第 1 章　Java 概述

根据原 Sun 公司"Java 白皮书"中对 Java 的定义，Java 是一种简单、面向对象、分布式、解释性、健壮、安全、结构中立、可移植、高性能、多线程、动态的语言。本章将主要介绍 Java 语言的概况。包括 Java 语言的发展概况、Java 语言的主要特点、Java 语言的开发环境和开发工具的使用方法。

1.1　Java 简介

1.1.1　Java 的产生与发展

Java 语言是由原 Sun Microsystems 公司开发的功能强大的纯面向对象的编程语言，使用它可在各种不同的计算机、不同操作系统平台的网络环境中开发软件。Java 是一种为网络量身定做的语言，是目前 Web 应用的主要开发语言。它改变了应用软件的开发模式，为迅速发展的信息世界增添了新的活力。

Java 最初源于 1991 年 Sun 公司内部一项名为 Green 的发展计划，该计划的目的是为家用电子产品开发一个分布式代码系统，这样可以把 E-mail 发给电冰箱、电视机等家用电器，对它们进行控制，和它们进行信息交流。一开始，该项目准备采用 C++语言，但 C++太复杂，安全性差。1991 年，Sun 公司的 Jame Gosling、Bill Joe 等人开发了一种新的语言 Oak（Java 的前身），Oak 是一种用于网络的精巧而安全的语言。Sun 公司曾依此投标一个交互式电视项目，但结果是被 SGI 打败。恰巧这时 Mark Andersen 开发的 Mosaic 和 Netscape 启发了 Oak 项目组成员，他们编制了浏览器，得到了当时 Sun 公司首席执行官的支持，从此开始进军 Internet。

1993 年之后，万维网（WWW）已如火如荼地发展起来。Sun 公司重新分析市场需求，Gosling 意识到 WWW 需要一个中性的浏览器，它不依赖于任何硬件和软件平台，是一种事实性较高、可靠安全、有交互功能的浏览器，于是 Gosling 决定用 Java 开发一个新的 Web 浏览器。实践证明 Sun 公司的这次市场决策是非常成功的。

1995 年 Sun 公司正式向 IT 业界推出了 Java 语言，这种语言具有安全、跨平台、面向对象、简单等显著特点，而这个时期以 Web 为主要形式的互联网应用正在迅猛发展，几乎所有程序员和软件公司对 Java 语言的出现都表现出了极大的关注，开发人员纷纷尝试用 Java 语言编写网络应用程序，他们的努力使 Java 语言朝着网络应用的方向飞速发展，Java 的地位也随之得到肯定。又经过一年的使用和改进，Java 1.0 版终于在 1996 年年初正式发布。

1999 年，Sun 公司重新组织了 Java 平台的集成方法，将 Java 2 平台分为三大块：J2SE、

J2ME、J2EE。该行为顺应了网络急速发展的潮流，对 Java 2 平台的发展起到了很好的催化作用，使得 Java 语言可以支持智能消费性电子产品的开发、各种应用程序的开发，尤其是 Web 应用程序的开发。

2000 年 5 月，J2SE 1.3 发布；2002 年 2 月，J2SE 1.4 发布，Java 的计算能力有了大幅提升。2004 年 9 月，J2SE 1.5 发布，成为 Java 语言发展史上的又一里程碑。为了表示该版本的重要性，J2SE 1.5 更名为 Java SE 5.0；2006 年 12 月，Sun 公司发布了 Java 6.0；2009 年 04 月 20 日，Oracle 公司用 74 亿美元收购了 Sun Microsystems，取得了 Java 的版权。2011 年 7 月 28 日，Oracle 公司发布 Java 7.0 的正式版；2014 年 3 月 19 日，Oracle 公司发布 Java 8.0 的正式版。随着版本的更新，Java 语言的功能也愈加完善与强大。

1.1.2 Java 语言的特点

Java 是一种基于面向对象的高级程序设计语言，它同时具备了面向对象程序设计和高级语言的所有优点与特性，如简单性、面向对象、分布式、安全性、可移植性、高性能、多线程和动态性等。下面介绍 Java 语言的几个重要特性。

1. 简单性

（1）Java 语法类似 C++，这使得 C++程序员学习 Java 语言更加容易。

（2）Java 语言抛弃了 C++语言中容易引发程序错误的内容，如指针、内存管理，使得使用 Java 语言比 C++语言更容易写出“无错代码”。

（3）Java 语言提供了丰富的类库和帮助文档，使编程更加容易。

（4）运行环境小巧。基本的 Java 的解释器和类的支持只有 40 KB，附加标准类库和线程的支持也只有 215 KB。

2. 面向对象

Java 语言是一种比 C++语言“还面向对象”的编程语言，Java 语言的设计集中于对象及其接口，它提供了简单的类似机制及动态的接口模型，是纯面向对象的编程语言，尤其适用于分布式环境。面向对象的语言都支持三个概念：封装、继承和多态，Java 语言也是如此。

（1）封装（encapsulation）：所谓封装，就是指隐藏其内部实现细节，只对外提供公共的访问接口。也就是说，将不需要对外提供的内容都隐藏起来，包括属性，仅提供公共方法对其访问。Java 语言的封装性很强，它没有全局变量。在 Java 语言中，除了简单的数值类型、字符类型和布尔类型外，大部分的成员都是类类型。而对于基本类型，Java 语言也提供了相应的对象类型，以便与其他对象交互操作。

（2）继承（inheritance）：继承是指一个对象直接使用另一个对象的属性和方法。Java 语言给用户提供了一系列的类，并且 Java 语言的类富有层次结构，子类可以继承父类的属性和方法。但是，与其他一些面向对象的编程语言所不同的是，Java 语言只支持单一继承，这样可以有效地避免冲突。在 Java 语言中，可以通过接口来实现多重继承。

（3）多态（polymorphism）：多态是指一个程序中同名的多个不同方法共存的情况，即一

个对外接口，多个内在实现方法。在面向对象程序中，多态的情况有多种，可以通过子类对父类方法的覆盖实现多态，也可以利用重载在同一个类中定义多个同名的不同方法来实现多态。多态的特点大大提高了程序的抽象程度和简洁性，同时，还最大限度地降低了类和程序模块之间的耦合性，使得它们不需了解对方的具体细节就可以很好地共同工作。这个优点，对程序的设计、开发和维护都有很大的好处。

3. 分布性

分布性指数据分布和操作分布。数据分布是指数据可以分散在网络的不同主机上；操作分布是指把一个完整的计算过程分散在不同的主机上处理。Java 包括一个支持 HTTP 和 FTP 等基于 TCP/IP 协议的子库，因此，Java 应用程序可凭借 URL 打开并访问网络上的对象，就像访问本地文件一样简单方便。Java 的分布性为在分布环境尤其是 Internet 下实现动态内容提供了技术途径。Java 语言具有强大的、易于使用的联网能力，非常适合开发分布式计算的程序。

4. 安全性

作为网络应用开发语言，安全是非常重要的。Java 的安全性可从两个方面得到保证。一方面，在 Java 语言中，指针和释放内存等 C++中具备的功能被删除，避免了非法内存操作。另一方面，当 Java 用来创建浏览器时，语言功能和浏览器本身提供的功能结合起来，使它更安全。Java 语言在机器上执行前，要经过很多次测试，如代码校验、检查代码段的格式、检测指针操作、对象操作是否过分及试图改变一个对象的类型。另外，Java 拥有多个层次的互锁保护措施，能有效地防止病毒的入侵和破坏行为的发生。

5. 可移植性

语言程序面临的一个主要问题是：操作系统的变化，处理器升级以及核心系统资源的变化，都可能导致程序出现错误甚至无法运行。而用 Java 写的程序不用修改就可在不同的软硬件平台上运行，可移植性是 Java 语言最大的优势。这样就能实现同样的程序既可以在 Windows 环境下运行，又可以在 Unix 或者 Linux 环境下不用修改就可以直接运行。Java 主要靠 Java 虚拟机（JVM）实现可移植性，即一次编写，随处运行（Write Once，Run Anywhere）。

6. 高性能

高级语言程序必须转换为机器语言程序才能执行，但不同的计算机系统所使用的机器语言不同。Java 为了实现"一次编译，随处运行"的目标，在编译时并不直接编译成特定的机器语言程序，而是编译成与系统无关的字节码，由 Java 虚拟机（JVM）来执行。当 JVM 解释执行 Java 程序时，Java 实时编译器（Just-In-Time，JIT）会将字节码译成目标平台对应的机器语言的指令代码。

早先的许多尝试解决跨平台的方案对语言的性能要求都很高。其他解释执行的语言系统，如 BASIC、PERL 都有无法克服的性能缺陷。然而，Java 却可以在非常低档的 CPU 上顺畅运行，这是因为 Java 的字节码是经过精心设计的，能够直接使用 JIT 编译技术将字节码转换成高性能的本机代码，从而得到较高的性能。

7. 多线程

Java 支持多线程编程，用 Java 编写的应用程序可以同时处理多项任务，在一定程度上提升了交互能力和实用性。Java 提供多线程功能使得在一个程序里可同时执行多个小任务。Java 在多线程处理方面性能超群，具有让设计者惊喜的强大功能，而且在 Java 语言中进行多线程处理很简单。多线程带来的最大的好处是具有更好的网上交互性能和实时控制性能，尤其是实现多媒体功能。

8. 动态性

Java 是一种动态的语言，Java 的动态特性是其面向对象设计方法的扩展。它允许程序动态地装入运行过程中所需要的类，这是 C++语言进行面向对象程序设计时所无法实现的。C++程序设计过程中，每当在类中增加一个实例变量或一种成员函数后，引用该类的所有子类都必须重新编译，否则将导致程序崩溃。同时 Java 通过接口来支持多重继承，使之比严格的类继承具有更灵活的方式和扩展性。

9. 中性结构

Java 编译器生成的是一种中性的对象文件格式，也就是说，Java 编译器通过伪编译后，将生成一个与任何计算机系统无关的"中性"的字节码。Java 的这种字节码经过了许多精心的设计，使得其能够很好地兼容于当今大多数流行的计算机系统，在任何机器上都易于解释，易于动态翻译成为机器代码。

1.1.3 Java 技术简介

目前，Java 开发技术主要包括以下三个方面。

（1）Java SE（Java 2 Platform Standard Edition）：以前的版本称为 J2SE，是 Java 平台的标准版，是用于工作站、PC 环境的 Java 标准平台。它体现了 Sun 公司的开放精神，被称为是"Internet 上的世界语"。

（2）Java ME（Java 2 Platform Micro Edition）：以前的版本称为 J2ME，是 Java 平台的精简版，是致力于消费产品和嵌入式设备的最佳解决方案。Java ME 是移动商务最佳的应用典范，无线通信、手机和 PDA 等小型电子装置均可采用 Java ME 作为开发工具及应用平台。它提供了 HTTP 等高级 Internet 协议，可以使移动电话以客户端/服务器方式直接访问 Internet 的全部信息，不同的客户端访问不同的文件。此外，Java ME 还能访问本地存储区，提供最高效率的无线交流。

（3）Java EE（Java 2 Platform Enterprise Edition）：以前的版本称为 J2EE，是 Java 平台的企业版，是一种利用 Java 平台来简化企业级解决方案的涉及开发、部署和管理相关复杂问题的体系结构。它提供了企业电子商务架构及 Web Services 服务，其优越的跨平台能力与开放的标准深受广大企业用户的喜爱。目前，Java EE 已经成为开发商创建电子商务应用的事实标准。

1.2 Java 工作原理

1.2.1 Java 虚拟机

Java 的安全性和可移植性两大特性依赖于 Java 的字节码（bytecode）。字节码是一套设计用来在 Java 运行的系统下执行的高度优化的指令集，该 Java 运行的系统称为 Java 虚拟机（Java Virtual Machine，JVM）。在其标准形式下，JVM 就是一个字节码解释器。

Java 虚拟机 JVM（Java Virtual Machine）在 Java 编程里面具有非常重要的地位。虚拟机的基本功能如下：

（1）通过 ClassLoader 寻找和装载 class 文件。

（2）解释字节码成为指令并执行，提供 class 文件的运行环境。

（3）进行运行期间垃圾回收。

（4）提供与硬件交互的平台。

Java 虚拟机是在真实机器中用软件模拟硬件的技术，代码被存储在.class 文件中。每个文件都包含最多一个 public 类。Java 虚拟机规范为不同的硬件平台提供了一种编译 Java 程序代码的规范，该规范使 Java 软件独立于平台，因为编译是针对作为虚拟机的“一般机器”，这个“一般机器”可用软件模拟并运行于各种现存的计算机系统，也可用硬件来实现。编译器在获取 Java 应用程序的源代码后，将其生成字节码，它是为 JVM 生成的一种机器码指令。每个 Java 解释器，不管它是 Java 技术开发工具，还是可运行 applets 的 Web 浏览器，都可执行 JVM。

JVM 的代码格式由紧缩有效的字节码构成。由 JVM 字节码编写的程序必须保持适当的类型约束。大部分类型检查是在编译时完成。任何从属的 Java 技术解释器必须能够运行任何含有类文件的程序，这些类文件应符合 Java 虚拟机规范中所指定的类文件格式。正是因为有虚拟机这个中间层，Java 才能够实现可移植性。虚拟机就好比是一个 Java 运行的基本平台，所有的 Java 程序都运行在虚拟机上，只要根据 JVM 规范描述将解释器移植到特定的计算机上，就能保证经过编译的任何 Java 代码能够在该系统上运行，也符合 SUN 公司提出的口号：“Write Once，Run Anywhere”（一次编写，处处运行）。Java 虚拟机位置如图 1-1 所示。

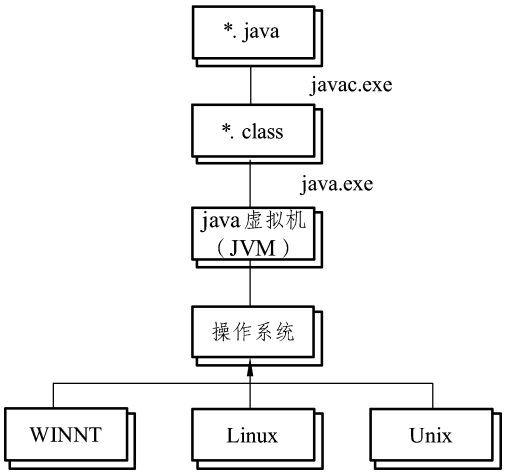

图 1-1 Java 虚拟机

1.2.2 内存自动回收机制

许多编程语言都允许在程序运行时动态分配内存，动态分配内存的过程根据语言句法的不同而有所变化，但总是要将指针返回到内存的起始位置，当分配的内存不再需要时，程序或运行环境应释放内存，把这些内存资源回收回来，加以再利用，从而节省资源，提高系统性能。

在 C、C++中，程序员负责释放内存。有时，这是一件很困难的事情。因为程序员并不总是事先知道内存应在何时被释放。当在系统中没有能够被分配的内存时，可导致程序瘫痪，这种程序被称作具有内存漏洞。

Java 编程语言解除了程序员释放内存的责任。它提供了一种系统级线程以跟踪每一次内存的分配情况。在 Java 虚拟机的空闲周期，垃圾收集线程检查并释放那些可被释放的内存。内存回收在 Java 技术程序的生命周期中自动进行，能够有效避免内存漏洞和内存泄露（内存泄露就是程序运行期间，所占用的内存一直往上涨，很容易造成系统资源耗尽而降低性能或崩溃）。

注意：

（1）Java 内存回收是一个自动的系统行为，程序员不直接控制内存回收的功能和行为。比如内存回收什么时候开始，什么时候结束，还有到底哪些资源需要回收等。

（2）存在跟内存回收相关的方法，比如：System.gc()等。但须记住，调用这些方法，仅仅是通知内存回收程序，至于内存回收程序运不运行，什么时候运行，都是程序员无法直接控制的。

（3）程序员可以通过设置对象为 null 来标识某个对象不再被需要了，这只是表示这个对象可以被回收了，并不是马上被回收。

1.3 Java 开发环境与开发工具

1.3.1 JDK 与 JRE

Java 的开发环境非常简单，只需一个 Java 开发工具包（Java SE Development Kit，JDK）和一个记事本程序就可以编写简单的 Java 程序。JDK 是 Sun 公司推出的 Java 开发包，包括了程序解释器、JVM、编译工具（javac.exe）、Java 执行程序（java.exe）等，是一个编写 Java 的 Applet 小程序和应用程序的程序开发环境。

运行 Java 程序需要先利用编译工具对其进行编译，然后再利用执行程序执行编译后的 class 文件。而这些操作的前提就是让 Java 程序知道这些工具的位置，这就需要开发者下载、安装 JDK，以及配置相应的环境。JDK 工具由官方免费提供，它包含在各种平台上运行的版本中，用户可以根据需要来下载。本书使用 JDK7 来做实例演练。

1. 安装 JDK

安装 JDK 需要以下 5 个步骤。

1）下载 JDK

读者可利用搜索引擎查找 JDK7.0 的安装文件，也可以去官方网站下载。下载链接如下。书中采用的是 jdk7.0 win64 版本，读者可根据自己的开发环境选择合适的版本。

http//www.oracle.com/technetwork/java/javase/downloads/java-se-jdk-7-download-432154.html

2）进入安装界面

双击已下载的应用程序进行安装，进入如图 1-2 所示界面。

图 1-2　JDK 安装向导界面

3）选择需要安装的功能

在图 1-2 中，单击“下一步”按钮，进入功能选择界面。该界面可以选择要安装的功能和更改默认安装目录。如图 1-3 所示。

图 1-3　功能和路径选择界面

4）选择“JRE”安装路径

单击图 1-3 中的“下一步”按钮，进行安装。进入“JRE”安装路径选择界面，如图 1-4 所示。

图 1-4　JRE 安装路径选择界面

5）安装完成

单击图 1-4 中的“下一步”按钮，安装“JRE”。当进度条走完，如无意外，则出现图 1-5 所示界面，表示已安装完成。

图 1-5　JDK 安装完成

单击图 1-5 中的“完成”按钮，至此，JDK 和 JRE 都已安装完成。虽然软件已安装完成，但依然不能利用记事本开发程序，因为还没有配置环境变量。

2. 环境变量的配置

JDK 安装完成后，通常不能马上用于开发程序，因为环境变量还没有配置。所谓环境变量，就是在操作系统中一个具有特定名字的对象，它包含了一个或多个应用程序使用的信息。如果安装完 JDK 之后，不配置 Java 的环境变量，那么在 DOS 命令行环境下就找不到 Java 的编译程序和 Java 的运行程序，也就不能在 DOS 环境下进行 Java 程序的编译与运行。Windows 7 设置环境变量的 3 个步骤：

1）打开系统属性界面

右击“计算机”，在弹出的菜单中单击“属性”选项，进入系统窗口。在“高级系统设置”选择卡里点击“环境变量”，对环境变量进行设置，Classpath 的变量值：.；%JAVA_HOME%\lib；%JAVA_HOME%\lib\tools.jar，如图 1-6 所示。

图 1-6　环境变量设置

2）打开系统变量编辑界面

Path 变量在系统里已存在，只增加变量值即可（加在原有变量值的最前面），变量值：%JAVA_HOME%\bin；%JAVA_HOME%\jre\bin。如图 1-7 所示为“编辑系统变量”界面。

3）设置 Path 变量值

在图 1-7 所示界面中的“变量值（V）”处添加 JDK 的安装路径信息。该路径是 JDK 安装路径下的 bin 目录。添加完成后，单击“确定”按钮，完成环境变量的设置。

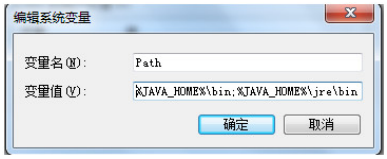

图 1-7　编辑系统变量窗口

经过这个处理后，需要验证环境变量是否配置成功。在“开始 ” 菜单中的“搜索程序和文件”中输入“cmd”命令并回车，弹出命令行窗口。在这个界面中键入“java -version”命令可以查看到安装的 JDK 版本信息；键入“java”命令，可以看到此命令的帮助信息；键入“javac”命令可以看到此命令的帮助信息。如果都能如愿看到，则说明环境变量配置成功，如图 1-8 所示。

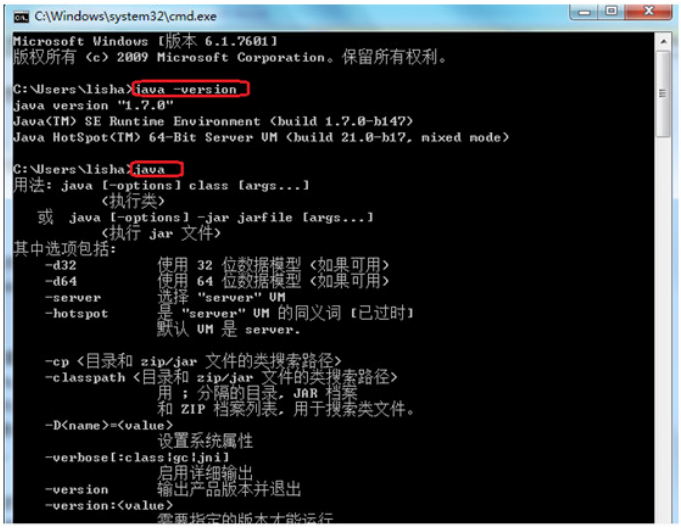

图 1-8　运行 javac 命令

注意：如果更改了系统环境变量的配置，必须重新打开 DOS 窗口，就是打开一个新的 DOS 窗口，环境变量的新配置才会有效，否则，DOS 窗口就还应用旧的环境变量的旧配置。

3. JDK 与 JRE 的关系

JDK 是整个 Java 的核心，包括了 Java 运行环境（JRE）、一些 Java 工具和 Java 的核心类库（Java API）。Java 的应用服务器实质上都内置了某个版本的 JDK。主流的 JDK 是 Sun 公司发布的 JDK，除了 Sun 之外，还有很多公司和组织都开发了自己的 JDK。JDK 提供 Java 程序的编译和运行命令，但没提供程序编辑环境。

JRE 是 Java Runtime Environment 的简称，不是开发环境，而是运行 Java 程序所必需的环境集合，它提供三个主要功能：

（1）加载代码：由 class loader 完成；

（2）校验代码：由 bytecode verifier 完成；

（3）执行代码：由 runtime interpreter 完成。

JRE 是个运行环境，JDK 是个开发环境。因此写 Java 程序的时候需要 JDK，而运行 Java 程序的时候就需要 JRE。因为 JDK 里面已经包含了 JRE，因此只要安装了 JDK，就可以编译 Java 程序，也可以正常运行 Java 程序。

1.3.2 集成开发工具 Eclipse

虽然记事本可以编写 Java 程序，但开发一个复杂的项目只用记事本是不现实的。为了提升开发效率，Java 的一些开发工具提供程序的开发环境，如：Eclipse、MyEclipse 等，它们都是建立在 JDK 的运行环境之上的。

Eclipse 是一个开放源代码的、基于 Java 的可扩展开发平台。就其本身而言，它只是一个框架和一组服务，用于通过插件组件构建开发环境。Eclipse 工具为软件开发提供了更好的灵活性，越来越受到 Java 程序开发者的青睐，是目前使用最多的一种 Java 开发工具。在本书中将采用 Eclipse 作为开发工具，下面简单介绍一下它的使用方法。

1. 第一步：新建 Java 项目

选择“File→New→Project...”，选择“Java Project”，单击“Next”按钮，便打开了“New Java Project”向导。在“Project name”中输入“Chapter1_1”，无须进行其他设置，直接单击“Finish”按钮，如图 1-9 所示。