第 1 章

Java 语言概述

本章主要内容

- Java 语言的特点；
- Java 虚拟机；
- Java 源文件（.java）与 Java 字节码文件（.class）；
- Java 应用程序的结构与主类。

Java 语言是一种简单易用、完全面向对象、与平台无关、安全可靠、主要面向 Internet 的开发工具，是一种完全面向对象的程序设计（Object Oriented Programming，OOP）语言。

1.1 Java 语言的诞生与发展

Java 语言诞生于 20 世纪 90 年代初期，从它正式问世以来，它的快速发展已经让整个 Web 世界发生了翻天覆地的变化。Java 语言的前身是 Sun Microsystems 公司（简称 Sun 公司，该公司于 2009 年 4 月被 Oracle 公司收购）开发的一种用于智能化家电的名为 Oak（橡树）的语言，它的基础是当时最为流行的 C 和 C++ 语言。但是，由于一些非技术上的原因，Oak 语言并没有得到迅速推广。直到 1993 年，WWW（万维网）迅速发展，Sun 公司发现可以利用 Oak 语言的技术来创造含有动态内容的 WWW 网页，于是已受人冷落了的 Oak 语言又被重新开发和改造，并将改造后的 Oak 语言改名为 Java 语言（Java 是太平洋上一个盛产咖啡的岛屿的名字）。终于，在 1995 年 Java 这个被定位于网络应用的程序设计语言被正式推出。

由于 Java 语言功能强大，其问世后不久，即被业界广泛接受，于是 IBM、Apple、DEC、Adobe、HP、Oracle、Toshiba、Netscape 和 Microsoft 等大公司均购买了 Java 语言的许可证。同时，众多软件开发商也开发了许多支持 Java 语言的产品。

随着 Java Servlet 的推出，Java 语言极大地推动了电子商务的发展。Java Server Page（JSP）技术的推出，更是让 Java 语言成为基于 Web 应用程序的首选开发工具。Internet 的普及和迅猛发展，以及 Web 技术的不断渗透，使得 Java 语言在现代社会的经济发展和科学研究中占据越来越重要的地位。

1.2 Java 语言的特点

Java 语言是一种跨平台、适合于分布式计算环境的面向对象编程语言。它具有很多特点，如简单易学、面向对象、平台无关性、分布式、可靠性、安全性、支持多线程、支持网络编程、编译与解释并存、可移植性、高性能、动态性等。下面介绍 Java 语言的几个重要特点。

1. 简单易学

Java 语言虽然衍生自 C++，与 C++ 相比 Java 是一个完全面向对象的编程语言。出于安全性和稳定性的考虑，Java 去掉了 C/C++ 支持的三个不易理解和掌握的数据类型：指针（pointer）、联合体（unions）和结构体（structs）。而 C/C++ 中联合体和结构体的功能，完全可以在 Java 中用类及类的属性等面向对象的方法来实现，这不但更加合理规范，而且还降低了学习难度。

2. 面向对象

Java 语言最吸引人之处，就在于它是一种以对象为中心、以消息为驱动的面向对象的编程语言。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。

3. 平台无关性

Java 是与平台无关的语言。所谓平台其实就是指由操作系统和处理器所构成的运行环境，与平台无关就是指应用程序的运行不会因为操作系统或处理器的不同而无法运行或出现错误。可以说平台无关性是指一个应用程序能够运行于各种不同的操作系统上，即使用 Java 语言编写的应用程序不用修改就可在不同的软硬件平台上运行。平台无关有两种：源代码级和目标代码级。Java 语言是靠 Java 虚拟机（JVM）在目标代码级实现平台无关性的。

4. 分布式

分布式包括数据分布和操作分布。Java 语言支持这两种分布性。Java 语言提供了一整套网络类库，开发人员可以利用类库进行网络程序设计，方便地实现 Java 语言的分布式特性。

5. 可靠性

Java 语言具有很高的可靠性。Java 解释器运行时实施检查，可以发现数组和字符串访问的越界等错误；另外，Java 语言提供了异常处理机制，可以把一组错误的代码放在一个地方，这样可以简化错误处理任务，便于恢复。

6. 安全性

Java 语言具有较高的安全性。当 Java 字节码进入解释器时，首先必须经过字节码校验器的检查；其次，Java 解释器将决定程序中类的内存布局；再次，类装载器负责把来自网络的类装载到单独的内存区域，避免应用程序之间相互干扰破坏；最后，客户端用户还可以限制从网络上装载的类只能访问某些文件系统。综合上述几种机制，使得 Java 成为安全的编程语言。

7. 支持多线程

Java 语言在两方面支持多线程：一方面，Java 环境本身就是多线程的，若干系统线程运行，负责必要的无用单元回收、系统维护等系统级操作；另一方面，Java 语言内置多线程机制，可以大大简化多线程应用程序开发。

8. 支持网络编程

Java 语言通过其提供的类库可以处理 TCP/IP，用户可以通过 URL 地址在网络上很方便地访问其他对象。

9. 编译与解释并存

Java 语言的编译器并不是把源文件（扩展名为 .java）编译成二进制码，而是将其编译成一种独立于机器平台的字节码（扩展名为 .class 的文件）。字节码可以被 Java 解释器所执行，由解释器将字节码再翻译成二进制码，使程序得以运行。

1.3 Java 语言规范

Java 语言有严格的使用规范，Java 语言规范是对 Java 程序设计语言的语法和语义的技术定义，如果编写程序时没有遵守这些规范，计算机就不能理解程序。Java 语言还为开发 Java 程序而预定义了类和接口，称为应用程序接口（Application Program Interface，API）。

目前 Java 技术主要包括如下三方面：

Java SE（Java Platform Standard Edition）：Java 平台的标准版，可以用于开发客户端应用程序，应用程序可以独立运行。

Java ME（Java Platform Micro Edition）：Java 平台的精简版，用于开发移动设备的应用程序。不论是无线通信还是手机，均可采用 Java ME 作为开发工具及应用平台。

Java EE（Java Platform Enterprise Edition）：Java 平台的企业版，用于开发服务器端的应用。例如，Java Servlet、JavaServer Pages（JSP）及 JavaServer Faces（JSF）等。

由于 Java SE 是基础，因此其他 Java 技术都基于 Java SE，Java SE 17 对应的 Java 开发工具包称为 JDK 17，本书采用 JDK 17 介绍 Java 程序设计。

1.4 Java 虚拟机

大部分计算机语言程序都必须先经过编译（compile）或解释（interpret）操作后，才

能在计算机上运行。然而，Java 程序（.java 文件）却比较特殊，它必须先经过编译的过程，然后再利用解释的方式来运行。通过编译器（compiler），Java 程序会被转换为与平台无关（platform-independent）的机器码，Java 称之为“字节码”（byte-codes）。通过 Java 的解释器（interpreter）便可解释并运行 Java 的字节码。图 1.1 说明了 Java 程序的执行过程。

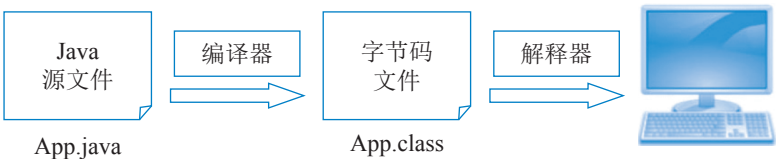

图 1.1　Java 程序的执行过程：先编译，后解释

字节码是 Java 虚拟机（Java Virtual Machine，JVM）的指令组，和 CPU 上的微指令码很相像。Java 程序编译成字节码后文件尺寸较小，便于网络传输。字节码最大的好处是可跨平台运行，即 Java 的字节码可以编写一次，到处运行。用户使用任何一种 Java 编译器将 Java 源程序（.java）编译成字节码文件（.class）后，无论使用那种操作系统，都可以在含有 JVM 的平台上运行。这种跨越平台的特性，也是让 Java 语言迅速普及的原因之一。

Java 虚拟机不是物理机器，而是一个解释字节码的程序，所以任何一种可以运行 Java 字节码的软件均可看作 Java 虚拟机（JVM），如浏览器与 Java 开发工具等皆可视为一部 JVM。很自然地，可以把 Java 的字节码看成是 JVM 上运行的机器码（machine code），即 JVM 负责将字节码解释成本地的机器码，所以说 JMV 就是解释器。所以从底层上看，JVM 就是以 Java 字节码为指令组的“软 CPU”。可以说 JVM 是可运行 Java 字节码的假想计算机。它的作用类似于 Windows 操作系统，只不过在 Windows 上运行的是 .exe 文件，而在 JVM 上运行的是 Java 字节码文件，也就是扩展名为 .class 的文件。JVM 其实就是一个字节码解释器。

1.5　Java 程序的结构

使用 Java 语言可以编写两种类型的程序：Application（应用程序）和 Applet（小程序）。但从 JDK 9 开始不推荐使用 Java 的 Applet 功能。从 JDK 11 开始 Java 对 Applet 的支持被删除，完全淘汰了 Applet。但 Java 中添加了一个 Applet 的替代器，称为 Java Web Start，它支持从 Web 页面动态下载应用程序功能。它是一种部署机制，对于不适合 Applet 的大型 Java 应用程序尤其有用。另外，从 JDK 11 开始 JavaFX 不再包含在 JDK 中，相反，这个 GUI 框架已经成为一个独立的开源项目，因为这些特性不再是 JDK 的一部分。

应用程序是从命令行运行的程序，它可以在 Java 平台上独立运行，通常称为 Java 应用程序。Java 应用程序是独立完整的程序，在命令行调用独立的解释器软件即可运行。另外，Java 应用程序的主类包含有一个定义为 public static void main(String[] args) 的主方法，

这个方法是 Java 应用程序的标志，同时也是 Java 应用程序执行的入口点（或称起始点），在应用程序中包含有 main() 方法的类一定是主类，但主类并不一定要求是 public 类。

一个复杂的应用程序可以由一个或多个 Java 源文件构成，每个文件中可以有多个类定义。下面的程序是一个 Java 应用程序文件。

说明：为了便于对程序代码的解释，本书在每行代码之前加一行号，它们并不是程序代码的一部分。

```
1  package ch01;                                  // 定义该程序属于 ch01 包
2  import java.io.*;                              // 导入 java.io 类库中的所有类
3  public class App1_1{                           // 定义类：App1_1
4    public static void main(String[] args) {    // 定义主方法
5      char c= ' ';
6      System.out.print(" 请输入一个字符：");
7      try{
8        c=(char)System.in.read();
9      }
10     catch(IOException s){}
11     System.out.println(" 您输入的字符是："+c);
12   }
13 }
```

从这个程序可以看出，一般的 Java 源程序文件由以下三部分组成：

- package 语句(0 个或 1 个)；
- import 语句(0 个或多个)；
- 类定义(1 个或多个)。

其中，package 语句表示该程序所属的包。它只能有一个或者没有。如果有，则必须放在最前面。如果没有，则表示本程序属于默认包。

import 语句表示引入其他类库中的类，以便使用。import 语句可以有 0 或多个，它必须放在类定义的前面。

类定义是 Java 源程序的主要部分，每个文件中可以定义若干类。

Java 程序中定义类使用关键字 class，每个类的定义由类头定义和类体定义两部分组成。类体部分用来定义属性和方法这两种类的成员，其中属性则类似于变量，方法类似于其他高级语言中的函数。类头部分除了声明类名之外，还可以说明类的继承特性，当一个类被定义为另一个已经存在的类（称为父类）的子类时，它就可以从其父类中继承一些已定义好的成员而不必自己重复编码。

类体中属性也称为域，包括常量、变量、对象、数组等独立的实体；类中的方法类似于函数的代码单元块。这两个组成成分通称为类的成员。在上面的例子中，App1_1 类中只有一个成员，即第 4 ～ 12 行定义的方法 main()。用来标志方法头的是方法名后面的一

对小括号，小括号里面是该方法使用的形式参数，方法名前面的 public 是用来说明这个方法属性的修饰符，其具体语法规定将在第 6 章中介绍。方法体部分由若干以分号“;”结尾的语句组成，并由一对花括号“{}”括起来，在方法体内部不能再定义其他方法。

同其他高级语言一样，语句是构成 Java 程序的基本单位之一。每条 Java 语句都以分号“;”结束，其构成必须符合 Java 语言的语法规则。类和方法中的所有语句应该用一对花括号括起来。即除 package 及 import 语句之外，其他执行具体操作的语句都只能存在于类的花括号之中。

比语句更小的语言单位是常量、变量、关键字和表达式等，Java 的语句就是由它们构成的。其中，声明常量与变量的关键字是 Java 语言语法规定的保留字，用户程序定义的常量和变量的取名不能与保留字相同。

Java 源程序的书写格式比较自由，如语句之间可以换行，也可以不换行，但养成一种良好的书写习惯比较重要。

注意： Java 是严格区分字母大小写的语言。书写时，大小写不能混淆。

一个应用程序中可以有多个类，但只能有一个类是主类。在 Java 应用程序中，这个主类是指包含 main() 方法的类，主类是 Java 程序执行的入口点。

本章小结

1. Java 程序比较特殊，它必须先经过编译的过程，然后再利用解释的方式来执行。即首先要将源程序（.java 文件）通过编译器将其转换为与平台无关的字节码（.class 文件），然后再通过解释器来解释执行字节码。

2. 字节码（byte-codes）最大的好处是可跨平台执行，可让程序“编写一次，到处运行”（write once,run anywhere）的梦想成真。

3. Java Application 称为 Java 应用程序。Java 应用程序是指可以在 Java 平台上独立运行的一种程序。

4. 每个应用程序都必须有一个主类，主类是程序执行的起始点，应用程序的主类是包含有 main() 方法的类，但应用程序的主类并不一定要求是 public 类。

习题 1

1.1 Java 语言有哪些特点？

1.2 什么是字节码？采用字节码的最大好处是什么？

1.3 什么是 Java 虚拟机？

1.4 什么是平台无关性？ Java 语言是怎样实现平台无关性的？

1.5 Java 应用程序的结构包含哪几方面？

1.6 什么是 Java 应用程序的主类？应用程序的主类有何要求？