，不仅仅是 new 一个对象那么简单。
- ✧ 封装创建逻辑变化，客户代码尽量不修改或尽量少修改。

常见的创建型设计模式有下列几种：

- ✧ 单例模式(Singleton Pattern)：一个类只有一个实例，而且自行实例化并向整个系统提供这个实例。
- ✧ 工厂方法模式(Factory Pattern)：在工厂方法模式中，工厂类成为了抽象类，实际的创建工作将由其具体子类来完成。工厂方法的用意是定义一个创建产品对象的工厂接口，将实际创建工作推迟到子类中，它强调的是“单个对象”的变化。
- ✧ 抽象工厂模式(Abstract Factory)：抽象工厂是所有工厂模式中最为抽象且最具有一般性的一种形态。抽象工厂可以向客户提供一个接口，使得客户可以在没有指定产品的具体类型的情况下，创建多个产品族中的产品对象，强调的是“系列对象”的变化。
- ✧ 建造者模式(Builder Pattern)：把构造对象实例的逻辑移到了类的外部，在类的外部定义了该类的构造逻辑。它把一个复杂对象的构造过程从对象的表示中分离出来，其直接效果是将一个复杂的对象简化为一个比较简单的目标对象，强调的是产品的构造过程。
- ✧ 原型模式(Prototype Pattern)：原型模式和工厂模式一样，同样对客户隐藏了对象创建工作具体的实现细节，但与通过对一个类进行实例化来构造新对象不同的是，原型模式通过复制一个现有对象生成新对象。

1.4.2 结构型

顾名思义，结构型模式讨论的是类和对象的结构，它采用继承机制来组合接口或实现(类结构型模式)，或者通过组合一些对象实现新的功能(对象结构型模式)。这些结构型模式在某些方面具有很大的相似性，但侧重点各有不同。

常见的结构型设计模式有以下几种：

- ✧ 代理模式(Proxy)：为其他对象提供一种代理以控制对该对象的访问。
- ✧ 装饰模式(Decorator)：动态地给一个对象添加一些额外的职责。就增加功能来说，装饰模式比生成子类更为灵活。
- ✧ 适配器模式(Adapter)：将一个类的接口变换成客户端所期待的接口，从而使原本因接口不匹配而无法在一起工作的两个类能够在一起工作。
- ✧ 组合模式(Composite)：也叫合成模式，将对象组合成树形结构以表示“部分-整体”的层次结构，使得用户对单个对象和组合对象的使用具有一致性。
- ✧ 桥梁模式(Bridge)：也叫桥接模式，将抽象和实现解耦，使得两者可以独立地变化。
- ✧ 外观模式(Facade)：也叫门面模式，要求一个子系统的外部与其内部的通信必须通过一个统一的对象进行，外观模式提供一个高层次的接口，使得子系统更易于使用。
- ✧ 享元模式(Flyweight)：是池技术的重要实现方式，使用共享对象可有效地支持大量的细粒度的对象。

1.4.3　行为型

行为型设计模式关注的是对象的行为，用来解决对象之间的联系问题。常见的行为型设计模式有以下几种：

- ✧ 模板方法模式(Template Method)：定义一个操作中的算法的框架，而将一些步骤延迟到子类中，使得子类可以在不改变算法的结构的情况下重新定义该算法的某些特定步骤。
- ✧ 命令模式(Command)：是一种高内聚的模式，将一个请求封装成一个对象，从而使用不同的请求把客户端参数化。对请求排队或者记录请求日志，可以提供命令的撤销和恢复功能。
- ✧ 责任链模式(Chain of Responsibility)：使多个对象都有机会处理请求，从而避免了请求的发送者和接受者之间的耦合关系。将这些对象连成一条链，并沿着这条链传递该请求，直到有对象处理它为止。
- ✧ 策略模式(Strategy)：也叫政策模式，定义一组算法，将每个算法都封装起来，并且使它们之间可以互换。
- ✧ 迭代器模式(Iterator)：提供一种方法访问一个容器对象中的各个元素，而又不需要暴露该对象的内部细节。
- ✧ 中介者模式(Mediator)：用一个中介对象封装一系列对象交互，中介者使各对象不需要显示相互作用，从而使其耦合松散，而且可以独立地改变它们之间的交互。
- ✧ 观察者模式(Observer)：也叫发布订阅模式，定义对象间的一对多的依赖关系，使得每当一个对象改变状态，则所有依赖于它的对象都会得到通知并被自动更新。
- ✧ 备忘录模式(Memento)：在不破坏封装性的前提下，捕获一个对象的内部状态，并在该对象之外保存这个状态。
- ✧ 访问者模式(Visitor)：封装一些作用于某种数据结构中的各元素的操作，它可以在不改变数据结构的前提下定义作用于这些元素的新的操作。
- ✧ 状态模式(State)：当一个对象内在状态改变时允许其改变行为，这个对象看起来像改变了其类型，状态模式的核心是封装，状态的变更会引起行为的变更。
- ✧ 解释器模式(Interpreter)：给定一门语言，定义它的文法的一种表示，并定义一个解释器，该解释器使用该文法表示来解释语言中的句子。

小　结

通过本章的学习，学生应该能够理解：

- ✧ 设计一个模式的过程就是将问题抽象化，忽略不重要的细节后发现问题的本质，并找到普遍适用的解决方案的过程。
- ✧ GoF 的《设计模式》提供了一套可复用的面向对象技术。

- ✧ 设计模式起源于建筑设计学。
- ✧ 设计模式的基本要素是：模式名称、问题、环境或初始环境、解决方案、效果、举例、末态环境、推理、其他有关模式、已知应用。
- ✧ 设计模式主要有 23 种，可以将这些模式划分为三大类型：创建型、结构型和行为型。
- ✧ 创建型包括：单例模式、工厂方法模式、抽象工厂模式、建造者模式和原型模式。
- ✧ 结构型包括：代理模式、装饰模式、适配器模式、组合模式、桥梁模式、外观模式和享元模式。
- ✧ 行为型包括：模板方法模式、命令模式、责任链模式、策略模式、迭代器模式、中介者模式、观察者模式、备忘录模式、访问者模式、状态模式和解释器模式。

练　习

1. 设计模式起源于_____。

   A. 机械设计　　B. 建筑工程设计

   C. 水利工程设计　　D. 工业电力化设计

2. 下面不属于设计模式的要素的是_____。

   A. 模式名称　　B. 问题　　C. 因素　　D. 环境

3. 目前有_____种设计模式，分为_____大类型。

   A. 23，3　　B. 24，3　　C. 3，23　　D. 3，24

4. 设计模式分为_____、_____和_____三大类型。

5. 简述设计模式的概念。