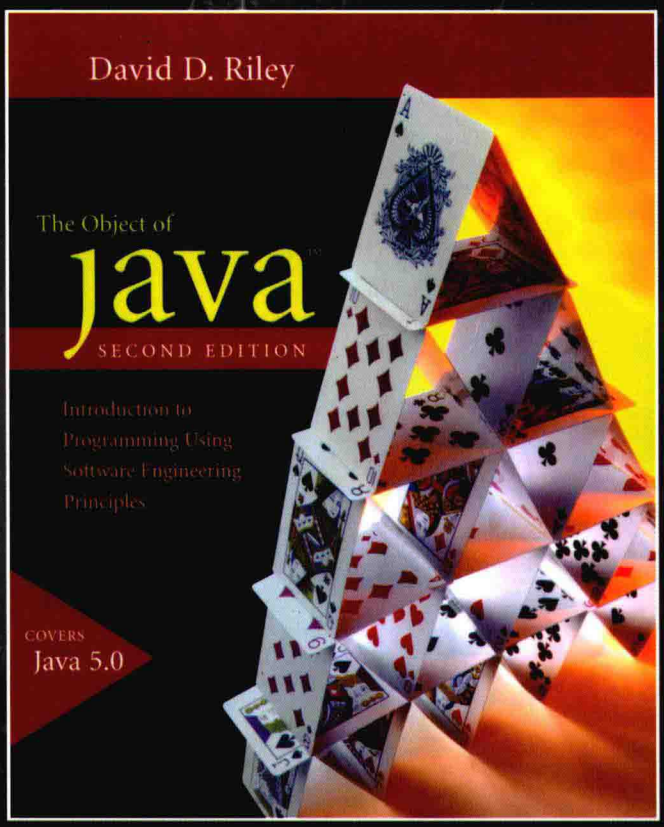
计 算 机 科 学 丛 书

原书第2版

Java程序设计

对象和软件工程方法

（美） David D. Riley 著　　苏钰涵 徐红梅 王琦 等译



The Object of Java

Introduction to Programming Using Software Engineering Principles

Second Edition

机械工业出版社
China Machine Press

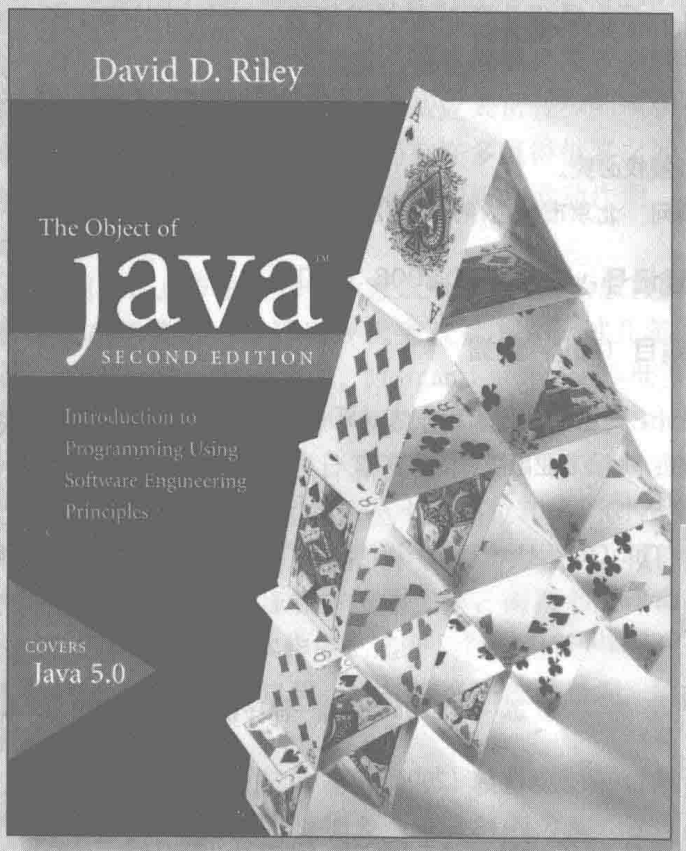
计 算 机 科 学 丛 书

原书第2版

Java程序设计

对象和软件工程方法

（美）David D. Riley 著　　苏钰涵　徐红梅　王琦　等译



The Object of Java

Introduction to Programming Using Software Engineering Principles

机械工业出版社
China Machine Press

本书采用“以对象为中心”的教学方法，介绍Java编程的技巧和面向对象方法。详细介绍对象和类、设计与实现、方法、基本数据类型、供应者类、 逻辑和选择、 继承、多态、重复、容器、数组、文件输入和输出、递归以及应用和Applet等内容。本书每章最后附有大量练习和编程练习，可以帮助读者巩固书中概念，注重重点难点，加深理解。附带光盘内容丰富、包括软件开发工具、案例分析模型、实例代码等。

本书适合作为计算机科学专业编程导论课程的教材或参考书。

本书版权登记号：图字：01-2006-3150

图书在版编目（CIP）数据

Java程序设计：对象和软件工程方法（原书第2版）/（美）莱利（Riley, D.）著；苏钰涵等译. －北京：机械工业出版社，2007.2

(计算机科学丛书)

书名原文：The Object of Java: Introduction to Programming Using Software Engineering Principles

ISBN 978-7-111-19989-2

Ⅰ.J… Ⅱ.① 莱… ② 苏… Ⅲ.Java语言－程序设计－教材 Ⅳ.TP312

中国版本图书馆CIP数据核字（2006）第151157号

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：陈冀康

北京诚信伟业印刷有限公司印刷 · 新华书店北京发行所发行

2007年2月第 1 版第 1 次印刷

184mm × 260mm · 28.5印张

定价：59.00元（附光盘）

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线（010）68326294

译者序

本书给我们的第一印象是这是一本关于Java的教材，但是等我们翻译完整本书之后，才领会到作者为什么会把它取名为The Object of Java，更确切地讲，作者是在利用Java谈对象，让读者乘着Java这艘飞船在面向对象的宇宙中遨游。

从内容的组织上看，本书与一般Java书也有所不同。作者首先简单介绍对象和类的概念，又从总体讲解设计与实现的基本问题。接下来才谈到基本类型、方法、逻辑和选择等程序设计语言的基本要素，并在其中穿插介绍继承、多态等面向对象概念。也许我们已经习惯于“语法－简单例子－复杂应用”的介绍模式，看到这种组织方式，一方面有耳目一新的感觉，另一方面也不免有些疑惑：为什么这么安排呢？

这个问题作者是靠事实来回答的，本书的手稿最初是作者所在学院一门计算机课程的教案，由于它的内容组织完全贴合教学，取得很大成功，促使本书第1版得以诞生；又由于第1版的畅销，加之Java新版本的出现，我们又有幸看到它的第2版，就是现在这本书。

本书吸引我们的不光是内容上的合理取舍、结构的精巧组织，还在于作者采用饶有趣味的示例来加深读者的理解。另外从随处可见的“软件工程提示”、“关闭的黑箱”、“打开黑箱”，以及每章最后的“Java检查员”，都能看出作者将他的经验、智慧通过这样的点点滴滴呈现给我们，确实让人备觉珍贵。

全书主要由苏钰涵、徐红梅、王琦翻译，其他参与翻译的人员还包括刘跃邦、王林绪、潘淼、王宇飞、王树春、韦群、林华君、赵蓓、刘立强、王志淋、蔡洪量、何跃强、丁小峰、苏金国、孙春娟、周兴汉、刘名臣、张练达等，全书由王林绪、潘淼等进行术语的审核，何跃强、丁小峰等提供技术问题支持，全体工作人员共同完成本书的翻译工作。本书最后由苏钰涵统稿。

我们深深地感谢我们的家人和朋友。在翻译过程中，他们给予我们莫大的关心、支持和帮助。

由于时间仓促，译者的水平有限，在翻译过程中难免会出现一些错误，恳请读者批评指正。

前　言

Java已成为现代软件开发的一个得力工具。因此，我们将这本书也精心设计为指导现代软件开发的一个工具。不论是Java还是本书，二者都有着面向对象范式所赋予的强大能力，并适当地结合可靠的软件工程实践方法。本书的目标是介绍编程技术、面向对象技术、软件工程技术以及Java技术，通过掌握这些技术，可以为将来在计算机科学领域的学习打下坚实的基础。

本书面向计算机科学专业的学生，可以作为编程导论课程（IEEE/ACM Curriculum 2001称这些课程为CS101或CS111）的教材。本书不要求读者以前写过计算机程序，重要的是要有一定程度的分析综合能力。只要学过三年的高中数学，就完全可以理解本书使用的记法。

本版的新特点

使用AWT和Swing

本书第1版主要依赖于作者提供的一个图形类类库。在Java 5中，使用标准Java类库就能非常容易地展示有关内容。基于这个变化，学生们不必再花时间去学习非标准的软件。本书仍然保留第1版采用的基本方法；本版还是会大量使用外部类，另外简单地介绍涉及GUI的应用，不过现在使用的是AWT和Swing类。

完全使用Java 5

本版充分结合Java 5的一些特性，在导论性质的编程课程中，这些特性极为重要。

泛型（Generics）：Java 5的一个重要改进就是引入了泛型数据类型。现在包和列表容器可以适当地定义为`Bag<ItemType>`和`SimpleList<ItemType>`，而不是`BagOfObject`和`SimpleListOfObject`。这就能提升数据容器的适用性，并且能显著提高类型安全性。

自动装箱（Autoboxing）：本版会展示Java 5对自动装箱和自动拆箱的支持。我们还是会介绍包装器类，不过由于Java 5中引入了自动装箱等重要的新语言特性，所以在包装器类上我们不再过多赘述。

枚举：许多编程语言中都有枚举数据类型，现在Java也开始支持枚举数据类型。在讨论如何编写你自己的类时，我们会介绍这个新的语言特性。另外还会介绍内置的`ordinal`和`name`方法，许多代码示例中都会用到枚举类型。

for循环：for循环已经得到改进，提供一种特殊版本来迭代处理集合。这种形式的循环是表述简单容器处理的一种更高级的方式。我们将针对列表和数组来讨论该版本的for语句。另外还会讨论这种特殊版本for语句的限制。

assert语句：以assert语句形式提供可测试的断言，这是一个重大的语言改进。在此会讨论assert语句，另外第7章介绍逻辑和选择时还会讨论assert对调试有何作用。

Scanner：`java.util.Scanner`是一个新增的类，它采用一种改进的类型安全方式，可以将文本形式的基本类型直接量转换为基本类型值。本书利用这个类来显示如何完成String、JTextField和文件的转换。使用Scanner可以更方便地处理基本类型的GUI输入。

更多地注重以对象为中心

本书的核心一直是面向对象。为了更清楚地反映这一点，第1章首先提供一些基本的例子，展示多个类的使用。这里我们没有举传统的“hello world”例子，而是分析一个模拟机器人小车的程序，这个程序更具面向对象特性。另外对第1版中的第2章和第3章也做了修改，阐述如何在适当的面向对象（OO）上下文中应用软件工程。

改进正交性

我们会竭尽所能让书中的示例类与Java标准一致。在此对文档稍有修改，从而能更好地用javadoc工具显示。包和列表类现在继承自标准Java接口，仍使用同样的`hasNext`、`next`、`add`和`remove`方法。前置条件和后置条件已经使用UML对象约束语言（Object Constraint Language）记法进行了重写。

更多完整的程序

要通过例子来学习编程。本书第1版的特点之一是其中包括70多个完整的程序。在本版中，完整的程序超过80个。其中新增的一些例子相当长，完全可以作为案例分析使用。

面向对象和Java

按James Gosling和Henry McGilton所说，“要想在日渐复杂的基于网络的环境中发挥作用，编程系统必须适应面向对象概念”。把重点放在面向对象上，这不仅要求要有一个“对象为先”的课程；还需要一种以对象为中心的方法。软件类、方法、继承和事件驱动代码在当今编程系统中应该像变量、循环和数组一样平常。以对象为中心的方法并不只是罗列对象术语，它寻求的目标是让软件开发人员能够采用面向对象思维。

以对象为中心的方法的优势之一在于它能兼容原来的命令式和函数式范式。要使用面向对象编程（object-oriented programming，OOP），程序员仍然必须编写赋值指令、传递参数、从函数（非void方法）返回值，还要熟练掌握所有基本控制结构。这说明，学生从OOP转向其他编程范式时，不会像从其他编程范式转向OOP时那样困难。

多年以来，我们一直在编程导论课程中使用OO方法，并积累了很多经验，本书的思想和方法就是基于这些经验得来的。最近，教师们已经开始使用Java编程语言。Java很适合入门课程，因为它采用一个合理的对象模型实现，而且不论是在学术界还是专业领域中Java都很常用。Java还有一个突出的好处，更使它增色不少：它采用类似于C的记法。这对于将来可能使用C、C++或C#的学生很有帮助。正如Gosling和McGilton所说，“Java编程语言的最初设计就是面向对象的”。

强调软件工程

就像完成好的作品一样，要想编写好的程序，需要有一定的技能，还要有一些准则。软件工程的指导原则和技术对于培养这种技能、建立有关准则至关重要。可以清楚地看到，本书自始至终都以软件工程为重点，特别是在阐述正确的软件工程技术时涵盖以下特点。

软件工程提示

各章分布有许多软件工程提示。这些软件工程提示提供了一系列软件开发人员“最佳实践”。软件工程提示可能建议一个语言构造采用什么格式来得到好的编程风格，也可能解释有经验的程序员如何解决一个常见的设计问题。

契约式编程

OOP充分展现了规范的重要性。方法的前置条件和后置条件以及类不变式对于表述代码的行为特别重要。我们会不断使用这些断言来描述例子，并定义示例类。契约式编程将在第2章介绍，并在后面经常使用。另外还讨论了逻辑表达式、循环不变式和特殊的断言记法。这种形式的类规范都以HTML格式提供。

模式

软件工程的设计模式原则提醒我们：软件开发技巧通常基于我们对常用结构的记忆。本书拓展了这种思想，在此包括经常遇到的代码表达式、指令和算法的模式以及作为编程模板的设计模式。本书强调这些模式是为了使读者能熟悉经常需要的解决方案，并了解如何使用以及何时使用。

软件测试

测试在所有好的软件工程模型中都是不可或缺的。我们专门用一些章节介绍测试，确保读者掌握基本的调试技能，并了解简单路径测试和黑盒测试。

Java检查员

软件工程研究得出一个结论：最有用的测试形式是非正式的桌面检查、代码审查和走查(walkthrough)。这些实践要求对“检查什么”、“审查什么”以及如何“走查”有基本的了解。每一章最后都有“Java检查员”一节来提供这些信息，这些小节不仅回顾各章的要求，还提供一些实用的解决策略（即“如何解决”）。

UML

图在面向对象设计和编程中起着关键的作用。本书穿插大量对象图来说明计算的运行时特性。还加入很多类图展示一个类的可见接口，并通过图来描述各个类之间的关系。另外，还提供了活动图显示控制流。

为了遵循软件工程主题，本书严格采用统一建模语言（Unified Modeling Language，UML）作为绘图记法。通过使用UML图，可以为学生展示已经在软件开发行业成为标准的记法。书中使用UML的一个子集，这在附录E中做了总结。

内容组织

我们在CS1课程中讲授OOP，并从中得到许多教训。无可置疑，最重要的一个发现就是：要以适当的顺序组织内容，这一点相当重要。当前，编程课程的讲授时间往往很少；每一分钟都不能浪费。我们做了很多尝试，根据不同的体验，最后以最利于学习的顺序来组织本书内容。

你可能会注意到，本书的目录很特别。由于我们采用以对象为中心的方法，这要求尽可能早地介绍关键的面向对象内容，以便在书中适当地应用。

第1章首先对类和对象进行简要的概述，概要介绍将要讨论的要点。该章使用一个面向对象示例程序来展示一些基本的记法和软件开发工具（编辑器、编译器和虚拟机），使读者对软件开发的一般过程有一个简单的认识。这一章的内容完全可以用一个课时讲完。

以对象为中心

第2章首先介绍基本的面向对象构造：方法调用。该章展示如何在对象上执行方法，然后分析这些指令的简单序列。我们还介绍对象的声明、实例化和赋值，并提供一个有助于展示对象绑定的代码模式：交换算法。该章还谈到编写初始程序所需的所有Java工具。

早期O-O策略

本书只有第3章混合了多方面的内容，其中包括一些基本软件设计技巧，还介绍一些额外的语言功能。在该章中，读者会看到在OOP上下文中解决编程问题的一种原型策略。假设每个学生刚开始都要写一些程序，原型策略有助于回答“从哪里开始”之类的问题。第3章还讨论了其他一些基本的软件开发技巧，如选择合适的标识符名，以及使用输出指令（`System.out.println`）来帮助调试。

方法

方法调用是OOP的主要指令，在第1章和第2章引入，并在第4章详细讨论。第4章还介绍如何创建方法，包括参数传递、局部变量和非void方法等问题。

基本类型

只有当读者对前4章中的对象、类和方法等内容有了一定的认识，我们才在第5章转向基本数据类型。我们发现，稍晚一些讨论基本类型（如数值表达式）可以避免早期的分心。通过这种方法，使得读者在遭遇Java采用非面向对象方式处理数字等问题之前，能相当轻松地处理引用数据。稍晚介绍基本类型还有一个好处，可以把基本类型变量看作是面向对象环境中的异类（尽管是合理的异类）。以后不会再花时间介绍基本类型表达式，这非常有利于学生熟悉引用数据/对象。

编写供应者类

在编写利用其他类的客户代码和编写这些外部类的供应者端代码之间看来存在显著的认知差异。第6章就是要帮助读者划清这个界限。该章不仅指出有关的设计决策，如一个特定方法要放在哪个类中，而且分析了适当的信息隐藏和封装的重要性。

控制结构

为什么不在第7章之前介绍选择和循环呢？答案是，较晚讨论这些控制结构能更早地讨论面向对象主题，因此能更多地在课程中使用面向对象。由于较晚地介绍控制结构，之前会很明显看出需要大量使用if指令和while循环等控制结构，相应地学生们会急切地学习这些有用的内容。在面向对象模型中，控制结构不再是编程技巧中的核心部分。不过，选择和重复仍然是很重要的编程工具，我们发现这样安排可以使学生充分接触有关内容，掌握这些构造。

较晚介绍选择和重复也是可能的，部分原因在于代码是事件驱动的。通过快速查看本书中的编程作业和示例，你就会相信，即使没有选择指令或循环，也完全有可能编写出有趣而且有难度的程序。

采用事件驱动编程还有其他一些好处。事件驱动代码与OOP很自然地相辅相成。我们可能都已经习惯了按钮、菜单和滚动条的行为（依赖于事件驱动控制）。实际上，对于伴随着图形用户界面成长起来的人来说，程序控制模型可能很糟糕。

继承

面向对象工具和技术的核心内容分3个阶段介绍：

1. 利用对象、类和方法（第1~4章）
2. 编写供应者端代码（第6章）
3. 使用继承（第8~9章）

较早介绍继承可能很好，但是我们发现，从大概课程一半的时候开始介绍继承效果最好，最为成功。采用这种时间安排，就能有足够的机会在许多课程项目中使用继承。

容器——包括数组

第11章力图把容器类的问题与数组的特定概念（第12章）区别开。列表固有的顺序特性使之比直接访问容器（如数组）更易于使用。更重要的是，列表是用面向对象方法实现容器（数据结构）的一个更好的例子。由于Java在语言中集成了数组记法，所以数组看上去与其他对象有所不同。第11章还涵盖泛型、Object类、包装器类等内容，并且讨论了一种很适用于列表的排序算法（插入排序）。

各章的顺序依赖关系

尽管在我们学院按照本书的顺序组织教学很成功，但是可能还存在许多其他因素，使得需要对顺序进行调整。因此，我们尽了最大努力，力争将各章之间的依赖关系减至最小。图0-1给出了各章之间的依赖关系。图中从A画至B的箭头指示：第B章很大程度上依赖于第A章中介绍的内容。

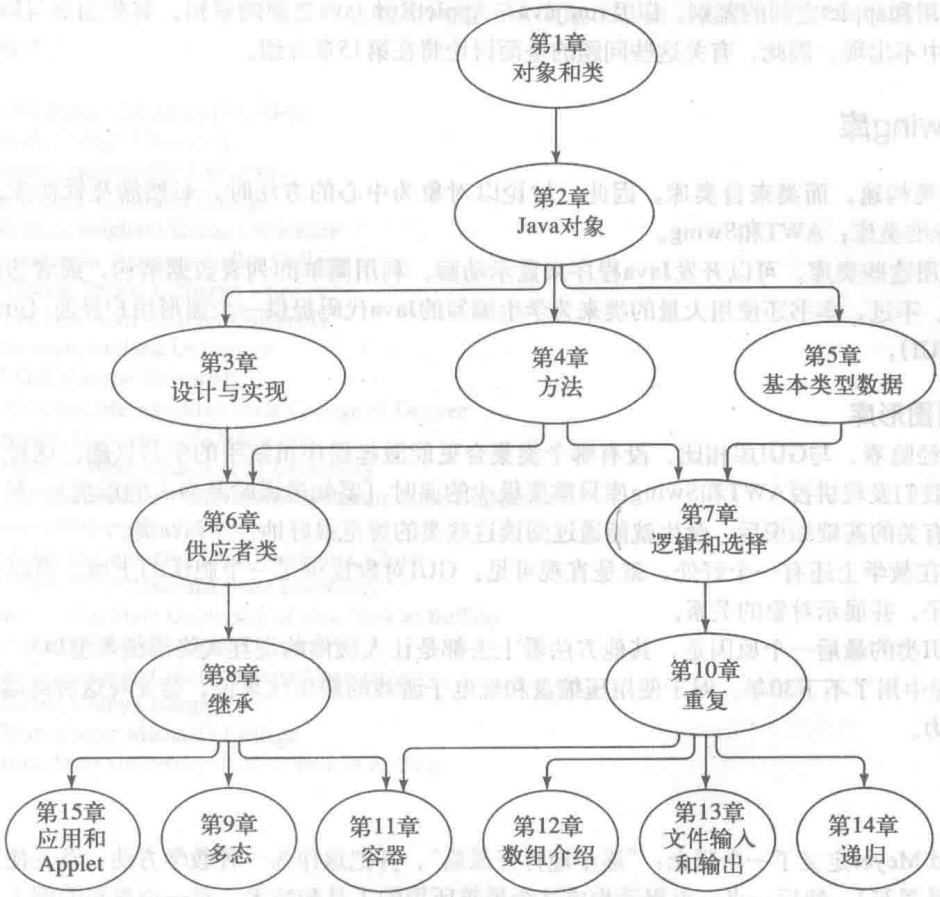


图0-1　各章的依赖关系

本书力图尽量全面地涵盖所有内容。在理想世界中，如果能把所有15章都在一门课里介绍完，那是再好不过了。不过这个任务很艰巨，我们怀疑大多数课程都可能会少讲或不讲后面的部分或全部章节。

对于想初步了解计算机硬件的人来说，本书提供了一个附录A。这个内容可以作为一个引言章节，或者可以作为课外阅读材料。之所以选择将这些内容放在附录中，是因为它不属于对象模型的范畴。

应用或Applet——由你来选择

本书中的程序都以一种灵活的方式编写，既可以作为Java应用执行，也可以作为Java applet执行。每个程序中都有一个名为Driver的类，这个类为程序定义了一个控制器对象。

以下所示的run.java 类可以把各个程序用作为一个应用。

```
public class run {
   public static void main(String args[]) {
      Driver driver = new Driver();
   }
}
```

如果你想把程序作为一个applet来运行，可以使用AppletRun.java 类。

```
import javax.swing.JApplet;
public class AppletRun extends JApplet {
    public void AppletRun() {
        Driver driver = new Driver();
    }
}
```

关于应用和applet之间的差别，以及run.java与AppletRun.java之间的差别，有关细节与Java有关，在前面的介绍中不出现。因此，有关这些问题的全面讨论将在第15章介绍。

AWT和Swing库

对象由类构建，而类来自类库。因此，讨论以对象为中心的方法时，必然涉及软件库。本书使用J2SE 5.0的标准类库：AWT和Swing。

通过使用这些类库，可以开发Java程序来显示动画、利用简单的列表数据结构，或者访问实际日历和时钟信息。不过，本书还使用大量的类来为学生编写的Java代码提供一个图形用户界面（graphical user interface，GUI）。

为什么使用图形库

从我的经验看，与GUI库相比，没有哪个类集合更能激起程序员新手的学习兴趣。这种动机如此强烈，以至于我们发现讲授AWT和Swing库只需要极少的课时（要知道课时是多么的珍贵）。在最初一到两个学时介绍有关的基础知识后，学生就能通过阅读这些类的规范很好地自学Java类。

图形类在教学上还有一个好处，就是直观可见。GUI对象提供了一个肥沃的土壤，可以很容易地建立对象的例子，并展示对象的关系。

使用GUI类的最后一个原因是，其他方法看上去都是让人厌倦的交互式终端流类型I/O，这些方法已经在CS1课程中用了不下30年。对于使用压缩盘和玩电子游戏的新生代来说，会发现这种终端I/O很别扭，很没有吸引力。

打开黑箱

Bertrand Meyer定义了一个概念："逐步地打开黑箱"，并把这作为一种教学方法，首先使用预定义的组件（也就是黑箱），然后一点一点揭示构成这个黑箱所用的工具和技术。举一个黑箱的例子，本书首先使用一个`run.java`静态类（前面已经提到）来实例化一个初始`Driver`对象，从而避免过早地为静态方法分心。我们让读者先不去管这个`run.java`类，直到后面真正开始解释静态方法，并显示`run.java` 类的代码时才会要求了解有关内容。

使用黑箱时，要求读者先要"认可"一个目标，为此在正文中会特别有一个"关闭的黑箱"小节，还会显示一个适当的图标。相应地，"打开黑箱"小节则指示了书中哪些位置上会揭示这些黑箱概念。

补充材料

所有读者都可以从http://www.aw.com/cssupport得到以下所有补充材料：

- 本书中所用示例程序的源代码（超过80个程序）。
- 对书中所分析的关键类/特性提供类图和规范（HTML形式）。

教师可以通过填写书后的《教学支持说明》来了解如何访问这些补充材料：

- 所有练习的一套完整答案。
- 编程练习示例解决方案的源代码。
- 对应各章内容的PowerPoint演示课件。

致谢

我非常感谢Barbara Barkauskas、Keith Burand和Kasilingam Periyasamy，感谢他们的信任、建设性的意见和无私的友谊。这些同事真的很棒，他们对这个项目信心十足（或者可能对其他教材有些失望），所以他们乐于采用本书第1版的手稿作为教材授课整整一年。这里要特别感谢Barb，因为她还答应作为本书的审校人员。她完全依照本书的内容讲授了这门课，因此能从特有的角度提供帮助。

我还要感谢Addison-Wesley为本书本版和第1版所集合的一群精兵强将，这些审校人员来自不同项目，

个个才能卓越。好的审校人员能探查内容、提出问题、给出建议并做出批评，这些学识渊博的计算机科学家们都做到了。

Thomas W. Bennet, Mississippi College
Glenn Blank, Lehigh University
Randy Bower, Jacksonville University
Jonathan Bredin, Colorado College
Robert Burton, Brigham Young University
Charles Costarella, Antelope Valley College
W. Sam Chung, Pacific Lutheran University
Eck Doerry, Northern Arizona University
Adrian German, Indiana University
Daniel P. Gill, Purdue University
Aaron J. Gordon, Metropolitan State College of Denver
Le Gruenwald, University of Oklahoma
H. Paul Haiduk, West Texas A & M University
Mark S. Hutchenreuther, California Polytechnic State University
Cerian Jones, University of Alberta
Michael A. Long, California State University, Chico
Blayne E. Mayfield, Oklahoma State University
Bina Ramamurthy, State University of New York at Buffalo
John M. Samaras, Valdosta State University
Carolyn J. C. Schauble, Colorado State University
Marc L. Smith, Colby College
John A. Trono, Saint Michael's College
Phil Ventura, State University of New York at Buffalo

Dave Riley

致 学 生

为什么你决定选这门课？你想成为一名计算机科学家吗？你是否想要提高你的分析问题和解决问题的技能？你是否想尝试一下看能不能把计算机编程作为你将来的职业？

不论出于什么原因，都应该认识到重要的一点，计算机科学的生命源泉就是软件开发。正因如此，很多计算机科学课程专门研究如何开发软件。本书就是要通过介绍软件开发将你引入计算机科学大门。

学习软件开发是一个需要众人参与的运动。临渊羡鱼，不如退而结网。如果只是读诗，你是学不会写诗的，如果只是读本书，或者只是听老师讲，你就无法学会如何创建软件。要亲手去做，亲自尝试。你会发现每章最后都建议你做一些编程作业。要做好准备，每章通过这样一些作业至少要完成一个程序。

解决问题是软件开发的核心。你的问题解决技能会随着开发软件的实践而提升。有时可能让人很灰心，有时又会让人很有激情。计算机科学家能在解决问题中找到成功的喜悦。通向解决方案的道路越困难，程序写成时的成就感就越大。

软件工程提示

通过多年的研究，当今对软件开发的理解已经成熟，现在我们把最佳实践称为“软件工程”。软件工程原则和过程可能很复杂，特别是在软件开发中。不过，软件工程确实是一组我们已知有效的过程，而且从别人的经验中学习总是明智的。

像所有工匠一样，软件工程师必须学习如何最佳地使用工具。在本书中，你会不时地看到突出显示的软件工程提示。这些提示提供“如何……”的建议。通常它们讨论软件工程师如何从多种方法或解决方案中加以选择。读有关章节时，应该阅读各个软件工程提示，这是一个很好的方法。

加入软件工程提示还有一个原因，即可以帮助你对将来可能遇到的软件工程问题有一定的认识。软件开发与解决其他问题一样，一般要遵循一个3步的过程：

1. 分析问题。
2. 设计一个解决方案。
3. 实现解决方案。

遗憾的是，在充分理解第3步之前，第1步和第2步很难（甚至是不可能）充分了解。因此，通常都以相反的顺序来讲授软件开发。换句话说，我们首先研究如何实现一个解决方案。软件工程提示简要介绍问题的分析和设计步骤，从中可以了解到软件开发中的一些重要关系。

Java检查员

实践证明，在软件工程中有一个意义非凡的实践做法，这就是检查。“设计审查”，“代码检查”，“结构化走查”以及“桌面检查”等等指的就是不同类型的检查。顾名思义，检查实际上就是非正式的分析，或仔细查看软件的某个方面。就像画家必须退一步看画里有没有瑕疵一样，软件工程师也要这么做，从软件退一步，查找问题，并找出可能的改进。Java检查员一节就是要提供一个指导，指出如何检查软件。与软件工程提示一样，利用某一章内容之前以及之后，最好看看Java检查员的提示。

逐步地打开黑箱

如果让你学习一种新语言，而且告诉你只能说完整的句子。这与使用某种现代编程语言开始开发软件（计算机程序）极为类似。难点在于，你几乎要从头就创建完整的程序。不过，如果不花些时间，要想学会有关拼法和文法的所有规则，那是不可能的。

术语

计算机行业中衍生出了极其丰富的专用词汇。早在几年前，如果你提到“Web”，大多数人都会认为

你说的是有关蜘蛛的事情。如今，我们说“给我发邮件”几乎与“给我打电话”同样频繁。

软件开发也有自己的术语。其中一些术语对你来说可能已经很熟悉了，但是大多数都可能是生面孔。在本书中，我们会标出最重要的词，这些术语第一次出现时会用粗体显示。另外每一章的最后还会对关键术语进行总结。

最后的说明

如果本书是你的一门课程的教材，那我要感谢你的老师选择了本书。不过，本书不是为教师写的，而是为学生所写。在Addison-Wesley的伙伴、我们系的一些好同事以及一些很棒的审校人员的帮助下，我尽力做到组织合理、表述清晰、文字通俗易懂。我在第一学期授课时用了本书的最初手稿，同时还采用另外一本很著名的教材，后来我的学生告诉我，他们更喜欢我的手稿（当然，学期末是我给他们打分，而不是那本教材的作者）。所以你才是评判本书质量的真正的法官。如果你认为它在计算机科学学习旅程上有所帮助，那我的努力就是值得的。

Dave Riley

目 录

* 星号表示选学内容。

⊖ 附录内容均放在光盘中，请查看随书光盘

第1章 对象和类

物质（object）追求比不上精神充实令人满足。

——Pliny the Younger

目标

- 提供对象的基本定义：对象是一个具有状态和行为的实体
- 展示对象和它们的类之间的基本差异和相似性
- 提供第一个例子，展示如何用对象和类构造一个简单程序
- 介绍类图的记法
- 解释编辑-编译-运行过程，并说明对于Java编程语言如何实现这个过程
- 介绍瀑布式软件开发模型的有关术语，并用这个模型描述软件开发中各个关键过程的作用

世界就是一个**对象**（object）集合。你（一个对象）今天早上起床后，可能会从橱柜（另一个对象）拿出一盒麦片（这也是一个对象），然后把麦片倒进一只碗（第4个对象）里。接下来，打开冰箱（同样也是一个对象），拿出一加仑牛奶（也是对象）来泡麦片。可以看到，对象在我们的“实际生活”中扮演着核心角色；同样，对象也是计算机程序设计的核心。

1.1 对象无处不在

分析对象时只要稍加留心，就可以得出结论，每个对象都包括以下两个方面：

- 对象的状态
- 对象的行为

对象的**状态**（state）由对象的特有特征组成。牛奶包里有多少牛奶、牛奶包的位置、牛奶包是否打开等等，这些都是牛奶包所特有的特征，可以构成其状态。当然，对象的状态可能会随时间发生变化。牛奶包打开时，其状态就会改变；如果牛奶流出来，状态也会变化。状态是可以改变的，这就意味着状态与时间有关。

冰箱也能表现出状态。例如，冰箱里的温度就是其状态的一个重要部分。如果冰箱的门关上，说明冰箱里的灯是灭的，这是冰箱状态的一部分。如果门打开，灯会亮，冰箱的状态就发生了改变。

行为（behavior）是指可以由对象完成的任务，或者可以在对象上完成的任务。通常，行为都采用操作形式[⊖]。例如，从牛奶包倒牛奶，这个动作就是牛奶包的一个很平常的行为。牛奶包还有其他一些常见的操作，包括打开牛奶包，以及把它放到冰箱里。如果要站在牛奶包上，或者要用它来画画，诸如此类的操作一般不是牛奶包的正常行为。

一般将对象归类为组，这也称为**类**（class）。牛奶包是一个特定的对象，它属于一个称为“牛奶容器”的类。类似地，冰箱也是一个特定对象，它属于冰箱类。

类定义了由所有类成员共享的状态和行为。操作（如倒牛奶）就是牛奶容器类的一部分，因为所有牛奶容器对象都有这个操作。

1.2 软件中的对象

程序（program）是一个指令集合。当执行程序时，它会在计算机上完成一个特定的任务。因此，编写程序的人称为**程序员**（programmer）。**软件**（software）和**代码**（code）这两个词是指一个或多个程序

⊖ 即通过操作来表述行为。——译者注