# 第 1 章　Java 语言概述

**本章导读**

- Java 语言的诞生
- 学习 Java 的必要性
- Java 的特点及与 C/C++之关系
- Java 程序开发
- JDK 1.6 编译器的新规定

在学习 Java 语言之前，读者应当学习过 C 语言，熟悉计算机的一些基础知识。读者学习过 Java 语言之后，可以继续学习与 Java 相关的一些重要内容。比如，如果希望从事编写和数据库相关的软件，可以深入学习 Java DataBase Connection（JDBC）；如果希望从事 Web 程序的开发，可以学习 Java Server Page（JSP）；如果希望从事手机应用程序的设计，可以学习 Java Micro Edition（Java ME）；如果希望从事与网络信息交换有关的软件设计，可以学习 eXtensible Markup Language（XML）；如果希望从事大型网络应用程序的开发和设计，可以学习 Java Enterprise Edition（Java EE），如图 1-1 所示。

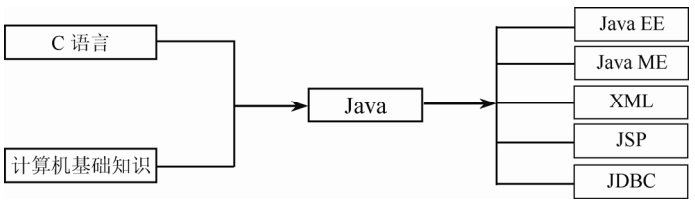


图 1-1　Java 的先导知识和后继技术

## 1.1　Java 语言的诞生

Java 诞生于 1995 年，是 Sun 公司组织开发的一门编程语言，主要贡献者是 James Gosling。开发 Java 语言的动力源于对独立于平台的需要，即这种语言编写的程序不会因为芯片的变化而发生无法运行或运行错误。当时，C 语言已无法满足人们的这一愿望，因为 C 语言总是针对特定的芯片将源程序编译为机器码，该机器码的运行与特定的芯片指令有关，在其他不同类型的芯片上可能无法运行或出现运行错误。芯片制造者、芯片使用者和软件编写者往往隶属于不同的公司。芯片制造者会不断地推出性能更好且价格更便宜的新型芯片，当有一种性价比更高的芯片出现时，芯片使用者就可能立即使用新的芯片，这些芯片可能安装在各种计算机或电子设备上，如果不能保证程序在新的芯片上正确运行，就可能出现难以发现的错误，

最终导致严重的后果，可能引起设备的毁坏等灾难性后果。所以，软件编写者必须针对新的芯片重新编译源程序，甚至需要对源程序进行必要的修改，这是令软件开发者最头痛的工作，还可能由于各种原因（如工作量的巨大等）而导致工作无法完成。

1990 年，Sun 公司成立了由 James Gosling 领导的开发小组，开始致力于开发一种可移植的、跨平台的语言，该语言能生成正确运行于各种操作系统、适应各种 CPU 芯片的代码。他们的精心钻研和努力促成了 Java 语言的诞生。Java 的快速发展得益于 Internet 和 Web 的出现，Internet 上有各种计算机，它们可能使用完全不同的操作系统和 CPU 芯片，但仍希望运行相同的程序，Java 的出现标志着真正的分布式系统的到来。

James Gosling 的办公室外面有一棵大橡树，他最初将 Java 语言命名为 oak，后来发现已经有一种计算机语言的名字叫 oak，最后决定为这种新的语言起名为 Java，其寓意是为世人端上一杯热咖啡。“Java”是印度尼西亚一个盛产咖啡的岛屿，中文译名是“爪哇”。

## 1.2　学习 Java 的必要性

Java 不仅可以用来开发大型的桌面应用程序，还特别适合 Internet 的应用开发。目前，Java 语言不但是一门正在被广泛使用的编程语言，而且已成为软件设计开发者应当掌握的一门基础语言。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，而且很多新的技术领域都涉及 Java 语言。因此，学习和掌握 Java 已成为共识，国内外许多大学将 Java 语言列入了本科教学计划。IT 行业对 Java 人才的需求正在不断增长，一些软件公司对其开发人员周期地进行 Java 的基础培训工作。在 IT 行业发达的北美洲，有将近 60%的软件开发人员使用 Java 完成他们的工作，Evans Data 公司在 2002 年做的一项调查中发现：在北美洲，Java 的使用率已经接近 C/C++。

2003 年，James Gosling 曾来北京，与中国的 IT 人士进行了交流，以下是对话的节选。

问：在近几年的发展过程中，很多编程语言都逐渐消失，Java 语言却越来越火，请问其中的原因是什么？

**James Gosling：**我认为，很多编程语言在发展中并不是消失，而是转移到了其他领域中，而 Java 的经久不衰取决于 Java 的技术基础。如果你问编程师，为什么会选择 Java，他会告诉你，Java 提供了多种功能，提供了方便的平台，是个足以吸引人的工具。我认为，推动 Java 最主要的因素是网络，Java 是以网络应用为基础的开发工具，这是它的强项。

问：在传统计算机领域中，Java 并不是十分大的平台，如 PC。而在其他领域，如移动领域，Java 发展迅速，Java 的未来发展方向是什么？

**James Gosling：**在 PC 领域，我并不认为 Java 不够强大。在 PC 领域，Java 有很多应用，这是表面上看不到的，主要因为微软花了大力气避免用户看到，实际上 Java 应用很广泛，如人工智能游戏。在其他领域，Java 更是应用广泛，如汽车、铁路机车上的即时控制系统，以及军用方面。

问：大家尊称您为 Java 之父，您能不能跟大家分享一下您在 Java 事业中最深的感受是什么？

**James Gosling：**当看到 Java 的客户通过 Java 完成了很多神奇的工作，如看到夏威夷火山上的观测台使用 Java 控制望远镜，看到荷兰健康医疗组织使用 Java 解决了保护隐私问题等，那真是一种奇妙的感觉。

## 1.3　Java 的特点

### 1．平台无关性

（1）平台与机器指令

无论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，因此这里所指的平台由操作系统（OS）和处理器（CPU）构成。与平台无关是指软件的运行不因操作系统、处理器的变化导致程序无法运行或出现运行错误。

所谓平台的机器指令，就是可以被该平台直接识别、执行的一种由 0 和 1 组成的序列代码。注意，相同的 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，因此每种平台都会形成自己独特的机器指令。比如，某平台可能用 8 位序列代码 1000 1111 表示一次加法操作，用 1010 0000 表示一次减法操作，另一平台可能用 8 位序列代码 1010 1010 表示一次加法操作，用 1001 0011 表示一次减法操作。

（2）C/C++程序依赖平台

现在来分析为何 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++语言提供的编译器对 C/C++源程序进行编译时，将针对当前 C/C++源程序所在的特定平台进行编译、连接，然后生成机器指令，即根据当前平台的机器指令生成机器码文件（可执行文件）。这样无法保证 C/C++编译器所产生的可执行文件在所有平台上都能正确地被运行，因为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。

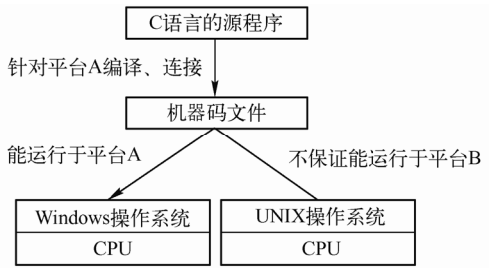


图 1-2　C/C++生成的机器码文件依赖平台

（3）Java 程序不依赖平台

与其他语言相比，Java 语言最大的优势就是它的平台无关性，这也是 Java 风靡全球的主要原因。Java 在平台之上再提供一个 Java 运行环境（Java Runtime Environment，JRE），该 Java 运行环境由 Java 虚拟机（Java Virtual Machine，JVM）、类库及一些核心文件组成。Java 虚拟机的核心是字节码指令，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，因为它不与特定的平台相关，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，如所有的虚拟机都将 1111 0000 识别、执行为加法操作。

与 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 源程序编译为称为字节码的一种“中间代码”。比如，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。

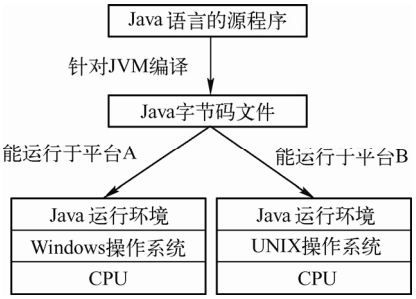


图 1-3　Java 生成的字节码文件不依赖平台

### 2．面向对象

面向对象编程是一种先进的编程思想，更加容易解决复杂的问题。面向对象编程主要体现在下列三个特性。

① 封装。面向对象编程的核心思想之一就是将数据和对数据的操作封装在一起。抽象即从具体的实例中抽取共同的性质形成一般的概念，如类的概念。人们经常谈到的机动车类就是从具体的实例中抽取共同的属性和功能形成的一个概念，那么一个具体的轿车就是机动车类的一个实例，即对象。一个对象将自己的数据和对这些数据的操作合理、有效地封装在一起，如每辆轿车调用“加大油门”改变的都是自己的运行速度。

② 继承。继承体现了一种先进的编程模式。子类可以继承父类的属性和功能，即继承了父类的数据和数据上的操作，又可以增加子类独有的数据和数据上的操作。比如，“人类”自然继承了“哺乳类”的属性和功能，同时增加了人类独有的属性和功能。

③ 多态。多态性是面向对象编程的又一重要特征。多态有两种。一种是操作名称的多态，即有多个操作具有相同的名字，但这些操作接收的消息类型必须不同。另一种是与继承有关的多态，是指同一个操作被不同类型调用时可能产生不同的行为。

Java 是面向对象的编程语言，本书将在第 4 章详细、准确地讨论类、对象、继承、多态、接口等重要概念。

### 3．多线程

Java 的特点之一就是内置对多线程的支持。多线程允许同时完成多个任务，使人产生多个任务在同时执行的错觉。目前，计算机的处理器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，远远超过了人们接收信息的速度，所以感觉好像多个任务在同时执行。C++没有内置的多线程机制，因此必须调用操作系统的多线程功能来进行多线程程序的设计。

### 4．安全

当用户准备从网络上下载一个程序时，最大的担心是程序中含有恶意的代码，如试图读取或删除本地机上的一些重要文件，甚至该程序是一个病毒程序等。当用户使用支持 Java 的浏览器时，可以放心地运行 Java Applet（Java 小应用程序），不必担心病毒的感染和恶意的企图，Java Applet 将限制在 Java 运行环境中，不允许它访问计算机的其他部分。本书将在第 12 章详细讲述 Java Applet。

### 5．动态

在学习了第 4 章后就会知道，Java 程序的基本组成单元就是类，有些类是用户自己编写的，有些是从类库中引入的。而类是运行时动态装载的，这就使得 Java 可以在分布环境中动态地维护程序及类库，而不像 C++那样，每当其类库升级之后，如果想让程序具有新类库提供的功能，就必须重新修改、编译程序。

## 1.4　Java 与 C/C++之关系

如果学习过 C++语言，读者会感觉 Java 很眼熟，因为 Java 中许多基本语句的语法与 C++类似，像常用的循环语句、控制语句等与 C++几乎一样，但不要误解为 Java 是 C++的增强版。Java 和 C++是两种完全不同的语言，它们各有各的优势，Java 语言和 C++语言已成为软件开发者应当掌握的语言。如果从语言的简单性方面看，Java 要比 C++语言简单，C++语言中许多容易混淆的概念或者被 Java 语言弃之不用，或者以一种更清楚、更容易理解的方式实现，如 Java 语言不再有指针的概念。Java 语言既易学又好用，但不要误解为这门语言很干瘪。读者可能很赞同这样的观点：英语要比阿拉伯语言容易学，但这并不意味着英语就不能表达丰富的内容和深刻的思想。

## 1.5　Java 运行平台

### 1．三种平台简介

Sun 公司要实现“编写一次，到处运行（write once，run anywhere）”的目标，就必须提供相应的 Java 运行平台。目前，Java 运行平台主要分为下列 3 个版本。

- Java SE（曾称为 J2SE）——Java 标准版或 Java 标准平台。Java SE 提供了标准的 JDK 开发平台，利用该平台可以开发 Java 桌面应用程序和低端的服务器应用程序，也可以开发 Java Applet。
- Java EE（曾称为 J2EE）——Java 企业版或 Java 企业平台，可以构建企业级的服务应用。Java EE 平台包含了 Java SE 平台，并增加了附加类库，以便支持目录管理、交易管理和企业级消息处理等功能。
- Java ME（曾称为 J2ME）——Java 微型版或 Java 小型平台。Java ME 是一种很小的 Java 运行环境，用于嵌入式的消费产品中，如移动电话、掌上电脑或其他无线设备等。

登录 Sun 公司的网站 http://java.sun.com，就能看到有关 Java SE、Java EE 和 Java ME 的介绍。上述 Java 运行平台都包括了相应的 JVM，JVM 负责将字节码文件（包括程序使用的类库中的字节码）加载到内存中，然后采用解释方式来执行字节码文件，即根据相应硬件的机器指令翻译一句，执行一句。

### 2．安装 Java SE 平台

学习 Java 应当从 Java SE 开始，因此本书基于 Java SE 来学习 Java。目前，Sun 公司已发布了 JDK 1.6，可以登录到其官网免费下载（如 jdk-6u3-windows-i586-p.exe），在“Popular Downloads”页面中选择“Java SE”→“JDK 6 Update”，单击“下载”即可。双击 jdk-6u3-windows-i586-p.exe 文件图标，将出现安装向导界面，接受软件安装协议，出现选择安装路径界面。为了便于今后设置环境变量，建议修改默认的安装路径。这里将默认的安装路径“C:\program Files\Java\Jdk1.6.0_3”修改为“E:\jdk1.6”，如图 1-4 所示。

注：在安装过程中会出现安装支持欧洲语言的 JRE（Java Runtime Environment）的界面，在该界面上不必更改默认的安装路径，使用默认的安装路径即可。JRE 的安装路径不可以与 JDK 的安装路径相同。

将 JDK 安装到 E:\jdk1.6 目录下会生成如图 1-5 所示的目录结构。现在就可以编写 Java 程序并进行编译、运行程序了，因为安装 JDK 的同时计算机就安装上了 Java 运行环境（JRE）。


图 1-4　JDK 的安装路径


图 1-5　JDK 的目录结构

JDK 主要目录内容如下。

① 开发工具：位于 bin 子目录中，包括工具和实用程序，可以开发、执行、调试和保存用 Java 编程语言编写的程序。

② Java 运行环境：位于 jre 子目录中，由 JDK 使用的 JRE 实现。JRE 包括 Java 虚拟机（JVM）、类库及其他支持执行用 Java 语言编写的程序的文件。

③ 附加库：位于 lib 子目录中，包括开发工具所需的其他类库和支持文件。

④ 演示 Applet 和应用程序：位于 demo 子目录中，Java 平台的编程示例（带源代码）。这些示例包括使用 SWING 和其他 Java 基类以及 Java 平台调试器体系结构的示例。

⑤ 样例代码：位于 sample 子目录中，某些 Java API 的编程样例（带源代码）。

⑥ C 头文件：位于 include 子目录中，支持使用 Java 本机界面、JVM 工具界面及 Java 平台的其他功能进行本机代码编程的头文件。

⑦ 源代码：位于 JDK 安装目录之根目录中的 src.zip 文件是 Java 核心 API 的所有类的 Java 编程语言源文件（即 java.*、javax.*和某些 org.*包的源文件，但不包括 com.sun.*包的源文件）。

（1）系统环境 path 的设置

JDK 平台提供的 Java 编译器（javac.exe）和 Java 解释器（java.exe）位于 Java 安装目录的\bin 子目录中，为了能在任何目录中使用编译器和解释器，应在系统特性中设置 path。对于 Windows XP，右键单击“我的电脑”，在弹出的快捷菜单中选择“属性”，弹出“系统特性”对话框，再单击“高级选项”，然后单击 “环境变量”按钮，添加系统环境变量。如果曾经设置过环境变量 path，可单击该变量进行编辑，添加需要的值，如图 1-6 所示。也可以在命令窗口（如 MS-DOS 窗口）中输入“path = E:\jdk1.6\bin;”。

（2）系统环境 classpath 的设置

JDK 的安装目录的\jre 子目录中包含 Java 应用程序运行时所需的 Java 类库，这些类库被包含在\jre\lib 中的压缩文件 rt.jar 中。安装 JDK 一般不需要设置环境变量 classpath 的值，如果读者的计算机安装过一些商业化的 Java 开发产品或带有 Java 技术的一些产品。安装这些产品后，classpath 的值可能会被修改。那么运行 Java 应用程序时，读者加载这些产品所带的老版本的类库，可能导致程序要加载的类无法被找到，使程序出现运行错误。读者可以重新

编辑系统环境变量 classpath 的值。对于 Windows 2000/2003/XP，右键单击“我的电脑”，在弹出的快捷菜单中选择“属性”，弹出“系统特性”对话框，再单击该对话框中的“高级选项”，然后单击“环境变量”按钮，添加如图 1-7 所示的系统环境变量。如果曾经设置过环境变量 classpath，可单击该变量进行编辑操作，添加需要的值。


图 1-6　设置环境变量 path

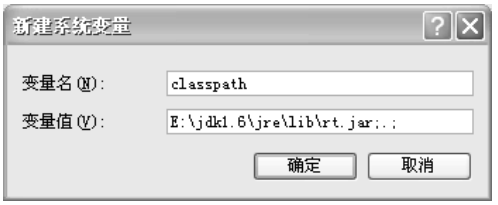


图 1-7　设置环境变量 classpath

对于 Windows 9x，用记事本编辑 autoexec.bat 文件，加入设置语句“set classpth=E:\jdk1.6\jre\lib\rt.jar;.;”即可，也可在命令行窗口（如 MS-DOS）中输入“set classpth=E:\jdk1.6\jre\lib\rt.jar;.;”。

环境变量 classpath 设置中的“.;”是指可以加载应用程序当前目录及其子目录中的类。

（3）仅仅安装 JRE

如果读者只想运行别人的 Java 程序，可以只安装 Java 运行环境 JRE。JRE 由 JVM、Java 的核心类及一些支持文件组成。读者可以登录 Sun 公司的官网免费下载 JRE。

（4）一些 IDE 开发工具

还有一些其他很好的 Java 程序 IDE 开发环境可用，包括来自 Sun、Borland、Symantec 公司的产品，如 Sun One、JBuilder 和 Eclipse 等，目前以 Eclipse 最流行。这些 IDE 产品都集成 JDK 作为主要部分。初学者应当使用 JDK 来开发 Java 程序，这不仅是学习 Java 最好的方式，还是掌握使用 IDE 开发工具的必要条件。IDE 开发环境适合于设计开发大型项目时使用，不适合于学习 Java 语言。

建议下载 Sun 公司的 Java 类库帮助文档，如 jdk-6-doc.zip。

## 1.6　Java 程序开发

用 Java 标准平台编译得到的字节码文件，可以在任何具有 Java 标准平台的计算机上正确地运行。开发一个 Java 应用程序需要经过三个步骤：编写源文件 → 编译源文件，生成字节码 → 加载运行字节码。

### 1．编写源文件

（1）源文件的结构

应使用文本编辑器（如 Edit 或记事本）来编写源文件，不可使用 Word 编辑器，因它包含不可见字符。Java 是面向对象编程，Java 应用程序的源文件由若干个书写形式互相独立的类组成。

【例 1-1】 编写 3 个类：A，B 和 Hello。

```
class A {
   void f() {
      System.out.println("I am A");
   }
}
```

```
class B {  }
public class Hello {
   public static void main (String args[ ]) {
      System.out.println("你好，很高兴学习 Java");
      A a=new A();
      a.f();
   }
}
```

class 是 Java 的关键字，用来定义类。public 也是关键字，说明 Hello 是一个 public 类，本书在第 4 章将系统讲述类的定义和使用。但现在必须知道：Java 应用程序的源文件由若干个书写形式互相独立的类组成，其中 class Hello 称为类声明，之后的第一个花括号和最后一个花括号以及它们之间的内容叫类体。

（2）应用程序的主类

Java 应用程序必须有一个类包含 public static void main(String args[])方法。这个类称为应用程序的主类。args[]是 main 方法的一个参数，是一个字符串类型的数组（String 的第一个字母是大写的），以后会学习怎样使用这个参数。

（3）源文件的命名规则

源文件的命名规则如下：① 如果源文件中包括多个类，那么只能有一个类是 public 类；② 如果有一个类是 public 类，那么源文件的名字必须与这个类的名字完全相同，扩展名是 .java；③ 如果源文件没有 public 类，那么源文件的名字只要与某个类的名字相同，并且扩展名是 .java 就可以了。

将上述源文件保存到 C:\1000 文件夹中，并命名为 Hello.java。注意，不可写成“hello.java”，因为 Java 语言是区分大小写的。

（4）良好的编程习惯

在编写程序时，一行最好只写一条语句。类体以及方法的花括号最好也独占一行，并有明显的缩进；也可把“{”位于类声明同行的末尾，“}”另起一行。对于空语句，“}”应紧跟在“{”之后。在编写代码时，应养成良好的编程习惯。

### 2．编译 Java 源文件

创建了源文件 Hello.java 后，就要使用编译器对其进行编译：

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

需要打开 MS-DOS 命令行窗口，进入 C:\，然后进入 C:\1000 目录，如图 1-8 所示。如果用户不曾设置过 path，需要在当前命令窗口中输入命令“path = E:\jdk1.6\bin; ”，然后才可以使用 javac 来编译源文件。

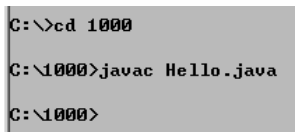


图 1-8　使用 javac 编译源文件

如果 Java 源程序中包含了多个类，那么用编译器（javac.exe）编译完源文件后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，其文件名与该类的名字相同。这些字节码文件被存放在与源文件相同的目录中。上述源文件编译成功后将得到 3 个字节码文件：A.class，Hello.class 和 B.class。如果对源文件进行了修改，那么读者必须重新编译，再生成新的字节码文件。

### 3．运行 Java 应用程序

Java 应用程序必须通过 Java 虚拟机中的 Java 解释器（java.exe）来解释执行其字节码文

件。Java 应用程序总是从主类的 main 方法开始执行。因此，必须如下运行 Java 应用程序：

```
C:\1000\>java Hello
```

运行效果如图 1-9 所示。


图 1-9　使用解释器运行程序

当 Java 应用程序中有多个类时，Java 命令执行的类名必须是主类的名字（没有扩展名）。

当使用解释器运行应用程序时，Java 虚拟机首先将程序需要的字节码文件加载到内存，然后解释、执行字节码文件。当运行上述 Java 应用程序时，Java 虚拟机仅仅将 Hello.class 和 A.class 加载到内存，B.class 并没有加载到内存，因为程序的运行并未用到类 B。当 Java 虚拟机将 Hello.class 加载到内存时，就为主类中的 main 方法分配了入口地址，以便 Java 解释器调用 main 方法开始运行程序。如果编写程序时错误地将主类中的 main 方法写成：public void main(String args[])，那么程序可以编译通过，但无法运行。本书将在第 4 章介绍 static 关键字的含义。如果主类中的 main()方法没用 static 修饰，Java 虚拟机将 Hello.class 加载到内存中时，就不会为这样的 main()方法分配入口地址，Java 解释器就无法找到 main()方法（实际上，源程序也就没有了主类）。

如果应用程序编译正确，也有正确的主类，但程序仍无法运行，一定是 classpath 的设置出现了问题。classpath 设置中一定要包括当前目录中的类，如

```
set classpath=E:\jdk1.6\jre\lib\rt.jar;.;
```

有关环境变量的设置见本章的 1.5 节。

使用 JDK 环境开发 Java 程序，需运行 MS-DOS 命令窗口。读者需要知道简单的 DOS 操作命令，如从逻辑分区 C 转到逻辑分区 D，需在命令行中输入“D:”，回车确定。

进入某个子目录（文件夹）的命令是：

```
cd  目录名
```

退出某个子目录的命令是：

```
cd..          或          cd/
```

例如，从子目录 book 退到父目录 like 的命令是“C:\like>book>cd..”。

我们再看一个简单的 Java 应用程序。也许读者还看不懂这个源程序，但应该知道怎样命名、保存源程序，怎样使用编译器编译源程序，怎样使用解释器运行程序。

**【源程序】**

```
public class Tom {
    int leg;
    String head;
    void cry(String s) {
        System.out.println(s);
    }
}
class Example {
    public static void main(String args[]) {
        Tom cat;
        cat=new Tom();
        cat.leg=4;
        cat.head="猫头";
        System.out.println("腿:"+cat.leg+"条");
        System.out.println("头:"+cat.head);
```

```
            cat.cry("我今天要和 Jerry 拼了");
        }
    }
```

我们必须把源文件保存起来并命名为 Tom.java。假设保存 Tom.java 在 C:\1000 下。

（1）编译源文件

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

（2）执行

```
C:\1000\>java Example
```

Java 命令后的名字必须是主类的名字，运行效果如图 1-10 所示。

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

C:\1000>java Example
腿:4条
头:猫头
我今天要和Jerry拼了

C:\1000>
```

图 1-10　程序运行效果

如果编译成功，目录 C:\1000 下就会出现 Tom.class 和 Example.class 这两个字节码文件。

## 1.7　JDK 1.6 编译器的兼容性

与 JDK 1.5 一样，JDK 1.6 的编译器 javac.exe 与 1.4 版本之前的编译器不同，不再向下兼容。也就是说，如果在编译源文件时没有特别约定，用 JDK 1.6 的编译器生成的字节码只能在安装了高于 JDK 1.6 或 JRE 1.6 的 Java 平台环境中运行。

可以使用“-source”参数约定字节码适合的 Java 平台。如果程序中并没有用到 JDK 1.6 的新功能，在编译源文件时可以使用“-source”参数，如“javac -source 1.2 文件名.java”。这样编译生成的字节码可以在 1.2 版本以上的 Java 平台运行。如果源文件使用的系统类库没有超出 JDK 1.1 版本，在编译源文件时应当使用-source 参数，取值 1.1，使得字节码有更好可移植性。

-source 参数可取的值有 1.6，1.5，1.4，1.3，1.2 和 1.1。

如果没有明显地使用“-source”参数，javac 默认使用该参数，并取值为 1.6。

## 问 答 题

1．发明 Java 语言的原因是什么？发明 Java 语言的主要贡献者是谁？

2．“Java 编译器将源文件编译生成的字节码是机器码”，这句话正确吗？

3．Java 应用程序的主类必须包含怎样的方法？

4．“Java 应用程序必须有一个类是 public 类”，这句话正确吗？

5．请叙述 Java 源文件的命名规则。

6．源文件生成的字节码在运行时都加载到内存中吗？

7．怎样编写加载运行 Java Applet 的简单网页。

8．JDK 1.6 编译器使用-source 参数的作用是什么？-source 参数的默认取值是什么？

## 作 业 题

1．参照例 1-1 编写一个 Java 应用程序，程序能在命令行中输出“撸起袖子加油干，The nation remains mobilized for brand new endeavors.”。