第1章

Java 语言开发环境

本章主要内容

- Java 虚拟机；
- Java 源文件（.java）与 Java 字节码文件（.class）；
- Java 开发工具的下载与安装；
- Java 源文件的命名规则。

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。它具有简单性、面向对象、分布式、可靠性、安全性、平台无关性、可移植性、高性能、多线程、编译与解释并存等特点。

1.1 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。目前 Java 技术平台主要包括 Java SE、Java ME 和 Java EE。

Java SE（Java Platform Standard Edition）：Java 平台的标准版，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。

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。

1.2 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），字节码是扩展名为 .class 的文件。通过 Java 的解释器（interpreter）便可解释并运行 Java 的字节码，Java 程序的执行过程如图 1.1 所示。

图 1.1 Java 程序的执行过程：先编译，后解释

说明：平台指由操作系统和处理器所构成的运行环境，与平台无关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错，即一个应用能够运行于各种不同的操作系统上。

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，无论使用哪种操作系统，都可以在含有 JVM 的平台上运行。

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 上所运行的机器码（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。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，即扩展名为 .class 的文件。JVM 其实就是一个字节码解释器。

1.3 Java 程序的结构

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，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)。

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。在类体中通常有两种组成成分：一种是域，包括常量、变量、对象、数组等独立的实体；另一种是方法，方法类似于其他高级语言中的函数。这两种组成成分统称为类的成员。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()。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 5 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。

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。

1.4 Java 开发工具

Java 开发工具（Java SE Development Kit，JDK）是 Java 程序开发的重要工具。JDK 是由 Java API、Java 运行环境和一组建立、测试工具的 Java 实用程序等组成。其核心是 Java API，API（Application Programming Interface）是 Java 提供的标准类库，供编程人员使用，开发人员需要用这些类来实现 Java 语言的功能。

1.4.1 JDK 的下载与安装

Oracle 公司提供了 Windows、macOS 和 Linux 等多种操作系统下的 JDK，用户可以根据自己的使用环境，从 Oracle 公司的网站上下载相应的 JDK 版本。本书使用的是 JDK 17 版本，操作系统使用的是 Windows 11 版本。

1. 下载 JDK

进入 Oracle 公司 Java SE 17 的下载网页后，根据自己所用的操作系统（Windows、macOS、Linux）的不同进行选择。Oracle 公司提供了 jdk-17_windows-x64_bin.exe、jdk-17_windows-x64_bin.msi 两种安装文件和 jdk-17_windows-x64_bin.zip 压缩安装包共三种安装方式。用户可以根据不同的需要选择不同的链接下载。本书的例子是在 Windows 系统的 64 位机器上开发的，下载的安装文件是 jdk-17_windows-x64_bin.exe。

2. 安装 JDK

下载得到 JDK 文件之后，双击 JDK 安装文件 jdk-17_windows-x64_bin.exe 即可进行安装，用户只需按 JDK 的安装步骤和提示进行安装即可，安装过程中用户可以选择欲安装的项目，但建议使用默认值。安装完毕后，将 JDK 安装到 C:\Program Files\Java\jdk-17.0.1 文件夹下，此文件夹称为 JDK 安装文件夹或安装路径。在该文件夹下有如下几个子文件夹：

bin：该文件夹存放的是 JDK 命令程序等。

conf：该文件夹存放的是一些可供开发者编辑的 Java 系统配置文件。

include：该文件夹存放支持本地代码编程与 C 语言程序相关的头文件。

jmods：该文件夹存放的是预编译的 Java 模块，相当于 JDK 9 之前的 .jar 文件。

legal：该文件夹存放的是有关 Java 每个模块的版权声明和许可协议等。

lib：该文件夹存放的是 Java 类库。

作为 JDK 的实用程序，工具库中的主要命令在 JDK 安装文件夹下 bin 子文件夹中，该子文件夹中包含了所有相关的可执行文件。下面是 bin 文件夹下的常用命令。

javac.exe：Java 编译器，将 Java 源文件转换为字节码文件。

java.exe：Java 解释器，执行 Java 程序的字节码文件。

javadoc.exe：根据 Java 源代码及注释语句生成 Java 程序的 HTML 格式的帮助文档。

javap.exe：把编译得到的字节码还原为源文件。

jdb.exe：Java 调试器，可以逐行执行程序、设置断点和检查变量。

jar.exe：创建扩展名为 .jar（Java Archive）的压缩文件，与 zip 压缩文件格式相同。

jmod.exe：创建扩展名为 .jmod 的压缩文件。

1.4.2 JDK 的操作环境

在使用 Java 编译与运行程序之前，必须先设置系统变量。所谓系统变量就是在操作系统中定义的变量，可供操作系统上的所有应用程序使用。若要使用 JDK17 的安装程序进行安装，则不需要人工设置路径 Path 和类路径 ClassPath 两个系统变量。但若要使用 JDK 压缩包 jdk-17_windows-x64_bin.zip 进行安装，就需要人工配置路径 Path 和类路径 ClassPath 两个系统变量。Path 系统变量的作用是设置供操作系统去寻找可执行文件（如 .exe、.com、.bat 等）路径的顺序，对 Java 而言即 Java 的安装路径，如果操作系统在当前文件夹下没有找到想要执行的程序或命令，操作系统就会按照 Path 系统变量指定的路径依次去查找，以最先找到的为准。Path 系统变量可以存放多个路径，路径与路径之间用分号“;”隔开。ClassPath 系统变量的作用与 Path 的作用相似，ClassPath 是 JVM 执行 Java 程序时搜索类文件（.class）的路径（类所在的文件夹）的顺序，以最先找到的为准。JVM 查找类的过程与 Windows 查找可执行文件的过程稍有不同，它默认不会在当前文件夹下查找，除非设置查找当前文件夹，否则只查找 ClassPath 系统变量指定的文件夹。即 JVM 除了在 ClassPath 系统变量指定的文件夹中查找要运行的类之外，是不会在其他文件夹下查找相应类的，由此可知 ClassPath 系统变量的作用就是告诉 Java 解释器在哪里找到 .class 文件及相关的库程序。因为本书使用的 JDK 17 是用安装文件进行安装的，安装程序默认将几个常用的开发工具（javac.exe、java.exe、javaw.exe 和 jshell.exe）自动复制到 C:\Program Files\Common Files\Oracle\Java\javapath 目录中，然后将该目录添加到系统变量 Path 中，所以不需配置路径 Path。同样也不用配置类路径系统变量 ClassPath，因为系统会自动找到 JRE 自带的 ClassPath，但若是使用第三方或用户自定义的类库，则还需要用户自己配置 ClassPath。所以不需配置路径 Path 和类路径 ClassPath 两个系统变量，Java 程序完全可以编译与运行。

1.4.3 JDK 帮助文档下载与安装

开发 Java 程序，除了需要 JDK 以外，拥有帮助工具也是很必要的。JDK 也提供了它的帮助文档，使用户在遇到问题时能快速得到解答，下面介绍 JDK 帮助文档的下载与安装操作。输入 Oracle 网站的网址 https://www.oracle.com/java/technologies/javase-jdk17-doc-downloads.html，即可进入 JDK 帮助文档下载页面进行下载。下载后得到 JDK17 帮助文档的压缩文件名为 jdk-17.0.1_doc-all.zip。可以将帮助文档 jdk-17.0.1_doc-all.zip 解压到先前安装 JDK 17 的文件夹中（也可以将其解压到其他文件夹中）。本书是解压到 C:\Program Files\Java\jdk-17.0.1 文件夹下。解压完成后，可以在该文件夹中看到 docs 子文件夹，打开它之后可看到 index.html 文件，双击即可打开帮助文档。

1.5 JDK 的使用

安装完 JDK 并设置好相应的系统变量后，就可以利用 JDK 来编译、运行 Java 程序了。下面介绍如何以最简单的方式来编写、编译与运行 Java 应用程序。在开始编写程序代码之前，先在硬盘 D 中创建一个名为 java 的文件夹，本书所有的例子均存储于 D:\java 文件夹下。

打开“文件资源管理器”窗口，在“此电脑”中选 D 盘的 java 文件夹，然后在下面的空白部分右击，在弹出的快捷菜单中选择“新建”选项，在弹出的二级菜单中选择“文本文档”选项。弹出如图 1.2（a）所示的询问打开该类文件所使用工具的对话框，在其中选择“记事本”选项，然后选中下方的“始终使用此应用打开 .txt 文件”的复选框。对于已经保存过的 .java 文件，若在其文件名上双击时会弹出如图 1.2（b）所示的窗格，同样选择“记事本”选项并勾选下边的“始终使用此应用打开 .java 文件”复选框即可，这样以后再以记事本打开 .java 文件时就不会再询问了。

（a）询问对话框　　（b）选择工具

图 1.2　选择打开文件所使用的工具

说明：目前在 Java 领域有很多优秀的集成开发工具，如 Eclipse IDE、NetBeans IDE、JCreator IDE、JDeveloper IDE 等，但还是建议初学者直接使用 Java SE 提供的 JDK，因为无论哪种集成开发环境都将 JDK 作为其核心，而且 IDE 界面操作复杂，主要是它还会屏蔽掉一些知识点，不利于初学者掌握基础知识。所以本书用 JDK 在命令行方式下直接编译与运行 Java 程序。

【例 1.1】编写一个 Java 应用程序（文件名为 App1_1.java），其功能是在命令行窗口中显示“Hello Java!”字符串。程序源文件代码如下：

```
1  //FileName: App1_1.java            简单的 Java 应用程序
2  public class App1_1{                                  //定义 App1_1 类
3    public static void main(String[] args){             //定义主方法
4      System.out.println("Hello Java !");
5    }
6  }
```

Java 应用程序源文件的命名规则：首先源文件的扩展名必须是 .java；如果源文件中有多个类，则最多只能有一个 public 类，如果有一个 public 类，那么源文件的名字必须与这个 public 类的名字相同（文件名字符的大小写可以与 public 类名的大小写不同）；如果源文件没有 public 类，那么源文件的名字由用户任意命名。

说明：

（1）当源文件中有 public 类时，在命名时虽然要求文件名与 public 类的名字相同，且可以不区分大小写，但良好的命名习惯应该是源文件名与 public 类名大小写完全相同。

（2）源文件名是由操作系统管理的，所以在使用 javac.exe 命令编译源文件时，文件名是不区分大小写的。

注意：包含有 main() 方法的类是 Java 应用程序的主类，主类无论是否是 public 类，但执行程序时必须输入主类名，即“java 主类名”，因为主类的 main() 方法是程序执行的起始点。

现在将源文件的内容输入记事本中，并把它存入 D:\java 文件夹内，根据 Java 对源文件命名规则的要求，必须将文件名命名为 App1_1.java，如图 1.3 所示。

在“另存为”对话框中文件名设为 App1_1.java，在“保存类型”下拉列表框内选择“所有文件”选项，如果此处选择“文本文件 (*.txt)”，将造成文件名称为 App1_1.java.txt，因而无法编译。在“编码”下拉列表框中选择 ANSI 选项。当单击“保存”按钮后弹出确认是否更改文件扩展名对话框，单击“是”按钮即可。

存好文件之后，接下来打开命令行窗口，并按下面的三个步骤来编译与运行 App1_1.java。

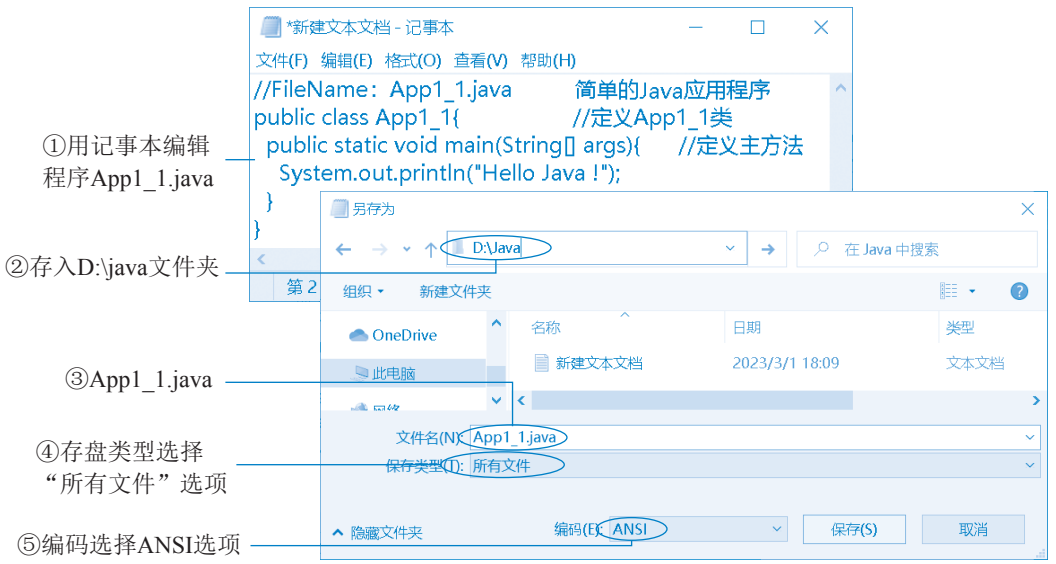

图 1.3 用记事本编写 Java 程序

（1）打开命令行窗口后，先将路径切换到保存 App1_1.java 的 D:\java 文件夹中，即在命令行窗口内输入：

```
d:
cd java
```

（2）切换好路径后，执行下面的命令来编译 App1_1.java。

```
D:\java > javac App1_1.java↙  // 带下画线的字符表示用户的输入，箭头表示按 Enter 键
```

在上面的命令中，javac 是用来编译其后给出的 Java 程序，编译好之后，在 D:\java 文件夹内发现一个与文件名 App1_1 相同但扩展名为 .class 的文件。这个文件也就是 byte-codes 文件，即字节码文件。

（3）编译好之后，执行下面的命令来运行字节码文件（即 App1_1.class）：

```
D:\java > java App1_1↙
```

则在命令提示符窗口输出

```
Hello Java!
```

注意：在运行字节码文件时，只需输入“java 主类名”即可，此处的主类名是指字节码的文件名，但不能把“.class”也输进去，即不能输入“java App1_1.class”来运行程序，这样将会造成错误。

如果源文件使用的编码字符集与运行环境命令行中的不同，则在编译源文件时需要在命令行中指定字符集选项。假设在图 1.3 中的第⑤步中选择的不是 ANSI 而是其他字符集，如 UTF-8，这时在命令行编译文件时需要在编译命令中给出字符集选项“-encoding UTF-8”，此时所使用的编译命令格式如下：

```
D:\java > javac -encoding UTF-8 App1_1.java
```

总之，当 .java 源文件所使用的字符集与命令行窗口的编码环境不一样时，在编译源文件时需要在编译命令中给出字符集选项“-encoding 字符集”，或者将源文件重新另存为与命令行窗口相同的字符集，即在图 1.3 中的第⑤步“编码”下拉列表中进行选择。

本章小结

1. JDK 的帮助文档（Java docs）与 Java 开发工具 JDK（Java Development Kit）同样是编写 Java 程序必备的工具。它们均可在 Oracle 公司的网站免费取得。

2. Java 应用程序源文件的命名规则：首先源文件的扩展名必须是 .java；如果源文件中有多个类，则最多只能有一个 public 类，如果有一个 public 类，那么源文件的名字必须与这个 public 类的名字相同（文件名字符的大小写可以与 public 类名的大小写不同）；如果源文件中没有 public 类，那么源文件的名字由用户任意命名。但需要注意的是，包含有 main() 方法的类是应用程序的主类，主类无论是否是 public 类，执行时必须输入主类名，即“java 主类名”，因为主类的 main() 方法是程序执行的入口点。

3. 因为 Java 程序是由类所组成的，所以在完整的 Java 程序中，至少要有一个类。

4. 当 .java 源文件所使用的字符集与命令行窗口的编码环境不一样时，在编译源文件时需要在编译命令中给出字符集选项“-encoding 字符集”。

习题

1.1 什么是 Java 虚拟机？

1.2 什么是平台无关性？ Java 语言怎样实现平台无关性？

1.3 Java 应用程序的结构包含哪几方面？

1.4 什么是 Java 应用程序的主类？应用程序的主类有何要求？

1.5 环境变量 Path 和 ClassPath 的作用是什么？

1.6 Java 应用程序源文件的命名有什么规定？

1.7 如何编译与命令行窗口字符集不同的 .java 源文件？