Addison Wesley

The Object of Java

Java 面向对象教程

应用软件工程原理编程

David D. Riley 著
贺 民 王朝阳 译

清华大学出版社

内 容 提 要

本书是采用 Java 语言讲授面向对象程序设计的教材。作者总结多年在 CS1 课程中应用 OOP 思想的宝贵经验，结合软件工程思想，提出了最有学习效率的叙述顺序和辅导方法。

本书首先强调了以对象为中心的思想，然后讨论了面向对象的策略，接着依次讲解了基本类型、提供者类、控制结构、继承、容器和数组等问题。整个讲解过程始终围绕软件工程的编程思想，应用各种常用的代码、算法及设计模式，力图将复杂的问题简单化、规范化。另外，还精挑细选出丰富的练习，帮助读者理解概念并提高实际编程能力。

本书的内容、结构完全满足高等院校 CS1 课程的教学要求，可以作为讲授面向对象程序设计的正式教材，也适用于 Java 程序员作为学习语言的主要参考书。

EISBN: 0-201-71585-6

the Object of Java

出版者：清华大学出版社（北京清华大学学研大厦，邮编 100084）

http://www.tup.com.cn

印刷者：北京市耀华印刷有限公司

发行者：新华书店总店北京发行所

开 本：正 16 印张：35.25 字数：880 千字

版 次：2003 年 5 月第 1 版 2003 年 5 月第 1 次印刷

印 数：0001～4000

盘 号：ISBN 7-89494-073-9

定 价：57.00 元（1CD）

序　言

为什么学习 Java 对象技术？是想从事计算机科学研究？还是希望提高分析和解决问题的能力？亦或是想尝试一下，确定计算机编程是否能成为自己未来的发展方向？

无论是出于何种目的，最重要的是要认识到软件开发是计算机科学的生命之源，这也是计算机课程均以软件开发为重点的原因所在。本书是软件开发的入门教材，可以作为研究计算机科学的敲门砖。

学习软件开发的过程重在参与。仅仅读诗，绝对不可能学会写诗，同样，仅仅看本书或听教师讲课，也不能学会编写软件，你要自己编写代码。本书每章的最后都有编程作业，你应当通过这些作业练习编写程序。

软件开发的实质就是解决问题。在开发软件时，解决问题的能力也会随之提高。编程的过程困难重重，有痛苦也有欢乐，计算机科学就是体验解决问题的乐趣。通往目标的路径越是艰险，解决问题之后就越有成就感。

S.1　软件工程提示

多年来，随着软件开发的发展，人们对这个过程的理解也日趋成熟，现在，我们将此过程的最佳实践经验叫做“软件工程”。软件工程的原则和过程可能比较复杂，尤其是在刚刚涉足软件开发时。然而，实际上，软件工程就是高效过程的集合，通过汲取别人的实践经验来学习是一种明智之举。

与任何艺术家一样，软件工程师必须学会如何最好地利用制作艺术品的工具。在本书中，随处可见“软件工程提示”，这些提示告诉读者“如何去做”。通常，提示中会讨论软件工程师如何从各种方法和解决方案中作出选择，在相关的章节阅读这些提示将非常有益。

软件工程提示　*在一两个软件项目中尝试了所介绍的技术之后，最好返回来再看看软件工程提示。*

通过软件工程提示，还可以正确地预测将来可能遇到的软件开发问题。软件开发与解决其他问题一样，一般遵从如下 3 个步骤：

（1）分析问题
（2）设计解决方案
（3）实现解决方案

遗憾的是，如果没有完全理解最后一步，完全理解前两步是非常困难的，因此，通常按照相反的步骤来讲解软件开发。换句话说，我们从学习如何实现解决方案开始。软件工程提

示通常会概括问题的分析与设计步骤，以完整地说明软件开发过程。

S.2 Java代码检查

在软件工程中，进行代码检查已被证明是一种非常有价值的实践活动。“设计审查”、“代码检查”、“结构化走查”以及“桌面检查”都是检查的不同形式。顾名思义，检查是一种非正式的审查方式，仔细测试软件的某个方面。画家完成绘画之后必须回过头来看看是否还存在着一些不够完善的地方，与此相同，软件工程师也要在编写软件后寻找问题所在或可改进的地方。“Java 代码检查”部分指导读者如何检查软件。在尝试文中提供的程序之前或之后，最好再看看“Java 代码检查”的内容。

S.3 连续打开黑箱

假设你要学习一门新语言，并且要求你必须只能说出完整的句子，这种情形非常类似于通过现代编程语言来开发软件，即计算机程序。困难在于，几乎从一开始你就要编写出完整的程序。然而，不花时间是不可能掌握任何语言的表示法和语法规则的。

密封黑箱 黑箱部分可以对你提供帮助。“密封黑箱”部分主要介绍某些语言功能或技术，你只需使用它们，而无需问如何使用和为什么使用。也就是说，只需将这些语言功能作为黑箱使用即可，无需关心是什么让这些黑箱工作。如果希望立即得到解决方案，在“密封黑箱”中，还提供了本书打开黑箱并揭示秘密的地方。通常最好是忽略那些参考信息，除非你已经学习过打开该黑箱的章节。

打开黑箱 生活中，人们乐于发现秘密。当打开黑箱时，我们将发现先前神秘事物的原因和工作方式。“打开黑箱”中说明此时可以打开这些黑箱，并告诉你对应的密封黑箱在什么地方。如果接受这些黑箱，你会发现学习编程很容易。

S.4 最后的说明

本书适于作为教材，本书的目标读者不是教师，而是学生。在 Addison Wesley Longman 公司中的优秀团队、我的同事以及一些优秀审查人员的帮助下，本书内容逻辑清晰、明白易懂。第一学期，我同时使用本书的手稿和一本著名的教材讲课，学生们告诉我他们更喜欢本书，你可以自己判断本书的质量如何。如果认为它可以帮助你顺利踏上计算机科学之旅，我会很高兴，因为我的努力得到了回报。

Dave Riley

前　言

本书主要介绍 Java 对象技术。Sun 公司将 Java 设计为现代软件的开发工具，与此类似，我们将本书定位为现代软件开发的指导工具。二者之间的共同点是，它们都以面向对象范型为动力，而合理的软件工程方面的实际应用又为其发展推波助澜。本书主要介绍编程技巧、面向对象技术、软件工程技术以及 Java 技术，为将来深入研究计算机科学打下坚实的基础。

按照 ACM/IEEE 的约定，本书面向 CS1 课程的学生，读者不需要具备任何计算机编程经验，但应当具备一定的分析能力。要理解本书中的所有表示法，还必须学完 3 年高中数学。

P.1　面向对象与Java

James Gosling 和 Henry McGilton 曾说过："为了适应越来越复杂的网络环境，编程系统必须采用面向对象的概念"。强调面向对象意味着，仅仅"对象优先"还不够，它需要"以对象为中心"的方法。在当今程序员编写的程序中，软件类、方法、继承以及事件驱动代码已经像变量、循环和数组一样随处可见。以对象为中心方法的内涵已经远远超出了对象本身，而且，通过该方法，软件开发人员可以学会用面向对象的思维方式思考问题。

以对象为中心的方法的一个显著优点是，它包含了以前的规则以及功能范型。使用面向对象编程（OOP，object-oriented programming）模式，仍然要编写赋值指令，传递参数，从函数返回值（非空方法），同时还要掌握所有的基本控制结构。这就意味着，从 OOP 转向其他编程方式时也许会轻而易举，而进行相反方向的转换却会遇到许多障碍。

本书作者曾在 CS1 课程中讲授面向对象（O-O）方法长达 6 年之久，积累了丰富的经验，在过去的 3 年中一直使用 Java 编程语言。由于 Java 能够很好地实现对象模型，并在学术领域和专业团体中普遍应用，所以，使用 Java 讲授面向对象方法是非常合适的。另外，Java 的表示方法与 C 语言类似，这对已学完 C++课程以及将来需要使用 C 或 C++语言的学生来说，非常有益。正如 Gosling 和 McGilton 所说："Java 编程语言的设计自始至终就是面向对象的。"

P.2　强调软件工程

与写作优美的文章类似，优秀的程序设计也需要技巧和规范，而在这方面，软件工程的

指导原则和技术起着关键的作用，因此本书自始至终着重强调了软件工程，这是本书的一大特色。

软件工程提示

各章内容中都有不少软件工程提示，这通常是软件开发人员的“实践经验集锦”。通过软件工程提示，可以了解如何调整语言结构的格式，从而体现良好的编程风格，另外，软件工程提示中解释了经验丰富的程序员如何解决常见的设计问题。

按约定编程

面向对象编程进一步强调了技术规范的重要性。方法的前提条件和后置条件以及类不变量对于传递代码的行为相当关键。本书在说明示例和定义示例类时，都统一使用这样的断言。第 2 章将介绍按约定编程，随后各章均包含这类内容。本书还讨论了逻辑表达式、循环不变量以及专用断言表示法。

模式

设计模式的软件工程原则常常提醒我们，软件开发技能一般以人们对常用结构的记忆为基础。本书拓展了此观点，讲述了各种常用的模式，如代码表达式、指令、算法以及作为编程模板的设计模式。这些模式都在文中重点讲述，以使读者熟悉这些常用且必备的解决方案，同时了解如何及何时应用这些模式。

软件测试

任何优秀的软件工程模型都必须有软件测试部分。我们将专门讲述测试内容，以便读者学会基本的调试技能，并掌握简单路径测试以及黑箱测试方面的知识。

Java代码检查

软件工程研究证明，最有用的测试可以有多种形式，如非正式的桌面检查、代码复查以及走查。实施这些测试需要知道“检查”和“复查”的内容以及如何“走查”等。本书每章都以“Java 代码检查”的形式提供此类信息。在各章的最后，不仅综述本章的重点内容，还在练习中进一步说明了如何应用这些概念解决实际问题。

UML

在面向对象的设计与编程中，图形具有举足轻重的作用。本书提供了大量的对象图，说明计算过程中的运行时特性。同时，使用了许多类图来描述类的可视接口以及多个类之间的关系。除此之外，还用活动图显示控制流。

为了正确表述软件工程方法，本书的所有图形表示法都来自统一建模语言（UML）。通过 UML 图表，可以向学生展示目前软件开发行业所用的标准符号。同时，附录 E 概括了本书使用的部分 UML 表示法。

P.3 主 题 顺 序

通过多年在CS1课程中讲授OOP，我们获得了宝贵的经验，发现传授知识时最重要的就是叙述顺序。而且，在学习过程中，最缺乏的就是辅导时间，而辅导时间非常重要。所以，在尝试了各种方法之后，我们发现，按照本书顺序安排课程，对学生最有帮助。

从目录便可以看出，本书内容的安排与众不同。这是因为，要掌握以对象为中心的方法，就要尽早介绍面向对象的关键内容，这样才能满足整个教学过程的需要。

本书第1章简要说明对象和类，使读者能够对以后的内容有一个概括的印象。首先，使用面向对象程序的例子来说明一些基本的表示法、软件开发工具（编辑器、编译器以及虚拟机），概括软件开发的整个过程。本章大约需要1课时。

以对象为中心

第2章首先介绍基本的面向对象构造，即方法调用。在演示了如何调用对象方法之后，说明简单的指令序列。本章还介绍了声明、实例化以及对象赋值。使用交换算法作为代码模式，则有助于说明对象绑定。第2章还介绍了编写初始程序所需的所有Java工具。

早期的面向对象策略

第3章介绍了基本的软件设计技术和其他一些语言工具。在本章中，可以学到两类解决OOP环境中编程问题的策略：自上而下设计和原型设计。利用这些策略，就会在编程时有的放矢，而不会感到无从下手了。第3章还讨论了其他基本的软件开发技巧，如选择合适的标识符名，以及通过输出指令（System.out.println）来帮助调试。

方法

第4章详细介绍第1章和第2章中简要说明的OOP基本方法调用，以及如何创建方法，包括参数传递、局部变量和非空方法之类的问题。

基本类型

第5章以前4章的内容为基础，介绍基本数据类型。我们已经发现，将基本数据类型的内容（比如数学表达式）放在后面介绍，有助于学生的理解。这样，读者在面临Java以非面向对象方式处理数字的问题之前，可以相对容易接受Java对引用数据的处理方式。在稍后对基本数据类型的介绍中可以看到，在面向对象环境中，基本变量显然与众不同。在这一章之后不再需要花费太多的时间来讲述基本表达式，这将改善学生对引用数据/对象的接受程度。

编写提供者类

编写利用其他类的客户代码与编写外部类的提供者代码截然不同。第6章帮助读者实现这方面的飞跃。本章不仅介绍最适于特定方法的类的设计决策，还将叙述隐藏和封装适当信息的重要性。

控制结构

为什么不在第 7 章和第 10 章之前介绍选择和循环结构呢？这是因为，推迟对控制结构的讨论可以使学生更早地接触到面向对象的主题，并应用于本课程的其余部分。晚一点介绍控制结构，有了前面的知识积累，可以更好地应用 if 结构和 while 循环，同时，学生会更加渴望接触到此类信息。在面向对象模型中，控制结构已经不再是主要的编程技巧了。然而，选择和循环仍然是重要的编程工具，并且，我们发现，这种安排方式有利于学生完全掌握这些结构。

推迟介绍选择和循环结构，部分原因在于事件驱动的代码。浏览本书的编程练习和示例之后，会得到这样的结论：不使用选择指令或循环结构进行编程也是可能的，这会非常有趣并极具挑战性。

进行大量的事件驱动编程还有其他好处。事件驱动的代码与 OOP 形影不离。我们都知道，按钮、菜单以及滚动条的行为均依赖于事件驱动的控件。的确，对于经常使用图形用户界面环境的用户而言，将很难适应程序控制模型。

继承

本书通过如下 3 个阶段介绍面向对象工具与技术的核心：

（1）利用对象、类和方法（第 1 章~第 4 章）。
（2）编写提供者端代码（第 6 章）。
（3）使用继承（第 8 章~第 9 章）。

尽早介绍继承会有很多好处，但我们发现，大致在课程中间介绍继承具有最佳效果。这样的安排可以使学生有充足的机会在多个课程项目中使用继承。

容器，包括数组

第 11 章介绍容器类问题，第 12 章说明数组的概念。相对于直接存取容器（如数组）而言，表内在的顺序特性使其更易于使用。更为重要的是，对于实现容器数据结构而言，表是以面向对象方法实现容器的更好示例。由于 Java 将数组表示法整合到该语言中，所以，数组与其他对象具有不同的外观。第 11 章还讨论了 Object 类、包装类以及适用于表的排序算法（插入排序）。

P.4 各章节内容的依赖性

虽然按上述顺序学习很合适，但依然有很多变化因素影响我们的决定。因此，必须尽量减少各主题之间的依赖性。图 P.1 说明了各重要章节之间的依赖性。在此图中，从 A 章到 B 章的箭头表明 B 章完全依赖于 A 章所提供的材料。

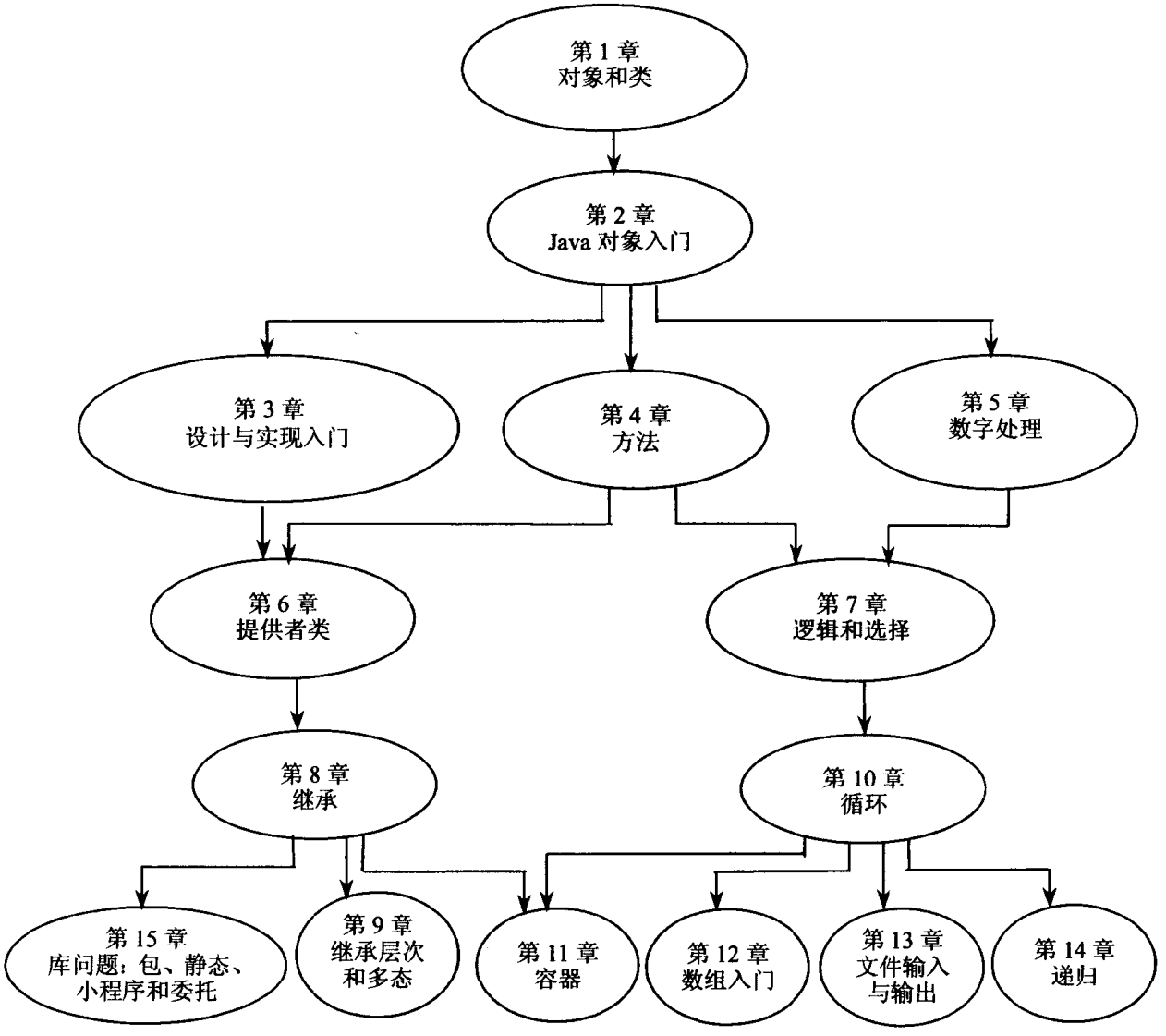

图 P.1 各章之间的依赖性

本书尽量满足 CS1 课程的需要。理想情况下，最好学完 15 章的全部内容。但通常，多数 CS1 课程不得不略去后面章节中的部分或全部内容。

此外，附录 A 介绍了计算机硬件的入门知识，可在课程的开始介绍此附录，或作为学生的课外阅读材料。由于此部分内容在对象模型的范围之外，所以将它放在附录中。

P.5 选择应用程序或小程序

本书的程序以如下方式编写：既可作为 Java 应用程序运行也可作为 Java 小程序运行。每个程序中都引入了一个 Director 类，为程序定义控制器对象。

每个程序还包括了两个单独的类：一个用于将程序作为应用程序来执行，另一个则将程序作为小程序执行。下面的 go.java 类可将程序作为应用程序来执行。

```
public class go {
  public static void main(String args[]) {
    Director director = new Director();
  }
}
```

如果要将程序作为小程序使用，则使用 appletStarter.java。

```
import java.applet.*;
  public class AppletStarter extends Applet {
    private Director  director;
    public AppletStarter()  {
        director = new Director();
  }
}
```

首先介绍应用程序和小程序（即 go.java 和 appletStarter.java）之间差别的详细信息对于学习 Java 没有什么好处。因此，有关此问题的完整讨论将推迟到第 15 章。

P.6 aLibrary（可用资源）

对象在类中建立，而类来自库。因此，如果不介绍软件库，以对象为中心的方法就不完整。然而，对于程序员来说，掌握指令系统的细节并没有迅速适应各种库代码的能力重要。阅读和使用库类，是拥有这一能力的最佳方法。使用库类首先要阅读类图和类说明。

本书介绍 Java 类库 aLibrary。通过 aLibrary，很容易开发显示动画的 Java 程序、利用简单的表数据结构或访问实时日历和时钟信息。然而，aLibrary 中的大量类主要用于为学生编写的 Java 代码提供图形用户界面（GUI）。

使用图形库的原因

就我们的经验而言，GUI 库最能激发初级程序员学习的动力，我们发现，讲述 aLibrary 根本无需花费很多的时间。一般来说，演示了 aLibrary 的基础之后，学生就可以通过说明自学这方面的内容。完整的 aLibrary 类图和类说明都是 HTML 格式，以具有方法的前提条件和后置条件的类不变量的形式提出，非常有利于学生自学。

图形类同时也具有可视的优势。利用 GUI 对象，可以很容易地将对象和对象之间的关系可视化。

使用 GUI 类的最后一条理由是，其他方法似乎只有乏味的交互方式的终端流类型 I/O，这种方法在 CS1 课程中已经使用了 30 年。但现在的学生是伴随光盘和电子游戏长大的，他们会发现，I/O 终端不自然且极其乏味。

不使用jdk库的原因

Java jdk 库（如 swing 和 awt）是为已有编程经验的程序员而不是入门级程序员而设计的。aLibrary 类则是教辅工具。熟练的程序员可以很快掌握复杂的 paint 方法、repaint 方法、update 方法和 Graphics 类，在应用这些方法时，即使绘制一个矩形都需要 swing/awt。对于初次编写程序的学生而言，更适于从 aLibrary 中实例化 ARectangle 类。对于经验丰富的程序员，通过 Java interfaces 进行事件委托是功能强大而灵活的工具，而 aLibrary 类中已经植入了事件处理程序，更适用于学习继承和方法覆盖。

虽然 aLibrary 是一个单独的库，但它并不能与 jdk 分离。每个图形 aLibrary 类都从某些 swing/awt 类继承而来，并且，aLibrary 保持与 swing/awt 的命名一致。这意味着，熟悉 aLibrary 库的程序员可以很容易地转向使用 swing/awt 库，有关信息，请参见附录 D。

P.7 黑箱揭秘

Bertrand Meyer 已经将“连续打开黑箱”的概念作为一种教学方法，这种方法首先使用预定义的组件（也就是黑箱），然后逐渐阐明构成黑箱的工具和技术。例如，本书通过使用 go.java 静态类实例化初始 Director 对象，以避免过早介绍静态方法可能给学生带来的困扰。读者开始可以忽略 go.java 类，后面我们再解释静态类和介绍 go.java 类的代码。与此类似，利用 A3ButtonWindow aLibrary 类，可以很方便地访问含有按钮的窗口。同时，在介绍 A3ButtonWindow 的实现作为继承示例之前，就已在程序示例中使用过它了。

在需要使用黑箱且希望读者作出某些假设时，本书提供“密封黑箱”部分。相应地，“打开黑箱”部分则解释本书提出的黑箱概念。

P.8 课 堂 检 验

在讲授 CS1 课程的几年时间中，我一直在课堂上使用上述教学方法。在过去的 3 个学期中，我一直使用本书手稿讲课并多次对手稿进行修改。更重要的是，我的 3 个同事在最后一个学期使用基本定稿的教材，效果很好。前来听课的学生水平参差不齐，从刚入学的新生，到已获得其他学位的学生，应有尽有。在 CS1 课程中，人数最多的是想主修计算机科学和信息系统专业的学生，但其他专业的学生也不少。

P.9 补充材料

下面列出与本书内容相关的补充材料，读者可从附带光盘中获得它们：

- 编程练习示例的所有源代码
- 本书示例程序的所有源代码（70 多个程序），而且都以应用程序和小程序两种形式给出
- 包括源代码的所有 aLibrary 类
- 所有 aLibrary 类的类图和类说明（以 HTML 格式）
- 所选 Java jdk 类的类图和类说明，包括 Byte、Character、Color、Double、Float、Integer、Long、Math、Object、Short、String 以及 Timer

Dave Riley

目　录

第 1 章

对 象 和 类

本章学习目的

- 提供对象的基本定义，此定义将对象的状态和行为作为一个整体
- 给出对象及其类之间的基本相似性与区别
- 提供一个初始示例，说明对象和类如何构成简单的程序
- 介绍类图的表示法
- 解释编辑 - 编译 - 运行过程及其 Java 编程语言的实现
- 介绍瀑布式软件开发模型的术语，并通过此模型来说明软件开发中各种关键过程的作用

世界是对象的集合体。当你（一个对象）清晨起床时，需要从食品橱（也是一个对象）中找到一盒麦片（另一个对象），然后，将麦片倒入碗里（第 4 个对象）。接着打开冰箱（对象）并取出牛奶（对象），将其倒到麦片上。正像在我们的日常生活中，对象处于中心位置一样，在计算机编程中，它们仍然充当中心角色。

1.1 对象随处可见

对象都具有如下特征：

- 对象状态
- 对象行为

对象的特殊特征组成了它的状态。容器中的牛奶数量、容器位置、是否打开此容器，都是构成牛奶容器状态的特征。当然，对象的状态会随时间而发生改变。当从容器中倒出牛奶时，牛奶数量发生了改变。状态具有可改变的特点，这表明状态与时间有关。

冰箱同时也有状态。比如，冰箱内部的温度就是它的一种重要状态。同样，关上冰箱门，冰箱灯熄灭，打开冰箱门时，冰箱灯将点亮。当然，冰箱的状态也随之变化。

行为是对象可以完成的任务，对对象执行的操作是行为的表现形式。比如，将牛奶从容器中倒出，这是针对牛奶容器的普通操作。其他常见的操作还包括将牛奶容器放到冰箱中，取下容器盖等。但是，坐在牛奶容器上，或者用容器来绘图，这些都不是此对象的典型行为。

通常可将对象分成组，即类。你手中的牛奶容器是具体的对象，但它属于“牛奶容器”类。与此类似，家庭中的冰箱也属于“冰箱”类。

类确定了所有类成员都具有的状态和行为。某个操作（如倾倒）属于牛奶容器类，是因为所有的牛奶容器对象都具有这一操作。

1.2 软件中的对象

程序是指令的集合，执行程序可以在计算机中完成特定的任务，所以，人们将编写计算机程序的人叫做程序员，术语软件和代码是一个或多个程序的集合，因此也可将程序员称为软件开发人员。

今天，经验丰富的软件开发人员最常用的编程策略为面向对象编程（OOP,object-oriented programming）。使用面向对象（O-O,Object-Oriented）策略的程序员首先要选择对象，利用这些对象共同解决给定的问题。

示例 1 说明了如何以 OOP 方式来开发特定的程序。软件开发人员首先要确定说明此程序所完成任务的程序需求。

示例 1：程序需求

编写可在窗口中显示灰色圆圈的程序。

软件工程提示 *面向对象的设计首先要选择对象。优秀的面向对象设计师一般都从程序需求中寻找名词，原因是这些名词一般都表示解决方案所需要的关键对象。*

示例 1 中的程序需求文档提出了两个重要的对象：圆圈和窗口。通过需求确定对象的一种方式就是找到需求中的名词，这里，圆圈和窗口就是这样的名词。

一旦程序员确定了程序中的对象，下一步就是找到或创建与每一对象相应的类。由于面向对象的程序代码都驻留于这些类中，所以，这些类至关重要。事实上，说面向对象程序由类的集合构成是正确的。

软件工程提示 *程序员通常应该寻找重用现有软件的机会，而不是浪费时间去创建它们。*

理想情况下，程序员应该重用现有的类，而不是编写新类的代码。程序员重用现有的软件在现实生活中也有同样的事例，如建筑师利用标准规格的板材来设计新建筑物，或者，工程师利用现有的火花塞设计新汽车引擎，而这些火花塞在多数五金商店都可以买到。

在示例 1 中，假设已经存在如下两类：窗口对象的 SimpleWindow 类以及圆圈对象的 GrayCircle 类。假设已经具备了合适的支持类，图 1.1 提供了可以完成示例 1 中任务的其余程序。