21世纪高等学校规划教材

Java MIANXIANGDUIXIANG CHENGXU SHEJI

Java面向对象程序设计

费贤举 主编
庄国强 蒋巍 程红林 副主编

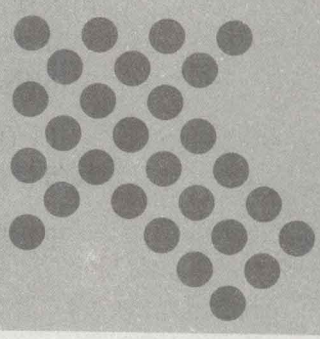
中国电力出版社
CHINA ELECTRIC POWER PRESS

21世纪高等学校规划教材

Java MIANXIANGDUIXIANG CHENGXU SHEJI

Java面向对象程序设计

主　编　费贤举
副主编　庄国强　蒋　巍　程红林
编　写　李亦飞
主　审　张广泉



中国电力出版社
CHINA ELECTRIC POWER PRESS

内 容 提 要

本书为21世纪高等学校规划教材。

本书共11章，主要内容包括概述、Java程序设计基础、Java常用系统类的应用、集合与泛型、Java面向对象程序设计、Java的异常处理、Applet小程序、图形用户界面应用、文件和流、多线程和动画编程、网络编程等。本书注重理论与实践相结合，注重基本知识的理解与基本技能的培养；以案例驱动和软件项目开发实例贯穿全书章节，强调理论与习题以及实验题相结合；内容丰富，结构安排合理，由浅入深，层次分明。

本书可作为普通高等院校Java语言程序设计、面向对象程序设计等课程教材，也可作为从事Java语言软件开发人员的参考用书。

图书在版编目（CIP）数据

Java面向对象程序设计／费贤举主编. —北京：中国电力出版社，2011.12

21世纪高等学校规划教材

ISBN 978-7-5123-2506-7

Ⅰ. ①J… Ⅱ. ①费… Ⅲ. ①JAVA语言－程序设计－高等学校－教材 Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2011）第269194号

中国电力出版社出版、发行

（北京市东城区北京站西街19号 100005 http://www.cepp.sgcc.com.cn）

汇鑫印务有限公司印刷

各地新华书店经售

*

2012年2月第一版 2012年2月北京第一次印刷

787毫米×1092毫米 16开本 17.5印张 424千字

定价**31.00**元

前　言

由于Java语言具有面向对象、与平台无关、安全、稳定和多线程等优良特性，自诞生以来已经发展成为当前社会最为炙手可热的计算机语言之一，被广泛应用在Internet、企业端、移动领域和桌面端开发等领域。

本书内容组织由浅入深，并不要求学生具备编程背景。本书以面向对象关键技术为突破口，让学生一开始就树立面向对象的编程思想；以典型案例为引导，通过精讲多练和培养学生程序调试能力，启发学生解题的思想，让学生掌握Java程序语言的概念和编程方法，以案例驱动和软件项目开发实例贯穿全书章节，强调理论与习题以及实验题相结合，为读者提供学会Java项目开发的最佳实践方案，同时紧扣面向对象与Java EE开发的最新技术。

此外，本教程配有电子课件及程序源代码，本书中用到的所有程序代码均在JDK1.5运行环境中利用Eclipse 3.3工具调试通过，供读者参考。

本书由常州工学院费贤举担任主编，庄国强、蒋巍、程红林担任副主编。费贤举编写了第1、5、7、8章，庄国强编写了第4、11章，蒋巍编写了第2、3、9章，程红林编写了第6、10章。李亦飞、王文琴和王文帮助编制了本书的部分程序的调试和校对工作。费贤举对全书进行了统稿工作。

本书由苏州大学张广泉教授担任主审。同时，本书编写过程中得到了计算机信息工程学院院长庄燕滨教授的鼓励和支持，还参阅了一些专家学者的文献资料。在此一并致谢！

由于时间有限，书中难免有错误和不妥之处，恳请广大读者提出宝贵意见。编者邮箱：feixj@czu.cn。

编　者

2011年11月

目 录

第1章　概　　述

——本章要点——

（1）了解Java语言的起源和发展历史。
（2）了解Java语言的应用现状、运行机制及学习Java语言的原因。
（3）理解面向对象编程的相关概念。
（4）熟悉Java运行和开发环境的构建过程。
（5）掌握Java程序开发的基本步骤和Application、Applet两类程序结构、编辑、编译和运行的方法。
（6）熟悉目前流行的Java IDE开发环境Eclipse的安装使用方法。
（7）为本课程后续的学习提供思想认识和工具环境基础。

1.1　了　解　Java

从1995年5月诞生至今，Java语言随着Internet的迅猛发展而成长壮大，现已成为Internet上的主流编程语言。J2ME、J2SE、J2EE三大平台强大的应用系统设计能力，使Java无处不在。

1.1.1　Java的起源

Java语言来源于Sun Microsystems公司的Green项目，该项目最初的目的是为家用消费电子产品开发一个分布式代码系统，以便用户将E-mail发送给电冰箱、电视机、烤箱等家用电器，对它们进行控制，并和它们进行信息交换。在项目研制初始阶段，项目组成员准备采用C++语言开发该系统，但是C++遇到了前所未有的挑战，因为它太复杂而且安全性差。所以最后项目组成员基于C++开发了一种新的语言Oak（橡树），这就是Java语言的前身。Oak是一种用于网络的精巧而又安全的语言，Sun公司曾依此投标了一个交互式电视节目，但结果是被SGI打败。正当Oak无家可归之时，Mark Ardreesen开发的Mosaic和Netscape启发了Oak项目组成员，他们用Java编制了HotJava浏览器，并得到了Sun公司首席执行官Scott McNealy的支持，开启了Java进军Internet的契机。

1994年，Internet的迅猛发展和WWW的快速增长促进了Java语言研制的进展，使得它逐渐成为Internet上受欢迎的开发与编程语言。1995年5月，Sun公司正式发布Java的第一个办公版本。从Oak问世到Java语言公开发布，许多人对Java的设计和改进作出了贡献，正是他们的贡献使最初的原型逐渐成熟。

Java平台由Java虚拟机（Java Virtual Machine，JVM）和Java应用编程接口（Application Programming Interface，API）构成。Java应用编程接口为Java应用提供了一个独立于操作系统的标准接口，可分为基本部分和扩展部分。在硬件或操作系统平台上安装一个Java平

台后，Java 应用程序就可运行。现在 Java 平台已经嵌入了几乎所有的操作系统。这样 Java 程序只需编译一次，就可以在各种系统上运行。

Internet 的迅速发展和万维网（WWW）应用的快速增长对 Java 的未来起了至关重要的作用，用 Java 编写的浏览器 HotJava 以及 Applet 小程序在 Web 上的应用，使得它逐渐成为 Internet 上受欢迎的开发、编程语言，一些著名的计算机公司纷纷购买了 Java 语言的使用权，如 Microsoft、IBM、HP、Netscape、Novell、Apple、DEC、SGI 等公司，同时许多公司希望将 Java 嵌入到各种操作系统中。

Java 从 1995 年的一个小的编程工具，到今天已经发展成为可驾驭智能卡、小型消费类器件到大型数据中心的 Java 2 平台，全球有众多的大型企业在采用 Java 开发自己的信息系统，多家公司从 Sun 公司获得了 Java 技术许可证。Java 开发者阵营拥有几百万位会员，Java 已进入主流计算模式。由此可见 Java 的发展速度是惊人的，它的发展历程如下。

1995 年 3 月，Sun 公司公布了 Java 的 Alpha 1.0a2 版本；

1995 年 3 月，Java 的第一个办公版本发布；

1996 年 1 月，Java 的第一个开发包 JDK v1.0 发布；

1997 年 2 月，Java 的开发包 JDK v1.1 发布；

1998 年 12 月，Sun 发布 Java 2 平台，JDK v1.2 发布，Java 2 平台的发布是 Java 发展史的新里程碑。

Java 分为三个体系：JavaME（Java 2 Platform Micro Edition），JavaSE（Java2 Platform Standard Edition），JavaEE（Java 2 PlatformEnterprise Edition）。

（1）J2ME——Java 2 Micro Edition，用于嵌入式 Java 消费电子平台。2000 年 12 月，Sun 公司宣布，它将推出 Java 2 平台 Micro（J2ME）开发版和使用于 Palm OS 平台的 MID（Mobile Information Device）规范概要。这些新品的推出将使数百万 Java 技术开发人员更容易为通用的 Palm OS 平台创建应用程序。此外，Sun 和 Palm 公司还将通过 JCP（Java Community Process）项目与业界的其他专家一起为个人数字助理（PDA）确定编程接口的技术规范。开发者因能把他们的 Java 技术经验用在 Palm OS 平台上配置解决方案而受益，终端用户因能将采用 J2ME 编写的应用程序用于新的企业、商务、娱乐和通信等解决方案而受益。有了相对于 Palm OS 平台的 Java 技术发展规划，开发商们将会拥有标准化的适用于业界应用的解决方案。

（2）J2SE——Java 2 Standard Edition，用于工作站、PC 的 Java 标准平台。在 Java 2 平台 JDK 1.2 发布之后，Sun 公司又相继发布了 J2SDK 1.3、J2SDK 1.3.3 和 J2SDK 1.4Beta2 版，目前，Java 最新版已到 1.6，即 Java 6.0。它们都是支持分布式计算免费的 Java 2 标准平台。Java 2 标准平台体现了 Sun 的开放精神，被称为“互联网上的世界语”，公布于互联网上供大家免费享用，甚至连代码也不保密，可以在网上免费下载。

（3）J2EE——Java 2 Enterprise Edition，可扩展的企业级应用 Java 2 平台。2001 年，Sun 公司在旧金山召开了关于 Java 2 平台企业版（J2EE）的新闻发布会，到目前，其最新的 J2EE SDK 1.4 可以从 Sun 公司网站下载。

J2EE 是分布式企业软件组件架构的规范，具有 Web 性能，具有更高的特性、灵活性、简化的集成性、便捷性，以及 J2EE 服务器之间的互操作性。目前已有多家取得 J2EE 技术许可的公司推出了基于 J2EE 技术的兼容性产品。这些公司通过了 J2EE 兼容性测试（CTS）

中的各项测试，满足了 J2EE 技术品牌中的所有要求。

1.1.2 学习 Java 语言的原因

Java 作为新一代的面向对象的程序设计语言，其平台无关性直接威胁到 Wintel 的垄断地位。一些著名的计算机公司纷纷购买了 Java 语言的使用权，如 IBM、Netscape、Novell、Apple、DEC、SGI、Oracle 等，甚至包括最不情愿的 Microsoft。

Java 语言被美国的著名杂志 PC Magazine 评为 1995 年十大优秀科技产品(计算机类就此一项入选)。微软公司总裁 Bill Gates 不无感慨地说："Java 是长时间以来最卓越的程序设计语言。" Sun 公司的总裁 Scott McNealy 认为，Java 为 Internet 和 WWW 开辟了一个崭新的时代。万维网（WWW）的创始人 Berners-Lee 说："计算机事业发展的下一个浪潮就是 Java，并且将很快会发生。"甚至有人预言：Java 将是网络上的"世界语"，今后所有的用其他语言编写的软件统统都要用 Java 语言来改写。根据 Java 之父 Gosling 2010 年公布的一份最新的 Java 调查报告中称，JRE（Java Runtime Environment）的每周下载量为 1500 万；共有 100 亿个 Java-enabled 的应用，10 亿个 Java-enabled 的桌面，1 亿个 Java-enabled 的 TV 设备，26 亿个 Java-enabled 的移动设备，55 亿个 Java 智能卡以及超过 650 万名 Java 开发者。自从 2010 年 Oracle 收购 SUN 后，同样也在致力于 Java 在桌面端、嵌入式、移动领域、高性能计算机及其他系统方面的发展。目前主流的手机移动开发平台 Android 也是采用 Java 语言开发。

Java 总是和 C++联系在一起，而 C++则直接派生自 C 语言，所以 Java 语言继承了这两种语言的许多特性，而且 Java 语言的产生与过去几十年中计算机编程语言的改进和发展密切相关。因为 Java 的语法是从 C 继承的，所以 Java 许多面向对象的特性受到 C++的影响。但是 Java 吸收了大量 C++中的优秀特性，同时除去了那些模糊、复杂、容易出错的特性以及 C 和 C++中影响程序稳健性的部分，如指针、内存申请和释放等，并引入了很多独特的高级特性。

作为一种程序设计语言，Java 是一种简单的、面向对象的、分布式的、解释型的、健壮安全的、结构中立的、可移植的、性能优异及多线程的动态语言。在面向对象的程序设计（OOP）中，使用 Java 语言的继承性、封装性、多态性等面向对象的属性可以较好地实现信息的隐藏、对象的封装，从而降低程序的复杂性，实现代码的复用，提高开发速度。

在计算机语言的发展过程中，人们会继续感受 Java 的影响和强大功能。它的许多革命性特点、结构和概念已成为所有新语言的基准。

受 Java 影响最显著的例子就是 C#了。C#是 Microsoft 公司所创建的，用来支持.NET 框架。C#与 Java 关系很密切，比如都使用相同的语法，均支持分布式程序设计，使用相同的对象模型等。当然，两者之间也存在一些差别，但是从总体来看非常相似。

下面分别从以下几个方面来讨论 Java 语言的特点，然后通过与 C、C++相比较进一步指出 Java 所具有的优势。

1. 简单性

Java 是一种面向对象的语言，通过提供最基本的方法来完成指定的任务。只需理解一些基本的概念，就可以用其编写出适合于各种情况的应用程序。Java 略去了运算符重载及多重继承等模糊概念，并且通过实现自动无用信息收集，极大地简化了程序员的内存管理工作。另外，Java 也适合于在小型机上运行。基本解释器及类的支持只有 40KB 左右，加上标准类库和线程的支持也只有 215KB 左右。

2. 面向对象

Java 语言的设计集中于对象及接口，提供了简单的类机制及动态的接口模型，对象中封装了状态变量及相应的方法实现了模块化和隐藏；类则提供了类对象的原型，并且通过继承机制，子类可以使用父类所提供的方法，从而实现了代码的复用。

3. 分布式

Java 是为 Internet 的分布式环境而设计的，因此它能处理 TCP/IP。事实上，通过 URL 地址访问资源与直接访问一个文件的差别并不太大。Java 还支持远程方法调用（Remote Method Invocation，RMI），使程序能够跨网络调用方法。

4. 可靠性

Java 在编译和运行程序时，均检查可能出现的问题，以消除错误。Java 提供自动无用信息收集来管理内存，防止程序员在管理内存时产生错误。通过集成的面向对象的异常处理机制，在编译时 Java 会提示未被处理但是又可能出现的异常，帮助程序员正确地选择以防止系统崩溃。另外，Java 在编译时还可以捕获类型声明中的许多常见错误，防止动态运行时出现不匹配问题。

5. 安全性

用于网络及分布环境下的 Java 必须要防止病毒的入侵，Java 不支持指针，一切对内存的访问都必须通过对象的实例变量来实现，从而可以防止程序员使用木马等欺骗手段访问对象的私有成员，并且可以避免由于指针误操作而产生的错误。

6. 平台无关性

Java 解释器生成与体系结构无关的字节码指令，只要安装了 Java 运行时系统，Java 程序即可在任意的处理器上运行。这些字节码指令对应于 Java 虚拟机中的表示，Java 解释器得到字节码后对其进行转换，使之能够在不同的平台上运行。

7. 可移植性

由于 Java 具有平台无关的特性，因此 Java 程序可以方便地移植到网络上的不同机器中。同时，Java 类库中也实现了与平台无关的接口，使得类库可以移植。此外，Java 编译器由 Java 语言实现，Java 运行时系统由标准 C 实现，从而使得 Java 本身也具有可移植性。

8. 解释执行

Java 解释器直接解释执行 Java 字节码，字节码本身携带了许多编译信息，使得连接过程更加简单。

9. 高性能

Java 字节码的设计使之能够很容易地直接转换成对应于特定 CPU 的机器码，从而得到较高的性能。

10. 多线程

多线程机制使应用程序能够并发执行多线程，程序员可以分别用不同的线程完成特定的行为，而不需要采用全局事件循环机制，这样就很容易地实现网络上的实时交互行为。

11. 动态性

Java 程序带有多种运行时类型信息，用于在运行时校验和解决对象访问问题。这使得在一种安全、有效的方式下动态地链接代码成为可能，同时对 Applet 环境的健壮性也十分重要，因为在运行的系统中，可以动态地更新字节码内的小段程序。

1.1.3 Java 语言的应用前景

Java 语言具有广泛的应用前景，大体上可以从以下几个方面来考虑其应用。

（1）所有面向对象的应用开发，包括面向对象的事件描述、处理及综合等。

（2）计算过程可视化及可操作化的软件的开发。

（3）动态画面的设计，包括图形图像的调用。

（4）交互操作的设计（选择交互、定向交互及控制流程等）。

（5）Internet 的系统管理功能模块设计，包括 Web 页面的动态设计、管理和交互操作设计等。

（6）Intranet（企业内部网）上的软件开发（直接面向企业内部用户的软件）。

（7）与各类数据库连接查询的 SQL 语句的实现。

（8）其他应用类型的程序。

根据 Oracle 公司最新 Java 战略及发展方向，如图 1-1 所示，可以看出 Java 语言的最新应用前景如下。

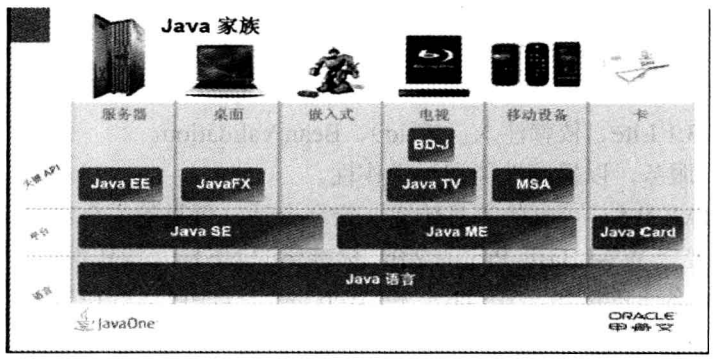


图 1-1　Java 战略及发展方向

（1）针对新型应用模型和硬件技术优化 Java。

1）提高 Java 开发效率。

2）为 Java 虚拟机增添模块化特性。

3）针对新处理器、内存和网络而优化。

4）改善性能、监控和诊断。

5）让 Java 虚拟机支持多种开发语言。

（2）Java 平台。计划要提供增强。

1）多核处理器、大内存、高速网络。

a）分支/合并（Fork/Join）框架和其他多线程增强。

b）超大型堆低停顿的垃圾回收机制。

c）消除 HotSpot 中的持久代（Permanent Generation）。

d）改进的网络功能——内置 Infiniband 支持、10Gb 以太网、SDP 和 SCTP。

e）新 I/O API——文件系统和异步 I/O 具有更好的操作系统互操作性。

2）Java VM 支持多语言。

a）InvokeDynamic 字节码提高动态语言的性能。

b）在多核处理器上自动扩展动态语言。

c）显著加速的 JavaScript 引擎。

（3）应用服务器设计目标。针对新型应用开发模型优化 Java 应用服务器。

1）通过依赖注入使应用服务器模块化。

2）为 Web 应用程序提供新型的轻型 Web Profile。

3）显著提升 POJO 和 EJB 编程效率。

4）改进 Java Web 服务，极大地提高性能和互操作性。

5）改善与脚本语言和动态语言间的互操作性。

6）使应用服务器模块化。

a）2009——基于参考实现 HK2 的微内核。

b）2010——企业 OSGI 规范，JPA、JNDI、JDBC、JTA、HTTP 服务。

c）2010——OSGI 和 JavaEE 混合编程模型。

7）为 Web 应用程序提供新的轻型 Web Profile。

a）2009——参考实现中提供的 JSR 316 Web Profile。

b）2010——集群化的 Web Profile。

8）提升 POJO 和 EJB 编程效率。

2009——EJB 3.1 Lite、依赖注入（Weld）、Bean Validation。

9）改进 Web 服务，以提高性能和互操作性。

a）2010——JAX WS、可靠的消息传递、安全会话、可靠的安全协议。

b）2010——遵守 WS-I Basic Profile 2.0，标准化的.NET 互操作性。

（4）Java 在设备上的应用。让 Java 和 Web 应用程序可运行在所有终端设备上，如图 1-2 所示。

图 1-2 Java 在设备上的应用前景

1）针对移动设备和语言特点改造 Java。

2）集成 Web 技术（HTML、JavaScript、CSS）。

3）新增访问硬件和操作系统功能的 API。

4）占用空间小、CPU 效率高的 Java 内核，适用于卡、电视、移动设备。

5）为各种 Java 设备提供一致的开发工具和模拟器。

1.1.4 Java 运作机制

一些编程语言（如 C++）的编译器是根据源文件，直接输出可执行的 EXE 文件。与之相

比，Java 编译器的输出并不是可执行的代码，而是字节码（byte code）。

所谓字节码，是一套能在 Java 虚拟机下执行的高度优化的指令集，在内存中也只不过是 1010……这样的字节编码。Java 虚拟机（Java Virtual Machine，JVM）从其表现形式来看，是一个字节码解释器，即可以解释执行字节码定义的动作。

从操作系统的级别层次来看，它是构架在不同操作系统上的，能屏蔽不同操作系统差异的基于软件的平台。虚拟机能保证用户看到其注重的代码的运行结果，而向用户屏蔽其不关心的代码在不同操作系统里的执行细节。这就好比在安装完声卡的驱动程序后，Windows 操作系统只向用户展示声卡的发声效果，而不是声卡的工作方式。

将 Java 程序解释成字节码，而不是最终的面向具体操作系统的可执行文件，可以让它运行在不同平台的虚拟机上，事实上，只要在各种操作系统上都安装不同的 Java 虚拟机就可以做到这点。所以，在特定的操作系统中，只要有支持 Java 功能的 jar 包存在，Java 程序就可以运行了。尽管不同平台的 Java 虚拟机和对应的支持 jar 包都是不同的（它们不应该相同），但它们的作用都是解释并运行 Java 字节码。因此，字节码的解释与运行机制是保证程序能很容易地在不同操作系统上运行。

图 1-3 所示为 Java 虚拟机的工作方式。

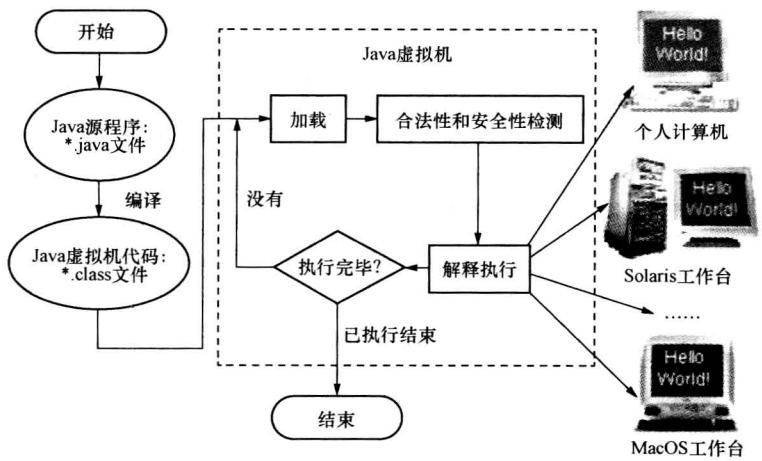


图 1-3　Java 虚拟机的工作方式

从 Java 解释执行的运行机制的角度上，能看到其跨平台的特性和保证其拥有跨平台的支持机制。

1.2　面向对象编程

目前，面向对象的思想被软件开发界广为追捧，其实，面向对象思想不是高深的理论，而是根据前人大量编程项目总结出来的一套分析和解决编程方面问题的方法。

以面向对象思想为指导，可以优化 Java 代码的结构，更可以让数据库访问模块变得富有弹性——即让代码和模块能更好地适应项目需求的频繁变化。

1.2.1 面向对象思想提出

计算机语言刚出现时，采用的编程语言是面向硬件的机器语言，即用一大堆计算机硬件能识别的 0 和 1 来编写代码。写这种机器代码需要很高的技巧，它们调试起来也比较困难，所以当时的程序员都是精英级别的人物。

为了提高编程语言的适用性，提升代码编写的效率，专家们开发了以汇编语言为代表的低级语言，即用一些助记符来代替晦涩的机器语言，例如用 add a b，来计算 $a+b$ 的值。这类语言直接与内存和其他硬件打交道，以使用最小系统资源和谋求最大硬件利用率为开发准则，在当时低配置的硬件环境下是有很大的实用意义的。

但是，用这种编程语言开发出来的代码的结构非常混乱，并且很难读懂。这样的情况可能在几百行规模的代码量下并不明显，但如果当项目规模扩大，代码量越来越多的时候，这个问题就突显出来了。更有甚者，作者自己在开发结束后，往往会忘记自己写的代码的结构和语意，这样的代码很难维护和阅读。

为了解决这类问题，提高代码的逻辑性，软件语言方面的专家提出了初步融合面向对象思想的结构化的面向过程语言。

结构化语言的重要特征是，在代码里分离了代码逻辑和程序数据，并使用了分支（IF）和循环（While 和 For）等关键字来优化代码的逻辑结构。由于结构化的语言使用子模块顺序调用的方式控制代码流程，所以又称为面向过程语言，以 C 语言为代表。

和汇编语言相比，面向过程语言能很大程度地改善代码结构，提高代码的逻辑性。具体表现为以下两点。

（1）由于在面向过程的语言里，可以把功能点以函数（过程）的形式封装，所以代码的重用性能会得到一定程度的提高，即后继项目可以通过调用具有相同功能的函数来利用现有的代码，这对项目的二次开发或者是功能升级是非常有利的。

（2）由于面向过程语言摒弃了使用 goto（或者是 jump）语句的跳转方式，而使用了分支和循环等结构性语句，所以代码的结构能得到大幅度的改善，从而能在一定程度上提高项目的可读性和可维护性。

然而，在面向过程语言的使用过程中，程序员们发现其依然有本质上的缺陷，具体表现如下。

（1）在模块间的函数调用时，由于无法屏蔽模块里不希望被访问修改的关键变量，这将导致模块间有很高的关联度，这对代码维护是非常不利的，往往会修改一处而触动全部代码。

（2）代码的重用级别仅仅是局限于函数级别的，这导致无法大量有效地利用已开发完成的代码成果，这对代码的重复性利用非常不利。

（3）函数的定义是针对具体动作和具体对象的。比如说，要定义打印的方法，如果使用的打印机型号不同，那就不得不根据不同的型号定义多个不同名的方法。也就是说，面对业务逻辑（业务上要做的事情）相同但业务逻辑处理对象不同的需求，必须要写成多个不同名的函数，这导致了代码过于注重细节，而无法考虑逻辑性、重用性和维护性等大局方面的问题。

为了解决上述问题，软件语言的专家们吸取面向过程语言中“结构化”和“模块化”等优秀方法，提出了面向对象的指导思想。

面向对象思想参照了现实生活中观察问题、解决问题的思维方式，将应用程序中的一切模型以类的形式，整合成对象，并通过归纳和抽象对象，提取出一套解决问题的方法和思路。

所以，面向对象思想可以通过优化模块间的结构和加固模块间的坚固性，来指导构造软件项目行业“摩天大楼”。

1.2.2 类与封装

类是基于面向对象思想编程语言的基础，程序员可以把具有相同业务性质的代码封装到一个类里，通过接口方法向外部代码提供服务，同时向外部代码屏蔽类里服务的具体实现方式。

对象是类的实例，类一般是个抽象名词，比如“人类”，而对象则是具体的物质存在，比如“张三”这个人。

在现实生活中，经常会遇到“类”和“对象”这些概念，比如封装了能实现“用微波加热材料”功能的微波炉类。这里，用户可以使用面板上的按钮，使用该类里提供的加热等服务，并且，由于该类实现了封装，所以在使用的时候，用户可以不用了解其中的微波加热原理以及实现机制。通过上面例子可以看出，类是对现实生活中真正存在的对象的描述，并且这些对象都具有共同的属性和行为。但是，根据不同的系统需求，同样的一种对象会被描述成具有不同属性和行为的类。比如，对于银行系统，人的这个类应该具有账号、存款余额的属性和存钱、取钱的行为，每个人都有自己不同的账号和相关金额，但是他们存钱取钱的流程是相通的；而对于电信系统，同样的人这个类应该具有手机号码、卡内余额的属性和充费、扣费的行为。因此，要注意类的定义和它所封装的行为是否能够正确地反映实际系统的需求。

1.2.3 抽象类与代码复用

抽象类也是类的一种，它同样拥有属性和行为，但是它与普通类最大的区别就在于抽象类里面允许有一些抽象的行为，抽象类只从概念的角度出发定义了类的行为，但是它没有描述行为的实现细节，而是把这个任务交给了它的子类来实现，它本身充当了父类的角色。因此，可以说子类不仅有从父类继承过来的属性和行为，而且它可以在父类原有内容的基础上做一些补充和发挥。父类只是做了一个概念的抽象。

图 1-4 说明了这种抽象的必要性。

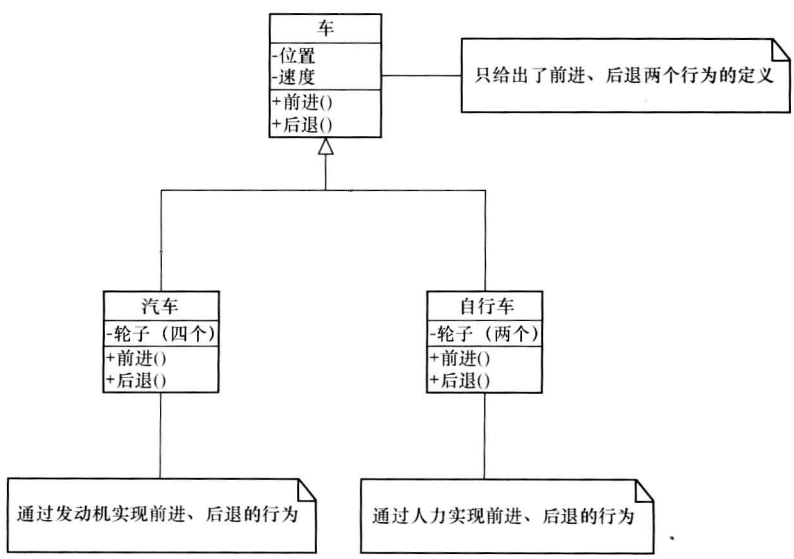


图 1-4　抽象类示意图

对于汽车、自行车这两个类来说，它们都是父类车的子类，具有位置、速度属性和前进、后退行为，但是汽车有四个轮子，自行车有两个轮子，汽车的运动是靠发动机来实现的，而自行车的运动是靠人力来实现的。因此，在车这个父类里面只定义了关于运动的两个方法，具体的实现细节交给了它的两个子类来完成。这样不仅避免了在两个相近的类之间定义重复的属性和行为，而且解决了允许不同的类具体的实现细节有所变化的问题。

抽象类可以帮助我们更加有效地解决代码重复问题，并且，它又不失灵活性，又同时达到了概念抽象的目的。这里又通过抽象类的概念引出了类之间，一个重要关系，就是子类与父类的继承，当然不仅抽象类可以作为父类，其他的一般类同样可以充当为某一子类的父类。只要它有这种继承的需求就可以这样来定义。

1.2.4 接口与功能抽象

接口的功能与抽象类相似，但是接口只能定义行为，这些行为的具体细节在实现了接口的类中描述，而抽象类还可以为这些类定义一些相同的属性，因此可以说，接口比抽象类要更加抽象。接口中只定义公共的行为，这些行为也就是实现它的那些类的功能，不同的类可以有不同的实现细节，这里的原理与抽象类相同。

图 1-5 说明了接口是如何做到功能抽象的。

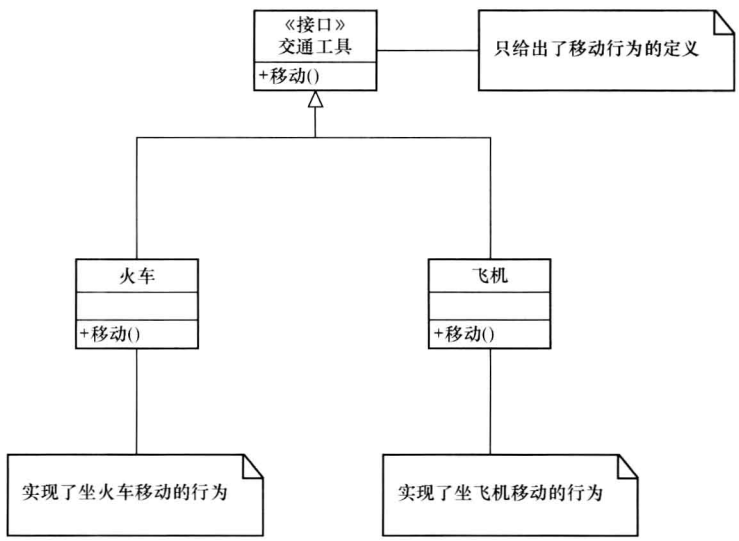


图 1-5 接口示意图

这里定义了交通工具这个接口，在这个接口里面定义了移动这个行为，但是行为的具体实现细节是在火车和飞机两个类里面完成的，并且是由不同的方式来实现的。假设某个用户有了一个从上海到北京的行程安排，并且由于时间紧，不能确定选择哪一种具体的交通工具，只知道要发生移动这个行为。那么，他可以先选择交通工具这个接口的移动行为，具体是坐火车还是坐飞机可以根据实际的情况来定。

接口是单单从行为的角度出发，把功能做了概念上的抽象，使得不同的类通过不同的机制实现名称相同的行为，方便使用者调用，隐藏了具体的实现细节。

1.2.5 多态与重载

多态是面向对象编程的一个重要特征，从字面意思来看就是指具有多种形式或形态的情