新编应用型系列技能丛书

Java 程序设计与实例

主　编：张文胜
副主编：任志宏　赵向梅
　　陈　宏　李　梅　王雅静

清华大学出版社
北　京

内 容 简 介

本书通过大量应用实例介绍了 Java 语言在应用程序开发过程中最常用的一些技术。从实际教学和市场对 Java 人才的需求出发进行编写，知识结构安排合理，由浅入深，循序渐进，通过生动有趣的案例介绍，提高学生的学习兴趣和动手实践能力。

本书可以作为高等学校的本科教材，也可以作为高职高专院校相关课程的教材及 Java 语言自学者的参考用书。

图书在版编目（CIP）数据

Java 程序设计与实例/张文胜主编．—北京：清华大学出版社，2015
（新编应用型系列技能丛书）
ISBN 978-7-302-40805-5

I. ①J…　II. ①张…　III. ①JAVA 语言-程序设计　IV. ①TP312

中国版本图书馆 CIP 数据核字（2015）第 162289 号

责任编辑：苏明芳
封面设计：刘　超
版式设计：刘艳庆
责任校对：王　颖
责任印制：

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
课件下载：http://www.tup.com.cn，010-62788951-223
印 刷 者：
装 订 者：
经　　销：全国新华书店
开　　本：185mm×260mm　印　　张：20.75　字　　数：489 千字
版　　次：2015 年 9 月第 1 版　印　　次：2015 年 9 月第 1 次印刷
印　　数：1～3000
定　　价：39.80 元

产品编号：055948-01

前　言

Foreword

Java 语言自 1995 年 5 月推出以来，历经近二十年的沉浮，在近十几年的世界编程语言排行榜中一直名列前三甲，拥有众多的拥趸，相关的论坛和社区不计其数。Sun 公司（2010 年被 Oracle 公司收购）在推出 Java 时就将其作为开放的技术，要求所设计的 Java 软件必须相互兼容，这一符合软件发展潮流的明智之举，使 Java 不仅“高端大气有深度”，而且“简约时尚国际范”，一路前行，渐次繁荣，到如今的硕果满园，在全球云计算和移动互联网蓬勃发展的产业环境下，Java 更具备了显著优势和广阔前景。

本书从实际教学和市场对 Java 人才的需求出发，合理安排知识结构，由浅入深，循序渐进，通过生动有趣的案例介绍，旨在提高学生的学习兴趣和动手实践能力，缩小高等院校在人才培养和软件公司在人才需求上的差距。

本书具有以下特色。

- 讲述由浅入深，结构清晰。本书内容从学生角度出发，理论联系实际，每个章节除了讲述知识点外，还配有相应案例指导学生实践，以提高学生的实际动手能力。
- 面向高等院校，目标是培养学生的工程应用能力。本书在教学方法上采用案例驱动与综合实训相结合的方式，写作特点是基于任务的认知过程，由案例程序得到基本知识点，再进行知识拓展，并以学生实际动手编写程序来完成一个知识单元的学习。最后一章是一个综合应用，将知识点分散的小案例综合为应用案例，有利于学生把知识点贯穿起来，形成系统性、完整性的项目体系。
- 提供立体化教材，并提供下载教学用课件PPT、课程案例源代码等，方便学生学习。

本书共 12 章，主要内容及各章节要求如下。

第 1 章　初识 Java：要求了解 Java 的发展历史和开发环境。

第 2 章　Java 语法基础：要求掌握 Java 常用的 8 种数据类型、运算符与表达式、if 和 switch 语句、3 种循环语句以及两个流程跳转语句。

第 3 章　类与对象：要求掌握面向对象编程中类最基本的特征，包括类的定义及实例化、Java 访问修饰符、this 关键字、包装类、拆箱和装箱以及封装。

第 4 章　继承与多态：要求掌握面向对象编程的两个重要内容——继承和多态。

第 5 章　常用类库和集合：要求掌握常用类库、集合的使用和泛型的使用。

第 6 章　GUI 编程：要求掌握 GUI 程序设计的基本类及相关技术。

第 7 章　输入/输出流：要求掌握 Java 的输入/输出流技术。

第 8 章　多线程与异常处理：要求掌握 Java 提供的多线程机制和异常处理机制。

第 9 章　JDBC 数据库操作：要求掌握 JDBC 技术基础知识和使用 JDBC 访问数据库

的操作。

第 10 章　Java 网络编程：要求掌握网络编程的基本概念、TCP 协议的网络编程、基于 UDP 协议的网络编程和 UDP 组播技术应用。

Note

第 11 章　反射与类加载器：要求掌握 Java 反射的基本概念和使用、类加载器和动态代理。

第 12 章　综合应用：通过设计一个小球的运动和弹跳的案例，综合掌握图形处理、JFrame 框架、继承机制和图形化界面的应用。

在学时设计上，总量控制为 72 学时，可分为教学 48 学时，实验 24 学时。

本书内容丰富、图文并茂、条理清晰，每个知识点都配有相应的实例，方便学生上机时间学习。本书由西安欧亚学院张文胜主持编写。第 1、3、10、11、12 章和附录由张文胜编写，第 2、4 章由李梅编写，第 5 章由陈宏编写，第 6、8 章由任志宏编写，第 7、9 章由赵向梅编写，张文胜、王雅静进行统一审稿，其中张文胜对全书内容进行了补充和完善，个人完成工作量 8.2 万字。此外，在编写本书的过程中，很多同事给予了很大的帮助，清华大学出版社的苏明芳老师也提出了很多意见，为这本书的出版付出了很多努力，在此，对他们表示衷心的感谢。

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。读者对本书有任何建议，可发送 E-mail 至 zhangwensheng@eurasia.edu。

编　者

目　录

Contents

Note

Note

Note

第 1 章
初识 Java

本章内容

- Java 简介
- Java 开发环境
- 简单的 Java 程序
- Java 程序的基本规则

Java 既是一种高级的面向对象的编程语言，也是一个平台，是由 Sun 公司（即 Sun Microsystems 公司）于 1995 年 5 月推出的 Java 程序设计语言和 Java 平台（即 Java EE、Java ME 和 Java SE）的总称。Java 自面世后非常流行，发展迅速。Java 技术具有卓越的通用性、高效性、平台移植性和安全性，广泛应用于个人 PC、数据中心、游戏控制台、科学超级计算机、移动电话和互联网，同时拥有全球最大的开发者专业社群。在全球云计算和移动互联网的产业环境下，Java 更具备了显著优势和广阔前景。

1.1 Java 简介

1.1.1 Java 的起源

Java 的发展历程充满了传奇色彩。最初，Java 是由 Sun 公司的一个执行“Green 计划”的研究小组开发出来的，该小组最初的目标是想用软件实现对家用电器的集成控制。开始时准备采用 C++语言来开发，但其太复杂，而且安全性差，最后基于该语言开发了一种新的语言 Oak，据说当时是小组成员之一、Java 之父 James Gosling 在苦思冥想为该语言命名时，正好看到了窗外的一棵橡树，Oak 在英文里是“橡树”的意思，所以给该语言命名为 Oak。它是一种用于网络的精巧而安全的语言，但是这个在技术上非常成功的产品在商业上却几近失败，可怜的 Oak 几乎濒临夭折的危险。

Internet 的诞生给 Oak 的发展带来了新的契机。在 Java 出现以前，Internet 上的信息内容都是一些乏味死板的静态文档，人们迫切希望能在静态网页中看到一些交互式的内容，开发人员也希望能够在 Web 上创建一类无须考虑软硬件平台就可以执行的应用程序，当然这些程序还要有极大的安全保障。对于用户的这种要求，传统的编程语言显得无能为力。

Sun 的工程师敏锐地察觉到了这一点，从 1994 年起，他们开始将 Oak 技术应用于 Web 上，并且开发出了 HotJava 浏览器的第一个版本。当 Sun 公司于 1995 年正式以 Java 这个名字推出该语言时，几乎所有的 Web 开发人员都感觉到：噢，这正是大家想要的。那么

Java 的名字又是由何而来呢？据说有一天，几位 Java 成员组的会员正在讨论给这个新的语言取什么名字，当时他们正在咖啡馆喝着 Java 咖啡（Java 是印度尼西亚爪哇岛的英文名称，因盛产咖啡而闻名），有个人灵机一动提议叫 Java，得到了其他成员的赞同，从此一个既好听又好记，具有强大生命力的编程语言 Java 诞生了。

Note

Java 语言中的许多库类名称与咖啡有关，如 JavaBeans（咖啡豆）、NetBeans（网络豆）以及 ObjectBeans（对象豆）等。Java 的标识也正是一杯正冒着热气的咖啡。Java 标志和 Java 的吉祥物 Duke 如图 1-1 所示。

图 1-1　Java 标志和吉祥物

1.1.2　Java 的发展历史

Java 的诞生是对传统计算机模式的挑战，对计算机软件开发和软件产业都产生了深远的影响，开始时，技术界没人认为 Java 会成功，因为它的对手太多而且都是“业界大佬”。尽管如此，经历了无数的风风雨雨后，Java 还是日益繁荣发展起来。让我们来看一看 Java 发展的时间表，回顾 Java 的历史轨迹。Java 历年发行的各版本及其特征如表 1-1 所示。

表 1-1　Java 历年发行版本及其特征

Java 发展纪年	发行的各版本及其特征
1995 年 5 月 23 日	Java 语言诞生
1996 年 1 月	第一个 JDK——JDK 1.0 诞生
1997 年 2 月 18 日	JDK 1.1 发布
1998 年 12 月 8 日	企业平台 J2EE 发布
1999 年 6 月	Sun 公司发布 Java 的 3 个版本：标准版（J2SE）、企业版（J2EE）和微型版（J2ME）
2000 年 5 月 8 日和 5 月 29 日	JDK 1.3 和 JDK 1.4 发布
2001 年 9 月 24 日	J2EE 1.3 发布
2002 年 2 月 26 日	J2SE 1.4 发布，自此 Java 的计算能力有了大幅提升
2004 年 9 月 30 日	J2SE 1.5 发布，成为 Java 语言发展史上的又一里程碑。为了表示该版本的重要性，J2SE 1.5 更名为 Java SE 5.0
2005 年 6 月	Java SE 6 发布，Java 的各种版本已经更名，取消其中的数字 2。J2EE 更名为 Java EE，J2SE 更名为 Java SE，J2ME 更名为 Java ME
2006 年 12 月	Sun 公司发布 JRE 6.0
2009 年 4 月 20 日	Oracle 公司以 74 亿美元收购 Sun 公司取得 Java 的版权；发布 Java EE 6
2011 年 7 月 28 日	Oracle 公司发布 Java SE 7
2014 年 3 月 18 日	Oracle 公司发表 Java SE 8

1.1.3　Java 成功的因素

Note

Java 在计算机行业里所扮演的角色和做出的卓越贡献，使其地位至今无法撼动的因素主要有以下几个方面。

1. 不屈不挠，逆境生存

面对竞争对手，尽管 Java 创造者在 Java 发展中有许多失策之处，但 Java 依然快速壮大，在服务器应用领域如鱼得水、硕果累累，在桌面应用领域满足了基本业务需求。虽然所有技术都需要在逆境湍流中前行，但是 Java 走得更难也更远。事实证明，Java 对许多应用都是一个优良选择。

2. 线程的魔力

Java 虚拟机的强项之一是多线程控制。JVM 针对大型多核机上线程的稳定性进行了优化。

3. Java 是初学者的语言

Java 作为一种编程语言，一开始就推动养成更好的编程习惯，虽然有较多的限制和规则，但是当初学者经过良好的训练后，就可以灵活处理并从中受益。

4. 跨平台兼容性

虽然 Java 并不是第一个提供跨平台兼容能力的语言，但目前 Java 已经成为最受欢迎的（跨平台语言）。

5. 芯片上的成功应用

Java ME 作为精简版的语言，和 VM 一样已经被广泛应用在许多所谓功能手机上，Android 平台都是基于 Java 构建起来的，并且应用 Android 系统的智能手机的销量超过了 iPhone 手机。

6. 蓝光（Blu-ray）标准

蓝光标准是围绕 Java 建立的，任何想在蓝光光碟中添加额外内容的人，必须得到其 javac 编译器版本。

7. Java 虚拟机

Java 虚拟机是按照运行 javac 编译器产生的那些代码的目的来设计和优化的，Java 虚拟机也可以运行其他代码，只要编译器产生标准的 Java 字节码，Java 虚拟机根本不关心使用的是哪种编程语言。

8. 开源

Sun 一直是开源领域中的领导者之一，在 2007 年完成了在 GPL 许可下公开大部分代码的工作，Java 程序员发布了很多开源许可的库和项目。

1.2 Java 开发环境

Note

1.2.1 JDK 开发工具

要开发 Java 应用程序，就要安装 JDK，这是进行开发的重要一步。那么，什么是 JDK？

JDK（Java Development Kit）是针对 Java 开发人员发布的免费软件开发工具包。没有 JDK，就无法编译 Java 程序，如果只想运行 Java 程序，要确保已安装相应的 JRE（Java Runtime Environment，Java 运行环境）。

本书使用的 JDK 是 1.8.0 版本，操作平台是 Windows 7/XP SP3 32 位，可在 Oracle 公司的网站（http://www.oracle.com/）上下载最新的安装程序。JDK1.8 的下载地址为 http://www.oracle.com/technetwork/java/javase/downloads/index.html。下载界面如图 1-2 所示。

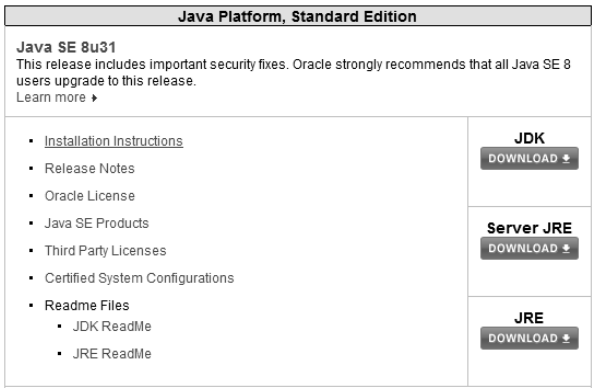
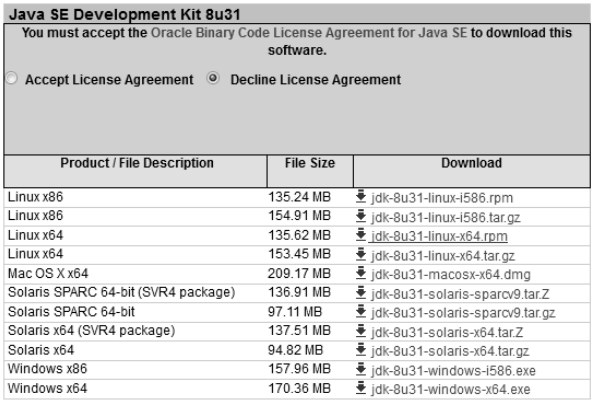


图 1-2 JDK 下载界面

单击 JDK DOWNLOAD 按钮，进入选择下载界面，如图 1-3 所示。根据用户的操作系统选择对应的 JDK，本书选用 jdk-8u31-windows-i586.exe，然后选中 Accept License Agreement 单选按钮，就可以下载了。



Product / File Description	File Size	Download
Linux x86	135.24 MB	jdk-8u31-linux-i586.rpm
Linux x86	154.91 MB	jdk-8u31-linux-i586.tar.gz
Linux x64	135.62 MB	jdk-8u31-linux-x64.rpm
Linux x64	153.45 MB	jdk-8u31-linux-x64.tar.gz
Mac OS X x64	209.17 MB	jdk-8u31-macosx-x64.dmg
Solaris SPARC 64-bit (SVR4 package)	136.91 MB	jdk-8u31-solaris-sparcv9.tar.Z
Solaris SPARC 64-bit	97.11 MB	jdk-8u31-solaris-sparcv9.tar.gz
Solaris x64 (SVR4 package)	137.51 MB	jdk-8u31-solaris-x64.tar.Z
Solaris x64	94.82 MB	jdk-8u31-solaris-x64.tar.gz
Windows x86	157.96 MB	jdk-8u31-windows-i586.exe
Windows x64	170.36 MB	jdk-8u31-windows-x64.exe

图 1-3 JDK 选择下载界面

小提示

如果是 Windows 64 位系统，最好选用 jdk-8u31-windows-x64.exe。

Note

在 Windows 平台上安装 JDK 的步骤如下。

（1）运行安装文件 jdk-8u31-windows-i586.exe，会弹出安装向导窗口，如图 1-4 所示。单击“下一步”按钮进入安装界面。

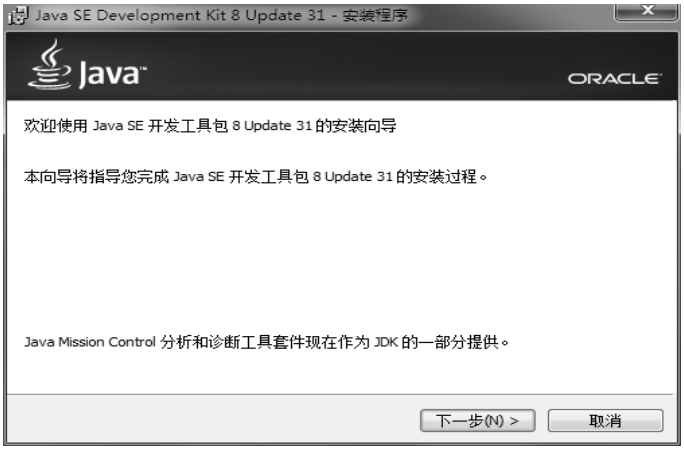


图 1-4　安装

（2）选择安装路径及安装内容，如图 1-5 所示。为了程序配置运行方便，如果需要改变安装路径，可以单击“更改”按钮，根据习惯重新配置安装路径，本书存放位置为 D:\Java\jdk1.8.0_31\。单击“下一步”按钮开始执行安装程序。

图 1-5　选择安装路径及安装内容

安装过程中，会询问 JRE 的安装路径，更改为和 JDK 安装在同一文件夹下，即 D:\Java\jre1.8.0_31，如图 1-6 所示。

安装成功后，出现如图 1-7 所示界面。

图 1-6　选择 JRE 安装路径

图 1-7　成功安装界面

Note

安装完 JDK 后，需要正确配置 Java 运行时必需的环境变量值，即 JAVA_HOME、PATH 和 CLASSPATH。无论是什么操作系统，都能够支持多个 JDK 版本的共存。大家可以根据应用程序的实际需求对不同的 JDK 版本进行管理。

正确管理 JDK 版本的方式，是在 JVM（Java 虚拟机）运行时指定 JDK 版本对应的环境变量。为了方便，我们往往为操作系统本身指定一个系统级别的环境变量。例如，Windows XP 操作系统的系统环境变量可以在“系统属性”的“高级”选项卡中找到，并在其中配置 JAVA_HOME、PATH 和 CLASSPATH 的值，步骤如下。

（1）右击“计算机”，在弹出的快捷菜单中选择“属性”命令，弹出“系统属性”对话框，选择“高级”选项卡，单击“环境变量”按钮，如图 1-8 所示。

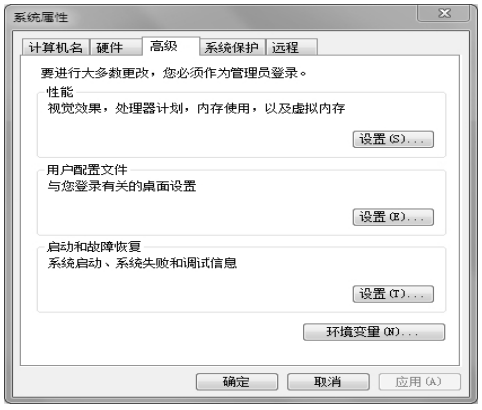


图 1-8　环境变量配置

（2）在弹出的环境变量对话框中，设置 3 项属性，即 JAVA_HOME、PATH 和 CLASSPATH（不区分大小写）。在没安装过 JDK 的环境下，PATH 属性是本来存在的，而 JAVA_HOME 和 CLASSPATH 是不存在的。若已存在则单击“编辑”按钮，若不存在则单击“新建”按钮。

① JAVA_HOME 的值指明 JDK 安装路径，如 JAVA_HOME= D:\Java\jdk1.8.0_31，如图 1-9 所示。

设置 JAVA_HOME 是为了方便引用，避免每次引用都输入很长的路径串，有时当 JDK

路径被迫改变的时候，仅需更改 JAVA_HOME 的变量值即可，有些第三方软件会引用约定好的 JAVA_HOME 变量。

② PATH 要添加的值为 PATH=%JAVA_HOME%\bin;，如图 1-10 所示。

图 1-9　JAVA_HOME 配置

图 1-10　PATH 配置

注意

配置 PATH 变量时，系统原来已有的值不要改，在值后面添加即可，用分号“;”隔开。如原来的值为 C:\WINDOWS，修改后为 C:\WINDOWS;%JAVA_HOME%\bin。

设置好 PATH 后就可以在任何路径下使用 Java 来执行命令了，当在命令提示符窗口输入代码时，操作系统会在当前目录和 PATH 变量目录里查找相应的应用程序并且执行。

③ CLASSPATH 要添加的值为 CLASSPATH=.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar，如图 1-11 所示。

注意

CLASSPATH 要添加的值前面要加“.”表示当前路径。CLASSPATH 变量设置的目的是为了程序能找到相应的.class 文件。

小提示

JDK 1.4 以前的版本，需要配置 CLASSPATH 环境变量，JDK 1.5 以后的版本不用配置 CLASSPATH 环境变量，也可正常编译和运行 Java 程序。

（3）测试安装是否成功。

单击 Windows 的“开始”菜单，选择“开始”→“运行”命令，弹出“运行”对话框，输入 cmd，按 Enter 键运行；或者按  （即键盘上有 Windows 窗口的键）+R 键，也会弹出“运行”对话框，输入 cmd，单击“确定”按钮即可。

在控制台下输入 java -version 和 javac -version 命令，当前 JDK 的运行版本会显示出来，如图 1-12 所示，说明环境变量配置成功。

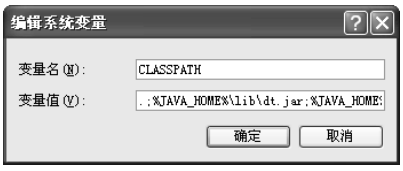


图 1-11　CLASSPATH 配置

图 1-12　JDK 配置成功

1.2.2 集成开发环境 Eclipse

Note

对 Java 开发者来说，得心应手的集成开发环境（Integrated Development Environment，IDE）令人如虎添翼、功力大增。所谓 IDE，就是把编写、编译、调试、运行集成在一个统一开发环境中的软件，并且还增加了许多提高开发效率的实用功能，如代码自动提示、自动编译、设置断点逐步调试、在 IDE 内部显示运行结果等。徒手开发好比刀耕火种，用 IDE 开发可谓进入了蒸汽时代。

Eclipse 是目前最受欢迎的跨平台的 Java 自由集成开发环境（IDE）。Eclipse 最初是由 IBM 公司开发的替代商业软件 Visual Age for Java 的下一代 IDE 开发环境，2001 年 11 月贡献给开源社区，现在由非营利软件供应商联盟 Eclipse 基金会（Eclipse Foundation）管理。2001 年 11 月 7 日，Eclipse 1.0 发布。目前已知的 Eclipse 各版本代号如下。

- Eclipse 3.1 版本代号 IO“木卫一，伊奥”。
- Eclipse 3.2 版本代号 Callisto“木卫四，卡里斯托”。
- Eclipse 3.3 版本代号 Eruopa“木卫二，欧罗巴”。
- Eclipse 3.4 版本代号 Ganymede“木卫三，盖尼米德”。
- Eclipse 3.5 版本代号 Galileo“伽利略”。
- Eclipse 3.6 版本代号 Helios“太阳神”。
- Eclipse 3.7 版本代号 Indigo“靛青”。
- Eclipse 4.2 版本代号 Juno“朱诺”。
- Eclipse 4.3 版本代号 Kepler“开普勒”。
- Eclipse 4.4 版本代号 Luna“卢娜，月神”。

在安装 Eclipse 之前，必须安装好 JRE 或 JDK（本身附有 JRE），Eclipse 基本上只要有 JRE 就可以运行。

进入 Eclipse 官方网页（网址为 http://www.eclipse.org/downloads/）选择能编写和执行 Java 程序的版本，图 1-13 所示的 3 个版本皆可。根据操作系统，选择 Windows 32 Bit，单击下载。本书使用的版本是 eclipse-jee-indigo-SR2-win32。

图 1-13　Eclipse 下载界面

小提示

SR 表示修正版或更新版，修正了正式版推出后发现的 Bug。

Note

Eclipse 是免安装软件，只需把 eclipse-jee-indigo-SR2-win32.zip 解压到指定目录下，运行 eclipse.exe 文件，出现 Workspace Launcher 对话框，提示选择 Workspace 的位置，Eclipse 会以 Workspace 为单位来管理工程，如图 1-14 所示。

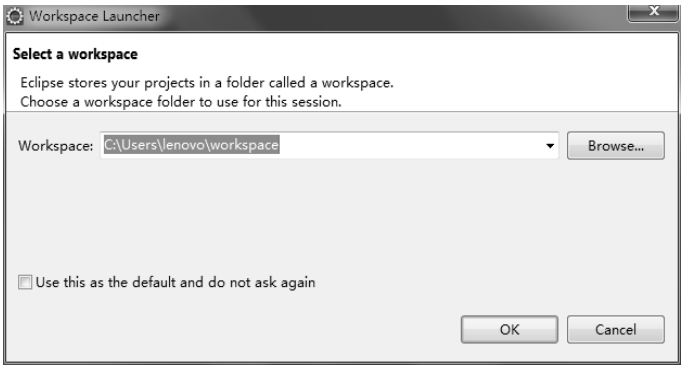


图 1-14　选择 Workspace 的位置

设置好 Workspace 后，出现 Welcome 主界面，单击右上方的 Workbench，进入 Eclipse 的 Workbench 界面。Workbench 是多个窗口的集合，每个窗口包含菜单栏、工具栏、状态栏，以及一个或者多个透视图，如图 1-15 所示。可以打开菜单栏的 Window 菜单，选择 Customize Perspective 命令，对开发视图进行编辑。

图 1-15　Workbench 界面

小提示

Eclipse 可以依据用户的习惯和喜好定制其视图和编辑器的布局。初学者常会不小心将布局弄乱，因此不知所措。其实，恢复初始界面的方法很简单：打开菜单栏的 Window 菜单，选择 Reset Perspective 命令即可恢复默认视图。

Note

1.3 简单的 Java 程序

1.3.1 实践案例一：创建 Hello World 程序

“Hello World!”程序是一个只在计算机屏幕上打印出“Hello World!”字符串的计算机程序，该程序通常是计算机程序设计语言初学者所要学习编写的第一个程序，它可以用来确定开发语言的编译器、程序开发环境以及运行环境已经安装完成。将“Hello World!”程序作为第一个编写的程序，已经成为一个传统。

用 Eclipse 开发 Java 项目分为以下几步。

（1）新建 Java 项目。

① 要在 Eclipse 中编写 Java 应用程序，首先要创建 Java 项目。选择 Eclipse 菜单的 File→New 命令，选择项目 Project，在弹出的窗口中，选择 Java Project 选项，单击 Next 按钮，如图 1-16 所示。

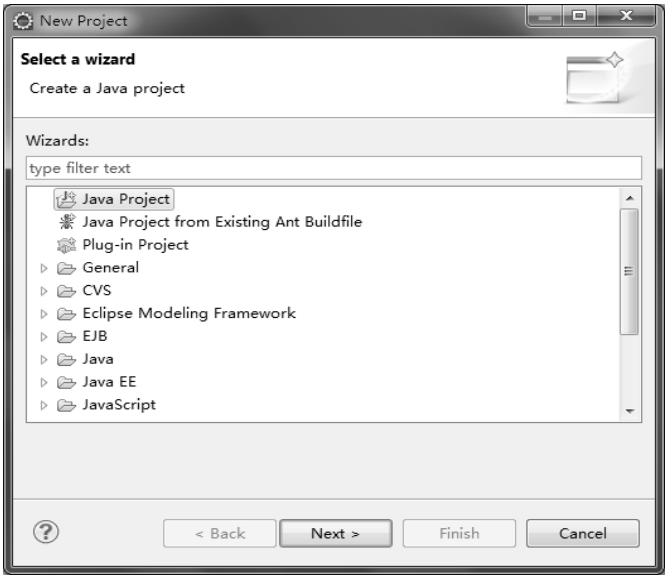


图 1-16 选择 Java Project 选项

② 在打开的 New Java Project 窗口中，填写项目名字 Ch1，不需要进行其他设置，直接单击 Finish 按钮，如图 1-17 所示。

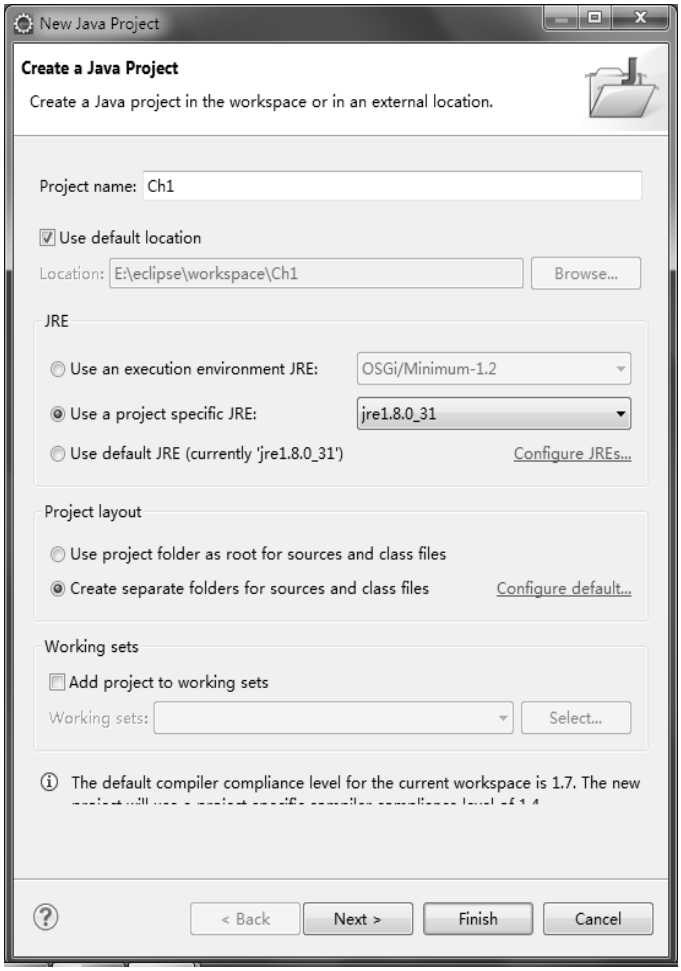


图 1-17　创建 Java 项目

当系统弹出 Open Associated Perspective 对话框时，单击 Yes 按钮。这是 Eclipse 为了方便开发项目定制了一些视图，如图 1-18 所示。

图 1-18　Eclipse 定制的视图

（2）新建 HelloWorld 类。

① 在 Eclipse 左侧 Project Explore 窗口中，选择工程 Ch1，右击，在弹出的快捷菜单

中选择 New→Class 命令，出现如图 1-19 所示的窗口。在 Package 文本框中填写包名 duke.example.ch1，Name 文本框中输入类名 HelloWorld，并且选中 public static void main(String[] args)和 Generate comments 复选框。然后单击 Finish 按钮。

Note

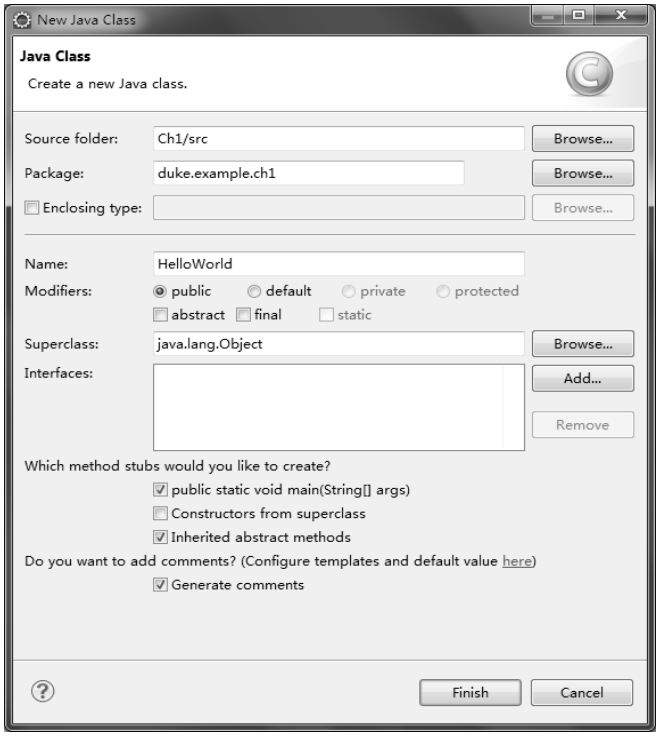


图 1-19　Eclipse 开发向导

② 在 Eclipse 中间的代码编辑区域，程序框架代码已经自动生成，这就是 Eclipse 的代码生成（Code Generation）特性。

（3）添加打印语句。

在自动生成的程序框架代码中，只需在 main()函数中添加如下代码。

```
1.  /**
2.  * 我的第一个 Java 程序
3.  */
4.  package duke.example.ch1;
5.  /**
6.  * @author Duke
7.  *
8.  */
9.  public class HelloWorld {
10.     public static void main(String[] args) {
11.             System.out.println("Hello World!");          //手工添加代码
12.     }
13. }
```

小提示

在 main()函数中，输入 syso，然后按 Alt+/键，则自动生成 System.out.println()打印函数。同理，如果忘记选 main()函数，只需输入 main，然后按 Alt+/键，则自动生成 main()函数。Eclipse 常用快捷键见附录 C。

Note

（4）运行 Java 程序。

现在直接在 Eclipse 中运行该程序，看看执行结果。选择菜单 Run→Run 命令，Eclipse 会弹出 Save and Launch 运行设置向导，单击 OK 按钮，即可在 Eclipse 下的 Console 中看到结果，如图 1-20 所示。

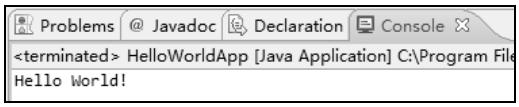


图 1-20　程序运行结果

1.3.2　实例分析

本节向读者介绍应用程序的构成，让读者了解 Java 应用程序文件的组成。展开 Package Explorer 视图窗口的 Ch1 项目，对应的程序文件结构如图 1-21 所示。

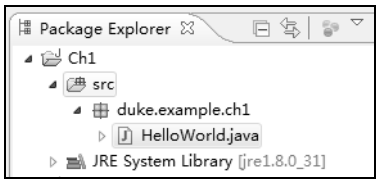


图 1-21　程序文件结构图

下面对每个文件夹及文件的作用进行介绍。

- Ch1：工程名或项目名，项目将映射到文件系统的一个目录结构中。
- src：用于存放应用程序的源代码。在图 1-21 中，duke.example.ch1 为应用程序包，HelloWorld.java 为应用程序的源代码。
- JRE System Library：用于存放工程运行所需的 JDK 1.8.0 的基础 jar 包的名字和所在路径。

在 Eclipse 的编辑器中是应用程序的源代码，文件名为 HelloWorld.java，见 1.3.1 节，下面对代码进行详细分析。

- 第 1～3 行：多行注释，包括起始符号“/*”、终止符号“*/”，以及之间的文字“我的第一个 Java 程序”，为注释内容，不被执行。
- 第 4 行：应用程序包名，定义为 duke.example.ch1，解决类的同名问题，由小写字母组成。
- 第 5～8 行：多行注释，显示作者等信息。
- 第 9 行：定义一个类，名字为 HelloWorld，public 修饰符表示这个类可以被公开访问。
- 第 10 行：定义了一个 main()方法，这是 Java 应用程序的入口方法，当运行 HelloWorld 应用时，Java 虚拟机（JVM）将从 HelloWorld 类中的 main()方法中的程序代码开始运行。

- 第 11 行：System.out 是指标准输出，println 是一个方法，作用是把字符串 Hello World!打印到标准输出区（也叫输出控制台）。单行注释，以符号“//”开始，到本行结束的所有字符均作为注释而被编译器忽略。

Note

小提示

Eclipse 编辑器中设置代码行号的方法是，按 Ctrl+F10 键，打开视图菜单（或在编辑器 Editor 的最左边右击，打开视图菜单），然后选中 Show Line Numbers 即可。

1.4 Java 程序的基本规则

1.4.1 程序的组织形式

Java 语言是面向对象的程序设计语言，Java 程序的基本组成单元是类（class），类体中又可包括属性与方法两部分。一个 Java 应用（Application）由一个或者多个 class 文件组成，其中一个类定义了 main()方法，该类称之为主类（在 Java 语言中没有主类、次类之分，这里之所以把带有 main()方法的类称为主类，是为了表达和理解的方便），即该类可以独立运行。可以在任何提供了 Java 解释器的环境中运行 Java Application，如集成开发环境 Eclipse。

1.4.2 实践案例二：创建完整的 Java 应用

按照案例一的步骤在已建立的 Java 项目 Ch1 和其包 duke.example.ch1 下，创建主类 MajorClass，以及次要类 SubClass1 和 SubClass2。对应的项目源代码如下。

```
1.  package duke.example.ch1;
2.
3.  public class MajorClass {                                   //定义一个主类
4.  public static void main(String[] args) {                    //定义主程序
5.              System.out.println("主类 main()方法运行");
6.              SubClass1.subMethod();
7.              SubClass2.subMethod();
8.  }                                                           //主程序结束
9.  }                                                           //主类结束
10.
11. class SubClass1 {                                          //定义一个次要类
12. public static void subMethod() {                           //定义方法
13.             System.out.println("次要类 1 的方法运行");
14.         }                                                   //方法结束
15. }                                                           //次要类结束
16.
17. class SubClass2 {                                          //定义一个次要类
```

```
18.    public static void subMethod() {                    //定义方法
19.            System.out.println("次要类 2 的方法运行");
20.        }                                                //方法结束
21.    }                                                    //次要类结束
```

在 Eclipse 下面的 Console 中看到结果如图 1-22 所示。

图 1-22　程序运行结果

1.4.3　实例分析包

展开 Package Explorer 视图窗口的 Ch1 项目，对应的程序文件结构如图 1-23 所示。

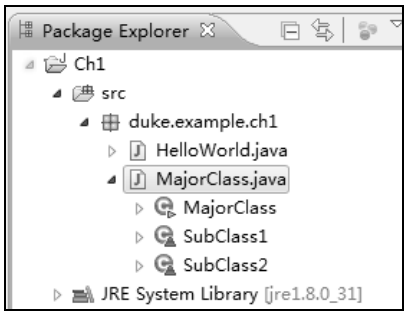


图 1-23　程序文件结构图

下面对每个文件夹及文件的作用进行介绍。

- Ch1：工程名或项目名，项目将映射到文件系统的一个目录结构中。
- src：用于存放应用程序的源代码。在图 1-23 中，duke.example.ch1 为应用程序包，MajorClass.java 为应用程序的源代码，MajorClass 为该源代码文件中的主类，SubClass1 和 SubClass2 为次要类。
- JRE System Library：用于存放工程运行所需的 JDK 1.8.0 的基础 jar 包的名字和所在路径。

在 Eclipse 的编辑器中是应用程序的源代码，文件名为 MajorClass.java，见 1.4.2 节，下面详细说明。

- 第 1 行：应用程序包名，定义为 duke.example.ch1。Java 编译器为每个类生成一个字节码文件，且文件名与类名相同。这就会带来一个问题：同名的类会发生冲突。包（package）便可管理类命名空间。一般来说，具有相同功能的类放在一个包中。包名为全小写的名词，中间可用点分隔开。

Note

- 第 3 行：定义一个主类，名字为 MajorClass，public 修饰符表示这个类可以被公开访问，且类名称必须与文件名称完全一致。注意，一个 Java 程序中至多只能有一个 public 类，也可以没有任何 public 类。
- 第 4 行：定义主类的 main()方法。main()方法是 Java 应用程序的入口方法，也就是说，程序在运行的时候，第一个执行的方法就是 main()方法，这个方法和其他的方法有很大的不同，比如方法的名字必须是 main，方法必须是 public static void 类型，方法必须接收一个字符串数组的参数等。
- 第 5 行：控制台输出字符串“主类 main()方法运行”。
- 第 6 行：执行次要类 SubClass1 的 subMethod()方法。
- 第 7 行：执行次要类 SubClass2 的 subMethod()方法。
- 第 11 行：定义一个次要类 SubClass1。在一个 Java 的源文件中，只能有一个 public class 的声明，但是允许有多个 class 的声明（一个 Java 程序中有几个 class 就会被编译成几个 class 文件），因此 SubClass1 类的声明不能为 public。
- 第 12 行：定义一个次要类 SubClass1 的静态方法 subMethod()，就是在控制台输出字符串“次要类 1 的方法运行”。
- 第 17～21 行：定义一个次要类 SubClass2 和其静态方法 subMethod()。

总结：一个 Java 源文件中最多只能有一个 public 类，当有一个 public 类时，源文件名必须与之一致，否则无法编译，如果源文件中没有一个 public 类，则文件名与类中没有一致性要求。Main()方法作为一个特殊的规范，与普通的方法有很大区别，限制很多，理解其原理需要学习 JVM 相关知识。

1.5 知识拓展——Java 虚拟机

1.5.1 什么是 JVM

JVM 是 Java Virtual Machine（Java 虚拟机）的缩写，Java 之所以能够崛起，JVM 功不可没。Java 虚拟机最初服务于让 Java 语言凌驾于平台之上，实现“编写一次，到处运行”的跨平台特性。而随着时间的推移，JVM 经过不同公司和团体以不同方式的实现，逐渐有更多除 Java 以外的语言登上了 JVM 这艘“船”。

JVM 其实是一种用于计算设备的规范，它是一个虚构出来的计算机，是通过在实际的计算机上仿真模拟各种计算机功能来实现的。Java 虚拟机包括一套字节码指令集、一组寄存器、一个栈、一个垃圾回收堆和一个存储方法域。JVM 屏蔽了与具体操作系统平台相关的信息，使 Java 程序只需生成在 Java 虚拟机上运行的目标代码（字节码），就可以在多种平台上不加修改地运行。JVM 在执行字节码时，实际上最终还是把字节码解释成具体平台上的机器指令来执行。编译虚拟机的指令集与编译微处理器的指令集非常类似。

1.5.2 查看字节码

Java 不同于一般的编译语言和直译语言。它首先将源代码编译成字节码，然后依赖各

种不同平台上的虚拟机来解释执行字节码，从而实现了“一次编写，到处运行”的跨平台特性。理解字节码以及 Java 编译器如何生成 Java 字节码，与学习汇编知识对于 C/C++程序员的意义一样。

Note

所谓字节码就是.class 文件，Java 文件编译后就会生成.class 文件，即字节码文件。如 1.3.1 节案例一中的 HelloWorld.class。在 workspace\Ch1\bin\duke\example\ch1 目录下可以找到这个.class 文件，可是我们并没有编译，怎么就生成了.class 文件？原来，Eclipse 在将源文件存盘时，自动帮我们编译好了。如果不使用 IDE 工具，就要自己手工编译源代码了。

要查看字节码文件，如果使用普通的文本编辑器，打开是乱码。可以使用专用工具，如可视化工具——字节码阅读器 jclasslib。

通过 jclasslib 查看 HelloWorld.class 字节码的界面如图 1-24 所示。

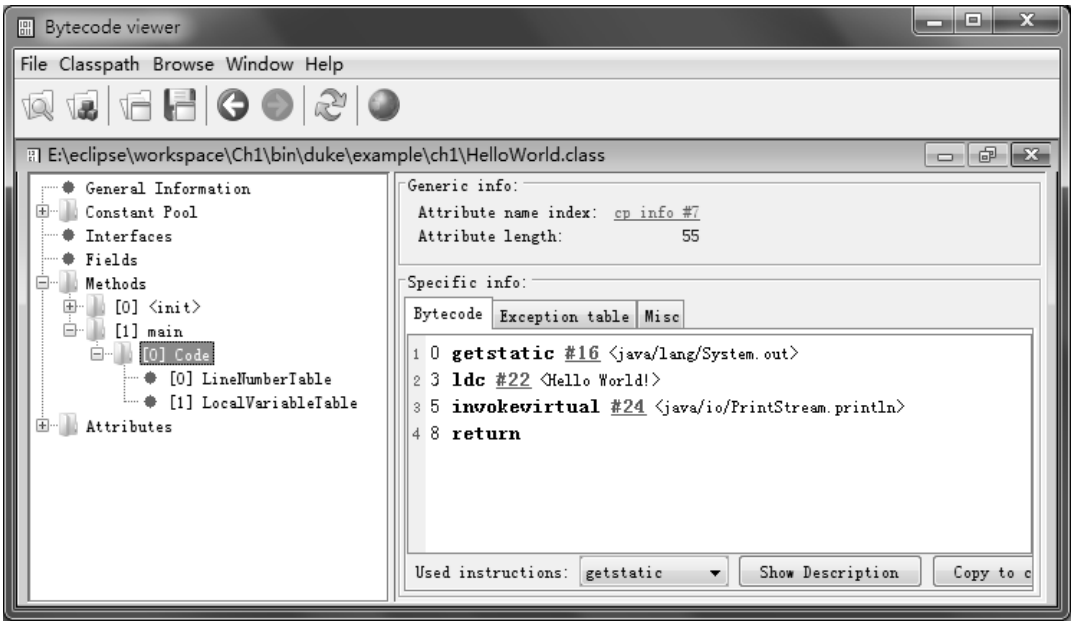


图 1-24　查看 HelloWorld.class 的字节码

1.5.3　JRE 和 JVM 的区别

JRE（Java Runtime Environment，Java 运行环境），也就是 Java 平台。所有的 Java 程序都要在 JRE 下才能运行。JDK 的工具也是 Java 程序，也需要 JRE 才能运行。为了保持 JDK 的独立性和完整性，在 JDK 的安装过程中，JRE 也是安装的一部分。所以，在 JDK 的安装目录下有一个名为 jre 的目录，专门用于存放 JRE 文件。

JVM 是 JRE 的一部分。它是一个虚构出来的计算机，是通过在实际的计算机上仿真模拟各种计算机功能来实现的。JVM 有自己完善的硬件架构，如处理器、堆栈、寄存器等，还具有相应的指令系统。Java 语言最重要的特点就是跨平台运行。使用 JVM 就是为了支持与操作系统无关性，实现跨平台。

1.6 想一想、练一练

Note

1.6.1 练习题

1. 填空题

（1）Java 既是一种高级的_____编程语言，也是一个平台。

（2）Java 不同于一般的编译语言和直译语言。它首先将源代码编译成字节码，然后依赖各种不同平台上的虚拟机来解释执行字节码，从而实现了_____的跨平台特性。

（3）Java 编写好的程序首先由编译器转化为_____，然后由 Java 虚拟机解释执行。

2. 选择题（单选）

（1）在 Java 中，负责对字节代码解释执行的是（　　）。

A. 应用服务器　　B. 虚拟机　　C. 垃圾回收器　　D. 编译器

（2）关于 public，下列哪一个表述是错误的？（　　）

A. 一个程序里只能有一个类被修饰为 public

B. 源文件名必须和用 public 修饰的类名相同

C. 若程序中没有任何 public 类，则文件名可任取

D. 一个 Java 程序中，必须有一个被 public 修饰的类

（3）编译 Java Application 源程序文件将产生相应的字节码文件，这些字节码文件的扩展名为（　　）。

A. java　　B. class　　C. html　　D. exe

（4）main()方法是 Java Application 程序执行的入口点，关于 main()方法的方法头以下哪项是合法的？（　　）

A. public static void main()

B. public static void main(String[] args)

C. public static int main(String[] arg)

D. public void main(String arg[])

（5）Java Application 源程序的主类是指包含有哪个方法的类？（　　）

A. main()方法　　B. toString()方法

C. init()方法　　D. actionPerfromed()方法

1.6.2 上机练习

1. 编写一个 Java 程序，显示自己的个人小档案。例如，在控制台输出如下信息。

姓名：黄小米

年龄：22 岁

性别：女

芳邻：李大头

爱好：睡觉，吃饭，打豆豆

2. 编写一个 Java 程序，要求由用户输入两个参数（如 Tom 和 Jerry），用“&”符号连接这两个参数并输出。

Note

提示：

（1）main()方法的参数为 String[] args，args 是用来接收参数的，args[0]存放第一个参数，args[1]存放第二个参数，依此类推。控制台输出第一个参数作为 System.out.println(args[0])。

（2）Eclipse 设置参数的方法：右击代码，在弹出的快捷菜单中选择 Run As→Run configurations→(x)=arguments 命令。然后在 Program arguments 里输入参数即可，参数之间用空格分隔开，如图 1-25 所示。

图 1-25　Eclipse 设置参数

（3）控制台输出，如图 1-26 所示。

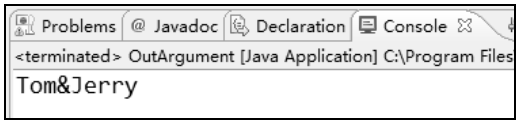


图 1-26　运行结果

第 2 章

Java 语法基础

本章内容

- 标识符与关键字
- 基本数据类型
- 数组
- 运算符与表达式
- 语句

本章介绍 Java 常用的 8 种数据类型，运算符与表达式，实现选择结构的 if 和 switch 语句，3 种循环语句 while、do…while、for，以及两个流程跳转语句 break 和 continue。对于 Java 语言中的基本数据类型和语句，在学习时需要熟记它们的作用，然后在实际项目中根据需要来实现程序的功能。

2.1 标识符与关键字

1. 标识符

（1）何为标识符

标识符就是一个名字，用来标识类名、变量名、方法名、接口名、类型名、数组名和文件名等的有效字符序列。

（2）标识符的命名规则

Java 语言的标识符由字母、数字、下划线和美元符号（$）组成，并且第一个字符不能为数字。

例如，Hello_java、Hello_11$和$123word 都是合法标识符；9world 和 consumer#都是非法标识符。

- Java 语言使用 Unicode 标准字符集，最多可以识别 65535 个字符。因此，Java 语言中的字母可以是 Unicode 字符集中的任何字符，包括拉丁字母、汉字、日文和其他许多语言中的字符。如“中国好声音”是合法标识符。
- 标识符不能是 Java 的关键字和保留字。非法的标识符如 this、goto。
- 在 Java 语言中标识符是区分大小写的，如果两个标识符的字母相同但是大小写不同，就是不同的标识符。

2. 关键字

关键字（keyword），也称保留字（reserved word），是指程序代码中规定用途的单词。

也就是说，只要在程序代码内部出现该单词，编译程序就认为是某种固定的用途。Java 关键字见附录 A。

Note

2.2　基本数据类型

Java 语言有 8 种基本数据类型，分别是 boolean、byte、short、int、long、float、double 和 char。

这 8 种基本数据类型按照用途可分为以下 4 大类型。

- 整数类型：byte、short、int、long。
- 字符类型：char。
- 浮点类型：float、double。
- 逻辑类型：boolean。

2.2.1　整数类型

整数类型用来存储整数数值，可以是正数、负数，也可以是 0。根据所占内存的大小不同，可以分为 byte、short、int 和 long 4 种类型，它们所占的内存和取值范围如表 2-1 所示。

表 2-1　整数类型所占的内存和取值范围

数 据 类 型	所占字节数	取 值 范 围
byte	1 个字节	−128～127，即-2^7～2^7-1
short	2 个字节	−32768～32767，即-2^{15}～$2^{15}-1$
int	4 个字节	-2^{31}～$2^{31}-1$
long	8 个字节	-2^{63}～$2^{63}-1$

1. byte 型

使用关键字 byte 来定义 byte 型变量，如 byte x=−12, tom=28, 漂亮=98;。byte 型是整型中所分配的内存空间最少的，只分配 1 个字节；取值范围也是最小的，为−128～127，使用时一定要注意，以免数据溢出产生错误。

2. short 型

short 型即短整型，使用 short 关键字来定义 short 型变量。系统给 short 型分配两个字节的内存，取值范围也比 byte 型大了很多，为−32768～32767。

3. int 型

int 型即整型，使用 int 关键字来定义 int 型变量。int 型变量取值范围很大，为−2147483648～2147483647，能满足一般需求，所以是整型变量中应用最广泛的。

4. long 型

long 型即长整型，使用 long 关键字来定义 long 型变量。long 是有符号的 64 位类型，

它的范围相当大，这使得大的、整个数字都被需要时，其非常有用。

注意

在对 long 型变量赋值时结尾必须加上 L 或者 l，否则将不被认为是 long 型。

Note

2.2.2 字符类型

char 型即字符类型，使用 char 关键字进行声明，用于存储单个字符，系统分配两个字节的存储空间。

在 Java 语言中，字符常量是用单引号括起来的单个字符，例如'A'，'B'，'6'，'#'，'a'，'b'，'+'，'*'等。

Java 语言还使用一种特殊形式的字符常量，即以反斜杠“\”开头，后跟一个或多个字符，具有特定的含义，不再执行原字符的功能，因而称为转义字符。例如'\n'就是一个转义字符，表示“回车换行”。Java 中转义字符如表 2-2 所示。

表 2-2 转义字符及其作用

转义字符	含义	ASCII 代码（十进制）
\n	换行，将当前位置移到下一行开头	10
\t	水平制表（跳到下一个 Tab 位置）	9
\b	退格，将当前位置移到前一格	8
\r	回车，将当前位置移到本行开头	13
\\	反斜杠字符“\”	92
\'	单引号字符	39
\"	双引号字符	34
\ddd	1～3 位八进制数 ddd 代表的字符	ddd （八进制）
\xhh	1～2 位十六进制数 hh 代表的字符	hh（十六进制）

说明

同 C 语言和 C++语言一样，Java 语言使用 Unicode 字符集作为默认的字符集，该字符集包含各国语言中常见的字符。

2.2.3 浮点类型

Java 语言中浮点类型分为单精度浮点类型（float）和双精度浮点类型（double）。它们具有不同的取值范围，如表 2-3 所示。

表 2-3 浮点类型取值范围

类型	内存空间（8 位等于 1 字节）	有效数字	数值范围
float	32	7～8	10^{-38}～10^{38}
double	64	15～16	10^{-308}～10^{308}

浮点型常量有小数和指数两种表现形式。需要特别注意的是 float 型常量后面必须带后缀 F 或者 f；double 型常量后面可以带后缀 D 或者 d，也可以省略。

2.2.4　逻辑类型

Note

逻辑类型只有两个值 true 和 false，分别代表布尔逻辑中的“真”和“假”。使用 boolean 关键字声明布尔类型变量，通常被用在流程控制中作为判断条件。

2.2.5　类型转换运算

Java 语言是一种强类型的语言。强类型的语言有以下几个要求。

- 变量或常量必须有类型。要求声明变量或常量时必须声明类型，而且只能在声明以后才能使用。
- 赋值时类型必须一致。值的类型必须和变量或常量的类型完全一致。
- 运算时类型必须一致。参与运算的数据类型必须一致才能运算。

但是在实际使用中，经常需要在不同类型的值之间进行操作，例如，要将 String 类型数据 123 转换为一个 int 整型变量 123，这就需要一种新的语法来适应这种需要，这种语法就是数据类型转换。

在数值处理这部分，计算机和现实生活中的逻辑不太一样，在现实生活中，1 和 1.0 没有什么区别，但是对于计算机来说，1 是整数类型，而 1.0 是小数类型，其在内存中的存储方式以及占用的空间都不一样，所以类型转换在计算机内部是必须的。

Java 对数据类型的转换有严格的规定：数据从占用存储空间较小的类型转换为占用存储空间较大的数据类型时，做自动类型转换；反之则必须做强制类型转换。

1. 自动类型转换

Java 中的 8 种基本类型可以进行混合运算，在运算过程中，对于不同类型的数据编译器首先自动完成类型转换，然后再进行计算。

自动类型转换要遵循一定的转换规则，从存储范围小的类型转换到存储范围大的类型。

具体规则为：byte→short(char)→int→long→float→double。也就是说 byte 类型的变量可以自动转换为 short 类型，示例代码如下。

```
byte b = 10;
short sh = b;
```

2. 强制类型转换

强制性数据类型转换一般格式如下。

```
（类型说明符）（表达式）
```

功能：将表达式转换成类型说明符指定的类型。

例如：

- (unsigned)i*3：在赋值时先将 i 的值转换成无符号型整数后，再乘以 3。
- (double)b：将 b 转换成 double 型。
- (int)(x+y)：将 x+y 的值转换成 int 型。
- (float)(5%3)：将 5%3 的值转换成 float 型。

Note

注意

（1）表达式要用括号括起来。

（2）由于基本数据类型中 boolean 类型不是数字型，所以基本数据类型的转换是除了 boolean 类型以外的其他 7 种类型之间的转换。

（3）小数强制转换为整数，采用的是“去 1 法”，也就是无条件地舍弃小数点的所有数字。例如：

```
double d = 3.10;int n = (int)d;
```

n 的值应为 3。

（4）整数强制转换为整数时取数字的低位，例如 int 类型的变量转换为 byte 类型时，则只取 int 类型的低 8 位(也就是最后一个字节)的值。

例如：

```
int n = 123;
byte b = (byte)n;
int m = 1234;
byte b1 = (byte)m;
```

则 b 的值还是 123，而 b1 的值为-46。b1 的计算方法如下：m 的值转换为二进制是 10011010010，取该数字低 8 位的值作为 b1 的值，则 b1 的二进制值是 11010010，按照机器数的规定，最高位是符号位，1 代表负数，在计算机中负数存储的是补码，则该负数的原码是 10101110，该值就是十进制的-46。

2.3 数　　组

在程序中，如果需要存储一个数值，则可以在代码中声明一个变量来进行存储，但有时为了程序操作方便，需要将一组相关的数值存储在一起，这就是数组出现的最初原因。

数组的作用就是存储一组相关的数据，从存储数值的角度考虑，其作用是和变量等价的。

实际使用时，数组名称是一个整体，类似学校里的班级名称。为了能够方便地访问数组中某个具体的值，可以对数组中的值进行强制编号，这个编号称作数组的下标，类似班级中每个学生的学号。在实际引用数组中的值时，使用数组名称和下标一起进行指定，类似于某班级学号为 *n* 的学生。

为了数组管理的方便，在语法上要求数组中存放的每个元素类型必须相同。数组中的每个具体的数值也称作数组元素。

在内存中，数组占用一块连续的内存区域，因为数组中每个元素的类型相同，则占用

的内存大小也一致，所以在访问数组中的元素时可以直接根据数组在内存中的起始位置以及下标来计算元素的位置，因此数组的访问速度很高。

在 Java 语言中，初始化数组时，必须指定数组的长度，而且一旦指定，长度就不能改变，除非再重新初始化该数组。

虽然数组从结构上来看，只是把以前语法中的多个变量存储在一起，通过数组名称组合上下标的方式进行使用，但这个简单的变化，极大地简化了程序算法的实现，所以说数组是对数据存储方式的很大革新。数组的另外一个变革就是下标可以使用变量表示，这样访问数组的值时会更加灵活，这一点也是理解数组的关键。

Note

数组的使用应注意以下几点。

- 数组中的元素类型必须相同。
- 数组的长度一旦指定即不能改变。
- 数组中的值通过数组名和下标组合起来进行访问。

2.3.1　一维数组

一维数组实质上是一组相同类型数据的集合，当在程序中碰到需要处理一组数据，或者传递一组数据时，可以使用一维数组来存储数据。

1. 一维数组的声明

声明一维数组有下列两种格式。

```
数组的元素类型        数组名[ ];
数组的元素类型[ ]     数组名;
```

例如：

```
int    array[];
char   flower[];
```

数组 array 可以存放 int 型数据，数组 flower 可以存放 char 型数据。

数组的元素类型可以是 Java 的任何一种类型。假如用户已经定义了一种 Student 类型数据，那么用户可以声明下面一个数组：

```
Student   stu[];
```

数组 stu 就可以存放 Student 类型的数据。

注意

与 C/C++不同，Java 不允许在声明数组中的方括号内指定数组元素的个数，若声明为以下两种格式，则会导致语法错误。

```
int array[10];
int[10]    array;
```

Note

2. 创建一维数组

声明数组后，还不能访问它的任何元素，因为声明数组仅仅是给出了数组名字和元素的类型，要想真正使用数组还要为其分配内存空间，即创建数组。为数组分配的内存单元称为数组的元素，分配内存空间时必须指明数组的长度。

为数组分配内存空间的格式如下。

```
数组名 = new 数组元素类型[数组元素的个数];
```

例如：

```
array = new int[5];                                    //array 数组每个元素默认值是 0
```

为数组分配内存空间后，数组 array 获得 5 个用来存放 int 型数据的内存空间。数组变量 array 中存放着这些内存单元的首地址，该地址称作数组的引用，这样数组就可以通过下标运算操作这些内存单元。

创建数组和分配内存不一定要分开执行，可以在创建数组时直接为变量进行赋值。语法如下。

```
数组元素类型  数组名[] = new 数组元素类型[数组元素的个数];
```

例如：

```
int array[] = new int[10];
```

3. 初始化一维数组

数组可以与基本数据类型一样进行初始化操作。数组的初始化可分别初始化数组中的每个元素。数组的初始化有以下两种形式。

```
int  array[] = new int[]{1,2,3,4,5};
```

或

```
int  array[] = {1,2,3,4,5};
```

4. 一维数组元素的引用

【例 2-1】使用 for 循环对数组元素赋值，然后再循环输出数组中的元素。

```
1. package duke.example.ch2;
2.
3. public class TraverseArray {
4.       public static void main(String[] args) {
5.              int[] array = new int[5];
6.              for (int i = 0; i < array.length; i++) {
7.                     array[i] = i;
8.              }
```

```
9.              System.out.println("数组中的元素：");
10.             for (int i : array) {
11.                 System.out.print(i + "    ");
12.             }
13.     }
14. }
```

Note

代码说明如下。

- 第 5 行：定义一维数组对象 array，初始化为默认值，int 型为 0。
- 第 6～8 行：使用 for 循环对数组元素赋值，对于一维数组，数组名.length 表示数组中元素的个数。
- 第 10～12 行：循环输出数组中的元素值，程序中 for 语句的语法如下。

```
for(声明循环变量：数组的名字)
{
    语句;
}
```

注意

（1）声明循环变量必须是变量声明，不能使用已经声明过的变量。
（2）作用：可理解为对于循环变量依次取数组的每一个元素的值。

【例 2-2】实现冒泡排序算法。

冒泡排序是一种简单的交换类排序。其基本思路是，从头开始扫描待排序的元素，在扫描过程中依次对相邻元素进行比较，将关键字值大的元素后移。每经过一趟排序后，关键字值最大的元素将移到末尾，此时记下该元素的位置，下一趟排序只需要比较到此位置为止，直到所有元素都已有序排列。

一般地，对 n 个元素进行冒泡排序，总共需要进行 n−1 趟。第 1 趟需要比较 n−1 次，第 2 趟需要比较 n−2 次……第 i 趟需要比较 n−i 次。

```
1. package duke.example.ch2;
2.
3. public class BubbleSort {
4. public static void main(String[] args) {
5.              int[] array = { 32, 25, 82, 5, 36 };
6.              for (int i = 0; i < array.length - 1; i++)
7.                      for (int j = 0; j < array.length - 1 - i; j++) {
8.                              if (array[j] > array[j + 1]) {
9.                                      int temp = array[j];
10.                                     array[j] = array[j + 1];
11.                                     array[j + 1] = temp;
12.                             }
13.                     }
14.             for (int i : array) {
```

```
15.                 System.out.print(i + " ");
16.             }
17.     }
18. }
```


代码说明如下。

- 第 5 行：定义一维数组 array，初始化给定值为 32, 25, 82, 5, 36。
- 第 6～12 行：n 个元素的数组进行 n−1 轮排序。
- 第 7 行：因为每一轮循环将确定一个数组元素的位置，因此每一轮的比较次数将会递减。
- 第 8～12 行：如果第 j 个元素比后面的相邻的元素大就交换。
- 第 14～16 行：排序后打印数组中的元素。

2.3.2 二维数组

在学校里，由于一个班的人数不多，所以按照顺序编号即可，当人数增多时，对于学校的学生来讲，编号时就要增加层次，如某班某学号。在部队中也一样，某师某团某营某连某排某班，这里的层次就比较深。为了管理数据的方便，一般要加深管理的层次，一层就代表一维。二维数组有两个层次，每一层对应一个下标。

二维数组常用于表示表，表中的信息以行和列的形式组织，第一个下标代表元素所在的行，第二个下标代表元素所在的列。

1. 创建二维数组

（1）二维数组的声明

声明二维数组有下列两种格式。

```
数组的元素类型       数组名[ ][ ];
数组的元素类型[ ][ ]   数组名;
```

例如：

```
int   map[][];
```

（2）创建二维数组

二维数组和一维数组一样，在声明之后必须用 new 运算符分配内存空间，例如：

```
int   myarr[][];
myarr=new int[3][4];
```

或者

```
int   myarr[][]=new int[3][4];
```

Java 采用“数组的数组”定义多维数组，一个二维数组是由若干个一维数组组成的，如上述创建的二维数组 myarr 就是由 3 个长度为 4 的一维数组 myarr[0]、myarr[1]和 myarr[2]构成。

注意

和 C 语言不同，Java 允许使用 int 型变量指定数组的元素个数，例如：

```
int    size=10;
double    a[]=new double[size];
```

Note

2. 二维数组的初始化

数组可以与基本数据类型一样进行初始化操作。数组的初始化可分别初始化数组中的每个元素。数组的初始化有以下两种形式。

```
int    array[][] = new int[]{1,2,3,4,5};
```

或

```
int    array[] = {1,2,3,4,5};
```

3. 二维数组举例

【例 2-3】根据数组中的值，在对应位置绘制指定的字符。规定 0 绘制空格，1 绘制星号（*），输出菱形。数组各元素的值如图 2-1 所示。

0	0	0	1	0	0	0
0	0	1	0	1	0	0
0	1	0	0	0	1	0
1	0	0	0	0	0	1
0	1	0	0	0	1	0
0	0	1	0	1	0	0
0	0	0	1	0	0	0

图 2-1　数组各元素值

```
1.    package duke.example.ch2;
2.
3.    public class OutDiamond {
4.        public static void main(String[] args) {
5.            int[][] map = { { 0, 0, 0, 1, 0, 0, 0 }, { 0, 0, 1, 0, 1, 0, 0 },
6.                    { 0, 1, 0, 0, 0, 1, 0 }, { 1, 0, 0, 0, 0, 0, 1 },
7.                    { 0, 1, 0, 0, 0, 1, 0 }, { 0, 0, 1, 0, 1, 0, 0 },
8.                    { 0, 0, 0, 1, 0, 0, 0 } };
9.            for (int row = 0; row < map.length; row++) {
10.               for (int col = 0; col < map[row].length; col++) {
11.                   switch (map[row][col]) {
12.                   case 0:                                    //判断数组中的值
13.                       System.out.print(' ');                 //输出空格
14.                       break;
```

```
15.                         case 1:                                    //判断数组中的值
16.                             System.out.print('*');                 //输出星号
17.                             break;
18.                         }
19.                     }
20.                     System.out.println();                          //换行
21.                 }
22.             }
23.         }
```

代码说明如下。

- 第 5～8 行：定义二维数组 map，并给定初始化值。
- 第 9～21 行：循环数组中的元素，判断数组中的值，根据值绘制对应的字符。

类似的代码在游戏开发中可以用来代表游戏中的地图数据，或者俄罗斯方块等益智游戏中地图块的值。

2.4 运算符与表达式

Java 提供了丰富的运算符，如算术运算符、关系运算符、逻辑运算符、位运算符等。本节将介绍这些运算符。

1. 算术运算符

Java 中的算术运算符主要有+（加）、-（减）、*（乘）、/（除）、%（求余），它们都是二元运算符。

优先级：*、/、% 高于 +、-。

结合性：自左向右。

2. 自增、自减运算符

自增、自减运算符是单目运算符，可以放在操作元之前，也可以放在操作元之后。操作元必须是一个整型或浮点型变量。其中，放在操作元前面的自增、自减运算符，会先将变量的值加 1（减 1），然后再使该变量参与表达式的运算；放在操作元后面的自增、自减运算符，会先使变量参与表达式的运算，然后再将该变量加 1（减 1），如下所示。

```
int   a=3;
int   b,c;
b=a++;              //先将 a 的值 3 赋给 b，再将 a 进行加 1 操作，a 的值变成 4
a=3;
c=++a;              //先将 a 进行加 1 操作，再将 a 的值 4 赋给 c，c 的值为 4
```

3. 关系运算符

关系运算符用来比较两个值的关系。关系运算符的运算结果是 boolean 型，当运算符

的对应关系成立时，运算结果是 true，否则是 false。关系运算符通常用在条件语句中作为判断的依据。

Java 中的关系运算符有>、>=、<、<= 、==、!=。其优先级及结合性如表 2-4 所示。

Note

表 2-4　关系运算符

运　算　符	优　先　级	用　　法	含　　义	结 合 方 向
>	6	op1>op2	大于	自左到右
>=	6	op1>=op2	大于等于	
<	6	op1<op2	小于	
<=	6	op1<=op2	小于等于	
==	7	op1==op2	等于	
!=	7	op1!=op2	不等于	

通过关系运算符形成的表达式称为关系表达式，如 x>20。

4. 逻辑运算符

逻辑运算符包括&&、||、!。其中&&、||为二目运算符，实现逻辑与、逻辑或；!为单目运算符，实现逻辑非。逻辑运算符的操作元必须是 boolean 型数据，逻辑运算符可以用来连接关系表达式。

逻辑运算符的用法和含义如表 2-5 所示。

表 2-5　逻辑运算符

运　算　符	优　先　级	用　　法	含　　义	结 合 方 向
&&	11	op1&&op2	逻辑与	自右到左
||	12	op1||op2	逻辑或	
!	2	!op	逻辑非	

通过逻辑运算符将操作元连接起来的式子称为逻辑表达式。逻辑运算的规则如表 2-6 所示。

表 2-6　用逻辑运算符进行逻辑运算

op1	op2	op1&&op2	op1||op2	!op1
true	true	true	true	false 自右到左
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

例如，3>8&&9>3 的结果为 false，3>8||9>3 的结果为 true。由于关系运算符的级别高于&&和||。所以，3>8&&9>3 相当于(3>8)&&(9>3)。

逻辑运算符&&和||，也称作短路逻辑运算符，这是因为当 op1 的值是 false 时，&&运算符在进行运算时不再去计算 op2 的值，直接就得出 op1&&op2 的结果是 false；当 op1 的值是 true 时，||运算符在进行运算时不再去计算 op2 的值，直接得出 op1||op2 的结果是 true。

Note

5. 赋值运算符

赋值运算符即“=”，是一个双目运算符，其功能是将右方操作数所含的值赋值给左方的操作数，语法如下。

```
变量名=表达式;
```

左方必须是一个变量，而右边所赋的值可以是任何数值或表达式，包括变量、常量或有效的表达式。

6. 位运算符

位运算符用于处理整型和字符型的操作数，对其内存进行操作。数据在内存中以二进制的形式存放，所以 Java 语言提供了直接操作二进制的运算符，即位运算。Java 语言中的位运算符主要有 4 种：&（位与）、|（位或）、^（异或）和～（按位取反），如表 2-7 所示。

表 2-7　位运算符

运　算　符	含　　义	用　　法	运 算 分 类
～	按位取反	～op1	按位运算
&	按位与	op1&op2	
|	按位或	op1|op2	
^	按位异或	op1^op2	
<<	左移	op1<<p2	移位运算
>>	右移	op1>>op2	
>>>	无符号右移	op1>>>op2	

（1）按位与运算

按位与运算符为“&”，是双目运算符。其运算规则是：如果两个操作数对应位都是 1，则结果位才是 1，否则为 0。如果两个操作数的精度不同，则结果的精度与精度高的操作数相同。例如：

a:　00000000　00000000　00000000　00000111

b:　10000001　10100101　11110011　10101011

a &b:　00000000　00000000　00000000　00000011

（2）按位或运算

按位或运算的运算符为“|”，是双目运算符。其运算规则是：如果两个操作数对应位都是 0，则结果位才是 0，否则为 1。如果两个操作数的精度不同，则结果的精度与精度高的操作数相同。例如：

a:　00000000　00000000　00000000　00000111

b:　10000001　10100101　11110011　10101011

a|b:　10000001　10100101　11110011　10101011

（3）按位非运算

按位非运算也称按位取反，运算符为“～”，是单目运算符，其运算规则是：将操作数对应二进制数中每位的 0 变成 1，1 变成 0。

Note

a:　00000000　00000000　00000000　00000111

~a:　11111111　11111111　11111111　11111000

（4）按位异或运算

按位异或运算的运算符是“ ^ ”，是双目运算符，其运算规则是：当操作数的二进制表示相同（同时为 0 或同时为 1）时，结果为 0，否则为 1。若两个操作数的精度不同，则结果数的精度与精度高的操作数相同。

a:　00000000　00000000　00000000　00000111

b:　10000001　10100101　11110011　10101011

a ^ b:　10000001　10100101　11110011　10101000

7. 移位运算符

Java 语言中的移位运算符有 3 种，其操作的数据类型只有 byte、short、char、int 和 long 5 种，3 种移位运算符如下。

（1）左移运算符（<<）

左移运算符，就是将左边的操作数在内存中的二进制数据左移右边操作数指定的位数，左边移空的部分补 0。

例如，48<<1，移位后的结果是 96。

（2）右移运算符（>>）

右移运算符，就是将左边的操作数在内存中的二进制数据右移右边操作数指定的位数，如果最高位是 0，左移空的位就填入 0；如果最高位是 1，右移空的位就填入 1。

例如，48>>1，移位后的结果是 24。移位过程如下。

48 对应的二进制数：　00000000 00000000 00000000 00110000

右移一位后的二进制数：　00000000 00000000 00000000 00011000

（3）无符号右移运算符（>>>）

无符号右移运算符，就是按二进制形式把所有的数字向右移动对应位数，低位移出(舍弃)，高位的空位补零。

例如：01100010 >>> 2，移位后的结果是:00011000。

2.5　语　　句

流程就是指程序执行的顺序，流程控制就是指通过控制程序执行的顺序实现要求的功能。流程控制对于任何一门编程语言来说都是至关重要的，它提供了控制程序步骤的基本

手段，是程序中语法和逻辑的结合。

2.5.1 分支语句

Note

条件分支语句指当指定表达式取不同的值时,运行的程序也发生相应的分支变化。这类语句在实际使用中，难点在于如何准确地定义条件。例如，实现程序登录功能时，如果用户名和密码正确，则进入系统，否则弹出“密码错误”这样的提示框。

在 Java 语言中，条件语句主要有 if 语句和 switch 语句。

1. if 语句

语法格式如下。

```
    if(条件表达式){
    语句序列;
}
```

语法说明如下。

（1）条件表达式：必须写出来，要求该表达式结果为 boolean 类型。它可以是一个布尔变量或常量，或者使用关系或布尔运算符的表达式。

（2）语句序列：可以是一条或多条语句，当表达式的值为 true 时执行这些语句。如语句序列中仅有一条语句，则可以省略条件语句中的大括号。

（3）if 语句的执行流程为：如果条件表达式成立，则执行其后的语句序列；如果条件表达式不成立，则不执行其后的语句序列，如图 2-2 所示。

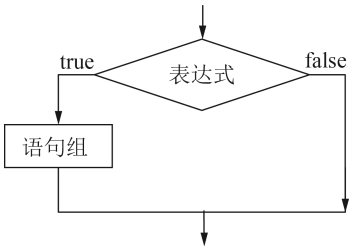


图 2-2 单分支选择结构

【例 2-4】条件判断语句。

```
1.    package duke.example.ch2;
2.
3.    public class IfSentence {
4.        public static void main(String[] args) {
5.            int a = 10;
6.            if (a >= 0)                          //判断变量 a 的值是否大于等于 0
7.                System.out.println("a 是正数");
8.            if (a % 2 == 0)                      //判断变量 a 是否为偶数
9.                System.out.println("a 是偶数");
10.       }
11.   }
```

代码说明如下。

- 第 6～7 行：第一个条件是判断变量 a 的值是否大于等于 0，如果该条件成立则输出相应内容。

❑　第 8～9 行：第二个条件是判断变量 a 是否为偶数，如果成立也输出相应内容。

2. if…else 语句

if…else 语句的一般格式如下。

```
if(表达式)        {  语句组 1  }
else              {  语句组 2  }
```

功能：如果表达式的值为真，则执行语句组 1，否则执行语句组 2。其流程图如图 2-3 所示。

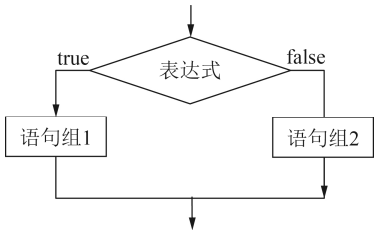


图 2-3　双分支选择结构

例如：

```
if（x>y）max=x;
else      max=y;
```

说明

在双分支语句中，else 必须与 if 配对使用，构成 if…else 语句，实现双分支选择。如果 else 空缺，即构成单分支结构。

【例 2-5】输入 3 个整型数据，求出其中的最大数和最小数。

```
1.    package duke.example.ch2;
2.
3.    public class IfElseSentence {
4.    public static void main(String[] args) {
5.            int a = 3, b = 3, c = 7, max, min;
6.            if (a > b)
7.            {
8.                max = a;
9.                min = b;
10.           } else {
11.               max = b;
12.               min = a;
13.           }
14.           if (max < c)
15.               max = c;
16.           if (min > c)
```

```
17.                 min = c;
18.             System.out.println("max = " + max);
19.             System.out.println("min = " + min);
20.     }
21.     }
```

Note

代码说明如下。

- 第 6～13 行：a 和 b 比较，max 取大数，min 取小数。
- 第 14～17 行：再和 c 比较，max 取大数，min 取小数。

3. if…else if 多分支选择语句

if…else if 多分支语句用于针对某事件的多种情况进行处理。语法如下：

```
if(表达式 1)          {语句序列 1}
else if(表达式 2)     {语句序列 2}
else if(表达式 3)     {语句序列 3}
…
else if(表达式 n)     {语句序列 n}
else                  {语句序列 n+1}
```

说明

（1）else if 是 else 和 if 两个关键字，中间使用空格隔开。

（2）表达式 1～n 都是 boolean 类型。

（3）else if 语句可以有任意多个。

（4）最后的 else 语句为可选。

（5）每个 else if 语句在书写时是有顺序的，在实际书写时，必须按照逻辑上的顺序进行书写，否则将出现逻辑错误。

功能：逐级判断表达式的值，以确定执行哪一组语句。若表达式 1 的值为 true 时，则执行语句序列 1，否则再判断表达式 2；若表达式 2 的值为 true 时，则执行语句序列 2，否则再判断表达式 3，依此类推。其执行过程如图 2-4 所示。

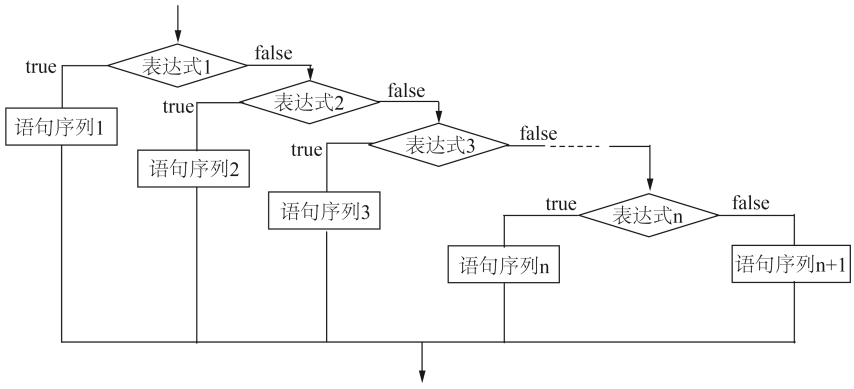


图 2-4　多分支选择结构

Note

【例 2-6】根据给定的学生成绩，将百分制的成绩转换为 A、B、C、D 和 E。

```
1.    package duke.example.ch2;
2.
3.    public class IfElseifSentence {
4.    public static void main(String[] args) {
5.                int score = 85;
6.                if (score >= 90) {
7.                      System.out.println('A');
8.                } else if (score >= 80) {
9.                      System.out.println('B');
10.               } else if (score >= 70) {
11.                     System.out.println('C');
12.               } else if (score >= 60) {
13.                     System.out.println('D');
14.               } else {
15.                     System.out.println('E');
16.               }
17.   }
18.   }
```

代码说明如下。

- 第 5 行：声明 int 型变量 score，并赋值 85。
- 第 8～16 行：else if 分别是 else 和 if 两个关键字，用于条件判断，else if 语句可以有任意多句，且最后的 else 为可选项，可有可无。

4. switch 语句

在编程中常见的一个问题就是检测一个变量是否符合某个条件。这种问题使用 if 语句也可以完成，但是较为烦琐。在 Java 中可以用 switch 语句以一个简单明了的方式来实现“多选一”的选择。语法如下：

```
switch(表达式)
{
case 常量表达式 1: 语句组 1
case 常量表达式 2: 语句组 2
…
case 常量表达式 n: 语句组 n
default: 语句组 n+1
}
```

功能：计算表达式的值，逐个与 case 中常量表达式的值比较，二者相等时，从该 case 标号后的语句组开始执行，执行完后不再比较，顺序执行其后所有语句组，直到结束。若表达式的值与所有 case 中常量表达式的值均不相等时，执行 default 标号后面的语句组。

说明

（1）表达式的数据类型除了 byte、short、char 和 int 这 4 种之外，还可以接受 String 和 enum 类型的参数。

（2）case 后面各常量表达式的值不能相同，否则会出错。case 后面允许有多个语句，可以不用“{}”括起来。

（3）default 子句可以省略不用。

（4）执行完一个 case 标号后的语句组后，流程转移到下一个 case 标号后的语句组继续执行。如果要求仅执行一个 case 标号后的语句组，可用 break 语句跳出 switch 结构，即：

```
switch(表达式)
{
case 常量表达式 1:      语句组 1;
                              break;
case 常量表达式 2:      语句组 2;
                              break;
…
case 常量表达式 n:      语句组 n;
                              break;
default:                      语句组 n+1;
}
```

Note

break 语句虽然可以独立使用，但通常主要用于 switch 语句中，控制程序的执行流程转移。在 switch 语句中，break 语句的作用是强制退出 switch 结构，执行 switch 结构之后的语句。其本质就是在单层循环结构体系中，强制退出循环结构。

【例 2-7】使用 switch 语句输出当前星期所对应的英语表达。

```
1.   package duke.example.ch2;
2.
3.   import java.text.*;
4.   import java.util.Date;
5.
6.   public class SwitchSentence {
7.       public static String getWeek(Date date) {
8.           // SimpleDateFormat 传入的参数：EEEE 代表星期，如“星期四”
9.           DateFormat f  week = new SimpleDateFormat("EEEE");
10.          return f  week.format(date).toString();
11.      }
12.
13.      public static void main(String[] args) {
14.          String today = "星期二";
15.          Date date = new Date();
16.          today = getWeek(date);
17.
18.          switch (today) {//判断执行哪一条语句，表达式的类型为 String
```

Note

```
19.                 case "星期一":
20.                     System.out.println("Today is Monday! ");
21.                     break;
22.                 case "星期二":
23.                     System.out.println("Today is Tuesday! ");
24.                     break;
25.                 case "星期三":
26.                     System.out.println("Today is Wednesday!");
27.                     break;
28.                     case "星期四":
29.                     System.out.println("Today is Thursday! ");
30.                     break;
31.                 case "星期五":
32.                     System.out.println("Today is Friday! ");
33.                     break;
34.                 case "星期六":
35.                     System.out.println("Today is Saturday! ");
36.                     break;
37.                 case "星期日":
38.                     System.out.println("Today is Sunday! ");
39.                     break;
40.                 default:
41.                     System.out.println("请输入合法的字符串! ");
42.                 }
43.             }
44.     }
```

代码说明如下。

- 第 7 行：声明获取日期中的星期的函数，根据日期取得星期几。
- 第 9 行：DateFormat 会根据计算机上的区域设定显示时间格式，EEEE 表示星期，MM 表示月份，dd 表示日期，而 yyyy 是公历年份，每个参数的设定都各有其意义。
- 第 18～42 行：在 switch（变量）这一行里，变量可以是整型、字符型、字符串和枚举类型，本例中是字符串类型。程序先读出这个字符串变量的值，然后在各个 case 里查找哪个值和这个变量相等，如果相等，就算条件成立，程序执行相应的分支，直到遇上 break 或者 switch 语句结束。

switch、case、break 和 default 的意思分别是开关，情况，中断，默认（值）。用一句话来说就是：根据开关值的不同，执行不同的情况，直到遇上中断；如果所有的情况都不符合开关值，那么就执行默认的分支。

2.5.2　循环语句

循环语句就是在满足一定条件的情况下反复执行某些操作。Java 中提供了 3 种常用的循环语句，分别是 while 语句、do…while 语句和 for 语句。

Note

1. while 语句

while 语句通过一个条件控制是否要继续反复执行某些语句，语法如下：

```
while(条件表达式){
        语句;
}
```

执行流程：在执行 while 语句时，首先判断循环条件，如果循环条件为 false，则直接执行 while 语句后续的代码，如果循环条件为 true，则执行循环体代码，然后再判断循环条件，一直到循环条件不成立为止。

【例 2-8】求 1～100 所有整数的和。

```
1.    package duke.example.ch2;
2.
3.    public class WhileSentence {
4.
5.    public static void main(String[] args) {
6.              int i = 1;
7.              int sum = 0;
8.              while (i <= 100) {
9.                    sum += i;
10.                   i++;
11.             }
12.             System.out.println("1+2+3+…+100=" + sum);
13.       }
14.   }
```

代码说明如下。

第 8～11 行：对于第一次循环，while 是先判断再执行大括号内的循环体。while 循环如果条件不成立，可能导致语句一次都不执行。

2. do…while 语句

do…while 语句与 while 语句类似，二者之间的区别是 while 语句先判断条件是否成立，再执行循环体。而 do…while 语句则先执行一次循环后，再判断条件是否成立。也就是说 do…while 循环体内的语句至少要被执行一次。语法如下：

```
do{
    语句;
}while(条件表达式);
```

【例 2-9】求 1～100 所有整数的和。

```
1.    package duke.example.ch2;
2.
3.     public class DoWhileSentence {
4.
```

```
5.      public static void main(String[] args) {
6.              int i = 1;
7.              int sum = 0;
8.              do {
9.                      sum += i;
10.                     i++;
11.             } while (i <= 100);
12.             System.out.println("1+2+3+…+100=" + sum);
13.         }
14.     }
```

Note

代码说明如下。

第 8～11 行：对于第一次循环，先执行大括号内的循环体，再判断条件。do…while 循环中，即使条件不成立，语句至少也会执行一次。其实在实际程序开发中，不常使用 do…while 循环语句。因为这种语句是先执行循环体再检测条件，所以会有一些危险数据不经检测就会被执行。建议使用 while 语句或者 for 循环语句来编写代码。

3. for 语句

for 循环是 Java 程序设计中最常用的循环语句之一。一个 for 循环可以用来重复执行某条语句，直到某个条件得到满足。语法如下：

```
for(表达式 1;表达式 2;表达式 3){
    语句序列
}
```

说明

表达式 1：初始化表达式，负责完成变量的初始化。

表达式 2：循环条件表达式，值为布尔类型，指定循环条件。

表达式 3：循环变量变化表达式，负责修改循环变量。

功能：首先执行表达式 1，给循环控制变量赋初值；然后判断表达式 2，如果表达式 2 的值为 true，即循环条件成立，则执行循环体语句；循环体语句执行结束后，计算表达式 3，修改循环控制变量。接着开始第二次循环，计算表达式 2，若值为 true，则继续循环，否则退出循环，如图 2-5 所示。

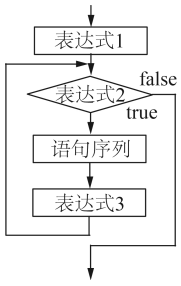


图 2-5　for 循环流程

【例 2-10】求 1～100 所有整数的和。

```
1.  package duke.example.ch2;
2.
3.  public class ForSentence {
4.
5.  public static void main(String[] args) {
6.          int i = 1;
7.          int sum = 0;
8.          for (i = 1; i <= 100; i++) {    //初始化表达式；判断表达式；递增表达式
9.              sum += i;
10.         }
11.         System.out.println("1+2+3+…+100=" + sum);
12.
13.     }
14. }
```

Note

代码说明如下。

第 8～10 行代码中 for 循环是专为那种多次执行的情况而设定的，也是 Java 中最常用的语法之一。定义一个 int 类型变量 i 来记录循环次数，每次判断(i<=100)之后使 sum+i，i 再自加一次。

【例 2-11】“水仙花数”指三位数中，每个数字的立方和和自身相等的数字，如 370，$3^3 + 7^3 + 0^3 = 370$，请输出所有的水仙花数。

```
1.  package duke.example.ch2;
2.
3.  public class ForFlower {
4.
5.  public static void main(String[] args) {
6.          int i, a, b, c;
7.          for (i = 100; i < 1000; i++) {
8.              a = i % 10;
9.              b = (i / 10) % 10;
10.             c = i / 100;
11.             if (a * a * a + b * b * b + c * c * c == i) {
12.                 System.out.println(i);
13.             }
14.         }
15.
16.     }
17.  }
```

代码说明如下。

- 第 7 行：for 循环所有三位数。
- 第 8 行：拆分出三位数字的个位数字。
- 第 9 行：拆分出三位数字的十位数字。