第 1 章　Java 概述

多年来，为什么 Java 语言一直在程序设计语言排行榜中独占鳌头，发展成为程序设计语言的常青树，而精通 Java 语言也为职业规划提供很多优势？让我们循序渐进，走进 Java，学习 Java，掌握 Java。

本章主要内容

- Java 与 C、C++
- Java 语言的特点
- Java 开发工具
- Java 程序的类型及其不同编程模式

【案例分析】

使用面向对象方法，描述现实世界中的一个实体——售报亭，如图 1.1 所示。

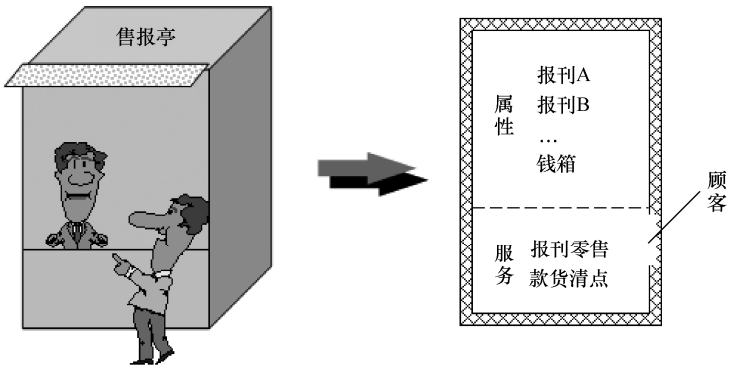

图 1.1　售报亭的对象封装

面向对象方法解决问题的思路是：从现实世界中的客观对象（如人和事物）入手，尽量运用人类的自然思维方式来构造软件系统。

在面向对象方法中，把一切都看成对象。把对象的属性和服务操作结合成一个独立的系统单位，其属性与操作刻画了事物的性质和行为，并尽可能隐蔽对象的内部细节，向外部只提供接口。软件对象是数据和方法的封装体。如图 1.1 所示，属性对应软件对象的数据，服务对应软件对象的方法。

在面向对象系统中，无论系统的构成成分，还是通过这些成分之间的关系而体现的系统结构，都可直接地映射问题域。这使得运用面向对象方法有利于我们正确理解问题域及系统责任。

1.1　Java 崛起

1991 年，美国 Sun Microsystems 公司启动了名为“Green Project”的研究项目，研究

解决家用电器的智能通信和控制问题。开发小组最初构想以当时颇为流行的 C++语言开发此项智能软件。后来，由于 C++语言本身的复杂性、安全性及平台移植方面的障碍等问题，因此项目组最后决定另辟蹊径，最终他们基于 C++重新开发了一套新的语言系统——Java 语言。

Java 语言的始创者是 Sun 公司的 James Gosling。起先，他根据办公室窗外的一棵橡树（Oak）将其命名为 Oak 语言。在申请注册时，因为命名冲突问题，后来将其改名为 Java 语言。

Java 语言可以称得上是一种精巧而安全的语言。然而，当时 Sun 公司一开始就遭遇了“智能化家用电器”市场的萧条。同时，Sun 公司以它投标一个自认为乐观的交互式电视项目时也折戟而归，未能成功。在这种情况下，Java 语言似乎生不逢时，Green 项目几乎走入了绝境。

可谓绝处逢生，峰回路转，1993 年万维网空前流行起来。Java 的发展转向了网络应用领域。

Java 语言具有平台无关性，使得 Java 程序适应了 Internet 上多样化的服务器站点环境，Java 程序既可以在 Windows 平台上运行，又可以在 Unix、Linux 等平台上运行。这造就了 Sun 公司宣传的“Write Once，Run Anywhere”（一次编写，随处运行）的优势。

1995 年 5 月，Sun 公司正式对外发布了 Java 语言，并随着互联网的飞速发展，逐渐确定了自己网络编程语言的地位。当年 Java 就被美国著名杂志《PC Magazine》评为十大优秀科技产品之一。1996 年 1 月，JDK 1.0 发布。2009 年 4 月，甲骨文公司收购 Sun 公司，取得了 Java 版权。

之所以命名为 Java 语言，有两种说法：其一，印度尼西亚有一个重要的岛屿——爪哇岛，盛产咖啡，开发人员起名 Java 寓意为世人端上一杯热腾腾的咖啡；其二，美洲俚语 Java 有咖啡之意。

Java 平台主要包括 Java SE（Standard Edition，早期的 J2SE）、Java EE（Enterprise Edition，早期的 J2EE)、Java ME（Micro Edition，早期的 J2ME）。Java SE 称为 Java 标准版或 Java 标准平台；Java EE 称为 Java 企业版或 Java 企业平台，用于构建企业级的服务应用；Java ME 为 Java 微型版，用于移动/嵌入式开发平台。其中，Java SE 是整个 Java 开发的基础。

总之，在计算机领域中很少出现过像现在所发生的 Internet/WWW/Java 这样的“火爆”现象。

延伸阅读：万维网和因特网

WWW 是环球信息网（World Wide Web）的缩写，简称为 Web，中文名称为“万维网”。万维网包括 WWW 服务器和 WWW 浏览器。万维网是一个资源空间，由“统一资源标识符”（URL）标识。这些资源通过超文本传输协议（Hypertext Transfer Protocol）传送给使用者，而后者通过点击链接来获得资源。

因特网（Internet）是当前全球最大的、开放的、由众多网络相互连接而成的计算机网络。万维网常常被视为因特网的同义词，其实万维网是依赖因特网运行的一项服务。万维网基于因特网，万维网被广泛应用于因特网之上。

延伸阅读：精神贵族——蒂姆·伯纳斯·李

蒂姆于 1955 年出生于英国伦敦，他是万维网的发明者。1989 年仲夏之夜，蒂姆成功开发出世界上第一个 Web 服务器和第一个 Web 客户机。并随后在 1990 年 12 月 25 日他成

功通过 Internet 实现了 HTTP 代理与服务器的第一次通信。2017 年，他因发明万维网、第一个浏览器和使万维网得以扩展的基本协议和算法而获得 2016 年度的图灵奖。他虽然放弃了万维网 http 的专利申请，但他成为了精神上最富有的人，是互联网的精神贵族，http 是他献给世界上每个人的互联网礼物。视频资料参阅央视纪录片《互联网时代》第一集（http://tv.cctv.com/ 2014/10/15/VIDA1413360557873609.shtml）。

1.2 Java 与 C、C++

随着程序规模的不断扩大，在 19 世纪 60 年代末期出现了软件危机，当时的程序设计范型都无法克服错误随着代码的增多而级数般地扩大的问题，这个时候就出现了一种新的程序设计范型——面向对象程序设计。

1.2.1 Java 和 C++

Sun 微系统公司的 Java 开发小组汲取了 C++的精华，并将其融入到 Java 中，同时舍弃了 C++的低效率和不便于程序设计人员使用的缺点。Java 小组也创造了一些新的特性，给予 Java 开发基于 Internet 的应用程序时所必需的动态性。

Java 的目的并不是改进 C++进而最终取代 C++，C++和 Java 这两种语言是用来解决不同问题的。Java 用来设计必须共存于不同机器的应用程序，即常常是基于 Internet 的基础之上。相反，C++用来开发在一台特定机器上运行的程序，尽管 C++程序被重新编译后能够在其他机器上运行。

Java 语言的许多基本结构与 C++是相似的，有时甚至是相同的。例如，Java 是一种面向对象编程语言，它用类来创建对象的实例，类具有数据成员和方法成员，这和 C++中的类是相似的。

但是 Java 没有指针，而在 C/C++编程语言中指针是基石。在 C++中正确使用指针能使程序富有效率，但是指针难以掌握，若使用不当则会导致运行错误。

Java 带有自动的垃圾自动回收机制，这是在 C/C++中没有的功能。垃圾自动回收机制是一个常规程序，它收集程序中不再使用的内存。这样，程序设计人员就不必编写代码来释放之前使用的内存。

在不同的平台上使用 C/C++程序使系统会对每种数据类型根据平台的不同进而分配不同的字节数。而在 Java 中，Java 会为各种数据类型分配合理的固定位数，且位数在每种平台上都不改变，这样便保证了 **Java 的平台无关性**。

C++中支持多重继承，一个类可以有多个父类，这种方式使 C++中的类可以使用多个父类的属性和方法，但其结构复杂，容易引起混乱。而在 Java 中，一个类只能有一个父类，但是可以实现多个接口，这样既达到了**多重继承的目的**，又保证了结构比多重继承更加清晰。

除此之外，与 C++不同，Java 中不支持结构和联合，不支持宏定义，不支持头文件，不支持友元，大大保证了 Java 程序的安全性。

1.2.2 Java 与 C

C 语言为面向过程的程序设计语言。面向过程程序设计语言在程序设计过程中都倾向于

面向行为。在 C 语言中，程序设计的单元是函数，C 编程人员着重于编写函数。执行同一个任务的一系列动作构成函数，一系列函数再构成程序。这种语言的主要问题是程序中的数据和操作分离，不能够有效地组成与自然界中的具体事物紧密对应的程序成分。

Java 是纯面向对象的程序设计语言，Java 语言中程序设计的单元是类，从类中创建实例对象。Java 编程人员着重创建用户自定义的类，每个类均可包含数据属性和若干操作数据的函数，一个类的函数部分称为方法。C 和 Java 编程与执行过程的区别如下。

Windows 下 C 语言开发过程如图 1.2 所示。C 语言程序在执行之前需要把程序编译成机器语言文件，程序执行效率高，依赖专门编译器，跨平台性稍差。

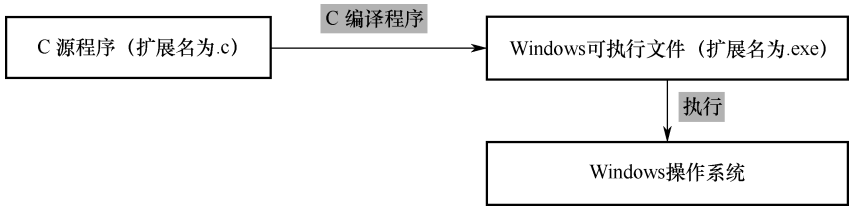

图 1.2　Windows 下 C 语言开发过程

Java 语言开发过程如图 1.3 所示。

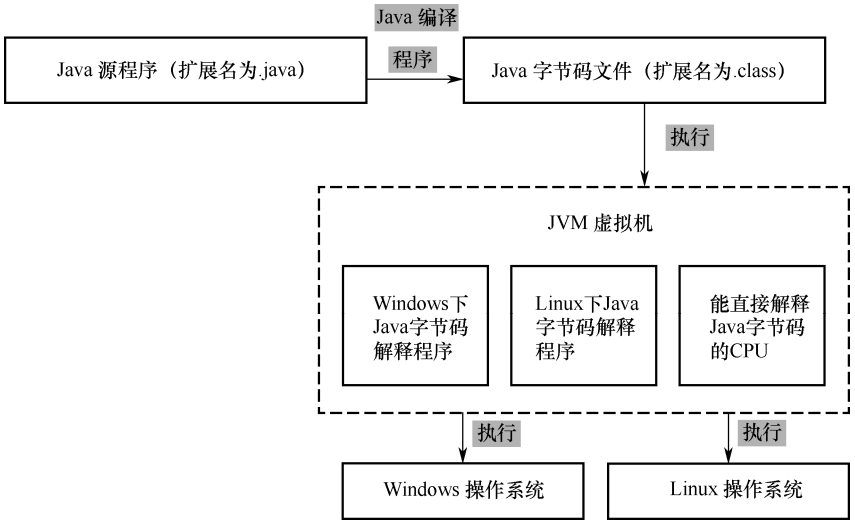

图 1.3　Java 语言开发过程

从图 1.2 和图 1.3 的比较中可以看出，Java 源程序编译后生成的字节码文件就相当于 C 源程序编译后 Windows 上的 exe 可执行文件，JVM（Java Virtual Machine）虚拟机的作用类似 Windows 操作系统。在 Windows 上运行的是 exe 文件，在 JVM 上运行的是 Java 字节码文件，即编译后生成的后缀为.class 的文件。

Windows 执行 exe 可执行文件的过程，就是从 exe 文件中取出一条条计算机指令交给 CPU 去解释、执行。字节码并不是机器指令，它与特定的平台无关，不能被任何平台直接识别、执行。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，将字节码翻译成所在平台的机器码，并让当前平台运行该机器码。可见，只要能实现特定平台下的解释器程序，Java 字节码就能通过解释器程序在该平台上运行，这是 Java

跨平台的根本特点。

Java 兼顾解释性与编译性语言的特点，Java 源文件转换成 class 字节码文件过程是编译型的，class 字节码文件在操作系统上运行的过程则是解释型的。Java 虚拟机充当了解释器的作用，C/C++都是编译型的语言，运行速度较快。

延伸阅读：程序设计语言发展脉络

计算机程序设计语言的发展是一个不断演化的过程，从最开始的机器语言到汇编语言再到各种结构化高级语言，最后发展到面向对象程序设计语言。

机器语言是第一代计算机语言，是最原始的编程语言，用二进制代码（0 或 1）书写，能被机器直接识别，二进制是计算机语言的基础。在计算机发展初期，软件工程师们只能用晦涩的机器语言来编写程序。**汇编语言**将一个特定指令的二进制串机器指令映射为简洁的英文助记符。例如，用“ADD”代表加法，用“MOV”代表数据传递等，它是比机器语言更“高级”的符号语言。**高级语言**是采用命令或语句的语言，屏蔽了机器的细节问题，提高了语言的抽象层次，如我们正在学习的 Java。

1.3 Java 语言特点及更新

Java 语言是一种彻底的面向对象的程序设计语言。作为一种纯粹的面向对象的程序设计语言，它非常适合大型软件的开发，同时它又简单易学。2005 年 6 月，Sun 公司发布 Java SE 6，此时，Java 的各种版本更名，J2EE 更名为 Java EE，J2SE 更名为 Java SE，J2ME 更名为 Java ME。2014 年 3 月，Oracle 公司发表 Java SE 8。Java 9 于 2017 年 9 月正式发布。Java 9 带来了很多的变化，其中最重要的改动是 Java 平台模块系统的引入。为了更快地迭代，以及跟进社区反馈，Java 的版本更新周期加快，Java 10 于 2018 年 3 月发布，最新特性大家可以参阅网上的最新资料。这里主要介绍 Java 语言核心特点。

1．面向对象

对象是程序的基本单元和构件。在面向对象的程序语言中，对象是类的实例，而类则是描述对象的模板。类是具有相同属性和服务的一组对象的抽象、一般描述。抽象是事物的泛化，抽象的目的是提取重要的特征而忽略不重要的细节。对象是现实世界中某个实际存在的事物，软件对象是数据和方法的封装体。类与对象的关系如同一个模具与用这个模具铸造出来的铸件之间的关系，如同自行车图纸与自行车的关系。

封装是面向对象的一个重要原则。它有两个含义，第一个含义是，把对象的全部属性和全部服务结合在一起，形成一个不可分割的独立单位（对象）；第二个含义也称为“信息隐蔽”，即尽可能隐蔽对象的内部细节，对外形成一个边界（或者形成一道屏障），只保留有限的对外接口使其与外部发生联系。这主要是指对象的外部不能直接地存取对象的属性，只能通过几个允许外部使用的服务（或称方法）与对象发生联系。

2．跨平台

这里所指的平台是由操作系统（OS）和处理器（CPU）所共同构成的平台。跨平台或与平台无关是指应用程序不因操作系统、处理器的变化而导致程序无法运行或出现运行错误。

用 Java 语言编写的程序，经过 Java 编译器编译后生成 Java 语言特有的字节码（Bytecode），

而不生成特定的 CPU 机器代码。字节码是一种中间码，它比机器码更抽象，因此跨平台性能更好。Java 字节代码运行在 Java 虚拟机（JVM，Java 语言解释器）上，Java 语言借助 Java 虚拟机，首先对 Java 编译后生成的字节码进行解释，虚拟机底层的运行系统把字节代码转化成实际的硬件调用，然后再执行它。

JVM是一种抽象机器，它附着在具体操作系统之上，本身具有一套虚拟机器指令，并有自己的栈、寄存器等。Java 虚拟机类似一个小巧而高效的 CPU，JVM 通常不在硬件上实现，它是通过软件仿真实现的，但是 Java 芯片的出现会使 Java 更容易嵌入到家用电器中。

JVM 是 Java 平台无关性的基础，Java 源代码先经过 Java 编译器生成 Java 虚拟机的字节码，再经过 Java 解释器将字节码转换成实际系统平台上的机器码，然后真正执行。任何一台机器只要配备了 Java 解释器，就可以运行字节码，而不管这种字节码是在何种平台上生成的。另外，Java 采用基于 IEEE 标准的数据类型。通过 JVM 保证数据类型的一致性，也就确保了 Java 的平台无关性。Java 产生的字节码与平台无关，这一点正是网络传输所需要的。

3．安全性

Java 将重点用于网络/分布式运算环境，确保建立无病毒且不会被侵入的系统。内存分配及布局由 Java 运行系统决定，字节码验证可以轻松构建防病毒、防黑客系统。

Java 最初设计的目的是应用于电子类消费产品，要求有较高的可靠性。Java 虽然源于 C++，但它消除了很多 C++的不可靠因素，可以防止很多编程出现错误。首先，Java 是强类型语言，要求显式的方法声明，保证了编译器可以发现方法调用错误，保证程序更加可靠；其次，Java 不支持指针，杜绝了内存的非法访问；再次，Java 解释器在运行过程中实时检查，可以发现数组和字符串访问的越界；最后，Java 提供了异常处理机制，便于程序及时发现运行错误。

由于 Java 主要用于网络应用程序开发，因此对安全性有着较高的要求。如果没有安全保证，那么用户从网络下载程序执行就会非常危险。

4．多线程

线程是操作系统的一种概念，被称为轻量级进程，是比传统进程更小的、可并发执行的单位。C 和 C++采用单线程体系结构，而 Java 提供了多线程支持。

一个线程是一个程序内部的顺序控制流。在 DOS 环境下，我们只能同时运行一个程序，也就是程序只有一条顺序控制流。即一部分程序因为某种原因不能执行下去的时候，整个程序就停止在那里，其他的操作就不能执行。进程的特点是每个进程都有独立的代码和数据空间，进程切换的开销大。线程是轻量的进程，同一类线程共享代码和数据空间，线程切换的开销小。通常，线程之间的切换是非常迅速的，使人们觉得好像所有的线程都在同时执行。但是在系统内部来看，线程仍是串行执行的，只不过由于操作系统可以快速、自动地进行切换，从而给人一种并发执行的感觉。

1.4 Java 程序的类型及其不同的编程模式

用 Java 书写的程序有两种类型：Java 应用程序（Java Application）和 Java 小应用程序

（Java Applet）。

Application 的基本编程模式：

```
class 用户自定义的类名  // 定义类
{
   public static void main(String args[ ] )   //定义 main( )方法
    {
       方法体
     }
}
```

Applet 的基本编程模式：

```
import java.awt.Graphics;  //引入 java.awt 系统包中的 Graphics 类
import java.applet.Applet;  //引入 java.applet 系统包中的 Applet 类
class  用户自定义的类名  extends Applet  //定义类
{
  public void paint(Graphics g)  //调用 Applet 类的 paint( ) 方法
   {
      方法体
    }
}
```

Applet 需要的 HTML 文件的最小集的格式：

```
<html>
<applet code=类名.class   width= 宽度   height=高度>
</html>
```

注意：HTML 标记包含在尖括号内，并且总是成对出现的，前面加斜杠表明标记结束。用<html>和</html>标记 HTML 文件的开始和结束，用<applet>和</applet>标记 Applet 的开始和结束。必须把以.class 结尾的字节码文件名嵌入到 HTML 文件中，这里的 HTML 文件应和字节码文件放在同一级目录下。另外，HTML 对字符大小写是不敏感的，参数值可加引号也可不加。

Java 应用程序必须得到 Java 虚拟机的支持才能够运行。Java 小应用程序则需要客户端浏览器的支持。Java 小应用程序运行之前必须先将其嵌入 HTML 文件的<applet>和</applet>标记中。当用户浏览该 HTML 页面时，Java 小应用程序将从服务器端下载到客户端，进而被执行。

综上所述，Applet 和 Application 是 Java 程序的两种基本类型，从源代码的角度来看，Applet 和 Application 有两个基本的不同点：

（1）一个 Applet 类必须定义一个从 Applet 类派生的类，Application 则没有这个必要。

（2）一个 Application 必须定义一个包含 main 的方法，以控制它的执行，即程序的入口。而 Applet 不会用到 main 方法，它的执行是由 Applet 类中的几个系统方法来控制的。

两者共同之处是：编程语法是完全一样的。

1.5 Java 开发工具入门

1.5.1 JDK 的下载、安装

作为初学者，学习 Java 最好直接选用 Java SE 提供的 JDK，各种集成开发环境一般系统界面相对复杂，还需要做相关配置，而且会屏蔽掉一些知识点。待 Java 编译、运行等命令已经熟练之后，再去尝试使用流行的 Java 集成开发环境。

提示：Eclipse 是一个开放源代码的，基于 Java 的可扩展开发平台。Eclipse 附带一个标准的插件集，包括 Java 开发工具（Java Development Kit，JDK）。使用或安装 Eclipse 前确保电脑已安装 JDK。Eclipse 下载地址：https://www.eclipse.org/downloads/，可以使用下载解压缩免安装，也可以采用安装版本。学习指导推荐“菜鸟教程”（http://www.runoob.com/eclipse/eclipse-tutorial.html）。

Java 开发工具使用 JDK，目前应用较多的版本是 JDK 8.0。JDK 软件包提供了 Java 编译器和 Java 解释器，但没有提供 Java 编辑器，初学者推荐使用 Windows 的“记事本”或其他高级编辑器，如 Notepad++、Notepad++是在微软视窗环境下的一个免费的代码编辑器，下载地址：http://notepad-plus-plus.org/。

JDK 是原 Sun 公司免费提供的，目前 Sun 公司被 Oracle 公司收购。最新 JDK 下载地址：http://www.oracle.com/technetwork/java/javase/downloads/index.html，如图 1.4 所示，点击按钮 Java。

图 1.4 JDK 下载页面

在下载页面中需要选择接受许可，并根据自己的电脑操作系统选择对应的版本，如图 1.5 所示。

下载 JDK 后采用默认安装即可，同时一并安装 JRE。一般默认电脑安装目录，如 C:\Program Files\Java\jdk1.8.0_65，版本号不同，jdk 后面数字号不同。

在安装 JDK 过程中，可以自定义安装目录等信息，一般情况下选择默认安装目录即可。默认安装完毕后，JDK 目录结构如图 1.6 所示，此处安装的是 JDK 8.0。

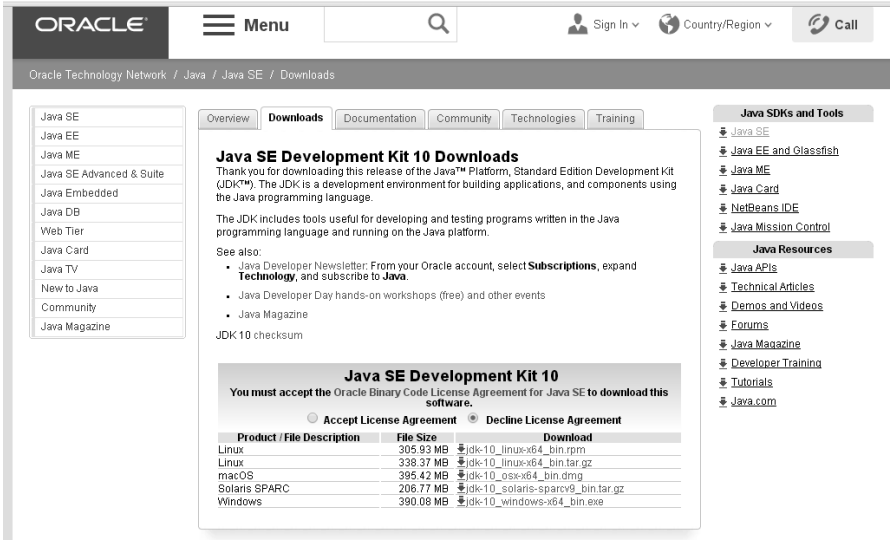

图 1.5　JDK 下载页面

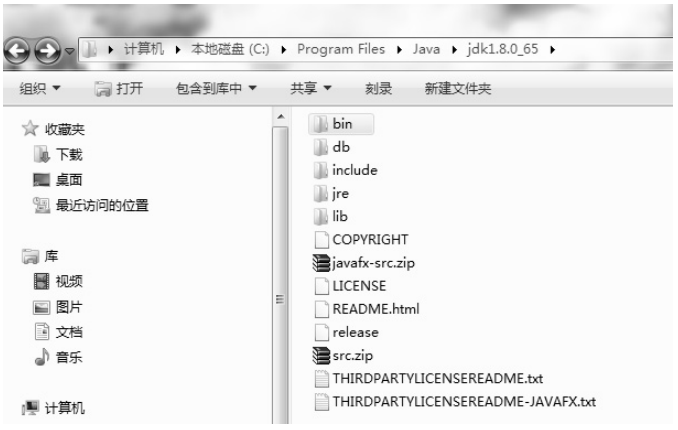

图 1.6　JDK 目录结构

1.5.2　配置环境变量

在计算机系统中可以定义一系列变量，这些变量可供操作系统中所有的应用程序使用，被称为系统环境变量。在学习 Java 的过程中，在 Windows 平台下，我们经常设置的环境变量是 path 和 classpath，path 和 classpath 分别指定了 JDK 命令搜索路径和 Java 类路径。

path 环境变量用于保存一系列路径，为操作系统提供所使用的应用程序搜索路径，也就是说，当操作系统在当前目录下没有找到想用的命令工具时，操作系统就会按照 path 环境变量指定的目录依次去查找，以最先找到的为准。path 环境变量可以存放多个路径，Windows 下路径和路径之间用英文分号“;”隔开。为保险起见，实验时一般将 jdk 路径放在最前面。

设置环境变量 classpath 的作用是告诉 Java 类装载器到哪里去寻找第三方提供的类和用户定义的共享类。在 classpath 环境变量中添加的“. ;”英文实心点代表 Java 虚拟机运行时的当前工作目录，英文分号进行路径之间的分隔。从 JDK 5.0 开始，classpath 环境变量不用

设置，Java 虚拟机会自动将其设置为当前目录“.”。

问题提示：安装 JDK 一般不需要设置环境变量 classpath 的值。当读者的计算机环境比较复杂时，即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，在安装这些产品后，这些产品所带的旧版本的类库，可能导致程序无法运行。在出现这种情况时，需要编辑 classpath 的值，增加 JDK 文件中 jre 文件夹中的 lib 文件夹中的 rt.jar 文件。

当采用 Windows 操作系统时，右键单击桌面上的“我的电脑”或“计算机”图标，然后选择菜单中的“属性”，这里以 Win7 系统为例，在出现的属性面板中选择“高级系统设置”→“高级”，如图 1.7 所示。然后单击“环境变量”按钮，打开环境变量面板，在这里可以看到上下两个窗口，上面窗口为“某用户的用户变量”，下面窗口为“系统变量”，如图 1.8 所示。

图 1.7　属性面板

图 1.8　环境变量面板

读者可以在图 1.8 中上下任意一个窗口进行设置，区别在于上面的窗口设置用于个人环境变量，只有以该用户身份登录时才有效，而下面窗口中的设置则对所有用户都有效。以设置系统变量为例，单击名为“path”的变量（若没有环境变量 path 选项，则在“用户变量”或“系统变量”中选择“新建”来添加），选择“编辑”，如图 1.9 所示，在打开的“编辑系统变量”窗口中的“变量值”输入框中添加新安装的 JDK 路径，应当在 path 原有值的末尾加上英文分号“;”，然后加上 Java 编译器和解释器等工具所在 bin 的路径（这里是 C:\Program Files\Java\jdk1.8.0_65\bin），然后单击“确定”。一般建议 JDK 路径放在变量值的最前面，这样检测效率高且防止其他应用自带老版本 JDK 的干扰。

图 1.9　编辑系统变量窗口

path 环境变量设置好后，在 DOS 下使用 cd 进入工作目录，程序就可以编译运行了。若偶尔有问题，则可以进一步在 DOS 下使用设置语句 set classpath=D:\mycode（以 mycode 为自己源代码存放的文件夹为例），再运行自己的源文件就可以了。

若用户在安装 jdk 时，选择了另外的 JDK 安装路径，则环境变量 path 和 classpath 要做相应的调整。

path 环境变量设置好后，需要重新打开 DOS 命令提示符界面，path 环境变量才起作用。环境变量设置完成后，可以在 DOS 窗口下进行测试。输入 javac 并按回车后，若出现 javac 的用法参数提示信息，则安装正确，如图 1.10 所示；也可以在 DOS 下使用 set 命令查看添加的系统路径，若没有找到系统路径，则要检查环境变量设置是否正确。

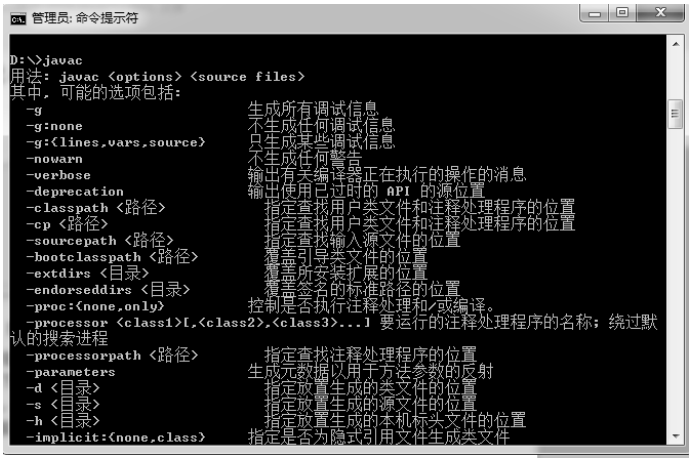

图 1.10　JDK 测试

问题提示：在编译器 javac 运行正常，而解释器 java 不能正常运行时，且提示的异常为“Exception in thread "main" java.lang.NoClassDefFoundError: Welcome”，其中 Welcome 是程序的主类名称，此时，请在“我的电脑”中打开 classpath，把英文实心点和分号“.;”添加到其变量值中。

1.5.3　JDK 开发工具简介

JDK 工具是以命令行方式应用的，即在 Windows 操作系统的 DOS 命令行提示符窗口中执行 JDK 命令。在 JDK 的 bin 目录下，存放着 Java 提供的一些可执行程序，为我们开发和测试 Java 程序提供了工具。在学习中，常用的 JDK 开发工具有以下 3 种：

- javac.exe：Java 语言的编译器，生成.class 字节码文件。
- java.exe：Java 程序执行引擎，Java 解释器，执行字节码文件。
- appletviewer.exe：JDK 自带的小应用程序浏览器。

1.6　Java 程序开发过程

1．开发过程简介

要编写和运行第一个 Java 程序，需要有文本编辑器和 Java 开发平台。编辑器可以使用 Windows 自带的记事本或其他高级文本编辑器，使用 JDK 作为开发工具和平台。Java 不断在更新，最新更新版本为 Java 10，本书的 JDK 采用 JDK 8.0。通常，初学者使用 Windows 环境中的记事本作为创建源文件的文本编辑器。

Java 源程序的开发步骤如图 1.11 所示，经过编写、编译和运行过程后，JVM 运行的是

Java 字节码，操作系统可以是不同的操作系统。

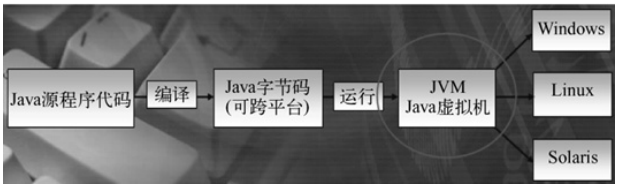

图 1.11　Java 源程序的开发步骤

要创建一个 Java 程序需要以下 3 个基本步骤：

（1）创建带有文件扩展名.java 的源文件。

（2）利用 JDK 自带的 Java 编译器生成文件扩展名为.class 的字节码文件。

（3）Application 程序利用 Java 解释器运行该字节码文件，Applet 利用 Java 自带浏览器运行嵌有字节码文件的 HTML 文件。

注意：在多个类的情况下，保存文件时一般要使用 public 修饰的类名作为文件名，刚开始学习书写简单单个类，往往使用默认修饰符的类，一般也使用主类的类名作为文件名，这样更方便，两者的后缀都为.java。因为记事本默认的扩展名是.txt，所以必须修改文件扩展名为.java，可在文件名的开始和扩展名的结尾处加上一对双引号后保存，或者不加双引号，保存类型选择 all files。

Java 编译器是 JDK 中的 javac.exe，将 Java 源程序编译成字节码文件，使用语法如下：

javac　类名.java

然后按回车键。若源程序没有错误，则屏幕上没有输出；否则将显示出错信息。

Java 解释器是 JDK 中的 java.exe，解释和执行 Java 应用程序，使用语法如下：

java　类名

然后按回车键。启动虚拟机，执行该类的 main 方法。主方法定义受 JVM 限制，有严格的格式要求。即

注意：这里不能带.class 后缀。在字节码文件编译生成后，将自动保存在与源程序同一级的目录下。

Java 的平台无关性就是因为每种计算机上都安装了一个合适的解释器，将不同计算机上的系统差别隐藏起来，使字节码面对一个相同的运行环境，实现了“编写一次，到处运行”的目标。

注意：随着 JDK 版本的升级，编译器 javac 后面的类名可以不是主类的名称，但是解释执行的时候必须是主类.class，因为文件名可以是任意类名，但是生成的 class 文件，解释从主类开始。

对于 Applet 程序来说，需要 HTML 文件的配合，使用语法如下：

appletviewer　HTML 文件名.html

然后按回车键。字节码文件嵌入 HTML 文件中，appletviewer 为 Applet 查看器（JDK 中的

appletviewer.exe），含有内置 Java 解释器。appletviewer 又称小浏览器，它仅显示相关 Applet 的属性，初学者使用方便。

2．创建 Java Application 程序示例

编写一个 Java Application 程序，过程简要描述如下：

首先，用户需要下载和安装 JDK。这里以 JDK 8.0 版本为例，暂且把程序源文件放置在 d 盘的自建 mycode 文件夹中，注意：Java 的 path 环境变量已经设置好。

其次，确定文本编辑器。在本例中，使用记事本，以 Windows 7 为例，从“开始”菜单项中选择“程序”→“附件”→“记事本”。当然，用户也可以选择其他文本编辑器。

【例 1.1】 实现第一个简单的应用程序：打印一行文字。

（1）在“记事本”中编写源程序

```
// 文件名: Welcome.java
public class Welcome {
    public static void main( String args[] )
    {
       System.out.println( "Welcome to Java Programming!" );
    } //结束 main 方法的定义
} //结束类 Welcome 的定义
```

（2）语法说明

程序中的“//”为单行注释符，只对当前行有效，表示该行是注释行。程序设计人员在程序中加入注释，用于提高程序的可读性，使程序便于阅读和理解。在程序执行时，注释行会被 Java 编译器忽略。多行注释以“/*”开始，以“*/”结束。

Java 程序由类或类的定义组成，类构成了 Java 程序的基本单元，创建一个类是 Java 程序的首要工作。Java 用关键字 class 标志一个类定义的开始，class 前面的 public 关键字代表该类的访问属性是公共的，表示这个类在所有场合中都可使用。一个程序文件中可以声明多个类，但仅允许有一个公共的类。class 后面是该类的类名，在本例中是 Welcome。

Application 中有一个显著标记就是必须定义一个 main( )主方法，而且严格按照例 1.1 中所示的格式定义其修饰符和命令行参数，用关键字说明 main 方法是 public，静态的 static，无返回值的 void，主方法的参数是字符串类型 String 的数组 args[]。一个类中可以声明多种方法，Java 应用程序自动从 main 主方法开始执行，通过主方法再调用其他方法。Java 语言的每条语句都必须以英文分号结束。

System.out 是标准输出对象，它用于在 Java 应用程序执行的过程中向命令窗口显示字符串和其他类型的信息。方法 System.out.println 在命令窗口中显示一行文字后，会自动将光标位置移到下一行（与在文本编辑器中按回车键类似）。

（3）编译运行程序

在源程序编写并保存好后，接着我们准备执行该程序。为此我们打开一个命令提示符窗口，用 cd\退回到根目录，如图 1.12 所示。

接着，通过 DOS 的 cd 命令，进入自己创建的目录 d:\mycode（读者可以使用自己创建的其他盘符的文件夹），如图 1.13 所示。

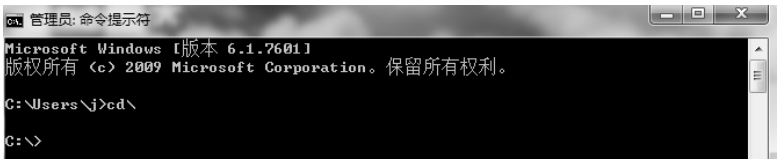

图 1.12　命令提示符窗口

图 1.13　显示 Welcome.java 文件

再在“命令提示符”窗口中输入 javac Welcome .java，按回车键，如图 1.14 所示。

图 1.14　编译 Welcome.java 文件

若此程序没有显示语法错误提示，则将生成一个 Welcome.class 字节码文件，自动保存在源文件同级目录下。

再在“命令提示符”中输入 dir 命令，就可以看到编译后生成的.class 文件，表明程序编译成功，如图 1.15 所示。

图 1.15　显示编译后的情况

运行字节码文件输入，java Welcome，按回车键，如图 1.16 所示。

此命令启动 Java 解释器，运行“.class”文件，字节代码被 Java 解释器解释执行。注意：解释命令不带.class 文件扩展名，否则解释器不能解释执行。解释器自动调用方法 main，然后通过 System.out.println 方法显示“Welcome to Java Programming!”。

图 1.16　运行程序并显示运行结果

3．创建 Java Applet 程序示例

Applet 也是一种 Java 程序，它一般运行在支持 Java 的 Web 浏览器内，有完整的 Java API 支持。Applet 是一种嵌入 HTML 文件中的 Java 程序，可以通过网络下载来运行。HTML 是超文本标记语言，它采用一整套标记来定义 Web 页。HTML 文件的扩展名为.html 或.htm。与从命令窗口执行 Java 应用程序不同，Applet 通过 JDK 的查看器 appletviewer 或支持 Java 的 Web 浏览器运行。

【例 1.2】　显示一行字符串的简单 Java Applet。

（1）在记事本中编写源代码

```
// 文件名: WelcomeApplet.java
// A first applet in Java
import javax.swing.JApplet;  // 加载系统类 JApplet
import java.awt.Graphics;    // 加载系统类 Graphics

public class WelcomeApplet extends JApplet {
   public void paint( Graphics g )
   {
      g.drawString( "Welcome to Java Programming!", 25, 25 );
   }  //结束 paint 方法的定义
}   //结束类 WelcomeApplet 的定义
```

（2）语法说明

Java 含有许多预定义的类或数据类型，这些类被归入 Java API（Java 应用程序编程接口，Java 类库）的各个包中。程序中使用 import 语句引入系统预定义类。程序中的两行加载语句告诉编译器 JApplet 类的位置在 javax.swing 包中，Graphics 类的位置在 java.awt 包中。当创建一个 Applet 小应用程序时，要加载 JApplet 类或 Applet 类。加载 Graphics 类是为了使程序能够画图（如线、矩形、椭圆和字符串等）。

注意：java.applet 中有一个传统的 Applet 类，它没有包括在 Java 最新的 GUI 构件 javax.swing 包中。

Java API 中的所有包存放在 java 目录中或 javax 目录中，这两个目录下还有许多子目录，包括 awt 目录和 swing 目录。注意：在磁盘上找不到这些目录，因为它们都存储在一个称为 JAR 的特殊的压缩文件中。在 JDK 安装结构中有一个名为 rt.jar 的文件，该文件包括了 Java API 里所有.class 文件。

与应用程序一样，每个 Java Applet 至少由一个类定义组成，但是用户几乎不必“从头开始”定义一个类。这是因为 Java 提供继承机制，使用户可以在已存在的类的基础上创建一个新类。程序中通过关键字 extends 实现，extends 前面为用户自定义的类，作为派生类或子类，extends 后面的类名为被继承的类，称为基类或父类或超类，如同此程序中的系统类 JApplet。通过继承建立的新类具有其父类的属性（数据）和行为（方法），同时增加了新功能（如在屏幕上显示“Welcome to Java Programming!”的能力）。

实际上，一个 Applet 需要定义 200 多种不同的方法，而在上面的程序中，我们只定义了一种 paint 方法。如果非要定义 200 多种方法，仅仅为了显示一句话，那么我们可能永远无法完成一个 Applet。使用 extends 继承 JApplet 类，这样 JApplet 的所有方法就已经成为 WelcomeApplet 的一部分。使用继承机制，程序设计人员不必知道所继承基类的每个细节，只需知道 JApplet 类具有创建一个 Applet 的能力即可。

注意：学习 Java 语言，一方面是学习用 Java 语言编写自己所需的类和方法；另一方面是学习如何利用 Java 类库中的类和方法。这样，有助于确保不会重复定义已提供的功能。

程序中重写了父类 JApplet 的 paint( )方法，其中参数 g 为 Graphics 类的对象。在 paint( )方法中，通过用 Graphics 对象 g 后的点操作符“.”和方法名 drawString 来调用 drawString( )方法，在坐标(25,25)窗口处输出字符串，其中坐标以像素点为单位，第一个坐标为 *x* 坐标，它表示距离 Applet 框架左边界的像素个数；第二个坐标为 *y* 坐标，它表示距离 Applet 框架上边界的像素个数。Applet 没有 main( )方法，这是 Applet 与 Application 的一个显著区别。JApplet 类的方法 paint 在默认情况下，不做任何事情。Welcome- Applet 类覆盖了或重新定义了这个行为，以使 appletviewer 或浏览器调用 paint 方法，在屏幕上显示一行字符串。

（3）用记事本编写与例 1.2 Java 源文件配合的 HTML 文件

```
<html>
<applet code="WelcomeApplet.class" width=400 height=50>
</applet>
</html>
```

HTML 标记是用尖括号括起来且成对出现的，加斜杠表明标记结束，用<html>和 </html>标记 HTML 文件的开始和结束，用<applet>和</applet>标记 Applet 的开始和结束。<applet>包含以下 3 个必需的参数。

- code：表示要打开的 Applet 字节码文件名。
- width：表示 Applet 所占用浏览器页面的宽度，以像素点为单位。
- height：表示 Applet 所占用浏览器页面的高度，以像素点为单位。

一般情况下，字节码文件和 HTML 文件处于同一个目录下；否则字节码文件的路径要在 code 中给出。

（4）编译运行

① 编译 Java 源文件，与例 1.1 一样，使用 javac 命令：javac WelcomeApplet.java，如图 1.17 所示。

② 运行时使用命令格式：appletviewer WelcomeApplet.html，如图 1.17 所示。appletviewer 是 JDK 工具，位于 JDK 安装路径/bin 中，作为 Java Applet 浏览器的 appletviewer 命令可在

脱离万维网浏览器环境的情况下运行 Applet。

```
D:\mycode>javac WelcomeApplet.java

D:\mycode>appletviewer WelcomeApplet.html
```

图 1.17　Java Applet 的操作步骤

③ Applet 运行结果如图 1.18 所示。这里需要注意，若运行环境是 Windows XP，则一切正常；若运行环境是 Windows 7 的 64 位操作系统，则会有提示“无法读取 appletviewer 属性文件”,但是运行结果可以出现,并且使用默认值。解决方法为当前目录运行“policytool.exe”“添加策略条目”再进一步“添加权限”等的设置，如图 1.19 所示。这里不详细展开，百度中有详细参考资料。

图 1.18　Applet 运行结果

图 1.19　添加权限

注意：Java 的跨平台不是没有任何条件的。Application 的运行是以各个平台上的虚拟机为前提条件的。Applet 也是如此。当需要用 Java 的 Swing 编写 Applet 时，Web 浏览器中需要安装支持它的插件。所以，为了在学习时调试 Applet 方便，本书示例均以 JDK 提供的 appletviewer 工具本地显示嵌有 Applet 的 HTML 文件。

4．良好的编程习惯

（1）所有的 Java 语句必须以英文分号“;”结束。

（2）Java 区分大小写，要注意关键字和标识符字母的大小写。

（3）花括号成对出现。在写左花括号时，要立即再写一个右花括号，这样有助于防止漏写右花括号。类名称后面的花括号表示类定义的开始和结束。

（4）习惯上，类名应以大写字母开头，变量以小写字母开头，变量名由多个单词组成，第一个单词后边的每个单词首字母应大写。当读一个 Java 程序时，寻找以大写字母开头的标识符，这些通常代表 Java 类。

（5）程序段中适当增加空白行会增加程序的可读性。在定义方法内容的花括号中，将整个内容部分缩进一层，使程序结构清晰，程序内容易读。编译器会忽略这些空白行和空格字符。

（6）在程序中，一行最好只编写一条语句。Java 允许一个长句分割写在几行中，但是不允许从标识符或字符串的中间分割。

（7）文件名与 public 类名在拼写及大小写上必须保持一致。

（8）若一个.java 文件含有多于一个 public 类，则这是错误的。

（9）不以.java 为扩展名的文件名是错误的。

（10）当运行 appletviewer 时，文件扩展名不是.htm 或.html 是错误的，这将导致无法使 appletviewer 装载 Applet。

1.7 实训

第 1 章　Java 概述-售报亭

阅读下列 Java 源文件，并回答问题。

```
public class Hello{
   void speakHello( ){
      System.out.println("I’m glad to meet you");
   }
}
Class HelloTest{
   Public static void main(String args[ ]){
        Hello he=new Hello( );
        he.speakHello( );
   }
}
```

（1）上述源文件的名称应该是什么？

（2）上述源文件编译后生成几个字节码文件？这些字节码文件的名称都是什么？

（3）在命令行执行 java Hello 会得到怎样的错误提示？执行 java HelloTest.class 会得到怎样的错误提示？执行 java HelloTest 会得到什么输出结果？

提示：文件名的命名一定是公共 public 类的名称，但是解释执行的时候只能是主类的类名。

习题 1

1．描述 Application 的执行流程。

2．描述 Applet 的执行流程。

3．请查阅资料，了解有关 SCJP 认证的情况。

4．编写运行一个简单的 Java Application，利用 JDK 软件包中的工具编译并运行这个程序，在屏幕上输出“Hello,world!”。

5．编写一个 Java Application 程序，分行显示字符串“Welcome to Java Programming!”中的 4 个单词。

6．编写一个简单的 Applet 程序，使其能够在浏览器中同样显示字符串“Hello,world!”。

问题探究 1

1．讨论目的：通过探究学习，使学生熟悉 Java 语言的产生背景；理解 Java 与 C、C++的关系；学习 Java 语言的特点及 Java 语言的行业应用等问题；使学生在获取资料、合作研讨、沟通技能等方面得到锻炼。

2．讨论准备：根据讨论目的和讨论内容的要求，实行学生小组长负责制，成员分工明确，收集相关材料，并进行研讨、分析梳理相关信息，最后制作 PPT 演示文稿等。

3．讨论内容：

（1）Java 产生和发展的深层原因是什么？探索并分析 Internet、Web 与 Java 渊源，进一步了解互联网精神贵族蒂姆·伯纳斯·李的故事。

（2）Java 与 C、C++的关系是什么？探索并分析程序设计语言的层次发展。

（3）Java 程序封装的意义是什么？如何理解 Java 的跨平台特点，分析 Java 的安全性和多线程特点。

（4）了解工业界对 Java 语言的评价，探索并分析 Java 目前的行业应用现状和 Java 对软件开发技术的影响。