工业和信息化“十三五”
人才培养规划教材

Java 高级程序设计|实战教程

Java Advanced Program Design

戴远泉 李超 秦争艳 ◎ 主编
董慧慧 麦琪琳 ◎ 副主编

用**项目**、**任务**讲解高级程序设计核心知识

精心选取知识点、技术点，**结构体系完整**

采用“**应用场景** - **相关知识** - **实现过程** - **拓展知识** - **拓展训练**”方式讲解

实例源于**真实项目**，类的设计符合 Java 编程思想

综合实训讲解**完整项目**需求分析 + 系统分析 + 系统设计 + 编码 + 测试

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

工业和信息化“十三五”
人才培养规划教材

Java 高级程序设计|实战教程

Java Advanced Program Design

戴远泉 李超 秦争艳 ◎ 主编
董慧慧 麦琪琳 ◎ 副主编

人 民 邮 电 出 版 社
北 京

图书在版编目（CIP）数据

Java高级程序设计实战教程 / 戴远泉，李超，秦争艳主编. -- 北京 : 人民邮电出版社，2018.10
工业和信息化“十三五”人才培养规划教材
ISBN 978-7-115-48965-4

Ⅰ. ①J… Ⅱ. ①戴… ②李… ③秦… Ⅲ. ①JAVA语言－程序设计－高等学校－教材 Ⅳ. ①TP312.8

中国版本图书馆CIP数据核字(2018)第168368号

内 容 提 要

本书是在读者初步掌握 Java 的基础知识和技能之后，进一步学习 Java 高级编程的指导用书，主要内容包括 Java 编码规范、Java 集合框架、Java 反射机制、Java 泛型机制、Java 序列化机制、Java 多线程机制、Java 网络编程、Java 数据库编程、综合实训等。本书能够帮助读者逐步领会 Java 的编程思想，并掌握 Java 的编程技能，为进一步学习 J2EE 框架技术奠定扎实的基础。

本书可作为应用型本科和高职高专院校计算机科学与技术、软件工程、软件技术等专业学生学习“Java 高级程序设计”课程的教材及 Java 应用开发培训班的教材，也可作为 Sun 公司 SCJP Java 认证考试的辅导用书。

◆ 主　　编　戴远泉　李　超　秦争艳
副 主 编　董慧慧　麦琪琳
责任编辑　桑　珊
责任印制　马振武

◆ 人民邮电出版社出版发行　北京市丰台区成寿寺路 11 号
邮编　100164　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
固安县铭成印刷有限公司印刷

◆ 开本：787×1092　1/16
印张：16.25　2018 年 10 月第 1 版
字数：451 千字　2018 年 10 月河北第 1 次印刷

定价：46.00 元

读者服务热线：(010)81055256　印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京东工商广登字 20170147 号

前言
Foreword

Java 是由 Sun Microsystems 公司于 1995 年 5 月推出的程序设计语言和 Java 平台的总称。

以 Sun 公司公布的 Java 框架结构为标准，Java 语言以 Java2 为中心，可分为以下 3 个组成部分。

（1）企业版 J2EE。该版本是以各大企业环境为中心而开发的一种以应用程序为主体的计算机网络平台，其中还包括 EJB、JSP 和 Servlet 3 个层次。

（2）标准版 J2SE。该版本中，Java 核心编程为图形用户界面的编程、工具包程序的编写以及数据库的程序编写。

（3）微型版 J2ME。该版本一直以消费品和各种嵌入式设备的网络应用平台为研究中心，主要涉及的领域为手机、手机中的各种无线游戏等，其核心技术为移动信息设备小程序。

Java 语言已是目前世界上流行的高级编程语言之一，正被广泛应用于计算机软件的开发，尤其是 Web 领域。自诞生以来，Java 迅速成为开发互联网应用程序的首选编程语言。

当前，应用型本科和高职高专院校开设的“Java 语言程序设计”课程相对应的教材主要讲述 Java 语言的基本语法（包括 Java 语言基础、数据类型、Java 类和对象），而高一级的软件工程专业普遍开设的 J2EE 课程对应的教材主要内容是 Servlet、JSP、SSH（Struts+Spring+Hibernate）及 SSM（Spring+SpringMVC+Mybatis）等企业级应用。从“Java 语言程序设计”到“J2EE 框架技术”等高级应用开发之间存在空白区，在课程开设及教学中，各门课程不能环环相扣，存在脱节现象。为了填补此空白区，本书在 Java 基础知识之上讲解了 Java 的高级技术和在实际 Java 项目的开发中所需的知识及其应用实例。在 Java 基础和 J2EE 应用之间起到了承前启后的作用。

本书特色如下。

（1）结构体系完整：本书体系完整，设计了 9 个应用领域，每个应用领域都是在实际软件开发中重要的或是频繁使用的知识点。

（2）实例源于真实：本书每个应用领域知识点对应的实例都源于或接近于真实项目，类的设计符合 Java 编程思想。

（3）讲解循序渐进：本书中对涉及的每个应用领域的知识点的讲解都由浅入深、循序渐进地展开。

（4）符合认知规律：本书采用“应用场景—相关知识—任务实施—拓展知识—拓展训练”的方式进行知识点的讲解，并配有课后小结、课后习题和上机实训。

本书内容如下。

本书设计了 9 个应用领域，每个应用领域的知识点都是在实际软件项目中得到大量应用的。

知识领域 1：Java 编码规范，讲解如何编写出符合规范、规则、惯例和模式的代码。

知识领域 2：Java 集合框架，讲解 List、Set 和 Map 等集合的使用。

知识领域 3：Java 反射机制，讲解 Java 反射机制的相关知识和应用。

知识领域 4：Java 泛型机制，讲解泛型的相关知识及应用，包括泛型类、泛型方法和泛型接口。

知识领域 5：Java 序列化机制，讲解序列化的相关知识及应用，包括对象序列化和 JSON 序列化。

知识领域 6：Java 多线程机制，讲解多线程的相关知识及应用，主要包括线程的创建和启动、线程的生命周期、线程的管理等。

知识领域 7：Java 网络编程，讲解网络编程的相关知识及应用，包括基于 URL 读取网页内容、基于 TCP 编程和基于 UDP 编程。

知识领域 8：Java 数据库编程，讲解基于 Java 的数据库编程，包括数据库访问技术、JDBC 连接数据库、执行 SQL 操作。

知识领域 9：综合实训，讲述了一个完整的实训项目——“餐饮管理系统”，使用软件工程的思想进行需求分析、系统分析、系统设计、编码、测试等过程，完成本项目。

本书由戴远泉、李超、秦争艳任主编，董慧慧、麦琪琳任副主编，书中的每个例程代码都经过反复调试和测试。

由于编者水平有限，书中难免存在疏漏之处，欢迎广大读者批评指正。

编者

2018 年 5 月

目录
Contents

PART01

知识领域1
Java编码规范

知识目标

理解Java编码规范，包括：文件后缀名、Java源文件样式约定、注释规范、命名规范等。

■ 能力目标

1. 熟练阅读Java源码。
2. 熟练使用Java编码规则编写Java代码。

■ 素质目标

1. 培养查阅科技文档的能力。
2. 培养团队协作的能力。

1.1 应用场景

一个大型的软件项目是由一个团队来完成的，每个程序员在给包、类、变量、方法取名的时候，如果根本没有一点约定，只是随心所欲，可能会带来一系列问题。

编码规范是程序编码所要遵循的规则，保证代码的正确性、稳定性、可读性。规范编码有以下作用。

1. 规范的代码可以促进团队合作

一个项目大多都是由一个团队来完成的，统一的风格使得代码可读性大大提高，可以让开发人员尽快而彻底地理解新的代码，最大限度地提高团队开发的合作效率。

2. 规范的代码可以减少 BUG（漏洞）处理

没有对输入输出参数的规范，没有规范的异常处理，没有规范的日志处理等，不但会导致我们总是出现类似空指针这样低级的 BUG，而且还很难找到引起 BUG 的原因。相反，在规范的开发中，BUG 不但可以有效减少，查找 BUG 也变得轻而易举。

3. 规范的代码可以降低维护成本

好的编码规范可以尽可能地减少一个软件的维护成本，并且几乎没有软件能在其整个生命周期中均由最初的开发人员来维护。

4. 规范的代码有助于代码审查

代码审查可以及时纠正一些错误，可以对开发人员的代码规范做出监督。代码规范不仅使得开发统一，减少审查监督，而且让代码审查有据可查，大大提高了审查效率和效果，同时代码审查也有助于代码规范的实施。

5. 养成代码规范的习惯，有助于程序员自身的成长

长期的规范性编码还可以让开发人员养成好的编码习惯，甚至锻炼出更加严谨的思维习惯。规范的代码更有利于帮助开发人员理解开发语言、理解模式、理解架构，能够帮助开发人员快速提升开发水平。

因此，程序设计的标准化非常重要，原因在于这能提高开发团队各成员的代码的一致性，使代码更易理解，这意味着更易于开发和维护，从而降低了软件开发的总成本。为实现此目的，和其他语言类似，Java 语言也存在非强制性的编码规范。

1.2 相关知识

Java 常见编码规范包括：文件后缀名、源文件样式约定、注释规范、命名规范等。定义此种规范的目的在于让项目中所有的文档格式的统一，增加可读性。

1.2.1 文件后缀名

这部分列出了常用的文件类别及其后缀名，如表 1-1 所示。

表1-1 Java程序使用的文件后缀名

文件类别	文件后缀名
Java 源文件	.java
Java 字节码文件	.class

其中两者最本质的区别在于，.java 文件是供虚拟机运行时执行的文件，而 .class 文件可以让你在任何一台安装了 Java 虚拟机的机器上运行。

1.2.2 源文件样式约定

Java 源文件必须按顺序由以下 3 部分组成。

- 版权信息。
- 包和引入语句。
- 类和接口声明。

1. 版权信息

版权和版本信息必须在 Java 文件的开头，其他不需要出现在 Javadoc 的信息也可以包含在这里。

例如：

```
/**
*Title:  确定鼠标指针位置类
* Description: 确定鼠标指针当前在哪个作业栏位中并返回作业号
* @Copyright: Copyright (c) 2017
* @Company: daiinfo
* @author: daiyuanquan
* @version: 1.0
*/
```

2. 包和引入语句

package 行要在 import 行之前，import 中标准的包名要在本地的包名之前，而且按照字母顺序排列。如果 import 行中包含了同一个包中的不同子目录，则应该用 * 来处理。

例如：

```
package com.hbliti.net.stats;
import java.io.*;
import java.util.Observable;
import com.hbliti.util.Application;
```

3. 类和接口声明

每个 Java 源文件都包含一个单一的公共类或接口。类或接口的各部分代码顺序如下：

（1）常量声明；
（2）静态变量声明；
（3）成员变量声明；
（4）构造函数部分；
（5）Finalize 部分；
（6）成员方法部分；
（7）静态方法部分。

表 1-2 描述了类和接口声明的各个部分以及它们出现的先后次序。

表1-2 类和接口声明的各个部分顺序

序号	类/接口声明的各部分	说明
(1)	类 / 接口文档注释 (/**……*/)	该注释中所需包含的信息
(2)	类或接口的声明	类头
(3)	类 / 接口实现的注释 (/*……*/) // 如果有必要的话	该注释应包含任何有关整个类或接口的信息，而这些信息又不适合作为类 / 接口文档注释
(4)	常量	公共的、静态的、不可改变的，必须赋初始值 (一旦赋值，不可改变)
(5)	类的静态成员变量	首先是类的公共变量，随后是保护变量，再后是包一级别的变量 (没有访问修饰符)，最后是私有变量
(6)	成员变量	首先是公共级别的，随后是保护级别的，再后是包一级别的 (没有访问修饰符)，最后是私有级别的
(7)	构造函数	应该用递增的方式写 (比如：参数多的写在后面)
(8)	成员方法	这些方法应该按功能，而非作用域或访问权限分组。例如，一个私有的类方法可以置于两个公有的实例方法之间。其目的是为了更便于阅读和理解代码

(1) 类 / 接口文档注释举例如下所示。

```
/**
* Title: 文件名称
* Description: 类内容的简介
*更新记录:
* 格式:[更新日期][修改的版本][操作人]内容
 * [2017-06-28][1.0][戴远泉]完善create方法。<br>
*
* Copyright: Copyright (c) 2017
* Company:   Daiinfo Co. Ltd.
* @version: 1.1
*/
```

(2) 类或接口的声明举例如下所示。

```
public class CounterSet extends Observable implements Cloneable{
…
…
 }
```

(3) 类 / 接口实现的注释。该注释应包含任何有关整个类或接口的信息，而这些信息又不适合作为类 / 接口文档注释。举例如下所示。

```
/**
* @see UserDao#save(User)
*/
public void save(User user) throws Exception{

}
```

(4) 类的 (静态) 变量。静态变量是基本数据类型，这种情况下在类的外部不必创建该类的实例就可以直接使用。举例如下所示。

```
class Value{
  static int c=0;
  Value(){
    c=15;
  }
  Value(int i){
    c=i;
  }
  static void inc(){
    c++;
  }
}
class Count{
  public static void prt(String s){
    System.out.println(s);
  }
    Value v=new Value(10);
    static Value v1,v2;
    static{
      prt("v1.c="+v1.c+"  v2.c="+v2.c);
      v1=new Value(27);
      prt("v1.c="+v1.c+"  v2.c="+v2.c);
      v2=new Value(15);
      prt("v1.c="+v1.c+"  v2.c="+v2.c);
    }
  public static void main(String[] args){
    Count ct=new Count();
    prt("ct.c="+ct.v.c);
    prt("v1.c="+v1.c+"  v2.c="+v2.c);
    v1.inc();
    prt("v1.c="+v1.c+"  v2.c="+v2.c);
    prt("ct.c="+ct.v.c);
  }
}
```

（5）成员变量举例如下所示。

```
/**
* Packet counters
*/
protected int[] packets;
```

public 的成员变量必须生成文档（JavaDoc）。protected、private 和 package 定义的成员变量如果名字含义明确的话，可以没有注释。

（6）构造函数举例如下所示。

```
public CounterSet(){
  this.size = 100;
}
public CounterSet(int size){
  this.size = size;
}
```

应该用递增的方式写（比如参数多的写在后面）。

（7）成员方法举例如下所示。

```
/**
* param r1 - ,,,,
* param r2 - ,,,,
* param r3
* param r4
*/
protected final void setArray(int[] r1, int[] r2, int[] r3, int[] r4)   throws Illeg-
alArgumentException{
    // Ensure the arrays are of equal size
,,,,
}
```

（8）toString 方法举例如下所示。一般情况下，每一个类都应该定义 toString 方法。

```
public String toString() {
...
}
```

1.2.3 注释规范

代码注释是架起程序设计者与程序阅读者之间的通信桥梁，可以最大限度地提高团队开发合作效率，也是提高程序代码可维护性的重要环节之一。所以我们不是为写注释而写注释。

1. 注释编写的原则

（1）注释形式统一。在整个应用程序中，使用具有一致的标点和结构的样式来构造注释。如果在其他项目中发现它们的注释规范与这份文档不同，按照这份规范写代码，不要试图在既成的规范系统中引入新的规范。

（2）注释内容准确简洁。内容要简单、明了、含义准确，防止注释的多义性，错误的注释不但无益反而有害。

2. 注释类型的基本划分

（1）基本注释必须要添加，包括以下几种。

- 类（接口）的注释；
- 构造函数的注释；
- 方法的注释；
- 全局变量的注释；
- 字段 / 属性的注释；

简单的代码做简单注释，注释内容不大于 10 个字即可，另外，持久化对象或 VO 对象的 getter、setter 方法不需加注释。

（2）特殊必加的注释包括以下几种。

- 典型算法必须有注释；
- 在代码不明晰处必须有注释；
- 在代码修改处加上修改标识的注释；
- 在循环和逻辑分支组成的代码中加注释；
- 为他人提供的接口必须加详细注释。

具体的注释格式自行定义，要求注释内容准确简洁。

3. 注释的格式

（1）单行（single-line）注释格式为“//……”。

（2）块（block）注释格式为“/*……*/”。

（3）文档注释格式为“/**……*/”。

（4）Javadoc 注释标签语法如下：

- @author　对类的说明，标明开发该类模块的作者；
- @version　对类的说明，标明该类模块的版本；
- @see　对类、属性、方法的说明，参考转向，也就是相关主题；
- @param　对方法的说明，对方法中某参数的说明；
- @return　对方法的说明，对方法返回值的说明；
- @exception　对方法的说明，对方法可能抛出的异常进行说明。

例如：构造方法注释如下。

```
public class OkButton extends Button {
  /**
   * 构造方法的描述
   * @param name
   *  按钮上显示的文字
   */
  public Test(String name){
     ……
  }
}
```

1.2.4　命名规范

命名指系统中对包名、目录（类名）、方法、常量、变量等标识符的命名。标识符的命名力求做到统一、达意、简洁，遵循驼峰法则。

统一是指对于同一个概念，在程序中用同一种表示方法。例如对于供应商，既可以用 supplier，也可以用 provider，但是我们只能选定一个使用，至少在一个 Java 项目中保持统一。

达意是指标识符能准确地表达出它所代表的意义，如 newSupplier，OrderPaymentGatewayService 等；而 supplier1、service2、idtts 等则不是好的命名方式。

简洁是指，在统一和达意的前提下，用尽量少的标识符。如果不能达意，宁愿不要太简洁。例如，theOrderNameOfTheTargetSupplierWhichIsTransfered 太长，transferedTargetSupplierOrderName 则较好，但是 transTgtSplOrdNm 就不好了。省略元音的缩写方式不要使用，我们的英语往往还没有好到看得懂奇怪的缩写。

用驼峰法则是指单词之间不使用特殊符号分割，而是通过首字母大写来分割。例如推荐用 SupplierName，addNewContract，而不是 supplier_name，add_new_contract。

1. 包名命名规范

包名按如下规则组成。

[基本包].[项目名].[模块名].[子模块名]……

例如：com.czpost.eims

```
com.hepost.eims.until...
```

不得将类直接定义在基本包下，所有项目中的类、接口等都应当定义在各自的项目和模块包中。

2. 类名命名规范

（1）首字母大写。

类名要首字母大写，例如可以用 SupplierService，PaymentOrderAction；不要用 supplierService，paymentOrderAction。

（2）添加有含义的后缀。

类名往往用不同的后缀表达额外的意思，如表 1-3 所示。

表1-3　类名命名后缀名含义

后缀名	意义	举例
Service	表明这个类是个服务类，里面包含了给其他类提供业务服务的方法	PaymentOrderService
Impl	这个类是一个实现类，而不是接口	PaymentOrderServiceImpl
Inter	这个类是一个接口	LifeCycleInter
Dao	这个类封装了数据访问方法	PaymentOrderDao
Action	直接处理页面请求，管理页面逻辑的类	UpdateOrderListAction
Listener	响应某种事件的类	PaymentSuccessListener
Event	这个类代表了某种事件	PaymentSuccessEvent
Servlet	一个 Servlet	PaymentCallbackServlet
Factory	生成某种对象工厂的类	PaymentOrderFactory
Adapter	用来连接某种以前不被支持的对象的类	DatabaseLogAdapter
Job	某种按时间运行的任务	PaymentOrderCancelJob
Wrapper	这是一个包装类，为了给某个类提供没有的能力	SelectableOrderListWrapper
Bean	这是一个 POJO	MenuStateBean

3. 方法名命名规范

方法的命名规范有：首字母小写，如使用 addOrder()，不要用 AddOrder()；动词在前，如使用 addOrder()，不要用 orderAdd()。

动词前缀往往表达特定的含义，如表 1-4 所示。

表1-4　方法名命名后缀名含义

前缀名	意义	举例
create	创建	createOrder()
delete	删除	deleteOrder()
add	创建，暗示新创建的对象属于某个集合	addPaidOrder()
remove	删除	removeOrder()
init	初始化，暗示会做些诸如获取资源的特殊动作	initializeObjectPool

续表

前缀名	意义	举例
destroy	销毁，暗示会做些诸如释放资源的特殊动作	destroyObjectPool
open	打开	openConnection()
close	关闭	closeConnection()<
read	读取	readUserName()
write	写入	writeUserName()
get	获得	getName()
set	设置	setName()
prepare	准备	prepareOrderList()
copy	复制	copyCustomerList()
modity	修改	modifyActualTotalAmount()
calculate	数值计算	calculateCommission()
do	执行某个过程或流程	doOrderCancelJob()
dispatch	判断程序流程转向	dispatchUserRequest()
start	开始	startOrderProcessing()
stop	结束	stopOrderProcessing()
send	发送某个消息或事件	sendOrderPaidMessage()
receive	接受消息或时间	receiveOrderPaidMessgae()
respond	响应用户动作	responseOrderListItemClicked()
find	查找对象	findNewSupplier()
update	更新对象	updateCommission()

find方法在业务层尽量表达业务含义，例如使用 findUnsettledOrders()，表达查询未结算订单，而不要使用findOrdersByStatus()。数据访问层，find、update等方法可以实现对数据表的Select（查询）、Insert（插入）、Update（更新）、Delete（删除）等操作，如findBySta－tusAndSupplierIdOrderByName(Status.PAID, 345)。

4. 常量命名规范

常量必须为大写单词，下划线分隔的命名方式。常量一定是 static final 的字段，但是不是所有的 static final 字段都是常量，例如：

```
static final int NUMBER = 5;
static final ImmutableList<String> NAMES = ImmutableList.of("Ed", "Ann");
```

5. 变量命名规范

非常量的变量（类变量和实例成员变量）名必须采用小写单词驼峰命名方式（lowerCamelCase）。变量命名通常使用名词和名词短语，如 computedValue、index。

1.3 任务实施

任务 计算员工的月工资

1. 任务需求

某公司分为多个部门，每个部门有一个经理和多个员工，每个员工根据职称发基本工资。员工的工资由基本工资、日加班工资、日缺勤工资等组成。具体需求如下所示。

- 员工的基本信息，包括部门、职务、职称以及工资记录等信息。
- 能记录员工的每一个职称信息，并授予相应的职称，系统在计算员工工资的时候选取职称对应的最高职称津贴。

2. 任务分析

问题域中涉及多个类，包括职员类 Staffer、经理类 Manager、测试类 TestEmployee。

- Staffer 类：通过此类封装定义计算职员基本工资方法。
- Manager 类：通过此类封装定义计算经理基本工资方法。
- TestEmployee 类：调用方法并实现结果输出。

其类图关系如图 1-1 所示。

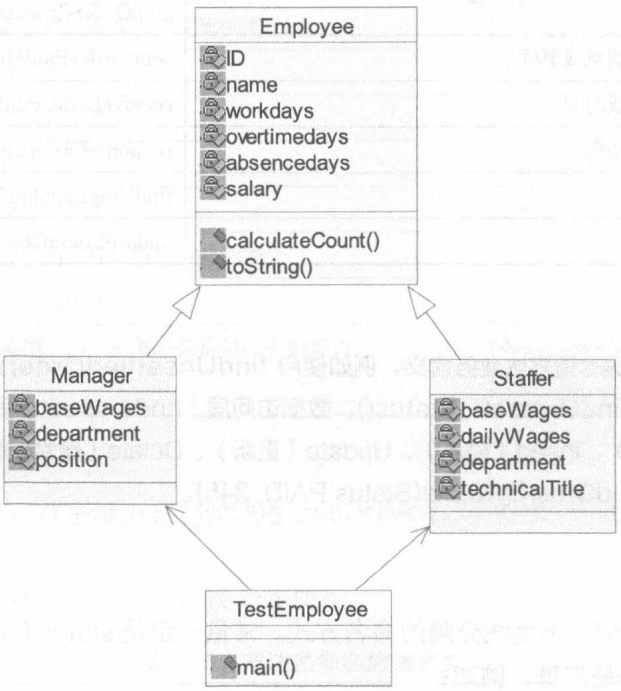

图1-1 类图关系

3. 任务实现

```
package com.daiinfo.seniorjava.ken1.implment;
/**
 *封装员工的信息和操作
```