目　　录

第1章　绪论

本章先向读者介绍Java语言的基础知识，再介绍如何使用命令行输出程序，最后介绍Eclipse开发工具的使用。

1.1　Java语言简介

Java语言是一门面向对象的编程语言，它不仅吸收了C++语言的各种优点，还摒弃了C++语言中难以理解的多继承、指针等概念，所以Java语言具有功能强大和简单易用两个特征。Java语言作为静态面向对象编程语言的代表，极好地实现了面向对象理论，允许程序员以优雅的思维方式进行复杂的编程。

1.1.1　Java语言平台无关性

1. 什么是平台

Java语言是可以跨平台的编程语言，那什么是平台呢？无论使用哪种编程语言编写的应用程序，都需要经过操作系统和处理器来完成程序的运行，所以这里的平台由操作系统（Operating System，OS）和处理器（Central Processing Unit，CPU）构成。与平台无关是指因操作系统和处理器不同造成的编译程序不能运行或运行错误。

2. Java语言跨平台的原理

C语言编译执行过程如图1-1所示。

如果您有使用C语言的开发经历，看到图1.1将会非常轻松。我们知道，只要是用标准C开发的程序，使用不同的编译器编译后的可执行文件是可以在对应平台运行的，例如，

Windows可以使用Visual C++编译，编译后的. exe文件就可以在Windows下运行；Linux下可以使用GCC编译，生成的可执行文件就可以在Linux上运行。不同平台的编译软件不能相互使用。

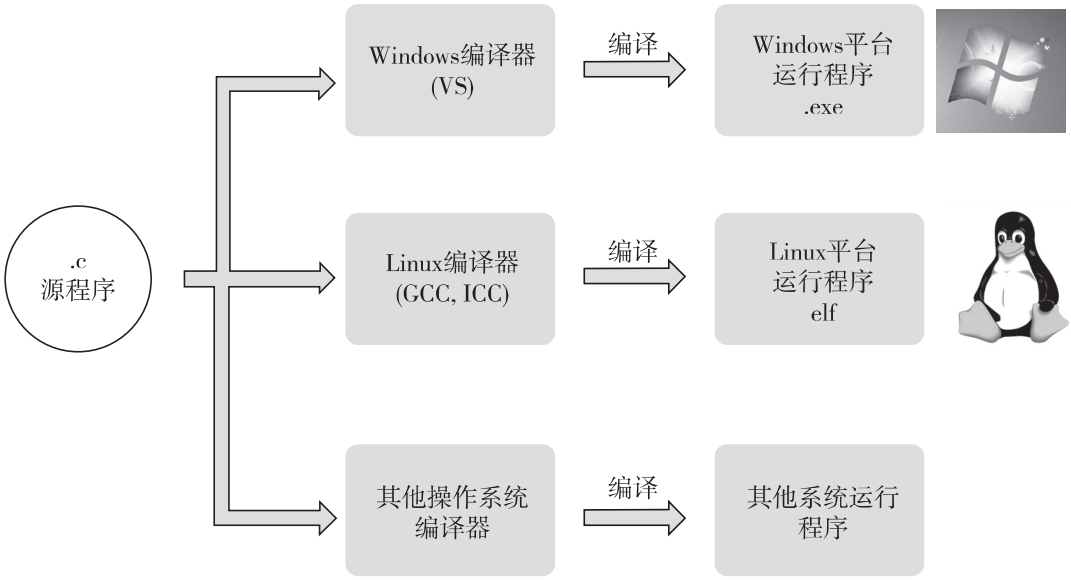

图1-1　C语言编译执行过程

Java程序实际上是在Java虚拟机（JRE是软件实现）中运行，Java虚拟机类似于一个模拟执行环境，在不同的操作系统上拥有不同的Java虚拟机实现，但是这些Java虚拟机遵循统一的规范来解释. class文件，并将. class文件中的指令转换为本地操作系统对应的指令，这样就实现了相同的. class文件。可以通过Java虚拟机转换为对应操作系统上的对应指令实现. class文件，也就是Java程序的跨平台性。Java语言代码执行过程如图1-2所示。

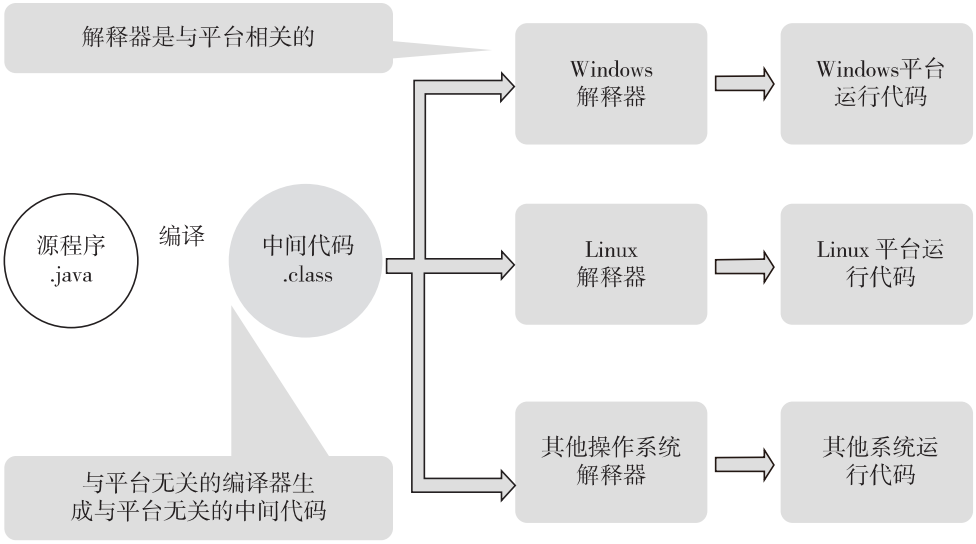

图1-2　Java语言代码执行过程

图1-2中的. java就是源程序，类似于C语言的. c，生成的中间代码是. class，通过Java虚拟机翻译成不同平台的机器执行代码。同一个. class文件在不同的虚拟机中会得到不同的机器指令（Windows和Linux的机器指令不同），但是最终的执行结果都是相同的。

第一种是编译执行，如上文中说到的C语言，它把源程序由特定平台的编译器一次性编译为平台相关的机器码，它的优点是执行速度快，缺点是无法跨平台；第二种是解释执行，如HTML、JavaScript，它使用特定的解释器，把代码一行行解释为机器码，类似于同声翻译，它的优点是可以跨平台，缺点是执行速度慢，暴露源程序；第三种是从Java语言开始引入的“中间代码+虚拟机”的方式，它既整合了编译语言与解释语言的优点，同时又如虚拟机那样可以解决如垃圾回收、安全性检查等这些传统语言中难以解决的问题，所以其后微软公司的. NET平台也使用了这种方式。

1.1.2　Java语言的发展历史

Java语言有二十几年的历史，已发展成为人类计算机史上影响深远的编程语言。Java语言所崇尚的开源、自由等精神，吸引了全世界无数优秀的程序员。事实上，从来没有一门编程语言能吸引这么多的程序员，也没有一门编程语言能衍生出如此多的开源框架。

Java语言是一门非常纯粹的面向对象的编程语言，它吸收了C++语言的各种优点，又摒弃了C++语言中难以理解的多继承、指针等概念，所以Java语言具有功能强大和简单易用两个特征。

从某种程度上来看，精通了Java语言的相关知识，相当于系统地学习了软件开发的相关知识，而不是仅仅学完了一门编程语言。

1. Java语言的由来

Java语言最开始只是Sun公司在1990年12月开始研究的一个内部项目。Sun公司的一位名叫帕特里克·诺顿的工程师被自己开发的C语言和C语言编译器搞得焦头烂额，因为其中的API极其难用。帕特里克决定改用NeXT，同时他也获得了研究公司的一个“Stealth计划”项目的机会。

“Stealth计划”后来改名为“Green计划”，詹姆斯·高斯林和麦克·舍林丹也加入了帕特里克的工作小组。他们和其他几个工程师一起在加利福尼亚州门罗帕克市沙丘路的一个小工作室里面研究开发新技术，瞄准下一代智能家电（如微波炉）的程序设计，Sun公司预料未来科技将在家用电器领域大显身手。团队最初考虑使用C语言，但是很多成员包括Sun公司的首席科学家比尔·乔伊，发现C语言和可用的API在某些方面存在很大问题。

工作小组使用的是内嵌类型平台，可以用的资源极其有限。很多成员发现C语言太复杂以至于经常被错误使用。他们还发现C语言缺少垃圾回收系统，以及可移植的安全性、分布程序设计和多线程功能。最后，他们想要一种易于移植到各种设备上的平台。

根据可用的资金，比尔·乔伊决定开发一种集C语言和Mesa语言搭成的新语言，在一份报告上，乔伊把它叫作“未来”，他提议Sun公司的工程师应该在C语言的基础上，开发一种面向对象的环境。最初，高斯林试图修改和扩展C语言的功能，他称这种新语言为C--，但是他放弃了。后来，他创造出一种全新的语言——Oak（橡树），以他的办公室外的树而命名。

就像很多开发新技术的秘密工程一样，工作小组没日没夜地工作，直到1992年的夏天，他们能够演示新平台的一部分了，包括Green操作系统、Oak的程序设计语言、类库和其硬件。最初的尝试是面向一种类PDA设备，被命名为Star7，这种设备有鲜艳的图形界面和被称为Duke的智能代理来帮助用户。1992年12月3日，这台设备进行了展示。

1992年11月，Sun公司的全资子公司——FirstPerson有限公司，被安排到了帕洛阿尔托。FirstPerson团队对建造一种高度互动的设备感兴趣，当时代华纳发布了一个关于电视机顶盒的征求提议书（Request for Proposal）时，FirstPerson改变了他们的目标，作为对征求意见书的响应，他们提出了一个机顶盒平台的提议。但是有线电视业界觉得FirstPerson的平台给予用户过多的控制权，所以FirstPerson的投标败给了SGI，与3DO公司的另外一项关于机顶盒的交易也没有成功。由于他们的平台不能在电视工业产生任何效益，因此该子公司被并回Sun公司。

1994年6～7月，约翰·盖吉、詹姆斯·高斯林、比尔·乔伊、帕特里克·诺顿、韦恩·罗斯因和埃里克·斯库米，在经历了一场历时三天的头脑风暴之后，团队的决定再一次改变了他们努力的目标，这次他们决定将该技术应用于万维网。他们认为随着Mosaic浏览器的到来，因特网正在向同样的高度、互动的远景演变，而这一远景正是他们在有线电视网中看到的。作为原型，帕特里克·诺顿写了一个小型万维网浏览器——WebRunner，后来改名为Hotjava。同年，Oak改名为Java。商标搜索显示，Oak已被一家显卡制造商注册，因此团队找到了一个新名字。这个名字是在很多成员常去的本地咖啡馆中杜撰出来的。名字是不是首字母缩写还不清楚，很大程度上来说不是。虽然有人声称是开发人员名字的组合：James Gosling（詹姆斯·高斯林）、Arthur Van Hoff（阿瑟·凡·霍夫）、Andy Bechtolsheim（安迪·贝克托克姆），或Just Another Vague Acronym（只是另外一个含糊的缩写）。还有一种比较可信的说法是这个名字是出于对咖啡的喜爱，所以以Java咖啡来命名。类文件的前四字节如果用十六进制阅读，分别为CA FEBA BE，就会拼出两个单词CAFE BABE（咖啡宝贝）。

1994年10月，HotJava和Java平台为公司高层进行演示。1994年，Java 1.0a版本已经可以提供下载，但是Java和Hotjava浏览器的第一次公开发布却是在1995年5月23日SunWorld大会上进行的。Sun公司的科学指导约翰·盖吉宣告Java技术的诞生。这个发布是与网景公司的执行副总裁马克·安德森的惊人发布一起进行的，其宣布网景将在其浏览器中包含对Java的支持。1996年1月，Sun公司成立了Java业务集团，专门开发Java技术。

1.1.3　Java语言的应用领域

如果你刚开始学习Java语言，你可能会思考Java语言会确切地运用在哪些地方。除了Minecraft（《我的世界》，一款沙盒游戏），你可能无法看到用Java语言编写的其他游戏吧？像Adobe Acrobat这样的桌面工具、Microsoft办公软件，这些都不是用Java语言编写的，甚至就连Linux或者Windows的操作系统也不是，那么人们到底在哪里使用了Java语言呢？Java语言到底有没有实际的应用程序呢？

1. 安卓App

如果你想知道Java语言应用在哪里，那么答案就会离你不远。打开你的安卓手机或者任何一款App，它们完全是用有谷歌Android API的Java编程语言编写的，这个API和JDK非常相似。前几年安卓刚开始起步，而如今已经有很多Java程序员从事安卓App的开发。顺便一提，虽然安卓使用了不同的Java虚拟机和不同的封装，但是代码仍然是用Java语言编写的。

2. 在金融服务行业的服务器应用

Java在金融服务业有着广泛的应用。很多全球性投资银行，如Goldman Sachs（高盛投资公司）、Citigroup（花旗集团）、Barclays（巴克莱银行）、Standard Charted（英国渣打银行）和一些其他银行，都用Java语言编写前台和后台的电子交易系统，结算、信息确认系统、数据处理项目和其他的项目。Java语言被运用于编写服务器端应用，但大多数没有前端，都是从一个服务器端（上一级）接收数据，处理数据后发向其他处理系统（下一级）。Java Swing由于能开发出图形用户界面的客户端供交易者使用而备受欢迎，但是现在C#语言正在快速地取代Swing的市场，这让Swing更有压力。

3. 网站应用

Java同样在电子商务和网站开发上有着广泛的应用。你可以运用很多RESTfull架构，这些架构是用Spring MVC、Struts 2.0和类似的框架开发出来的。甚至简单的Servlet、JSP和Struts在各种政府项目也备受欢迎，许多政府、医疗、保险、教育、国防和其他部门的网站都是建立在Java语言之上的。

4. 软件工具

很多有用的软件和开发工具都是运用Java语言编写和开发的，例如Ecilpse、InetelliJ Idea和Netbans IDE。这些都是经常使用的用Java语言编写的桌面应用程序，就如上面所说，Swing曾经在图形用户界面的客户端开发领域非常流行，它们大多数应用在金融服务领域以及投资银行。虽然现在Java Fx正在逐渐地流行起来，却仍然无法替代Swing，但C#语言已经在大部分金融领域中代替了Swing。

5. 交易系统

第三方交易系统是金融服务行业的一大部分，同样也是使用Java语言编写的。例如Murex这种受欢迎的交易系统，运用于与许多的银行前端连接，同样也是用Java语言编写的。

6. J2ME App

虽然iOS和Android的到来几乎扼杀了J2ME的市场，但是仍然有很多的低端手机在使用J2ME。曾经有段时间大部分的游戏和手机应用都是利用MIDP和CLDC，或者J2ME部分平台编写的，以适用于Android系统。J2ME依然在蓝光、磁卡、机顶盒等产品中流行着。App之所以如此流行是因为对于所有的低端手机，App仍然适用于J2ME。

7. 嵌入式领域

Java在嵌入式领域也有广泛的应用。你只需要130KB就能够使用Java技术（在一块小的芯片或者传感器上），这显示了这个平台的可靠性。Java最初是为了嵌入式设备而设计的。事实上，这也是Java最初的一项“编写一次，到处运行”主旨的一部分。

8. 大数据技术

Hadoop和其他的大数据技术也在不同程度上使用着Java，例如Apache的基于Java的Hbase、Accumulo（开源）以及ElasticSearch。但是Java并没有占领整个领域，还有其他大数据技术，例如MongoDB就是使用C++语言编写的。如果Hadoop和ElasticSearch逐渐发展，那么Java就能有潜力在大数据技术领域上得到更大的发展空间。

9. 高频交易领域

Java平台已经大大提高了其性能特点和JITS，并且Java也拥有C++级别的传输性能。所以，Java也流行于编写高并发系统。虽然Java的传输性能比不上C++，但用户可以不用考虑Java的安全性、可移植性和可维护性等问题（Java内部已经实现好了），而且Java有着更快的运行速度。安全性等问题会使一个没有经验的C++程序员编写的应用程序变得更加缓慢和不可靠。

10. 科学应用

现在Java经常是科学应用的默认选择，包括自然语言处理。这最主要的原因是Java比起C++或者其他语言有更高的安全性、可移植性、可维护性，而且Java有着更好的高级并发工具。

20世纪90年代，由于Applet的原因，Java在互联网中占据了重要地位，但是这几年，因为Applet的沙箱模型存在各种安全隐患，Applet越来越失去人气，导致如今桌面Java和Applet几乎都消失了，但是Java仍然作为软件行业默认的应用开发语言，且在金融服务业、投资银行和电子商务领域有着广泛的应用。任何学习Java的人都认为自己有着光明的未来。另外，Java 8加强了一个概念——在未来的几年，Java将继续统治着软件开发领域。

1.1.4　Java语言的地位

1. 网络地位

网络已经成为信息时代最重要的交互媒介，那么基于网络的软件设计就成为软件设计领域的核心。Java的平台无关性让Java成为编写网络应用程序的佼佼者. 而且Java也提供了许多以网络应用为核心的技术，使得Java特别适用于网络应用软件的设计与开发。

2. 语言地位

Java语言面向对象编程，并涉及网络、多线程等重要的基础知识，是一门很好的面向对象语言。通过学习Java语言不仅可以掌握使用对象来完成某些任务及面向对象编程的基本思想，而且会为今后进一步学习设计模式奠定一个较好的语言基础。C语言无疑是非常基础和实用的语言之一。目前，Java语言已经获得了和C语言同样重要的地位，即其不仅是一门正在被广泛使用的编程语言，而且已经成为软件设计开发者应当掌握的一门基础语言。

3. 需求地位

目前，由于很多新技术领域都涉及Java语言，例如用于设计Web应用的JSP、设计手机应用的Android等，使得IT行业对Java人才的需求正在不断地增长。人们可以经常看到许多培训或招聘Java软件工程师的广告，所以，掌握Java语言及其相关技术意味着会拥有较好的就业前景和工作酬金。

1.2　Java语言的特性

Java语言是一种跨平台、适用于分布式计算环境的面向对象的编程语言。具体来说，它具有如下特性：简单性、面向对象、分布性、编译和解释性、稳健性、安全性、可移植性、高性能、多线程性和动态性等。

1. 简单性

Java语言的设计类似于C++语言，但是为了使语言小和容易熟悉，设计者们把C++语言中许多可用的特征去掉了，这些特征是一般程序员很少使用的。例如，Java不支持go to语句，代之提供break和continue语句以及异常处理。Java还剔除了C++的操作符过载（overload）和多继承特征，并且不使用主文件，免去了预处理程序。因为Java没有结构，

数组和串都是对象，所以不需要指针。Java能够自动处理对象的引用和间接引用，实现自动的“无用单元收集”，使用户不必为存储管理问题烦恼，能将更多的时间和精力花在研发上。

2. 面向对象

Java语言是一种面向对象的语言。对程序员来说，这意味着要注意其中的数据和操纵数据的方法（method），而不是严格地用过程来思考。在一个面向对象的系统中，类（class）是数据和操作数据的方法的集合。数据和方法一起描述对象（object）的状态和行为。每一对象是其状态和行为的封装。类是按一定体系和层次安排的，使得子类可以从超类继承行为。在这个类层次体系中有一个根类，它是具有一般行为的类。Java程序是用类来组织的。

Java还包括一个类的扩展集合，分别组成各种程序包（package），用户可以在自己的程序中使用。例如，Java提供产生图形用户接口部件的类（java. awt包），这里awt是抽象窗口工具集（abstract window toolkit）的缩写，处理输入输出的类（java. io包）和支持网络功能的类（java. net包）。

3. 分布性

Java语言支持在网络上应用，它是分布式语言。Java语言既支持各种层次的网络连接，又以Socket类支持可靠的流（stream）网络连接，所以用户可以产生分布式的客户机和服务器。

网络变成软件应用的分布运载工具，Java程序只要编写一次，就可到处运行。

4. 编译和解释性

Java编译程序生成字节码（byte-code），而不是通常的机器码。Java字节码提供对体系结构中性的目标文件格式，代码设计成可有效地传送程序到多个平台。Java程序可以在任何实现了Java解释程序和运行系统（run-time system）的系统上运行。

在一个解释性的环境中，程序开发的标准连接阶段大大消失了。如果说Java还有一个连接阶段，它只是把新类装进环境的过程，它是增量式的，轻量级的过程。所以，Java支持快速原型和容易试验，它将导致快速程序开发。这是一个与传统的、耗时的“编译、连接和测试”形成鲜明对比的精巧的开发过程。

5. 稳健性

Java语言原来是用作编写消费类家用电子产品软件的语言，所以它被设计成编写具有高可靠性和稳健性的软件。Java语言消除了某些编程错误，用它写可靠软件相当容易。

Java语言是一个强类型语言，它具有允许扩展编译时检查潜在类型不匹配问题的功能。Java语言要求显式的方法声明，它不支持C语言风格的隐式声明。这些严格的要求保证编译程序能捕捉调用错误，确保程序的可靠性。

可靠性方面最重要的增强之一是Java的存储模型。Java不支持指针，它消除重写存储

和讹误数据的可能性。类似地，Java自动的“无用单元收集”预防存储泄漏及其他有关动态存储分配和解除分配的有害错误。Java解释程序也执行许多运行时的检查，诸如验证所有数组和串访问是否在界限之内。

异常处理是Java中使得程序更稳健的另一个特征。异常是某种类似于错误的异常条件出现的信号。使用try-catch-finally语句，程序员可以找到出错的处理代码，这就简化了出错处理和恢复的任务。

6. 安全性

Java的存储分配模型是它防御恶意代码的主要方法之一。Java没有指针，所以程序员不能得到隐蔽起来的内幕和伪造指针去指向存储器。更重要的是，Java编译程序不处理存储安排决策，所以程序员不能通过查看声明去猜测类的实际存储安排。编译的Java代码中的存储引用在运行时由Java解释程序决定实际存储地址。

Java运行系统使用字节码验证过程来保证装载到网络上的代码不违背任何Java语言限制。这个安全机制部分包括类如何从网上装载。例如，装载的类是放在分开的名字空间而不是局部类，预防恶意的小应用程序用它自己的版本来代替标准Java类。

7. 可移植性

Java语言使得语言声明不依赖于实现的方面。例如，Java显式说明每个基本数据类型的大小和它的运算行为（这些数据类型由Java语法描述）。

Java环境本身对新的硬件平台和操作系统是可移植的。Java编译程序也用Java编写，而Java运行系统用ANSI C语言编写。

8. 高性能

Java语言是一种先编译后解释的语言，所以它不如全编译性语言快。但是有些情况下性能是很重要的，为了支持这些情况，Java设计者制作了“及时”编译程序，它能在运行时把Java字节码翻译成特定CPU（中央处理器）的机器代码，也就是实现了全编译。

Java语言字节码格式在设计时考虑这些“及时”编译程序的需要，所以生成机器代码的过程相当简单，它能产生相当好的代码。

9. 多线程性

Java语言是多线程语言，它提供支持多线程的执行（也称为轻便过程），能处理不同任务，使具有线程的程序设计很容易。Java的lang包提供一个Thread类，它支持开始线程、运行线程、停止线程和检查线程状态的方法。

Java的线程支持也包括一组同步原语。这些原语是基于监督程序和条件变量风范，由C.A.R. Haore开发的广泛使用的同步化方案。利用关键词synchronized，程序员可以说明某些方法在一个类中不能并发地运行。这些方法在监督程序控制之下，确保变量维持在一个一致的状态。

10. 动态性

Java语言适应于变化的环境，它是一个动态的语言。例如，Java语言中的类是根据需要载入的，甚至有些是通过网络获取的。

1.3 Java编程规范

建立一个可行可操作的编程标准、约定和指南，以规范人们的代码开发工作，提高代码的可读性，提高系统的健壮性、稳定性、可靠性。通过遵循这些程序设计标准，作为一个Java软件开发者的生产效率会有显著提高。经验表明，若从一开始就花时间编写高质量的代码，则在软件开发阶段，对代码的修改要容易很多。最后，遵循一套通用的程序设计标准将带来更大的一致性，使软件开发团队的效率明显提高。

1.3.1 包的命名与注释

1. 包的命名

包的命名都是由小写的英文字母组成，每个包名称之间用点号分隔开来，如java.lang和javax. servlet. http等。

全局包的名字用所在机构的Internet保留域名开头，如com. sun和com. oracle等。

我们的命名约定：

com. excellence. common：excellence下的通用文件，可能被以后不同项目通用的类文件。

com. excellence. exoa4j：exoa2004forjava项目的Java分模块分类存放。

在com. excellence. exoa4j下面，我们按照模块分开各自的包。

2. 注释包

应保证有一个或者多个外部文档以说明相应机构所创建的包的用途。对于每个包，应说明如下内容。

（1）包的基本原理。其他开发者需要了解一个包是用来做什么的，这样他们才能判断是否可以用它；如果是一个共享包，他们可以判断是否需要改进或是扩展它。

（2）包中的类。在包中要包含一个类和接口的列表，每个类或接口用简短的一行文字来说明，以便让其他的开发者了解这个包中含有什么。

技巧：生成一个以包名命名的HTML文件，将它放到包的适当的目录中去。这个文件应具有扩展名．html。

1.3.2　类、接口的命名及注释

1. 命名类

标准Java类约定使用完全的英文描述符，所有单词的第一个字母要大写，并且单词中大小写混合。

类名应是单数形式。

示例：

```
Customer Employee    Order    OrderItem    FileStream    String
```

相关的类如果使用了某些已经约定的开发模式，应该使用相关约定的命名，如LoggerFactory、UserDAO等。

在同一个模块中的相关类，可以采用同样开头的名称以区分它们的相关性，如UserValue、UserDAO等。

2. 命名接口

采用完整的英文描述符说明接口封装，所有单词的第一个字母大写。习惯上，名字后面加上后缀able、ible或者er。

示例：

```
Runnable    Contactable    Compressible    Prompter
```

3. 命名编译单元

编译单元在这种情况下是一个源码文件，应被赋予文件内定义的主要的类或接口的名字。用与包或类相同的名字命名文件，大小写也应相同。.java应作为文件名的扩展名。

示例：

```
Customer.java
Singleton.java
DeptInfo.java
```

4. 类及接口的注释

以下信息应写在文档注释中紧靠类的定义的前面。

（1）类的目的和作用。开发者需要了解一个类的一般目的，以判断这个类是否满足他们的需求。开发者应养成一个注释与类有关的任何事项的习惯。

（2）已知的问题。如果一个类有任何突出的问题，应说明出来，以让其他开发者了解这个类的缺点／难点。此外，还应注明为什么不解决该问题的原因。

（3）类的开发／维护历史。通常要包含一个历史记录表，列出日期、类的作者和修改概要。这样做的目的是让进行维护的程序员了解过去曾对一个类所做的修改，是谁做了什么样的修改。

（4）版权信息。

对于以上几点，其中类的目的和作用是必须要填写的。采用javadoc形式标志的一个示例如下：

```
/**
* <p>类说明：LoggerFactory，创建Logger的工厂类</p>
*  < p> Copyright  2003:  Excellence Network Co., LTD< /p>
* @ author fjy
* @ version1.0
* @ since 2017-06-10
* @modified fjy 2017-06-13修改了××××
* @modified fjy 2017-06-23修改了××××
*/
```

1.3.3 成员函数的命名及注释

1. 命名成员函数

成员函数的命名应采用完整的英文描述符，大小写混合使用，所有中间单词的第一个字母大写。成员函数名称的第一个单词常常采用一个具有强烈动作色彩的动词。

示例：

```
openAccount ( )      printMailingLabel ( )      createUser ( )      delete ( )
```

这种约定常常使人一看到成员函数的名称就能判断它的功能。虽然这种约定要使开发者多做一些输入工作，而且函数名常常较长，但是可以提高代码的可理解性。

2. 获取函数

获取函数作为一个成员函数，返回一个字段的值。除了布尔字段之外，还应采用get作为字段的前缀，布尔字段采用is作为前缀。

示例：

```
public String getUserCode ( )  {
   return userCode;
)
public String getUserName ( )  {
   return userName;
)
```

```
getFirstName ( )
getAccountNumber ( )
isPersistent ( )
isAtEnd ( )
```

遵循这个命名约定，显然，成员函数将返回对象的字段，布尔型的获取函数将返回布尔值“真”或者“假”。这个标准的另一个优点是它遵循Beans Development Kit（BDK）对获取成员函数采用的命名约定（DES97）。它的一个主要的缺点是get是多余的，需要额外地录入。

按照JBuilder中bean / properties工具生成的设置和获取函数。

3. 设置函数

设置函数也称为变值函数，是可以修改一个字段值的成员函数。无论何种字段类型，都要在字段名的前面加上set前缀。

示例：

```
public void setUserCode （String userCode）  {
   this.userCode=userCode;
}
public void setUserName （String userName）  {
   this .userName=userName;
}
setFirstName （String name）
setAccountNumber （int accountNumber）
setReasonableGoals （Vector  goals）
setPe rsis tent （boolean  is Persis tent）
setAtEnd （boolean isAtEnd）
```

按照这种命名约定，显然是一个成员函数设定一个对象的字段值。这个标准的另一个优点是：它遵循Beans Development Kit（BDK）对设置函数采用的命名约定（DES97）。它的一个主要的缺点是set是多余的，需要额外地录入。

按照JBuilder中bean / properties工具生成的设置和获取函数。

4. 命名构造函数

构造函数是在一个对象初次生成时，完成所有必需的初始化的成员函数。构造函数与它所属类的名字总是相同的。例如，类Customer的构造函数是Customer（ ）。注意大小写一致。

示例：

```
Customer ( )
```

SavingsAccount ( )

PersistenceBroker ( )

这个命名约定由Sun公司设定，必须严格遵守。

5. 成员函数的可见性

良好的程序设计应该尽可能地减小类与类之间耦合，所遵循的经验法则是：尽量限制成员函数的可见性。如果成员函数没必要公共（public），就定义为保护（protected）；没必要保护，就定义为专用（private），如表1.1所示。

表1-1　成员函数的可见性

可见性	Java关键字	说　明	正确用法
公共	public	公有成员函数可被任何其他对象和类的成员函数调用	当该成员函数必须被该函数所在的层次结构之外的其他对象和类访问时
受保护	protected	被保护的成员函数可被它所在的类或该类的子类的任何成员函数调用	当该成员函数提供的行为被它所在类的层次结构内部而非外部需要时
专用	private	私有成员函数只可以被该类所在的其他成员函数调用，该类的子类不可以调用	当该成员函数所提供的行为明确针对定义它的类时。私有成员函数常常是重新分配要素的结果。重新分配要素又称为“重组”，指类内其他成员函数封装某一个特定行为的做法
缺省	无关键字，简单地使其为空白	成员函数对于同一包中的其他所有类实际上都是公共的，但是对该包外部的类是专用的。有时，它称为包可见性或友好的可见性	这是一个有趣的功能，但要小心使用。一般不建议使用

6. 成员函数的参数

成员函数的参数在采用javadoc@param标识的头文件中注释。应说明：

（1）参数用来做什么。需要注释出参数用来做什么，以便其他开发者了解使用参数的上下文。

（2）任何约束或前提条件。如果一个参数的值域不能被成员函数接收，则应让调用者知道。可能一个成员函数只接收正数，或者字符个数小于5的字符串。

（3）示例。如果应传递的参数不明显，那么应该在注释中给出一个或多个例子。

1.3.4　字段、属性的命名及注释

field这个词在这里指的是字段，Beans Development Kit （BDK）称为“属性”（DES97）。字段是说明一个对象或者一个类的一段数据。字段可以是像字符串或者浮点

数这样的基本数据类型，也可以是一个对象，例如一个消费者或者一个银行账户。

1. 命名字段

应采用完整的英文描述符来命名字段（GOS96，AMB98），以便使字段所表达的意思一目了然。像数组或者矢量这样是集合的字段，命名时应使用复数来表示它们代表多值。

示例：

```
firstName      zipCode unitPrice      discountRate      orderItems（复数）
```

2. 命名组件（部件）

应采用完整的英文描述符命名组件（接口部件），名字的扩展名是组件类型名。这让用户容易区分一个组件的目的和它的类型，容易在一个表里找到各个组件（许多可视化的编程环境在一个Applet程序或者一个应用程序中提供了一个所有组件的列表。如果所有名字都是类似于button1、button2或&，则容易混淆）。

示例：

```
okButton     customerList     fileMenu     newFileMenuItem
```

3. 字段都应很好地加以注释，以便其他开发者理解它

要想有效地注释，以下部分需要说明。

（1）字段的说明。需说明一个字段，才能使人了解如何使用它。

（2）注释出所有采用的不变量。字段中的不变量是指永远为“真”的条件。例如，字段dayOfMonth的不变量可能是它的值只能在1～31（显然，可以用基于某一年里的某个月份来限制这个字段值，使其变得更加复杂）。通过说明字段值的限制条件，有助于定义重要的业务规则，使代码更易理解。

示例：

```
// 终执行的SQL语句
protected String  strSQL=null;
// 需要连接数据源的名称
private String  dsName;
```

1.3.5　局部变量命名及注释

一般说来，命名局部变量遵循与命名字段一样的约定，即使用完整的英文描述符，任何非开头的单词的第一个字母都要大写。

但是为了方便起见，对于如下几个特殊的局部变量类型，这个约定可以放宽。

（1）流。

（2）循环计数器。

（3）异常。

1. 命名流

当有一个单输入和/或单输出流在一个成员函数中被打开、使用和关闭时，通常的约定是对这些流分别采用in和out来命名（GOS96）。对于既用于输入又用于输出的流，采用inOut来命名。

一个常用的取代这种约定的方法是分别采用inputStream、outputStream和ioStream这样的名字，而不是in、out和inOut，虽然这与Sun公司的建议相抵触。

2. 命名循环计数器

因为局部变量常用作循环计数器，并且它为C/C++所接受，所以在Java编程中，可以采用i、j或k作为循环计数器（GOS96）。若采用这些名字作为循环计数器，要始终使用它们。

概括起来说，i、j、k作为计数器时，它们可以很快地被输入，也可以被广泛地接受。

3. 命名异常对象

因为在Java代码中异常处理也非常普遍，所以字母e作为一般的异常符被广泛地接受。

如果有多个异常同时出现，应该使用该类型异常全称的缩写表示。例如：

catch（SQLException sqlexception）

4. 声明和注释局部变量

在Java中声明和注释局部变量有以下几种约定。

（1）一行代码只声明一个局部变量。这与一行代码应只有一个语句相一致，并使得对每个变量采用一个行内注释成为可能。

（2）用一个行内注释语句说明局部变量。行内注释是一种紧接在同一行的命令代码后，用符号//标注出来的单行注释风格（也称“行末注释”）。应注释出一个局部变量用于做什么、在哪里适用、为什么要用等，使代码易读。

例如：

```
int count=1;                    //记录循环次数
int pageSize=5;                 //页面的大小
```

1.4 搭建 Java 开发环境

对 Java 有了一个基本了解后，现在就来学习一下如何进行 Java 程序开发。Java 开发环境的搭建相对其他语言可能有些复杂，因为 Java 本身提供了很多机制，从而方便学习。

Java的开发环境可以用 JDK 来代表，在本节中就来看一下如何下载、安装和配置 JDK。

1.4.1　下载 JDK

JDK 是 Sun 公司提供的一种免费的 Java 软件开发工具包，里面包含了很多用于 Java 程序开发的工具，最常用的是编译和运行工具。现在就先来看一下该工具包的下载， JDK 的下载可以通过 Sun 官方网站或者通过搜索引擎下载。

1.4.2　安装

JDK下载 JDK 后，双击下载的 EXE 文件，即可开始安装 JDK。首先是弹出许可证协议窗口，其中给出了 Sun 公司的一些开发协议，单击其中的“接受”按钮，就会弹出如图 1–3 所示的“自定义安装”窗口。

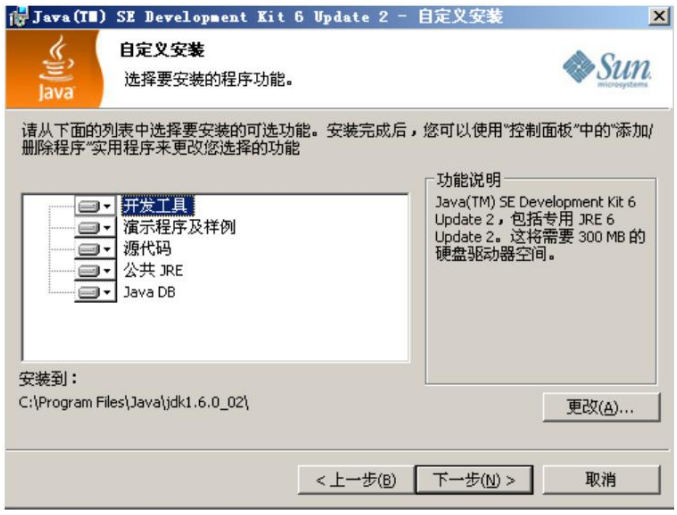

图1–3 “自定义安装”窗口

在窗口中可以选择要安装的 Java 组件和 JDK 文件的安装路径。这里可以采用默认安装Java 的所有组件并在 C 盘安装。在后面的配置中，也将按照默认安装进行配置。单击“下一步”按钮后就开始安装 JDK，稍后，单击窗口中的“完成”按钮就正式完成了 JDK 的安装。

1.4.3　配置 JDK

下载和安装 JDK 后，只是完成 Java 开发环境搭建的前半部分，最关键的部分还是配置JDK。配置 JDK 的目的是能够在命令提示符中运行 JDK 中的命令，例如编译和运行。配

置JDK 的操作步骤如下。

（1）在“我的电脑”桌面图标上，单击鼠标右键，在弹出菜单中选择“属性”→“高级”→“环境变量”命令，弹出“环境变量”窗口，在该窗口中就可以进行环境变量的设置，如图 1–4 所示。

（2）单击“系统变量”选项组中的“新建”按钮，弹出如图 1–5 所示的“新建系统变量”窗口。

图1–4　环境变量设置

图1–5　新建系统变量

（3）在“变量名”文本框中输入“PATH”，在“变量值”文本框中输入“C:\Program Files\Java\jdk1.6.0_02\bin;”，注意要以分号结尾。

（4）重复步骤（3）的操作，在“变量名”文本框中输入“CLASSPATH”，在“变量值”文本框中输入“C:\Program Files\Java\jdk1.6.0_02\lib\tools.jar;”。单击“确定”按钮。

（5）配置 JDK 完成之后，这时候就可以测试一下是否配置正确。选择“开始”→“运

行”命令，弹出“运行”命令框，如图 1–6 所示。

（6）在“运行”命令框中输入“cmd”，进入命令提示符界面。在该界面中输入javac命令，如果出现如图 1–7 所示的结果，则表示 JDK 配置成功；如果提示错误，则表示配置失败，就需要重新配置，查找哪一步发生了错误。

图1–6　“运行”命令框

图1–7　测试

1.5　开发 Hello World 程序

搭建 Java 的开发环境后，现在是否已经迫不及待地想开发一个 Java 程序了？在本节中就来开发一个非常简单的输出“Hello World”内容的程序。通过该程序来演示 Java 程序的编写、编译和运行，从而了解 Java 程序的开发过程。

1.5.1　编写 Java 程序

开发 Java 程序，首先要编写一个 Java 程序。在 F 盘下，新建一个文本文档，将初始