前言

PREFACE

Java语言是当前应用非常广泛的一种面向对象的程序设计语言，从大型复杂的企业级系统到小型移动设备系统开发，随处都可以看到Java活跃的身影。当前，大部分高校计算机相关专业都开设了Java程序设计相关课程。

Java程序设计从基础到高级需要一个长期学习的过程，对于初学者来说，需要打好基础，树立信心、循序渐进，同时注重编程习惯的培养。本书针对应用型人才培养，强调实践动手能力，从需求出发，以案例驱动的形式进行组织。书中的案例都是精心设计的，只有读者实际动手敲过代码，调试过代码后才能熟能生巧，知其所用，才能真正掌握这些代码，感受到编写Java程序的乐趣。

全书共分为13个项目。

项目1主要介绍了Java语言的特点、JDK的安装及Eclipse开发环境，实现第一个Java程序的开发。

项目2介绍Java语言的语法基础，如果读者有C语言基础，学习难度不大，但是对JVM内存划分需要仔细体会。

项目3和项目4是Java语言面向对象思想的最重要部分，详细讲解了类和对象的关系、封装、继承和多态等相关内容，只有理解了面向对象的编程思想，才能真正掌握Java语言的精髓。同时讲解了单例模式、简单工厂模式的实现。

项目5介绍了Java对异常的处理。

项目6详细讲解了Java对多线程技术的支持。其中生产者和消费者模型是难点，通过它可以深刻理解线程间通信。

项目7讲解了包装类、字符串相关类和System类。其中字符串相关类非常重要，读者要尽量掌握相关方法，并理解正则表达式。

项目8讲解了时间处理、随机数和Math类的用法，难度不大。

项目9讲解了集合类，要重点掌握各种集合的创建，增、删、改、查，以及遍历的方法。

项目10讲解File类和输入输出流，编写程序免不了要和文件打交道，涉及的类较多，同

时讲解了装饰模式的实现。

项目 11 讲解 JDBC 编程，安装了 MySQL，详细讲解了 JDBC 开发过程。

项目 12 讲解反射的相关知识。反射是 Java 的高级特性。掌握反射的本质和应用，有助于将来的框架学习。

项目 13 给出了一个简单的 Java Web 程序开发示例，帮助读者理解 Java 在服务器端的开发应用。

本书由文华学院林爱武、南通理工学院宋伟、哈尔滨远东理工学院齐晶薇担任主编；由文华学院张采芳、张翼、黄金刚、仇亚萍、田笛担任副主编；南通理工学院孙溢洋、崔庆华参编。全书由文华学院林爱武审核并统稿。

为了方便教学，本书还配有电子课件等资料，任课老师可以发邮件至 hustpeiit@163.com 索取。

由于编者水平有限，加之时间比较仓促，书中难免有疏漏和不妥之处，恳请广大读者朋友批评指正。

编者

2021 年 5 月

目录

CONTENTS

项目1 Java语言简介及开发环境

1.1 Java 语言简介

◆ 1.1.1 Java 语言特点

Java 语言是 Sun Microsystems 公司(2009 年被 Oracle 公司收购)于 1995 年 5 月推出的一种可以编写跨平台应用软件、完全面向对象的程序设计语言。时任 Sun 公司副总裁的詹姆斯·高斯林 (James Gosling)是 Java 语言的主要设计师,被公认为"Java 之父"。在全球云计算和移动互联网的产业环境下,Java 具备了显著优势和广阔前景,成为当前最流行的编程语言之一。

Java 语言具有很多的优点,以下罗列出其中几个主要的特点。

1) 完全面向对象

Java 语言作为一门面向对象的程序设计语言也继承了面向对象的诸多好处,例如代码扩展、代码复用等。面向对象的编程使得程序间的耦合度更低,内聚性更强。

2) 简单性

Java 语言的语法比较简单,风格类似于 C++,但是比 C++简单。例如,Java 丢弃了 C++中运算符重载、多重继承等模糊难懂的概念。Java 增加了引用类型来代替指针,同时提供了自动垃圾回收机制管理内存,使程序员不用像 C++那样操心内存管理。

3) 安全性

Java 主要用于网络应用程序的开发,Java 通过自己的安全机制来有效防止病毒程序的产生和下载程序对本地系统的威胁破坏。例如,Java 程序在运行前会对字节码进行安全检查,确保程序不存在非法访问本地资源、文件系统的可能,保证了程序在网络间传送的安

全性。

4）跨平台运行

Java 语言最大的优势在于与平台无关性，也就是可以跨平台使用。绝大多数的编程语言都是不可以跨平台使用的。所谓的平台，我们可以理解为操作系统。比如，C 语言在 Windows 系统下编译的 *.exe 文件在其他系统下是无法运行的。在不同的操作系统下可运行文件是不同的，所以同样功能的软件我们需要编写出多份适用于不同平台上的代码，造成重复开发，严重影响了开发效率。

但是 Java 语言不同，因为 Java 程序不是直接运行在操作系统上，而是在 JVM 中运行。JVM 是 Java virtual machine(Java 虚拟机)的缩写，它是虚构出来的计算机，是通过模仿实际计算机的各种功能实现的。也就是说，对于实际计算机中的某些功能，JVM 也可以实现。JVM 是 Java 跨平台使用的根本。

所以 Java 的编译程序只需要在 JVM 中生成目标代码(字节码)文件，就可以在不同的平台上直接运行了(不用修改)，当然我们的操作系统中必须要有适合该系统的 Java 虚拟机。JVM 在执行字节码时，会把字节码解释为具体平台的机器指令，这也说明了 Java 既是编译型语言(编译为字节码)也是解释型语言。

Java 这种“一次编写，到处运行”(write once，run anywhere)的特性大大降低了程序开发和维护的成本。再次强调，不同平台的 JVM 有不同的实现，例如，Windows 平台有 JVM for Windows，而 Linux 平台有 JVM for Linux，对于开发者而言，不需要关心使用的平台是什么。

Windows、Linux、macOS 操作系统的 Java 虚拟机效果如图 1-1 所示。

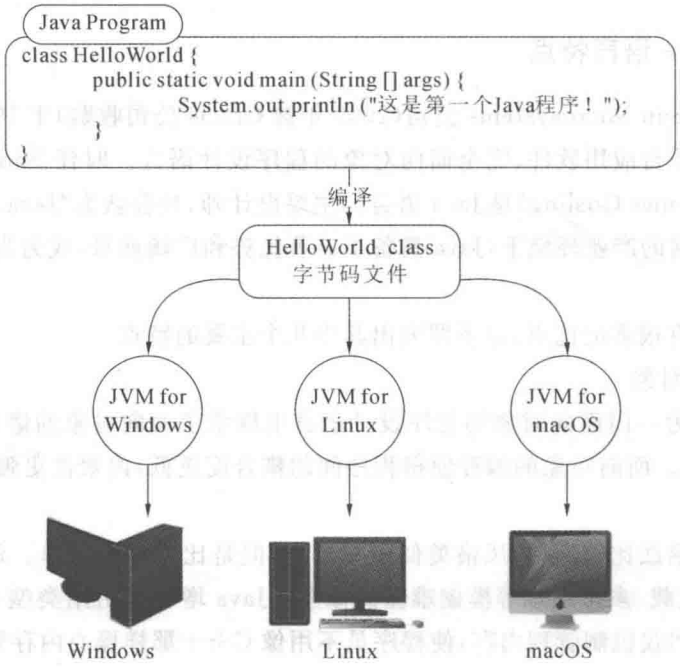

图 1-1　不同平台的 Java 虚拟机

5）支持多线程

多线程是现代程序设计中必不可少的一种特性，多线程处理能力使得程序具有更好的

交互性、实时性。Java 支持多线程开发，控制线程的运行，并且使用同步机制保证对共享数据的正确操作。因此，Java 对多线程的支持使其能够成为服务器端的开发语言之一。Java 开发人员可以方便地写出多线程的应用程序，从而提高程序的执行效率，实现网络上的实时交互行为。

1.1.2 Java 技术平台

Oracle 公司根据应用领域的不同将 Java 技术划归为三个平台，即 Java SE、Java EE 和 Java ME。

1) Java SE

Java SE(Java platform, standard edition)是 Java 平台标准版的简称，用于开发普通桌面应用程序。Oracle 公司为 Java SE 提供了一套开发环境 JDK(Java development kit)，包括了 Java 语言最核心的类库，例如集合、IO、JDBC、网络编程等。Java EE 和 Java ME 都是从 Java SE 的基础上发展起来的，因此，无论将来想从事哪个方面的程序开发，都应该从 Java SE 开始学习。

2) Java EE

Java EE(Java platform, enterprise edition)是 Java 平台企业版的简称，能够帮助程序员开发和部署可移植、健壮、可伸缩且安全的服务器端 Java 应用程序。Oracle 公司为 Java EE 提供了开发包 SDK(software development kit，包含相关 JDK)。Java EE 建立于 Java SE 之上，具有 Web 服务、组件模型以及通信 API 等特性，可以用来实现企业级的面向服务体系结构(SOA)和 Web 2.0 应用程序。

3) Java ME

Java ME(Java platform, micro edition)是 Java 微型版的简称。Java ME 为在移动设备和嵌入式设备(比如手机、PDA、电视机顶盒和打印机)上运行的应用程序提供一个健壮且灵活的环境。Java ME 包括灵活的用户界面、健壮的安全模型、许多内置的网络协议以及对可以动态下载的联网和离线应用程序的丰富支持。基于 Java ME 规范的应用程序只需编写一次，就可以用于许多设备，而且可以利用每个设备的本机功能。

1.2 JDK 的使用

1.2.1 JDK 的安装

本书通过 JDK 7.0(内部版本号为 1.7)来学习 Java SE 的相关知识。读者需要从 Oracle 公司官方网站(http://www.oracle.com/technetwork/java/javase/downloads/index.html)或者通过搜索引擎下载与自己平台匹配的 JDK 安装文件。本书采用的是 32 位版的 Windows 操作系统的 JDK 安装文件“jdk-7u15-windows-i586.exe”，该安装文件在 64 位版的 Windows 操作系统下也可以安装。当然，采用其他更高版本的 JDK 也是可以的。

JDK 的安装和卸载说明如下：

(1) 安装前，本书在 D 盘下新建文件夹 JavaDevelop，用于集中管理 Java 开发用到的相关软件和文件，建议软件都不要安装在带有中文或者空格的目录下。

(2) 双击 JDK 安装文件，进入 JDK 安装向导界面，如图 1-2 所示。

图 1-2 JDK 7.0 安装向导界面

点击【下一步】按钮进入 JDK 自定义安装界面，如图 1-3 所示。

图 1-3 JDK 自定义安装界面

在图 1-3 中，点击【更改】按钮，更改安装目录为 D:\JavaDevelop\jdk1.7.0_15\，如图 1-4 所示。

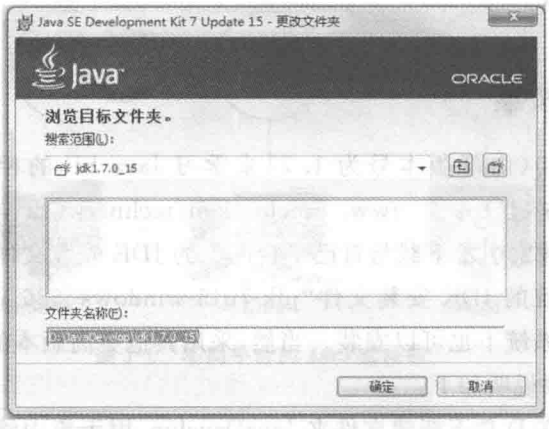

图 1-4 更改 JDK 安装目录

更改 JDK 安装目录后，点击【确定】按钮，然后在“公共 JRE”下拉选项中选择“此功能将不可用。”，如图 1-5 所示。

图 1-5　选择不安装公共 JRE

JRE 是 Java 运行环境(Java runtime environment)的英文缩写，由于开发工具中有一个 JRE，因此可以选择不安装。点击【下一步】按钮后便开始安装 JDK，安装完毕后进入图 1-6 所示的界面。点击【关闭】按钮后完成 JDK 的安装。

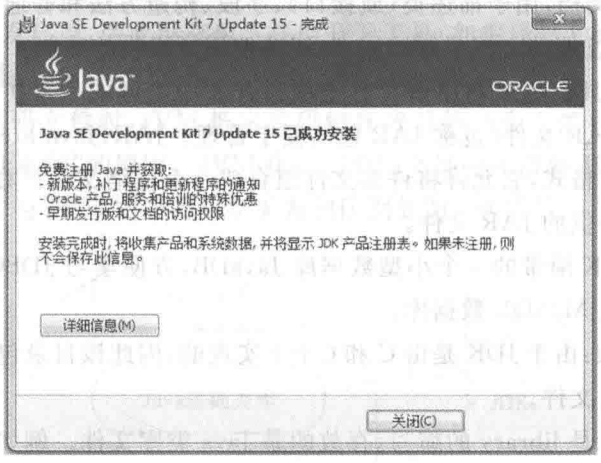

图 1-6　JDK 安装成功

(3) 如果由于更换版本等原因需要卸载已安装好的 JDK，不能直接删除 JDK 的安装目录。正确的做法是通过“控制面板”里的“添加/删除程序”卸载，或者通过专业的软件工具卸载。

1.2.2　JDK 目录说明

JDK 安装成功后，会在安装目录下生成若干子目录和文件，如图 1-7 所示。

其中比较重要的目录或文件的含义说明如下：

(1) bin 目录：该目录存放 Java 开发需要的编译、运行等工具，是可执行的程序。常用工具说明如下：

图 1-7 JDK 的目录结构

javac. exe:Java 编译器。可以将编写好的 Java 源文件(扩展名为. java)编译成 Java 字节码文件(扩展名为. class)。

java. exe:Java 运行工具。它会启动一个 Java 虚拟机(JVM)进程,专门负责运行 Java 编译器生成的字节码文件。

javadoc. exe:Java 文档生成工具。可以将 Java 程序中的文档注释提取出来,自动生成 HTML 格式的帮助文档,用于描述类(或接口)、字段、构造方法和普通方法。当程序文档化后,使用者就可以通过帮助文档了解该程序,正确使用所提供的功能。

jar. exe:打包工具。可以利用该工具把项目开发中编译的一堆扩展名为. class 的字节码文件打包成一个 JAR 文件(也称 JAR 包),便于管理。JAR(Java archive,Java 归档)是一种与平台无关的文件格式,它允许将许多文件组合成一个压缩文件。在程序开发时,经常会导入和使用第三方提供的 JAR 文件。

(2) db 目录:JDK 附带的一个小型数据库 JavaDB,方便学习 JDBC 时不用额外再装数据库。本书采用的是 MySQL 数据库。

(3) include 目录:由于 JDK 是由 C 和 C++实现的,因此该目录包含需要引入的一些 C 语言开发时用到的头文件。

(4) lib 目录:lib 是 library 的缩写,存放的是 Java 类库文件。例如 rt. jar,rt 是 runtime 的缩写,是 Java 程序在运行时必不可少的文件,里面包含了 Java 程序开发时常用的类,包含 java. lang 包、java. util 包、java. io 包等。

(5) src. zip 文件:src 文件夹的压缩包,存放 Java 核心类的源代码。注意,rt. jar 中存放的是编译后的. class 文件,通过 src. zip 文件可以查看相关类文件的源代码。

(6) jre 目录:JRE 是 Java 运行时环境(Java runtime environment)的缩写,目录结构如图 1-8 所示。

JRE 是运行 Java 程序所必需的环境的集合,包括 Java 虚拟机(JVM)、Java 核心类库和支持文件。但是,它不包含开发工具,如编译器、调试器和其他工具。因此,jre 目录里有 java. exe,但是没有 javac. exe 等开发时才用到的工具。

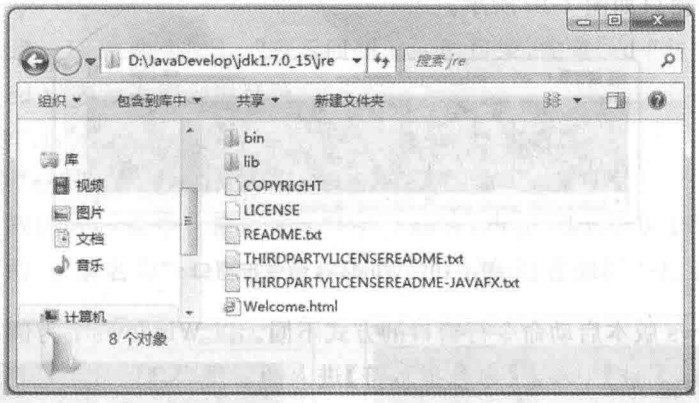

图 1-8　jre 目录结构

1.2.3　JDK、JRE 和 JVM 的关系

学习 Java 时，经常会提到 JDK、JRE 和 JVM，它们之间的关系说明如下：

(1) JDK 是 Java 开发工具包，是整个 Java 的核心，包括了 Java 运行环境 JRE、一堆 Java 工具和 Java 基础类库。如果要开发程序，必须安装 JDK。编译后的 Java 程序必须要有 JRE 才能运行，JDK 里自带了 JRE。

(2) JRE 是 Java 运行时环境，包含 JVM 标准实现及 Java 核心类库。JRE 是 Java 运行环境，并不是一个开发环境，因此没有包含任何开发工具(如编译器和调试器)。普通用户只需安装 JRE 即可运行 Java 字节码文件。

(3) 在执行字节码文件时，JVM 把字节码解释成具体平台上的机器指令执行，是 Java 能够“一次编译，到处运行”的原因。JVM 执行字节码文件时还需要 JRE 下类库的支持。

总之，JDK、JRE 与 JVM 之间的主要关系和区别如图 1-9 所示。

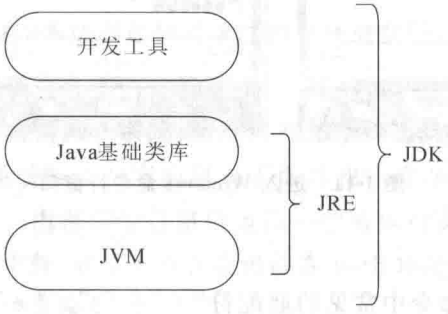

图 1-9　JDK、JRE 与 JVM 之间的关系

1.2.4　Windows 命令行窗口操作

当使用记事本和 JDK 开发 Java 程序时，就会和 Windows 命令行窗口(即 DOS 窗口)打交道，下面介绍如何打开命令行窗口以及常用的 DOS 命令。

1. 如何打开命令行窗口

JDK 中提供的 javac. exe、java. exe 等可执行文件都可以在 Windows 命令行窗口运行。

Windows 命令行窗口如图 1-10 所示。

图 1-10　Windows 命令行窗口

不同 Windows 版本启动命令行窗口的方式不同。以 Windows 7 为例，点击屏幕左下角的【开始】菜单，在【附件】中双击【命令提示符】进入命令行窗口。或者在【搜索程序和文件】栏中输入“cmd”，然后按 Enter 键（或者鼠标左键点击左上角出现的“cmd. exe”）即可进入 Windows 命令行窗口，如图 1-11 所示。

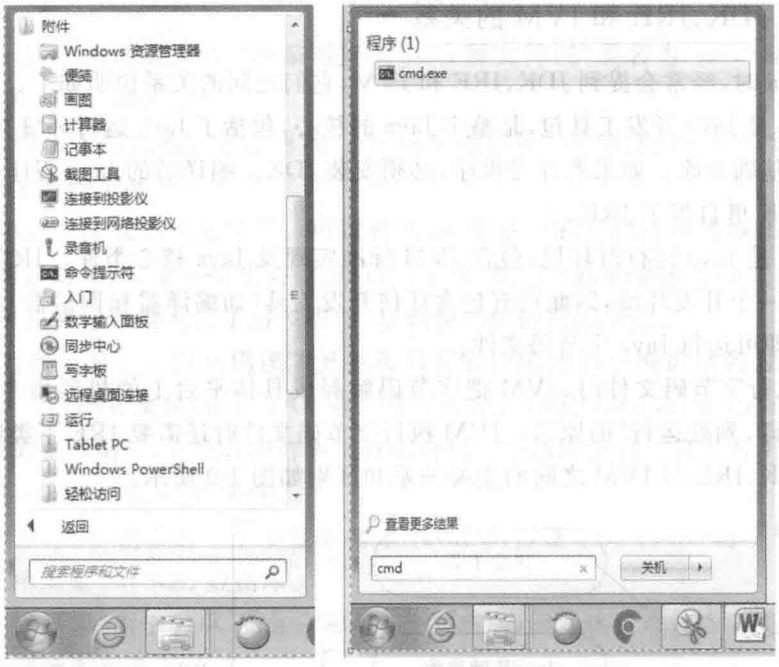

图 1-11　进入 Windows 命令行窗口

2. 常用 DOS 命令

首先，介绍一下 DOS 命令中常见的通配符“*”和“?”。“*”表示一个字符串，“?”代表一个字符。例如：“jdk*”表示以 jdk 开头的所有文件夹或文件；“*.java”表示扩展名为 java 的所有文件；“? s*.*”表示第二个字母为 s 的所有文件。

其次，命令行窗口中几个常用的 DOS 命令如下：

（1）切换盘符。例如，要回到 D 盘，则在命令行窗口中敲入“d:”，然后按 Enter 键即可。

（2）显示一个目录下的文件和子目录。DOS 命令为“dir”，dir 是 directory（目录）的缩写，可以结合通配符“*”使用，例如 dir *.java。

（3）目录的进入。DOS 命令为“cd”，cd 是 change directory（改变目录）的缩写，可以结合“dir”和通配符“*”使用。通过“dir”显示当前有哪些目录，通配符“*”可以方便使用者不

用敲完整的目录名。

(4) 目录的回退。DOS 命令“cd..”表示回到上一层目录,注意“cd ”与“..”之间是有空格的。DOS 命令“cd \”表示返回到根目录。

(5) DOS 命令“cls”表示清屏。

(6) DOS 命令“exit”表示退出命令行窗口。

例如,要求在 Windows 命令行窗口进入“D:\JavaDevelop\jdk1.7.0_15\bin”目录,同时显示以“java”开头的、扩展名为“exe”的文件。DOS 命令操作过程如图 1-12 所示。

图 1-12 DOS 命令的演示

1.2.5 系统环境变量设置

系统环境变量相当于给系统设置的一些参数,具体起什么作用和具体的环境变量相关。通常,在使用 JDK 前需要配置两个系统环境变量,即 path 和 classpath(不区分大小写)。下面分别介绍在 Windows 7 操作系统中如何设置这两个环境变量。

1. 设置 path 环境变量

path 环境变量的作用是告诉操作系统,当要求操作系统运行一个程序而没有告诉它程序所在的完整路径时,操作系统除了在当前目录下面寻找该程序外,还应当在 path 环境变量里配置的那些路径下寻找。当在命令行窗口运行一个可执行文件时,操作系统首先会在当前目录下查找是否存在该文件,如果不存在会继续在 path 环境变量中定义的路径下寻找这个文件,如果仍未找到,系统会报错。

为了保证在 Windows 命令行窗口任意路径下均可执行 JDK 里的“javac.exe”“java.exe”等命令,需要将这些命令所在的目录添加至 path 环境变量。具体步骤如下。

(1) 打开环境变量窗口。

鼠标右键点击【计算机】,在下拉菜单中选择【属性】,在弹出的窗口中选择【高级系统设置】,在弹出的窗口中选择【高级】标签,然后点击【环境变量】按钮,如图 1-13 所示。

(2) 设置 path 环境变量。

首先,新建 JAVA_HOME 变量。在系统变量下点击【新建】按钮,在弹出的“新建系统变量”对话框里输入变量名“JAVA_HOME”,变量值“D:\JavaDevelop\jdk1.7.0_15”,如图

图 1-13　环境变量窗口

1-14 所示。

然后，修改“Path”变量。选中“Path”系统变量，点击【编辑】按钮修改，在原有内容前插入“%JAVA_HOME%\bin;”，如图 1-15 所示。

最后，一路点击【确定】按钮完成设置。

图 1-14　新增 JAVA_HOME 系统变量

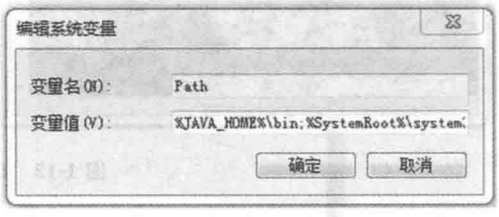

图 1-15　修改“Path”系统环境变量

(3) 查看并验证 path 系统环境变量。

打开 Windows 命令行窗口，执行“set path”命令查看当前 path 值，如图 1-16 所示。

从 path 取值来看，Java 命令所在的目录已经添加成功，放在最前面是为了优先查找。

在 Windows 命令行的任一目录下测试“javac. exe”“java. exe”等命令，如果能显示该命令的帮助信息，则证明 path 系统环境变量设置成功，如图 1-17 所示。

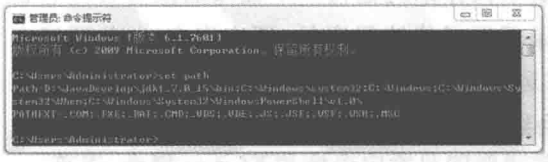

图 1-16　查看 path 系统环境变量

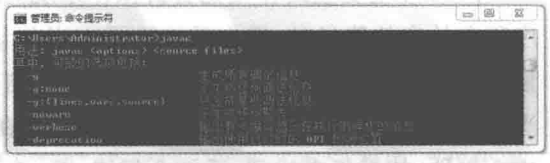

图 1-17　验证 path 系统环境变量

2. classpath 环境变量设置

classpath 环境变量也用于保存一系列路径(没有打包的文件，就用目录；已经打包的，必须指定到打包文件的名称)，它和 path 环境变量的查看与配置方式完全相同。当 Java 虚拟机需要运行一个类时，会在 classpath 环境变量中所定义的路径下寻找所需要的 class 文件。

需要注意的是，从 JDK 5 开始，如果 classpath 环境变量没有进行配置，那么 Java 虚拟机会自动搜索当前路径下的类文件，并且自动加载“dt. jar”和“tools. jar”包中的 Java 类，因此，从 JDK 5 开始，可以不设置 classpath 环境变量。

1.2.6 第一个 Java 程序

在安装好 JDK、设置好系统环境变量后就完成了 Java 开发环境搭建，下面开始学习第一个 Java 程序："Hello World!"。

【例 1-1】

通过记事本编写一个 Java 源程序，文件名为"HelloWorld.java"。在 Windows 命令行窗口里编译该 Java 源程序，并运行编译后的字节码文件(.class 文件)，在命令行窗口打印"Hello World!"。

步骤 1：编写 Java 源程序。

在目录"D:\JavaDevelop\chapter1"目录下新建一个文本文档，名称为"新建文本文档.txt"，然后重命名为"HelloWorld.java"。注意，如果在新建文本文件时名称里后缀没有发现扩展名".txt"，这说明文件的扩展名被系统隐藏了，需要让文件名显示出扩展名。具体方法是：先找到【文件夹和搜索选项】，选择【查看】标签页，在"高级设置"中将"隐藏已知文件类型的扩展名"选项前的"√"取消掉，然后点击【确定】按钮退出，如图 1-18 所示。

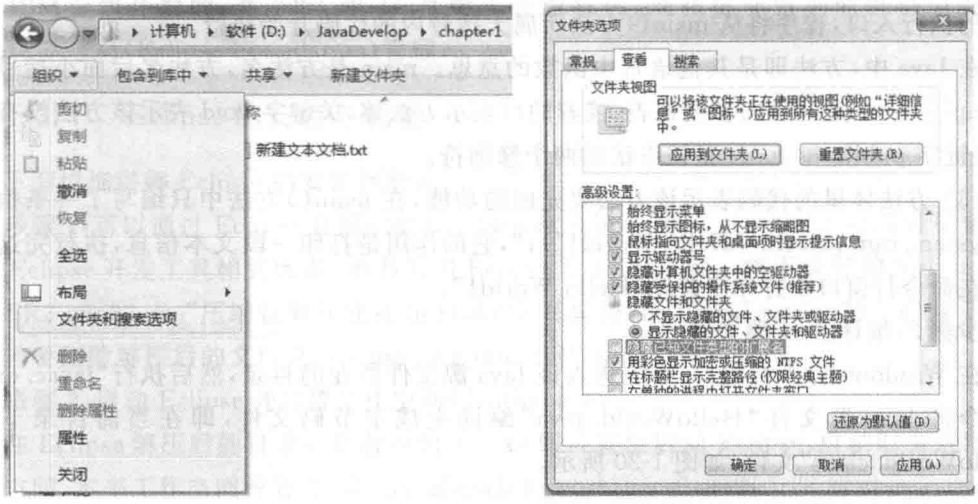

图 1-18　显示已知文件类型的扩展名

在编写 Java 代码时，需要注意以下几点：

(1) 程序中出现的空格、括号、分号等符号必须采用英文半角格式，否则程序编译会报错。

(2) 熟练掌握如下快捷键，开发过程中时常用到。

全选：Ctrl+A　　复制：Ctrl+C　　粘贴：Ctrl+V

剪切：Ctrl+X　　撤销：Ctrl+Z　　保存：Ctrl+S

(3) 建议 Java 源文件的名称和类名保持一致。

(4) Java 语言严格区分大小写，不要敲错。

(5) 程序中的括号都是成对出现的，注意配对问题。建议敲完左括号后马上敲右括号，然后再填写括号内的内容，避免出现少括号的问题。

(6) 程序中建议用 Tab 键缩进，目的是使得整个代码整齐，方便阅读。

用系统自带的记事本将 HelloWorld.java 文件打开，编写 Java 源程序，如图 1-19 所示。

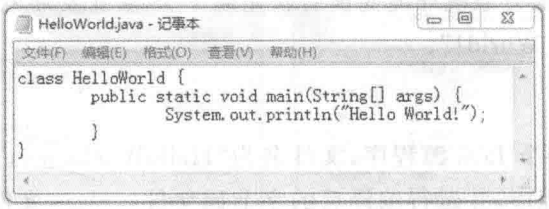

图 1-19 HelloWorld.java 源文件

例 1-1 中的代码实现了第一个 Java 程序，下面对其中的代码做简单解释。

(1) Java 程序最基本的单位是类，所以首先要定义一个类。class 是一个关键字，它用于定义一个类，所有的代码都需要在类中书写。HelloWorld 是类的名称，简称类名。class 关键字与类名之间需要用空格、制表符、换行符等任意的空白字符进行分隔。类名之后要写一对大括号，它定义了当前这个类的管辖范围。

(2) "public static void main(String[] args){}"定义了一个 main()方法，该方法是 Java 程序的执行入口，程序将从 main()方法所属大括号内的代码开始执行。

在 Java 中，方法即是其他语言中函数的意思。main 是方法名，方法名后面小括号里的"String[] args"是方法的参数列表，接着的{}表示方法体，关键字 void 表示该方法没有返回值，关键字 public 和 static 是该方法的两个修饰符。

(3) 方法体里的代码表示该方法要完成的功能，在 main()方法中只编写了一条执行语句"System.out.println("Hello World!");"，它的作用是打印一段文本信息，执行完这条语句会在命令行窗口中打印字符串"Hello World!"。

步骤 2：编译 Java 源文件。

在 Windows 命令行窗口，首先进入该 Java 源文件所在的目录，然后执行"javac.exe"编译命令将 Java 源文件"HelloWorld.java"编译生成字节码文件，即在当前目录下生成"HelloWorld.class"文件，如图 1-20 所示。

图 1-20 编译 Java 源文件

如果 Java 源文件在编译过程中出现编译错误，则需要根据提示修改源代码，然后再重新编译。

步骤 3：运行 Java 程序。

通过 Java 命令运行编译后的"HelloWorld.class"文件，在命令行窗口显示程序运行结果，如图 1-21 所示。