chapter1 Java概述

Java的位置如下图所示

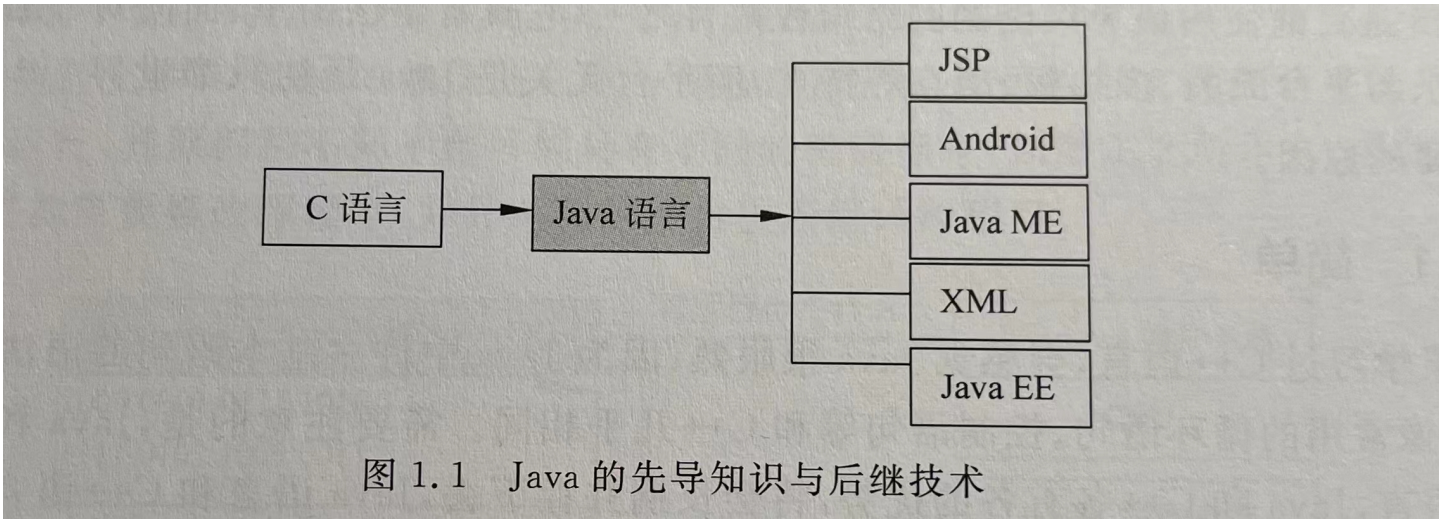

图 1.1　Java 的先导知识与后继技术

其中，
JSP-和Web设计相关、Java Me-手机程序设计、XML-数据交换技术相关、Java EE-网络中间件设计相关

1.Java的地位

- Java具有面向对象、与平台无关、安全、稳定和多线程等优良特性
- Java不仅可以用来开发大型的应用程序，而且适合Internet的开发

网络；Java的平台无惯性让Java成为编写网络应用程序的佼佼者，且Java也提供了许多以网络应用为核心的技术
需求： Web应用的JSP、设计手机应用的Android、嵌入式开发的Java ME等

2.Java的特点

- **语法简单**
  Java中许多语句和C++相同，但Java抛弃了许多C++中许多容易混淆的概念（如Java中不再有指针的概念），或者以一种更清楚、更容易理解的方式实现
- **面向对象**
- **平台无关**
  Java语言的出现源自于**对独立于平台语言的需要**，希望这种语言能编写出嵌入各种家用电器等设备的芯片上且易于维护的程序。

C和C++都有一个缺点：只能对特定的CPU芯片进行编译，所以一旦设备更换了芯片，就不能保证程序运行正确。

Java和其他语言相比，最大的优势就是**平台无关性**

> （DEF）即由Java语言编写的软件能在执行码上兼容，能在所有的计算机上运行，不因操作系统和处理器的变化而发生无法运行或错误的情况

》》Java可以在计算机的操作系统上再提供一个Java运行环境，该运行环境由Java虚拟机、类库以及一些核心文件组成。

》》只要平台提供了Java运行环境，Java编写的软件就能在其上运行

具体解释如下图：

因每个平台都会形成自己独特的机器指令（0、1序列）。不同的CPU和不同的操作系统所形成的平台的机器指令可能是不同的

表示减法指令。

❷ C/C++程序依赖平台

现在分析一下为何 C/C++语言编写的程序可能因为操作系统的变化、处理器升级导致程序出现错误或无法运行。

C/C++针对当前 C/C++源程序所在的特定平台对其源文件进行编译、连接，生成机器指令，即根据当前平台的机器指令生成可执行文件，那么，可以在任何与当前平台相同的平台上运行这个可执行文件。但是，不能保证 C/C++源程序所产生的可执行文件在所有的平台上都能正确地被运行，其原因是不同平台可能具有不同的机器指令（如图 1.2 所示）。因此，如果更换了平台，可能需要修改源程序，并针对新的平台重新编译源程序。

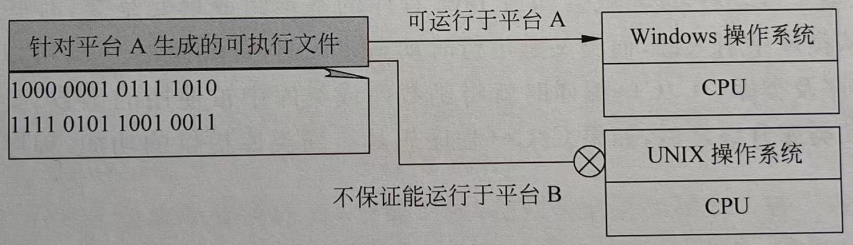

图 1.2　C/C++生成的可执行文件依赖平台

.

❸ Java 虚拟机与字节码

Java 虚拟机的核心是所谓的字节码指令，即可以被 Java 虚拟机直接识别、执行的一种由 0、1 组成的序列代码。字节码并不是机器指令，因为它不和特定的平台相关，不能被任何平台直接识别、执行。Java 针对不同平台提供的 Java 虚拟机的字节码指令都是相同的，例如所有的虚拟机都将 1111 0000 识别、执行为加法操作。

和 C/C++不同的是，Java 语言提供的编译器不针对特定的操作系统和 CPU 芯片进行编译，而是针对 Java 虚拟机把 Java 源程序编译成称为字节码的“中间代码”，例如，Java 源文件中的“+”被编译成字节码指令 1111 0000。字节码是可以被 Java 虚拟机识别、执行的代码，即 Java 虚拟机负责解释运行字节码，其运行原理是：Java 虚拟机负责将字节码翻译成虚拟机所在平台的机器码，并让当前平台运行该机器码，如图 1.3 所示。

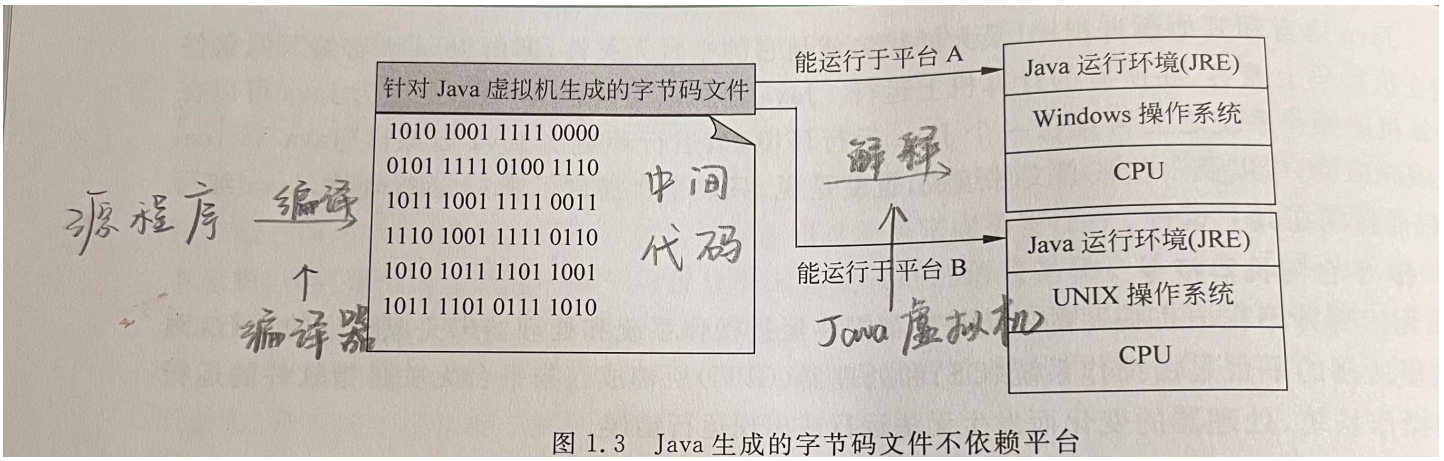

图 1.3　Java 生成的字节码文件不依赖平台

在一个计算机上编译得到的字节码文件可以复制到任何一个暗转了Java运行环境的计算机上直接使用。
字节码再由Java虚拟机负责解释运行，即Java虚拟机负责将字节码翻译成本地计算机的机器码（有特异性），并把机器码交给本地的操作系统来运行

> 【**程序**--(编译，不针对操作系统，而是Java虚拟机》所有系统中间代码都相同)--->
> --->**中间代码**（字节码文件）--(解释（翻译一句执行一句），Java虚拟机针对操作系统翻译成机器指令)-->
> --->**机器指令**】

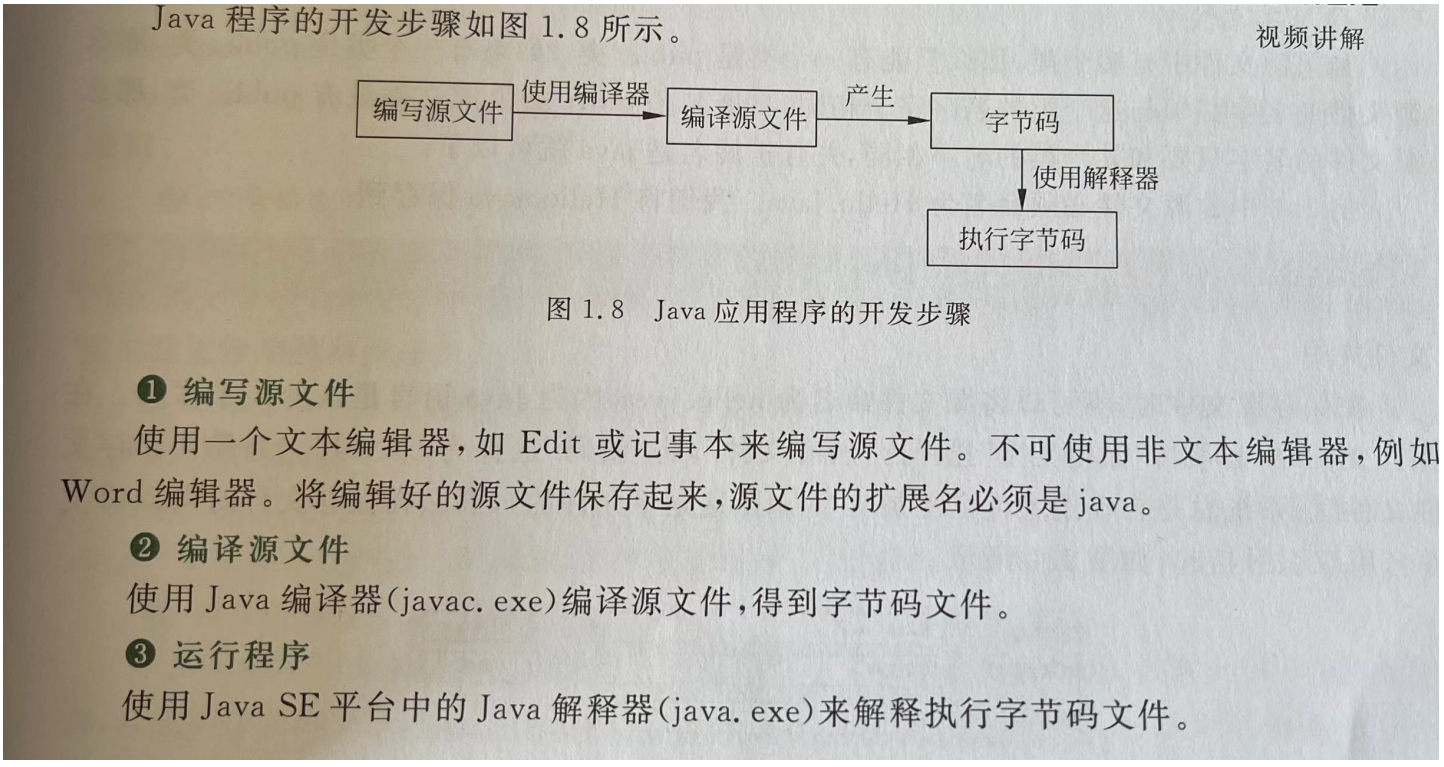

图 1.8　Java 应用程序的开发步骤

❶ 编写源文件

使用一个文本编辑器，如 Edit 或记事本来编写源文件。不可使用非文本编辑器，例如 Word 编辑器。将编辑好的源文件保存起来，源文件的扩展名必须是 java。

❷ 编译源文件

使用 Java 编译器(javac. exe)编译源文件，得到字节码文件。

❸ 运行程序

使用 Java SE 平台中的 Java 解释器(java. exe)来解释执行字节码文件。

- **多线程**
  多线程允许同时完成多个任务。但是计算机在同一时刻只能执行一个线程，但处理器可以在不同的线程之间快速切换。由于处理器速度非常快，远远超过了人接受信息的速度，所以给人的感觉好像多个任务在同时执行。

Meanwhile，C++中没有内置多线程机制，因此必须调用操作系统的多线程功能来进行设计（如Linux中的make -j4命令，使用虚拟机的4个内核多线程完成作业）

- **动态**
  Java程序的基本组成单元是类，有些类自己编写、有些类是从库中引用的，而【类是运行时动态装载的】
  可以让【Java可以在分布环境中动态地维护程序以及类库，不需要从新编译、修改】
  Meanwhile，C++中编译时就将函数库或类库同时生成机器码，那么每次修改类库的时候，C++程序若想要拥有新功能，程序就必须重新编译、修改

3. JDK的安装

Java Development Kit (同时安装了Java的运行环境)
为了保障Java的平台无关性（Java虚拟机对中间代码在不同机器上解释成相同的代码（形同的代码可以在不同的机器上解释成相同的含义）），就必须提供Java的运行环境

- Java SE（选择！！）
  Java标准版，提供了标准的JDK，可以开发Java桌面应用程序和服务器应用程序。
- Java EE
  Java企业版，可以构建企业级的服务应用【包含了Java SE ，并增加附加类库，以便支持目录管理、交易管理等功能】
- Java ME
  Java微型版，用于嵌入式系统中（移动电话、无线设备等）

4.简单的Java程序

4.1 编写和保存

- 源文件中如果有多个类，那么只能有一个类是public类！
- 如果有一个类是public，**【那么源文件的名字必须与这个类的名字完全相同】**
- 如果源文件没有public类，那么源文件的名字只要和某个类的名字相同，且扩展名是.java就可以啦！

在保存源文件时，不可以将源文件命名为 hello.java，因为 Java 语言是区分大小写的。在保存文件时，必须将“保存类型”选择为“所有文件”，将“编码”选择为“ANSI”。如果在保存文件文件时，系统总是自动给文件名尾加上“.txt”（这是不允许的），那么在保存文件时可以将文件名用双引号括起，如图 1.9 所示。

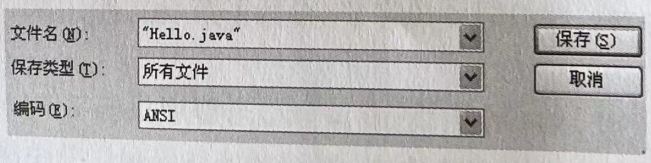

图 1.9　保存源文件

【小心自动添加了.txt呢！】

4.2 编译（要先设置环境变量，让编译器能找到这个文件）

编译器javac

在编写了hello.java的源文件之后，使用Java编译器(javac.exe)对其进行编译

step1：打开命令行窗口，进入源文件所在的文件夹

进入某个子目录（文件夹）的命令是，“cd 目录名”--和linux是一样的呢

字节码文件（.class文件）

如果源文件包含多个类，编译源文件将生成多个扩展名为class的文件，每个扩展名是class的文件中只存放一个类的字节码，其文件名和该类的名字相同。这些字节码文件被存放在和源文件相同的目录中。

tip：若源文件有语法错误，不生成字节码文件

若源文件有修改，则重新编译，再生成新的字节码文件

ex：编译hello.java文件

》生成hello.class student.class两个文件

tip：JDK5之后的编译器和之前版本的编译器有很大不同，不能再向下兼容。

高版本编译的字节码文件不能在低版本的Java运行环境中使用，但低版本可以在高版本的环境中使用

编译多个源文件

【法一】一次性使用javac一次编译多个源文件，只需要空格+多个文件名

```
C:\1000>javac Car.java Person.java
```

【法二】编译某个目录下的全部java源文件，使用通配符“*”代表各个源文件的名字来编译全部的源文件

```
C:\1000> javac *.java
```

4.3 运行

应用程序的主类

DEF：

一个Java程序必须要有一个类（至少一个，可以多个）含有【public static void main(String args[])方法】
称这个类是应用程序的主类。【S必须大写！！】

其中，args[]是main方法的一个参数，是一个字符串类型的数组
ex:hello.java中的主类是hello

```
public class Hello
{
  public static void main(String args[])
  {
    System.out.println("hello world");
    Student stu= new Student();
    stu.speak("we are future");
  }
}
class Student
{
  public void speak(String s)
  {
    System.outprintln(s);
  }
}
```

解释器java
使用java解释器(java.exe)来解释执行其字节码文件。【java程序总是从主类的main方法开始执行】
》》需要进入主类字节码所在目录，例如C:\chapter1 ，然后使用java解释器来运行主类的字节码（即.class文件）
【.java的名字是由public类决定。.class文件的名字是由public static void main()来决定的】

```
C:\chapter1\>java Hello
```

【当java应用程序中有多个类时，java解释器执行的类名必须是主类的名字！！（不能用扩展名.class）】

> 流程：
> **字节码文件**--java虚拟机 --> **内存**--java解释器-->**解释**

tip:
错误1：Exception in thread "main"java.lang.NoClassFondError
【没有正确地输入主类名】
错误2：程序通过编译，但无法运行
【主类中的方法缺少了static!】

以下为编写、编译、运行一个程序的完整过程

例 1.2

```
public class Rect {
    double width;                                    //长方形的宽
    double height;                                   //长方形的高
    double getArea(){ //返回长方形的面积
        return width * height;
    }
}
class Example1_2 {                                   //主类
    public static void main(String args[]) {
        Rect rectangle;
        rectangle = new Rect();
        rectangle.width = 1.819;
        rectangle.height = 1.5;
        double area = rectangle.getArea();
        System.out.println("矩形的面积:" + area);
    }
}
```

❶ 命名保存源文件

必须把例 1.2 中的 Java 源文件命名保存为 Rect.java(回忆一下源文件命名的规则)。假设将 Rect.java 保存在 C:\chapter1 下。

❷ 编译

```
C:\chapter1\> javac Rect.java
```

如果编译成功,chapter1 目录下就会有 Rect.class 和 Example1_2.class 两个字节码文件。

❸ 执行

```
C:\chapter1\> java Example1_2
```

java 命令后的名字必须是主类的名字(不包括扩展名)。

对上述 Java 程序进行编译、运行的操作步骤如图 1.12 所示。

```
C:\chapter1>javac Rect.java

C:\chapter1>java Example1_2
矩形的面积:2.7285
```

图 1.12 注意主类的名字

5.Java应用程序的基本结构

- Java以类作为“基本单位”，一个java程序就是由若干个类组成。
- 一个java应用程序（工程）只有一个主类
- 一个java程序可以将它使用的各个类分别存放在不同的源文件中，也可以把它使用的类存放在一个源文件中。
  同时，从编译的角度看，每个源文件都是一个独立的编译单位，当程序需要修改某个类时，只需要重新编译该类所在的源文件即可，不必重新编译其他类所在的源文件！

ex:以下为多文件编译的例子

例 1.3

Circle.java

```
public class Circle {
    void printArea(double r) {
        System.out.println(r * r * 3.1416926);
    }
}
```

Rectangle.java

```
public class Rectangle {
    void printArea(double a,double b){
        System.out.println(a * b);
    }
}
```

MainClass.java

```
public class MainClass {
    public static void main(String args[]) {
        Circle circle = new Circle();
        circle.printArea(100);
        Rectangle rect = new Rectangle();
        rect.printArea(100,65);
    }
}
```

假设上述 3 个源文件都保存在：

```
C:\chapter1
```

在命令行窗口进入上述目录，并编译 MainClass.java：

```
javac MainClass.java
```

编译 MainClass.java 的过程中，Java 系统会自动地编译 Circle.java 和 Rectangle.java，这是因为应用程序要使用 Circle.java 和 Rectangle.java 源文件产生的字节码文件。编译通过后，C:\chapter1 目录中将会有 Circle.class 、Rectangle.class 和 MainClass.class 3 个字节码文件。

--- 因为是3个文件，每个文件中的类都被定义成了public。所以3个文件都得命名成类ming.java
--- 因为MainClass中调用了另外两个java程序，所以编译MainClass之后，系统会自动编译其他两个java文件
--- 因为主函数的MainClass之中，所以最后一步解释器解释的时候应该是
C:\1000>java MainClass

6.注释和风格

注释

// 单行

/* */ 多行

编程风格

1、Allmans风格--独行 风格

```
class Allmans
{
  public static void main(String args[])
  {
    ...
  }
}
```

【代码量小的时候用！代码布局清晰，可读性强】

2、Kernighan风格--行尾 风格

```
class Kernighan{
  public static void main(String[]){
    ...
  }
}
```

【代码量大的时候用！】

如果此时仍要用独行风格，会导致代码左半部分出现大量的左、右大括号，导致代码清晰度下降！