计算机应用案例教程系列

Java开发案例教程

王晓娟　王　超　刘　涛◎编著

清华大学出版社

北　京

内 容 简 介

本书以通俗易懂的语言、翔实生动的案例全面介绍 Java 基础知识和相关技术。全书共分 13 章，内容包括 Java 语言概述，Java 基础语法，分支结构，循环结构，方法的使用，数组的概念和应用以及如何将数组作为方法的参数使用，字符串的定义及操作方法，类和对象的概念及使用，继承、多态与接口，Applet 编程，GUI 编程，I/O 编程，线程的概念以及 Java 多线程程序的创建和使用。

本书提供配套的素材文件以及云视频教学平台等资源的 PC 端下载地址，以方便读者扩展学习。本书具有很强的实用性和可操作性，可作为高等院校计算机相关专业的教材，同时也是广大初中级计算机用户的首选参考书。

本书对应的电子课件和实例源文件可以到 http://www.tupwk.com.cn/teaching 网站下载，也可以通过扫描前言中的二维码推送配套资源到邮箱。

图书在版编目(CIP)数据

Java 开发案例教程 / 王晓娟，王超，刘涛编著. —北京：清华大学出版社，2021.2
计算机应用案例教程系列
ISBN 978-7-302-57521-4

Ⅰ. ①J… Ⅱ. ①王… ②王… ③刘… Ⅲ. ①JAVA 语言－程序设计－高等学校－教材 Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2021)第 025247 号

责任编辑：胡辰浩
封面设计：高娟妮
版式设计：妙思品位
责任校对：成凤进
责任印制：丛怀宇

出版发行：清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者：小森印刷霸州有限公司
经　　销：全国新华书店
开　　本：185mm×260mm　　**印　　张：**18.25　　**字　　数：**467 千字
版　　次：2021 年 4 月第 1 版　　**印　　次：**2021 年 4 月第 1 次印刷
定　　价：79.00 元

产品编号：076400-01

前言

熟练使用计算机已经成为当今社会不同年龄段人群必须掌握的一门技能。为了使读者在短时间内轻松掌握计算机各方面应用的基本知识，并快速解决生活和工作中遇到的各种问题，清华大学出版社组织了一批教学精英和业内专家特别为计算机学习用户量身定制了这套“计算机应用案例教程系列”丛书。

丛书和配套资源

选题新颖，结构合理，内容精炼实用，为计算机教学量身打造

本丛书注重理论知识与实践操作的紧密结合，同时贯彻“理论+实例+实战”3 阶段教学模式，在内容选择、结构安排方面更加符合读者的认知规律，从而达到老师易教、学生易学的目的。丛书采用双栏紧排的格式，合理安排图与文字的占用空间，在有限的篇幅内为读者提供更多的计算机知识和实战案例。丛书完全以高等院校及各类社会培训学校的教学需要为出发点，紧密结合学科的教学特点，由浅入深地安排章节内容，循序渐进地完成各种复杂知识的讲解，使学生能够一学就会、即学即用。

配套资源丰富，全方位扩展知识能力

本丛书配套的素材文件和云视频教学平台等资源，可通过在 PC 端的浏览器中下载后使用。用户也可以通过扫描下方的二维码推送配套资源到邮箱。

(1) 本书配套素材和云视频教学平台的下载地址如下。

http://www.tupwk.com.cn/teaching

(2) 本书配套资源的二维码如下。

扫码推送配套资源到邮箱

在线服务，疑难解答，贴心周到，方便老师定制教学课件

便捷的教材专用通道(QQ：22800898)为老师量身定制实用的教学课件。老师也可以登录本丛书的信息支持网站(http://www.tupwk.com.cn/teaching)下载图书对应的电子课件。

本书内容介绍

《Java 开发案例教程》是本丛书中的一种，该书从读者的学习兴趣和实际需求出发，合理安排知识结构，由浅入深、循序渐进，通过图文并茂的方式讲解 Java 语言的相关知识和编

程技术。全书共分 13 章，各章主要内容如下。

第 1 章介绍 Java 语言的概况。

第 2 章介绍 Java 语言的基础语法，解释变量及基本数据类型的含义。

第 3 章介绍分支结构及 Java 语言对应的实现语句。

第 4 章介绍循环结构及 Java 语言对应的实现语句。

第 5 章介绍方法的使用、递归算法以及变量的作用域。

第 6 章介绍数组的概念和应用以及如何将数组作为方法的参数使用。

第 7 章介绍字符串的定义以及 String 和 StringBuffer 类型字符串的操作方法。

第 8 章介绍面向对象技术中的类、对象以及包的概念和使用。

第 9 章介绍类的继承、多态以及抽象类和接口。

第 10 章介绍 Java Applet 的开发和 HTML 的知识。

第 11 章介绍图形用户界面技术以及 AWT 组件集。

第 12 章介绍基于流的 Java 输入输出技术。

第 13 章介绍线程的概念以及 Java 多线程程序的创建和使用。

读者定位和售后服务

本丛书为所有从事计算机教学的老师和自学人员而编写，是一套适合于高等院校及各类社会培训机构的优秀教材，也可作为广大初中级计算机用户的首选参考书。

如果您在阅读图书或使用计算机的过程中有疑惑或需要帮助，可以登录本丛书的信息支持网站(http://www.tupwk.com.cn/teaching)，本丛书的作者或技术人员会提供相应的技术支持。

本书由佳木斯大学的王晓娟编写第 1～3、11、12 章，王超编写第 4～6、10、13 章，刘涛编写第 7～9 章。由于作者水平有限，本书难免有不足之处，欢迎广大读者批评指正。我们的邮箱是 huchenhao@263.net，电话是 010-62796045。

“计算机应用案例教程系列”丛书编委会

2020 年 11 月

目录

第1章

初识 Java

本章将对 Java 进行初步的介绍，使读者对 Java 的特点有所了解，并通过一个 Java 程序对 Java 的开发环境和开发步骤进行具体的讲解，帮助初学者建立学好 Java 语言的信心。

1.1 程序设计语言

Java 语言是目前使用最为广泛的编程语言之一，是一种简单、面向对象、分布式、解释、健壮、安全、与平台无关且性能优异的多线程动态语言。

1.1.1 Java 语言的发展历程

Java 语言的前身是 Oak 语言。1991 年 4 月，Sun 公司(已被 Oracle 公司收购)以 James Gosling 为首的绿色计划项目组(Green Project)计划开发一种分布式系统结构，使其能够在各种消费类电子产品上运行。项目组成员在使用 C++编译器时发现了 C++的很多不足之处，于是研发出 Oak 语言来替代 C++，但仅限于 Sun 公司内部使用。

1994 年下半年，由于 Internet 的迅速发展和 Web 的广泛应用，工业界迫切需要一种能够在异构网络环境下使用的语言，James Gosling 项目组在对 Oak 语言进行小规模改造的基础上于 1995 年 3 月推出了 Java 语言，并于 1996 年 1 月发布了包含开发支持库的 JDK 1.0 版本。该版本包括 Java 运行环境(JRE)和 Java 开发工具箱(Java Development Kit，JDK)，其中 JRE 包括核心 API、集成 API、用户界面 API、发布技术及 JVM(Java 虚拟机)5 个部分，而 JDK 包括编译 Java 程序的编译器(javac)。在 JDK 1.0 版本中，除 AWT 外，其他的库并不完整。

1997 年 2 月，Sun 公司发布了 JDK 1.1 版本，为 JVM 增加了即时编译器(JIT)。与传统的编译器编译一条指令并等待其运行完之后再将其释放掉不同的是，JIT 将常用的指令保存在内存中，这样在下次调用时就没有必要再编译了。继 JDK 1.1 版本后，Sun 公司又推出了数个 JDK 1.x 版本。

虽然在 1998 年之前，Java 被众多的软件企业采用，但由于当时硬件环境和 JVM 技术尚不成熟，Java 的应用很有限。那时 Java 主要应用于前端的 Applet 以及一些移动设备。然而这并不等于 Java 的应用只限于这些领域。1998 年是 Java 迅猛发展的一年，在 1998 年，Sun 发布了 JSP/Servlet、EJB 规范并且将 Java 分成了 J2EE、J2SE 和 J2ME，还标志着 Java 已经吹响了向企业、桌面和移动 3 个领域进军的号角。

1998 年 12 月，Sun 公司发布了 JDK 1.2 版本。JDK 1.2 版本是 Java 语言发展过程中的一个关键阶段，从此 Sun 公司将 Java 更名为 Java 2。

JDK 1.2 版本可分 J2EE、J2SE 和 J2ME 三大应用平台。JDK 1.2 版本的 API 分成了核心 API、可选 API 和特殊 API 三大类。其中核心 API 是由 Sun 公司制定的基本的 API，所有的 Java 平台都应该提供；可选 API 是 Sun 为 JDK 提供的扩充 API；特殊 API 是用于满足特殊要求的 API。同时，JDK 1.2 版本增加了 Swing 图形库，其中包含各式各样的组件。

从 JDK 1.2 版本开始，Sun 以平均每两年一个版本的速度推出新的 JDK。Java 在发展的 20 多年时间里，经历了无数的风风雨雨，现在 Java 已经成为一种相当成熟的语言。Java 平台吸引了数百万的开发者，在网络计算遍及全球的今天，Java 已被广泛应用于移动电话、桌面计算机、蓝光光碟播放器、机顶盒甚至车载，有 30 多亿台设备使用了 Java 技术。

1.1.2 Java 语言的特点

作为一种面向对象且与平台无关的多线程动态语言，Java 具有以下特点。

1. 语法简单

Java 语言的简单性主要体现在以下三个方面。

(1) Java 的风格类似于 C++，C++程序员可以很快掌握 Java 编程技术。

(2) Java 摒弃了 C++中容易引发程序错

误的地方，如指针和内存管理。

(3) Java 提供了丰富的类库。

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。面向对象可以说是 Java 最重要的特性。Java 语言的设计完全是面向对象的，不支持类似 C 语言那样的面向过程的程序设计技术。Java 支持静态和动态风格的代码继承及重用。单从面向对象的特性看，Java 类似于 SmallTalk，但其他特性，尤其是适用于分布式计算环境的特性远远超越了 SmallTalk。

3. 分布式

Java 从诞生起就与网络联系在一起，它强调网络特性，内置 TCP/IP、HTTP 和 FTP 协议栈，便于开发网上应用系统。因此，Java 应用程序可凭借 URL 打开并访问网络上的对象，访问方式与访问本地文件系统完全相同。

4. 安全性

Java 的安全性可从两个方面得到保证。一方面，在 Java 语言里，像指针和释放内存等 C++中的功能被删除，避免了非法内存操作。另一方面，当 Java 用来创建浏览器时，语言功能和一些浏览器本身提供的功能结合起来，使 Java 更安全。Java 程序在机器上执行前，要经过很多次测试。Java 的三级安全检验机制可以有效防止非法代码入侵，阻止对内存的越权访问。

5. 健壮性

Java 致力于检查程序在编译和运行时的错误。除了运行时异常检查之外，Java 还提供了广泛的编译时异常检查，以便尽早发现可能存在的错误。类型检查可帮助用户检查出许多早期开发中出现的错误。Java 自己操纵内存减少了内存出错的可能性。Java 还实现了真数组，避免了覆盖数据的可能，这项功能大大缩短了开发 Java 应用程序的周期。同时，Java 中对象的创建机制(只能使用 new 操作符)和自动垃圾收集机制大大减少了因内存管理不当引发的错误。

6. 解释运行效率高

Java 解释器(运行系统)能直接运行目标代码指令。Java 程序经编译器编译，生成的字节码已经过精心设计并进行了优化，因此运行速度较快，克服了以往解释性语言运行效率低的缺点。Java 使用直接解释器可在 1 秒内调用 300000 个进程。翻译目标代码的速度与 C/C++相比没什么区别。

7. 与平台无关

Java 编译器会将 Java 程序编译成二进制代码，也就是字节码。字节码有统一的格式，不依赖于具体的硬件环境。

平台无关类型包括源代码级和目标代码级两种类型。C 和 C++属于源代码级，与平台无关，还意味着用它们编写的应用程序不用修改，只需要重新编译就可以在不同平台上运行。Java 属于目标代码级，与平台无关，主要靠 Java 虚拟机(Java Virtual Machine，JVM)来实现。

8. 多线程

Java 提供的多线程功能使得在一个 Java 程序里可同时执行多个小任务。线程有时也称作小的进程，是从一个大的进程里分出来的小且独立的进程。Java 由于实现了多线程技术，因此相比 C 和 C++更健壮。多线程带来的更大的好处是更好的交互性能和实时控制性能。当然实时控制性能还取决于系统本身(UNIX、Windows、Macintosh 等)，在开发难易程度和性能上都比单线程好。比如在上网时，大家都会觉得为调一幅图片而等待是一件令人烦恼的事情，而在 Java 里，可使用单线程来调一幅图片，同时可以访问 HTML 里的其他信息而不必等待。

9. 动态性

Java 的动态性是 Java 面向对象设计方法

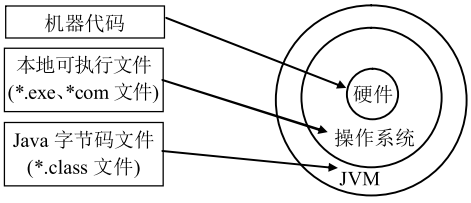
的进一步发展。Java 允许程序动态装入运行过程中所需的类，这是 C++语言进行面向对象程序设计时无法实现的功能。在 C++程序设计过程中，每当在类中增加一个实例变量或一种成员函数后，引用该类的所有子类都必须重新编译，否则将导致程序崩溃。Java 编译器不是将对实例变量和成员函数的引用编译为数值引用，而是将符号引用信息在字节码中保存下来传递给解释器，再由解释器在完成动态链接后，将符号引用信息转换为数值偏移量。这样一来，存储器中生成的对象不是在编译过程中确定的，而是延迟到运行时由解释器确定，因此在对类中的变量和方法进行更新时不至于影响现存的代码。解释执行字节码时，这种符号信息的查找和转换过程仅在新的名字出现时才进行一次，随后代码便可以全速执行。在运行时确定引用的好处是可以使用已更新的类，而不必担心影响原有代码。如果程序引用了网络上另一系统中的某个类，那么该类的所有者也可以自由地对该类进行更新，而不会使任何引用该类的程序崩溃。如果系统在运行 Java 程序时遇到了不知道怎样处理的功能程序，Java 能自动下载所需的功能程序。

1.1.3 Java 虚拟机(JVM)

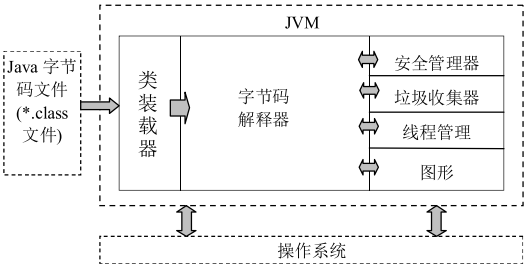
虚拟机是一种对计算机物理硬件计算环境的软件实现。虚拟机是一种抽象机器，内部包含解释器(Interpreter)，可以将其他高级语言编译为虚拟机的解释器可以执行的代码[我们称这种代码为中间语言(Intermediate Language)]，从而实现高级语言程序的可移植性与平台无关性(System Independence)，无论是运行在嵌入式设备上还是包含多个处理器的服务器上，虚拟机都执行相同的指令，使用的支持库也具有标准的 API 和完全相同或相似的行为。

Java 虚拟机依附于具体的操作系统，本身具有一套虚拟的机器指令，并且有自己的栈、寄存器等运行 Java 程序必不可少的机制。编译后的 Java 指令并不直接在硬件系统的 CPU 上执行，而是在 JVM 上执行。JVM 提供了解释器来解释 Java 编译器编译后的程序。任何一台机器只要配备了解释器，就可以运行这个程序，而不管这种字节码是在何种平台上生成的。

JVM 是编译后的 Java 程序和硬件系统之间的接口，程序员可以把 JVM 看作虚拟处理器。JVM 不仅解释执行编译后的 Java 指令，而且还进行安全检查，JVM 是 Java 程序能在多平台间进行无缝移植的可靠保证，同时也是 Java 程序的安全检查引擎，如下图所示。



JVM 由多个组件构成，包括类装载器(Class Loader)、字节码解释器(Bytecode Interpreter)、安全管理器(Security Manager)、垃圾收集器(Garbage Collector)、线程管理(Thread Management)及图形(Graphics)，如下图所示。



(1) 类装载器：负责加载类的字节码文件，并完成类的链接和初始化工作。类装载器首先将要加载的类名转换为类的字节码文件名，并在环境变量 CLASSPATH 指定的每个目录中搜索字节码文件，把字节码文件读入缓冲区。其次将类转换为 JVM 内部的数据结构，并使用校验器检查类的合法性。如果类是第一次被加载，就对类

中的静态数据进行初始化。最后加载类中引用的其他类，把类中的某些方法编译为本地代码。

(2) 字节码解释器：字节码解释器是整个 JVM 的核心组件，负责解释执行由类装载器加载的字节码文件中的字节码指令集合，并通过 Java 运行环境(JRE)由底层的操作系统实现操作。可通过使用汇编语言编写解释器，重组指令流以提高处理器的吞吐量，在最大程度上使用高速缓存及寄存器等措施来优化字节码解释器。

(3) 安全管理器：根据一定的安全策略对 JVM 中指令的执行进行控制，主要包括那些可能影响底层操作系统的安全性或完整性的 Java 服务调用，每个类装载器都与某个安全管理器相关，安全管理器负责保护系统不受由加载器载入系统的类企图执行的违法操作的侵害。默认的类装载器使用信任型安全管理器。

(4) 垃圾收集器：垃圾收集器用于检测不再使用的对象，并将它们占用的内存回收。Java 语言并不是第一种使用垃圾收集技术的语言。垃圾收集是一种成熟的技术，早期的面向对象语言 LISP、SmallTalk 等已经提供了垃圾收集机制。理想的垃圾收集应该回收所有形式的垃圾，如网络连接、I/O 路径等。在 JVM 中，垃圾收集的启动方式可分为请求式、要求式和后台式。请求式是通过调用 System.gc( )方法请求 JVM 进行垃圾收集的。要求式是指当使用 new( )方法创建对象时，如果内存资源不足，则请求 JVM 进行垃圾收集。后台式是指通过一个独立的线程检测系统的空闲状态，如果发现系统空闲了多个指令周期，就进行垃圾收集。

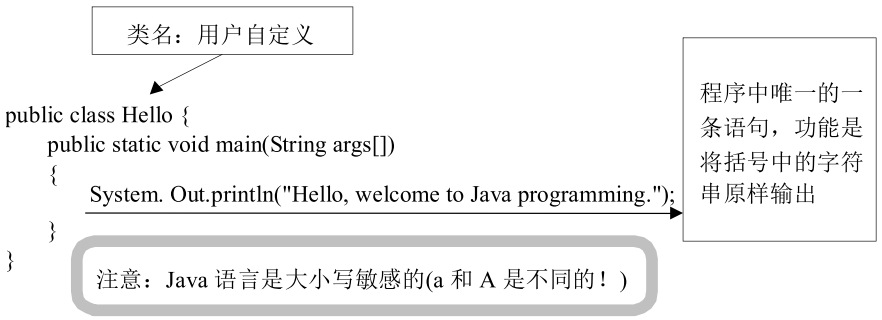
1.2　第一个 Java 程序

Java 程序有两种类型：Java 应用程序(Java Application)和 Java 小程序(Java Applet)。虽然二者的编程语法完全一样，但后者需要客户端浏览器的支持才能运行，并且在运行前必须先嵌入 HTML 文件的<applet>和</applet>标签对中。当用户浏览 HTML 页面时，将首先从服务器端下载 Java 小程序，进而被客户端的 Java 虚拟机解释和运行。由于 Java 小程序与 HTML 联系紧密，且编程相对复杂，因而放在后面章节中进行介绍，这里只以 Java 应用程序为例进行说明。下面就来看看第一个完整的 Java 程序。

```
public class Hello {
    public static void main(String args[])
    {
        System.out.println("Hello,welcome to Java programming.");
    }
}
```

Java 源程序是以文本格式存放的，文件扩展名必须为.java，对于上面的程序，我们将其保存为 Hello.java 文件，这里有个非常细小但必须注意的问题：文件名必须与(主)类名一致，包括字母大小写也要一致，通常在定义类时，类名的第一个字母大写，所以在正确编辑以上代码后，存储时也要确保文件名正确，否则后面就不能编译通过，也就运行不了了。所有的 Java 语句必须以英文的“;”结束，编写程序时千万注意别误输入中文的“；”，二者的意义是截然不同的，中文的“；”是不能被编译器识别的。另外，Java 是大小写敏感的，编写程序时也要注意区分其他关键字和标识符中的大小写字母。

下图对上述 Java 示例程序的组成做了简要描述，此时，大家可能还不太理解，但这没有关系，硬着头皮看下去，后面还会详细地对各个要素进行剖析。



图解第一个 Java 程序

在上面的图中，除了类名的定义和唯一的一条程序语句之外，其他部分可以被看作模板，照抄即可，注意其中的配对大括号以及字母的大小写。下面对整个程序稍作解释，读者不必完全理解，只要了解并注意模仿即可。

上述程序首先使用关键字 class 声明了一个新类，类名为 Hello，这是一个公共类，整个类定义已用大括号{}括起来。Hello 类中有一个 main()方法，其中 public 表示访问权限，表示所有的类都可以调用(使用)这个方法；static 指明这是一个静态的类方法，可以通过类名直接调用；void 则指明 main()方法不返回任何值。对于 Java 应用程序来说，main()方法是必需的，而且必须按照如上格式定义。Java 解释器在没有生成任何实例的情况下，将以 main()方法作为程序入口。Java 程序中可以定义多个类，每个类中也可以定义多个方法，但是最多只能有一个公共类，main()方法也只能有一个。在 main()方法的定义中，圆括号中的 String args[ ]是传递给 main()方法的参数，参数名为 args，它是 String 类的一个实例，参数可以为零个或多个，每个参数以“类名 参数名”的形式指定，多个参数之间用逗号分隔。在 main()方法的实现部分(大括号中的代码)，只有一条语句：System.out.println("Hello,welcome to Java programming."); ，它用来实现字符串的输出，这条语句与 C 语言中的 printf 语句和 C++中的 cout<<语句具有相同的功能。

比较简单的 Java 应用程序的模板如下：

```
public class 类名 {
    public static void main(String args[])
    {
        //你的程序代码！
    }
}
```

在此，我们总结一下初学者应注意的事项：

(1) 用类名后面的大括号标识类定义的开始和结束，而 main()方法后面的大括号则用来标识方法体的开始和结束。Java 程序中的大括号都是成对出现的，因而在写左大括号时，最好也把右大括号写上，这样可以避免漏掉，否则可能会给程序的编译和调试带来不便。有些初学者经常在这方面犯错，花了很多时间查错，最后才发现原来是大括号不配对。

(2) 通常，我们习惯将类名的首字母大写，而变量则以小写字母打头，变量名由多个单词组成时，除第一个单词外的每个单词的首字母应大写。

(3) 应适当使用空格符和空白行来对

程序的语句元素进行间隔，从而增强程序的可读性。一般在定义方法的大括号中，将整个方法体的内容部分缩进，使程序结构清晰，一目了然。编译器会忽略这些间隔用的空格符及空白行，也就是说，它们仅仅起到增强程序可读性的作用，而不对程序产生任何影响。

(4) 在编辑程序时，最好一条语句占一行。另外，虽然 Java 允许一条长语句分开写在几行中，但前提是不能从标识符或字符串的中间进行换行。另外，文件名与 public 类名在拼写和大小写上必须保持一致。

(5) 一个 Java 应用程序必须包含且仅包含一个 main()方法，以控制程序的运行。对于复杂的程序，除了 main()方法之外，可能还会有其他方法，这将在后面章节中介绍。

1.3　Java 程序开发工具

能用来编写 Java 源程序的工具软件有很多，只要是能编辑纯文本(注意：Word 文档不是纯文本)的都可以，比如 notepad(记事本)、wordpad(写字板)、UltraEidt、EditPlus 等。对于 Java 软件开发人员来说，一般倾向于使用一些 IDE(集成开发环境)来编写程序，以提高效率、缩短开发周期。下面我们给大家介绍一些比较流行的 IDE(尽管本书讲解的知识并不一定都用到，但将来大家可能会用到)。

1) Borland 的 JBuilder

有人说 Borland 的开发工具都是里程碑式的产品，从 Turbo C、Turbo Pascal 到 Delphi、C++ Builder 等都十分经典。JBuilder 是第一个可以开发企业级应用的跨平台开发环境，它支持最新的 Java 标准，它的可视化工具和向导使得应用程序的快速开发变得非常轻松。

2) IBM 的 Eclipse

Eclipse 是一种可扩展、开放源代码的 IDE，由 IBM 出资组建。Eclipse 框架灵活、易扩展，因此深受开发人员的喜爱，目前它的支持者越来越多，大有成为 Java 第一开发工具之势。

3) Oracle 的 JDeveloper

JDeveloper 的第一个版本采用的设计方案购自 JBuilder，不过后来已经完全没有 JBuilder 的影子了，现在的 JDeveloper 不仅是很好的 Java 编程工具，而且是 Oracle Web 服务的延伸。

4) Symantec 公司的 Visual Cafe

很多人都知道 Symantec 公司的安全产品，但很少有人知道 Symantec 的另一项堪称伟大的产品：Visual Cafe。有人认为 Visual Cafe 如同当年 Delphi 超越 Visual Basic 一样，今天，它也同样超越了 Borland 的 JBuilder。

5) IBM 的 Visual Age

Visual Age 是一款非常优秀的集成开发工具，但用惯了微软开发工具的读者在开始时可能会感到非常不适应，因为 Visual Age 采取与微软截然不同的设计方式，为什么会这样呢？那是因为蓝色巨人 IBM 怎么能跟着微软的指挥棒转呢！

6) Sun 公司的 NetBeans 与 Sun Java Studio

以前称为 Forte for Java，现在 Sun 公司将其统一称为 Sun Java Studio，出于商业考虑，Sun 将这两个工具合在一起推出，不过它们的侧重点是不同的。

7) Sun 公司的 Java WorkShop

Java WorkShop 是完全使用 Java 语言编写的，并且是当今市场上销售的第一个完整的 Java 开发环境。目前 Java WorkShop 支持 Solaris 操作环境的 SPARC 和 Intel 版以及 HP UX 等操作系统。

综上所述，可以用来进行 Java 开发的工具有很多。在计算机开发语言的历史中，从来没有哪种语言像 Java 这样受到如此众多

厂商的支持，有如此多的开发工具，Java 初学者就如初入大观园的刘姥姥，看花了眼，不知该如何选择。的确，这些工具各有所长，没有绝对完美的，让人很难做出选择。但是需要记住的是，它们仅仅是集成的开发环境，而在这些环境中，有一样东西是共同的，也是最核心和最关键的，那就是 JDK(Java Development Kit)，中文意思是 Java 开发工具集。JDK 是整个 Java 的核心，包括了 Java 运行环境(Java Runtime Envirnment)、一堆 Java 工具和 Java 基础类库(rt.jar)等，所有的开发环境都需要围绕 JDK 来进行。事实上，对于初学者而言，我们的建议是：JDK + 记事本就足够了，因为掌握 JDK 是学好 Java 的第一步，也是最重要的一步。首先用记事本编辑源程序，然后用 JDK 编译、运行 Java 程序。这种开发方式虽然简陋，但却是学好 Java 语言的最好途径。

1.4 Java 程序开发步骤

在学习 Java 语言之前，必须了解并搭建好所需的开发环境。要编译和执行 Java 程序，JDK 是必需的。下面具体介绍下载和安装 JDK、配置环境变量以及编译和运行程序的方法。

1.4.1 下载和安装 JDK

Java 运行平台主要分为以下 3 个版本。

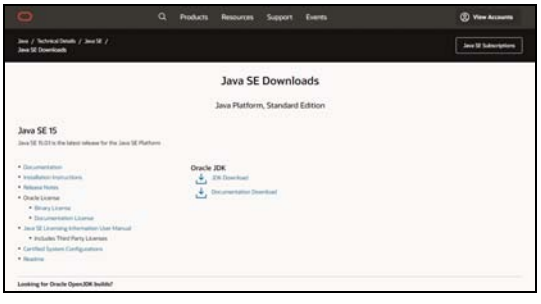
(1) Java SE：Java 标准版或 Java 标准平台。Java SE 提供了标准的 JDK 开发平台。

(2) Java EE：Java 企业版或 Java 企业平台。

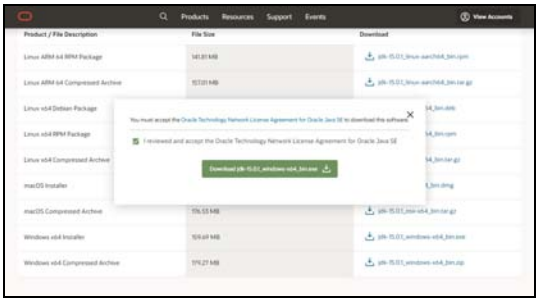
(3) Java ME：Java 微型版或 Java 小型平台。

自 JDK 6.0 开始，Java 的 3 个应用平台被称为 Java SE、Java EE 与 Java ME(之前的旧名称是 J2SE、J2EE、J2ME)。

最新的JDK需要从Oracle公司的官网进行下载。用户可以打开浏览器，输入网址 https://www.oracle.com/java/technologies /javase-downloads.html，在打开的下载界面上单击 JDK Download 链接，见下图。



然后在 JDK 的下载列表中根据操作系统选择适当的 JDK 版本，在弹出的对话框中勾选许可协议复选框后，即可下载 JDK。



JDK 的版本更新速度比较快，用户在下载 JDK 时选择最新版本的 JDK 即可。

下面我们讲解在 Windows 10 中安装 JDK 的方法。

(1) 双击已下载完毕的 JDK 安装文件，将打开如下图所示的欢迎对话框，此时单击【下一步】按钮。

(2) 在打开的【目标文件夹】对话框中，建议用户不要更改 JDK 的安装路径，其他设置保持默认选项，然后单击【下一步】按钮。

(3) 成功安装 JDK 后，将打开如下图所示的【完成】对话框，单击【关闭】按钮完成安装操作。

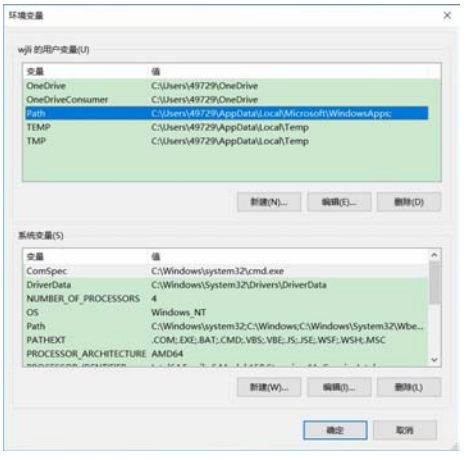
在安装 JDK 时，不要同时运行其他安装程序，以免出现错误。

1.4.2　配置环境变量

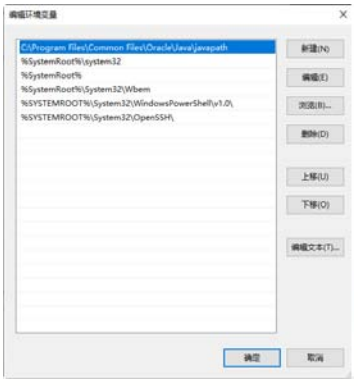
配置环境变量主要是为了“寻径”，也就是让程序能够找到它需要的文件，所以设置的内容就是一些路径。在 Windows 操作系统中，配置环境变量的具体操作如下。

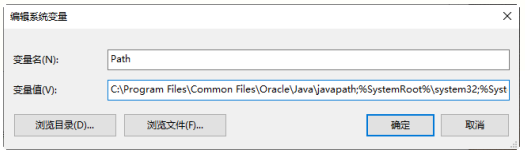
(1) 右击桌面上的【此电脑】图标，在弹出的快捷菜单中选择【属性】命令，在打开的【系统】对话框的左侧单击【高级系统设置】链接，将打开如右上图所示的【系统属性】对话框，单击对话框右下角的【环境变量】按钮。

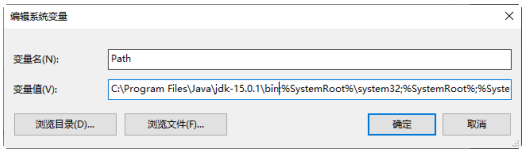
(2) 在打开的【环境变量】对话框中，在【系统变量】列表框中双击 Path 变量，如下图所示。



(3) 在打开的【编辑环境变量】对话框中，单击【编辑文本】按钮，对 Path 变量的值进行修改。先删除原变量值最前面的“C:\Program Files\Common Files\Oracle\Java\javapath;”，再输入“C:\Program Files\Java\jdk-15.0.1\bin;”(也就是已安装的 JDK 的 bin 文件夹目录)，修改前后的效果如下页左上两图所示。

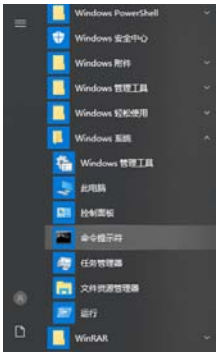




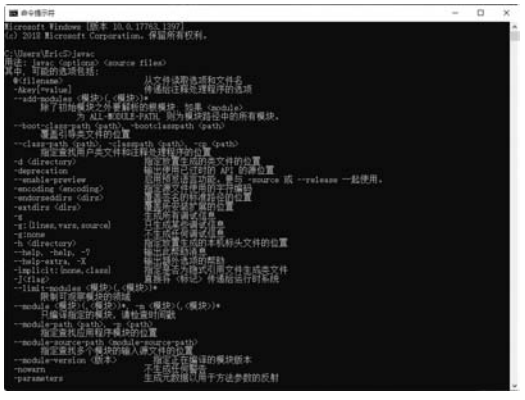


(4) 修改完毕后，逐步单击对话框中的【确定】按钮，依次退出上述对话框后，即可完成在 Windows 中配置 JDK 的相关操作。

JDK 配置完毕后，需要确认其是否匹配准确。在 Windows 中测试 JDK 环境时，需要先单击桌面左下角的【开始】图标，在弹出的【开始】菜单中选择【命令提示符】选项。



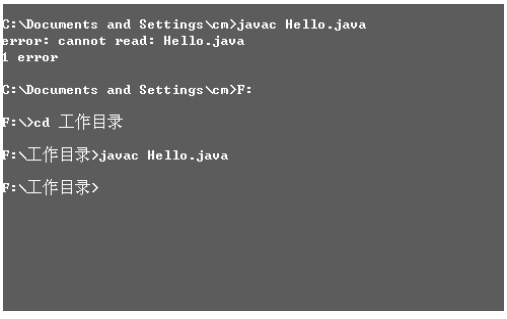
在打开的【命令提示符】对话框中输入 javac，按 Enter 键，将显示如下图所示的 JDK 编译器信息，其中包括修改命令的语法和参数选项等内容，这说明 JDK 环境已经搭建成功。



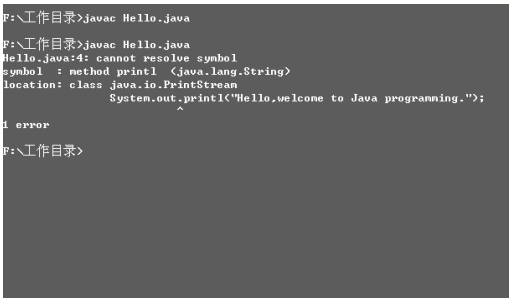
1.4.3 编译和运行程序

配置好环境变量后，就可以在命令行模式下编译和运行 Java 程序了。下面以前面介绍的第一个 Java 程序为例来说明编译过程。假定 Hello.java 程序存放在“F:\工作目录”文件夹中。打开“命令提示符”对话框，输入 javac Hello.java 命令，对源程序进行编译。

从上图可以看到，产生了编译错误，提示找不到源程序，解决办法是切换到“工作目录”，然后执行 javac Hello.java 命令。



此时，源程序编译成功，系统将在“工作目录”下生成字节码文件 Hello.class，这是一个二进制格式的文件，供解释运行时使用。由于程序一般都不太可能一次编写成功，尤其对于初学者更是如此；因此，当试图编译编写有错误(主要是语法错误)的源程序时，系统将在“命令提示符”对话框中用^符号将可能出错的地方指示出来，并给出适当的提示信息，方便程序员查找错误并改正。下页左上图显示了程序编译失败时的情形。



上图中的出错信息提示我们，方法名 printl 不能被识别，原因是在编辑源程序时，println 方法在录入时漏掉了最后一个字符 n。有些初学者可能会问，“我刚学 Java，怎么知道是 println 还是 printl 呢？”其实这也没什么理由，系统就是这么命名的，大家记住就行了。事实上，学习一门新的编程语言时，语法和一些常用的方法(功能)都是需要适当记忆的。另外，有时候，一个错误可能会引发后续一系列连锁错误，因此当大家在编译程序的过程中出现非常多的错误时，不要灰心，正确的做法是从第一个错误开始，逐个查找并改正，很可能仅修改了几处后，程序就已经编译通过了。

编译成功后，就可以运行 Java 程序了，命令为 java Hello。注意：java 命令和字节码文件名(不含扩展名.class)之间至少要有一个空格符，然后按回车键，如下图所示。

上图显示已成功执行了字节码文件，程序中只有一条 System.out.println()输出语句，输出内容为“Hello,welcome to Java programming.”。

另外，有些初学者还经常碰到这样的情形：上次编译和运行成功的程序，后来再次运行时却失败了，如下图所示。

在上图中，当试图运行 Hello 字节码文件时，运行失败了。细心的读者会发现，这次命令的执行路径是 C:\Documents and Settings\cm，与原来的“F:\工作目录”不一样了。原来的路径保证了可以找到字节码文件，而现在路径不一样了，当然就找不到了，因此系统提示：

Exception in thread "main" java.lang.NoClassDefFoundError: Hello

解决上述问题的办法就是将“工作目录”的路径也添加到 classpath 环境变量中，这样，不管当前路径是什么，都能找到相应的字节码文件。这一点在前面已经提到过，对于初学者来说，务必注意。

至此，可以对 Java 程序的开发步骤做一次简单总结，主要步骤如下：

(1) 下载 JDK 软件并安装。

(2) 配置相应的环境变量(Path 和 ClassPath)。

(3) 编写 Java 源程序(使用文本编辑器或集成开发环境 IDE)。

(4) 编译 Java 源程序，得到字节码文件(javac *.java)。

(5) 执行字节码文件(.java 字节码文件名)。

下面列举一下有助于初学者排除困惑的几个注意事项：

开发 Java 程序时，开发人员必须用到 JDK，而运行或使用 Java 程序时，只需要有

JRE(Java Runtime Environment，Java 运行时环境)即可。一般在安装 JDK 时，JRE 也跟着一起安装了。因此，对于不开发 Java 程序的普通用户来说，只要从网络上下载专门的 JRE 软件并进行安装，即可运行 Java 程序。

编译型语言 C/C++可以将源程序直接编译成操作系统可以识别的可执行文件，不需要经过二次编译。但是对于 Java，第一次编译后生成的是 Java 自己的可执行文件(.class 文件)，在执行时，需要使用 Java 虚拟机读取.class 文件中的代码，一行一行加以解释。

Java 虚拟机可以理解为以字节码为机器指令的虚拟计算机，对于不同的运行平台，有不同的虚拟机。Java 的虚拟机机制屏蔽了底层运行平台之间的差异，真正实现了“一次编译，随处运行”。

1.5 上机练习

目的：掌握 Java 语言的上机环境配置，学会编写简单程序。

内容：按以下步骤进行上机练习。

(1) 安装 JDK 集成环境，安装成功后，配置 Path、ClassPath 环境变量，让用户在任何目录下均可使用 Java 的系统资源。创建工作目录 C:\java，Java 源程序、编译后的字节码文件都将存放在这个目录中。

(2) 在 Windows 中启动记事本。

(3) 用记事本编辑如下源程序。

```
//   HelloWorldApp.java
public class HelloWorldApp{
      public static void main(String args[]){
      System.out.println("Hello World!");
   }
}
```

(4) 保存源程序。需要注意的是，保存源程序时，程序名要与主类名一致。因此，这里使用 HelloWorldApp.java 作为源程序的文件名。记事本默认的扩展名是.txt，因此要给文件名加引号。把源程序保存到目录 C:\java 中。

(5) 编译程序。打开“命令提示符”对话框，键入如下命令：

```
C:\WINDOWS>cd \java <CR>
```

进入源程序所在目录 C:\java。其中<CR>表示回车。

键入如下命令，把 HelloWorldApp.java 编译成字节码文件。

```
C:\JAVA>javac HelloWorldApp.java <CR>
```

如果编译成功，将在 C:\java 目录中生成字节码文件 HelloWorldApp.class。

(6) 运行程序。

进入 HelloWorldApp.class 所在目录 C:\java，键入如下命令就可以运行程序了。

```
C:\JAVA>java HelloWorldApp <CR>
```

(7) 查看程序的运行结果。