魏魏交大　百年书香
www.jiaodapress.com.cn
bookinfo@sjtu.edu.cn

策划编辑　杨　洋
责任编辑　胡思佳
封面设计　张瑞阳

Java面向对象
程序设计实战教程

扫描二维码
关注上海交通大学出版社
官方微信

ISBN 978-7-313-30033-1
9 787313 300331 >
定价：45.00元

河北省『十四五』职业教育规划教材

Java面向对象程序设计实战教程

主编　张　昕　胡丽霞

上海交通大学出版社

河北省“十四五”职业教育规划教材

Java面向对象
程序设计实战教程

Java MIANXIANG DUIXIANG CHENGXU SHEJI SHIZHAN JIAOCHENG

主编　张　昕　胡丽霞

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

Java面向对象程序设计实战教程

Java MIANXIANG DUIXIANG CHENGXU SHEJI SHIZHAN JIAOCHENG

主　编　张　昕　胡丽霞
副主编　黄丙根　佟　超
参　编　张　娜　刘丹丹　刘文锋

上海交通大学出版社
SHANGHAI JIAO TONG UNIVERSITY PRESS

内容提要

本书共9个项目，包括Java程序设计概述、Java语言基础、流程控制与数组、面向对象、异常、Java集合、JDBC数据库编程、流与文件和多线程。每个项目按照“课前知识学习—课中学习任务—课后任务提升—程序人生”的框架结构编排。

本书既适合作为高等职业院校Java程序设计课程的教材，也可作为相关爱好者的参考用书。

图书在版编目(CIP)数据

Java面向对象程序设计实战教程 / 张昕，胡丽霞主编. — 上海 ：上海交通大学出版社，2023.12

ISBN 978-7-313-30033-1

Ⅰ. ①J… Ⅱ. ①张… ②胡… Ⅲ. ①JAVA语言—程序设计—高等职业教育—教材 Ⅳ. ①TP312.8

中国国家版本馆CIP数据核字(2023)第257505号

Java面向对象程序设计实战教程

Java MIANXIANG DUIXIANG CHENGXU SHEJI SHIZHAN JIAOCHENG

主　　编：张　昕　胡丽霞

出版发行：上海交通大学出版社　　地　　址：上海市番禺路951号

邮政编码：200030　　电　　话：021-64071208

印　　制：三河市骏杰印刷有限公司　　经　　销：全国新华书店

开　　本：787 mm×1 092 mm　1/16　　印　　张：14.5

字　　数：300千字

版　　次：2023年12月第1版　　印　　次：2023年12月第1次印刷

书　　号：ISBN 978-7-313-30033-1　　电子书号：ISBN 978-7-89424-723-0

定　　价：45.00元

Preface
前 言

随着高等职业教育的迅速发展，项目导向、任务驱动、基于工作过程系统化的课程开发理念得到普遍认同。为全面贯彻党的教育方针，落实党的二十大精神，坚持为党育人、为国育才，全面提高人才自主培养质量，及时反映新时代课程教学改革的成果，满足高等职业院校计算机类专业的教学需要及相关人员的在岗培训需求，编者特编写了本教材。

编者总结了多年的工作和教学经验，针对高职学生的培养目标和学习特点，采用科学严谨、深入浅出、图文并茂的“融媒体”教材形态，从培养专业能力出发，以为后续课程服务、专业服务职业岗位（群）为宗旨，按照“理实一体化”的教学模式编排内容。

本教材具有以下特点。

1. 全面反映新时代教学改革成果

本教材以《职业院校教材管理办法》（教材〔2019〕3 号）和《教育部关于职业院校专业人才培养方案制订与实施工作的指导意见》（教职成〔2019〕13 号）为指导，以课程建设为依托，全面反映新时代产教融合、校企合作、教育信息化等方面的教学改革成果，以培养职业能力为主线，将探究学习和创新能力的培养贯穿教材始终，充分融合了不断创新与发展的“教学做合一”和“理实一体化”的教学组织与实施形式。教材的编写弱化了“教学材料”的特征，强化了“学习资料”的功能。

2. 以学生为中心，融入“互联网 +”思维

本教材按照“以学生为中心，以学习成果为导向，促进自主学习”的思路开发设计，可充分调动学生的学习主动性，培养其树立正确的职业价值观。

为了充分发挥“互联网 +”的优势，教材配备了二维码学习资源。学生用手机扫描教材中的二维码即可获得在线数字课程资源支持。同时，编者为方便教师授课还提供了配套的数字教材。

新形态一体化教材便于学生即时学习和个性化学习，有助于教师借此创新教学模式。

3. 编写体例、形式和内容适合职业教育特点

本教材采用模块化设计结构，以“任务”为驱动，强调“理实一体、学做合一”，符合学生认知规律，更加突出实践性。

4. 提炼思政教育元素，强化教材全方位育人功能

本教材特设“程序人生”模块，将编程知识的学习与思政元素相结合，把思想政治工作贯穿教育教学全过程，落实立德树人根本任务，科学合理拓展专业教材在思想教育方面的广度、深度和温度，体现时代发展与产业升级的最新要求，旨在培养德智体美劳全面发展的社会主义建设者和接班人。

5. 校企合作开发教材，实现校企协同“双元”育人

本教材紧跟产业发展趋势，意在满足行业人才需求，及时将产业发展的新技术、新工艺、新规则融入教材，反映典型岗位（群）职业能力要求，并吸纳行业企业高级技术人员、能工巧匠等深度参与教材编写。

本教材在编写团队深入校企合作的基础上开发完成，案例来源真实可靠。青软创新科技集团股份有限公司的高级工程师刘文锋具有多年的软件开发工作经验和社会学员培训经验，对教材中的案例进行了审核，并为教材编写提供了诸多宝贵意见。

本教材由廊坊职业技术学院张昕、胡丽霞担任主编，由黄丙根、佟超担任副主编，张娜、刘丹丹、刘文锋参与编写。

本教材在编写过程中参阅了国内外有关专家和学者在编程方面的一些新理念和新成果，在此深表感谢。

由于编者水平有限，书中不足之处在所难免，恳请广大读者批评指正，以便于我们在今后的修订和重印过程中及时改正。

编　者

Contents

目 录

项目 1

Java 程序设计概述

知识目标

- 了解 Java 的发展历程。
- 配置和使用 Java 开发环境。
- 理解 Java 的运行机制。

能力目标

能够安装和配置 Java 程序的开发环境，能够编写简单的 Java 程序，能调试简单程序。

素质目标

培养学生编程的基本素养、自学意识。培养学生的爱国主义情怀，树立学生的课程自信、学习自信。

近年来，中国在技术领域取得了巨大成就，在语言编程方面，Java 在中国已经成了一种强大的技术力量，它不仅是一种在中国广泛使用的编程语言，也是在世界上流行的编程语言之一。从移动应用到企业软件，从金融系统到游戏开发，Java 都是一个非常受欢迎的选择。据统计，全球有超过 9 亿台设备上运行着 Java，其中包括计算机、手机、平板电脑等。Java 受欢迎的第二个原因是其可移植性和可扩展性，这使得 Java 成为开发跨平台软件的首选语言。Java 是一个开源项目，这意味着它的源代码可以被任何人查看、修改和分发。开源使得 Java 的发展得到了全球开发者的支持和贡献，从而推动了其技术的不断发展。

由于 Java 的广泛应用和具有的开源特性，其应用前景非常广阔，越来越多的公司和组织使用 Java 来开发各种应用程序，从而推动了 Java 技术的不断发展和创新。

1.1 课前知识学习

Java 是由 Sun Microsystems 公司于 1995 年 5 月推出的 Java 程序设计语言和 Java 平台的总称。Java 是一种可以编写跨平台应用软件的面向对象的程序设计语言，也是现在使用最广泛的编程语言之一。

1991 年 4 月，由詹姆斯·高斯林(James Gosling)领导的绿色计划(Green Project)开始启动，这个计划的产品就是 Java 语言的前身 Oak(橡树)。随着 1995 年互联网潮流的兴起，Oak 迅速找到了最适合自己发展的市场定位并蜕变成为 Java 语言。

1995 年 5 月 23 日，Oak 语言改名为 Java，并在 SunWorld 大会上正式发布 Java 1.0 版本。Java 语言第一次提出了“Write Once，Run Anywhere”的口号。

1996 年 1 月 23 日，JDK 1.0 发布，Java 语言有了第一个正式版本的运行环境。JDK 1.0 提供了一个纯解释执行的 Java 虚拟机实现(Sun Classic VM)。JDK 1.0 版本的代表技术包括 Java 虚拟机、Applet、AWT 等。

1997 年 2 月 19 日，Sun 公司发布了 JDK 1.1，Java 技术的一些最基础的支撑点(如 JDBC 等)都是在 JDK 1.1 版本中发布的，JDK 1.1 版本的代表技术有 JAR 文件格式、JDBC、JavaBeans、RMI。

1998 年 12 月 4 日，JDK 迎来了一个里程碑式的版本 JDK 1.2，Sun 在这个版本中把 Java 技术体系拆分为 3 个方向，分别是面向桌面应用开发的 J2SE(Java 2 Platform，standard edition)、面向企业级开发的 J2EE(Java 2 Platform，enterprise edition)和面向手机等移动终端开发的 J2ME(Java 2 Platform，micro edition)。

2002 年 2 月 13 日，JDK 1.4 发布。JDK 1.4 是 Java 真正走向成熟的一个版本，Compaq、Fujitsu、SAS、Symbian、IBM 等著名公司都有参与甚至实现自己独立的 JDK 1.4。许多主流应用(Spring、Hibernate、Struts 等)能直接运行在 JDK 1.4 之上。

2006 年 12 月 11 日，JDK 1.6 发布，并启用 Java SE 6、Java EE 6、Java ME 6 的命名方式。

2009 年 4 月 20 日，Oracle 公司宣布正式收购 Sun 公司，Java 商标从此正式归 Oracle 公司所有。

2014 年 3 月 18 日，Oracle 公司发布 Java SE 1.8。2017 年 9 月 21 日，JDK 1.9 发布。2018 年 3 月 14 日，JDK 1.10 发布。

2018 年 3 月，Oracle 公司正式宣布 Java EE 成为历史名称，曾经拥有无数光辉的 Java EE 产品线(至今仍使用较为广泛的 JDBC、JMS、Servlet 等组件)被 Oracle 公司扫地出门，全部打包赠送给 Eclipse 基金会，并且不能使用 Java 商标，并更名为 Jakarta EE。

2018 年 9 月 25 日，JDK 11 发布，Oracle 也调整了 JDK 的授权许可证，把以前的商业许

可证授权给 OpenJDK，官方宣布同时发布两个 JDK，一个是 Oracle OpenJDK，另一个是 OracleJDK，共享的大部分源码近乎一致，个人均可免费使用。

2019 年 3 月 20 日，JDK 12 发布，RedHat 接手了 OpenJDK 8 和 OpenJDK 11 的管理和维护权。在 JDK 12 中包含了 8 个 JEP。

2019 年 9 月 17 日，JDK 13 发布，这个版本主要通过改善 Java SE 平台和 JDK 的性能、稳定性和安全性来提高开发人员的生产力。

2020 年 3 月 17 日，JDK 14 发布，这个版本主要是对 JDK 历史版本的一些增强，也引入了一些新增的功能。

2020 年 9 月 15 日，JDK 15 发布，按照规划路线，JDK 14 也停止更新，JDK 15 虽然不是长期支持(long time support，LTS)版本，但也引入了一些开创性的功能和对早期版本功能的优化。

2021 年 3 月 16 日，JDK 16 发布，这部分依旧是一些功能的优化升级。

2021 年 9 月 14 日，JDK 17 发布，这也是在 JDK 11 之后的一个 LTS 版本，主打安全和稳定的特性，并且官方支持到 2029 年 9 月。这个版本包含了 14 个 JEP 更新。

2023 年，以 TIOBE(TIOBE 是一家荷兰的编程软件质量评估公司，每月发布一份编程语言排行榜)编程语言排行榜来看，Java 的排名有所下降，2023 年 6 月，Java 的排名一度降至第 4 名，造成这种趋势的原因之一是在 Java8 版本以后引入付费许可模式，尽管 Java 的受欢迎程度有所下降，但其在软件开发中的地位不可替代，在软件开发领域的重要性不可忽视。同时，Java 语言本身也在不断发展和改进，以适应不断变化的开发需求和市场环境。

1.2　课中学习任务

Java 既是编译型语言，又是解释型语言。所以 Java 源文件首先需要通过 javac 编译生成后缀名为 class 的字节码文件(与平台无关，只面向 JVM)，然后使用 Java 虚拟机将字节码解释成特定平台上的机器码运行。虽然不同平台上的 JVM(Java virtual machine，Java 虚拟机)不同，但是都提供了相同的接口。


Java 开发环境搭建-i5eg1p

1.2.1　任 务 准 备

1.2.1.1　JDK 简介

JDK(Java development kit)是 Java 语言的软件开发工具包，要运行 Java 程序，首先要下载并安装 JDK，若没有 JDK，则无法编译 Java 程序(指 Java 源文件)。JDK 中包含了 Java 的运行环境(JVM+Java 系统类库)和 Java 工具。

JDK 包含的基本组件如下。

(1)javac:编译器,将源程序转换成字节码。

(2)jar:打包工具,将相关的类文件打包成一个文件。

(3)javadoc:文档生成器,从源码注释中提取文档。

(4)jdb:debugger,查错工具。

(5)java:运行编译后的 Java 程序(class 后缀)。

(6)appletviewer:小程序浏览器,一种执行 HTML 文件上的 Java 小程序的 Java 浏览器。

(7)Javah:产生可以调用 Java 过程的 C 过程,或建立能被 Java 程序调用的 C 过程的头文件。

(8)Javap:Java 反汇编器,显示编译类文件中的可访问功能和数据,同时显示字节代码的含义。

(9)Jconsole:Java 进行系统调试和监控的工具。

1.2.1.2 下载 JDK

可以从 Oracle 官网下载 JDK,以下提供的链接是 JDK 1.8 的下载地址 https://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html,下载后安装即可,如图 1-1～图 1-3 所示。

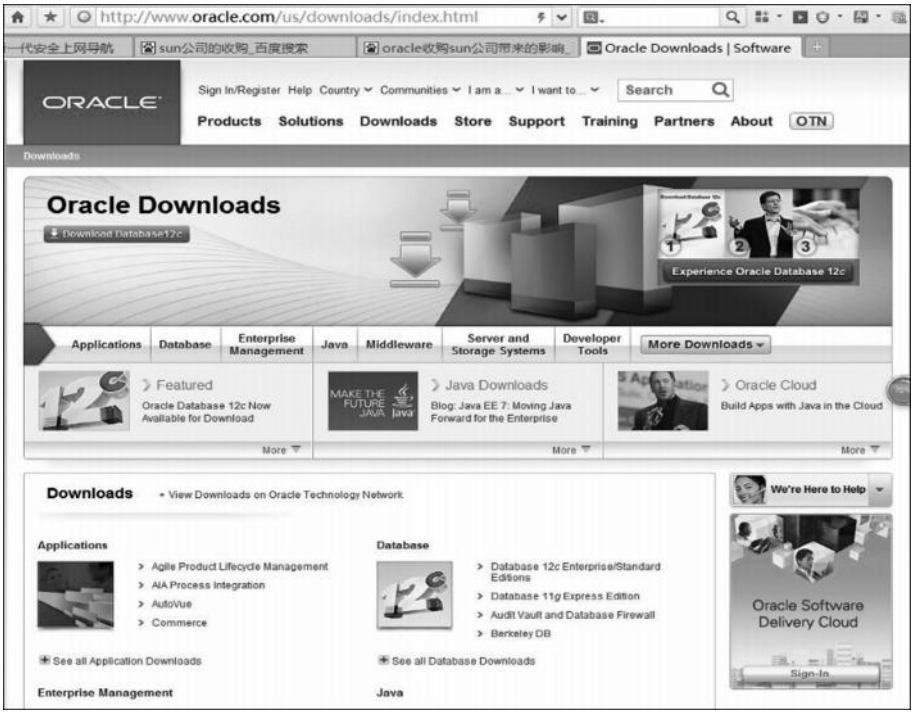

图 1-1 Oracle 官网下载页面

图 1-2　JDK 1.8 下载页面

图 1-3　JDK 文件内容

1.2.2　任 务 实 施

1.2.2.1　配置环境变量

Java 开发环境搭建是否成功的关键-ajtpbh

安装好 JDK 后，需要把它的地址添加到环境变量中。在“控制面板”中单击“高级系统设置”链接，如图 1-4 所示，打开“系统属性”对话框，切换到“高级”选项卡，单击“环境变量”按钮，如图 1-5 所示，在弹出的“环境变量”对话框中新建一个系统变量，变量名为 JAVA_HOME，变量内容为 JDK 的安装地址，如图 1-6 所示。这样本地机的环境就配置好了，如图 1-7 所示。

图 1-4　控 制 面 板

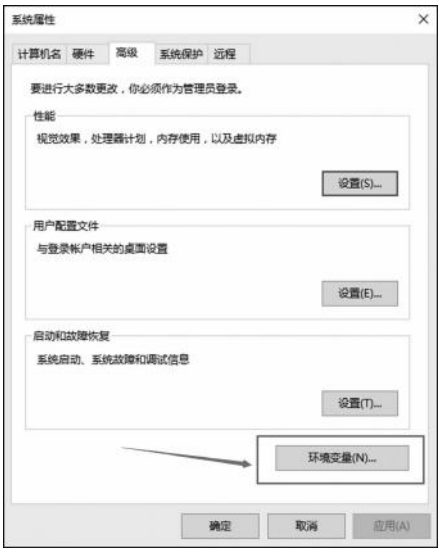

图 1-5　“系统属性”对话框

图 1-6 “环境变量”对话框

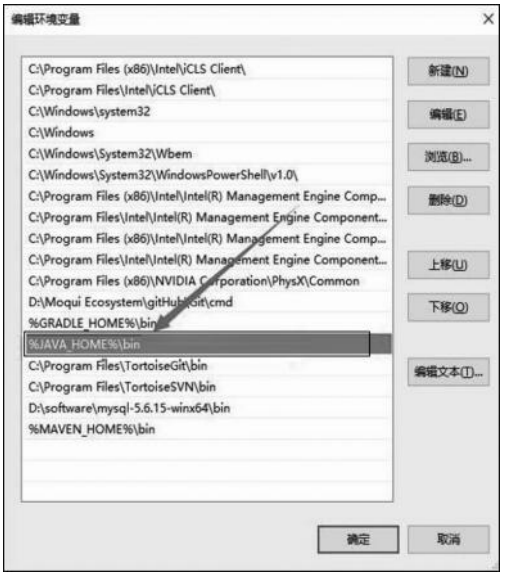

图 1-7 “编辑环境变量”对话框

1.2.2.2 下载和安装 Eclipse

本书选用的集成开发环境(integrated development environment,IDE)为 Eclipse，可以从其官网 http://www.eclipse.com 下载和安装，如图 1-8 所示。

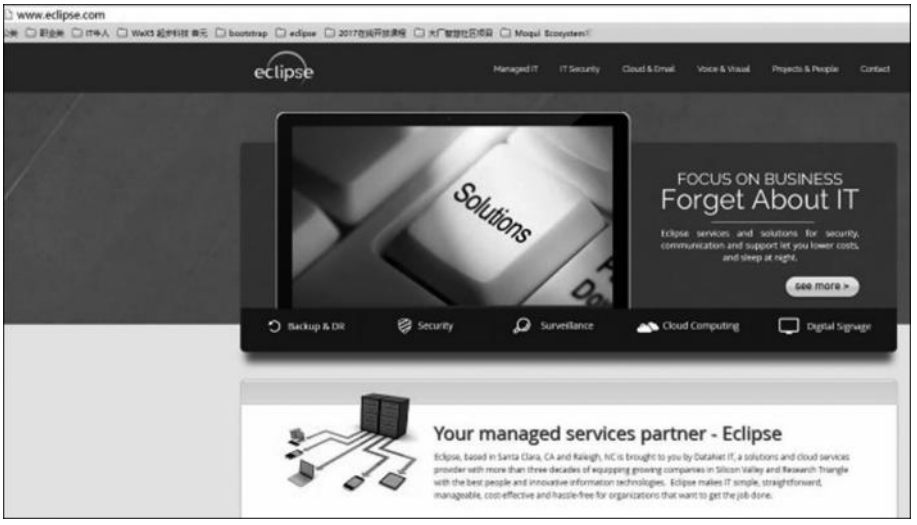

图 1-8 Eclipse 官网

1.2.2.3 编写和运行第一个 Java 程序

在 Java 环境配置好后，就可以使用 Eclipse 编写并运行 Java 程序了。

一般而言，Java 程序分为 Java 应用程序(Java Application)和 Java 小程序(Java

Applet)。Java Application 是一个完整的应用程序,打包成 jar 文件后,可以独立运行;而 Java Applet 一般被看作一个软件组件,作为动态网站的一部分。

本书以 Java Application 为主进行 Java 知识的讲解。

1. 第一个 Java 程序

要编写和运行一个 Java 应用程序,就必须先创建一个 Java 项目。在 Eclipse 中执行“File”→“New”→“Java Project”命令,如图 1-9 所示,并在打开的“New Java Project”窗口中输入项目名(如 Project),则创建好了一个项目,如图 1-10 所示。

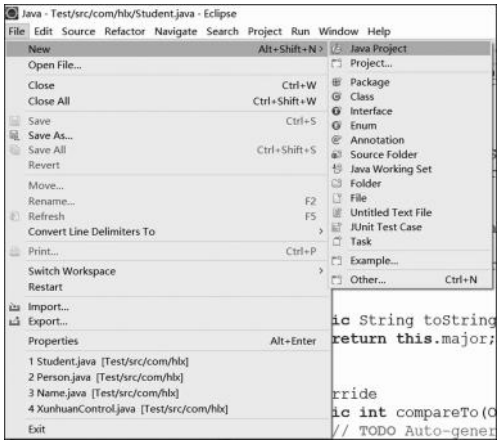

图 1-9　创建 Java 项目

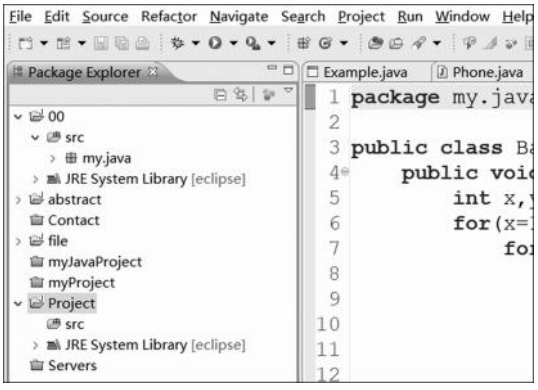

图 1-10　项 目 窗 口

在 Project 的 src 上右击,在弹出的快捷菜单中选择“New”→“Package”选项,在弹出的对话框中输入包名(如 first. com),如图 1-11 所示,在包上右击,在弹出的快捷菜单中选择“New”→“Class”选项,即可创建类,如图 1-12 所示。

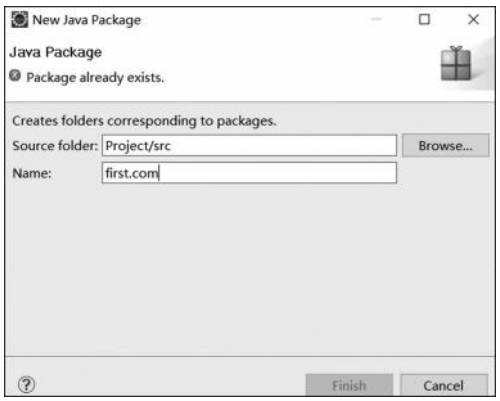

图 1-11　创建 Package

图 1-12　创　建　类

这样就可以输入 Java 代码了。

在程序执行过程中，首先由编译器将程序员编写的源代码转换成字节码，而后由 Java 虚拟机将字节码转换成 x86 系列 CPU 使用的本地代码，然后由 x86 系列 CPU 负责实际的处理，即 Java 虚拟机是一边把 Java 字节码逐一转换成本地代码一边运行的。Java 源文件编译和解释的过程是编译器—虚拟机—CPU。

2. 理解 Java 的运行机制

Java 语言的运行和其他高级语言相同，遵循了从源代码转换成机器码的过程。Java 的源代码（*.java 文件）经过 Java 编译器转换成特定 CPU 架构的机器码。也正是因为这一中间物，Java 才有所谓的跨平台特性。在 Windows 平台上编译好的字节码复制到 linux 平台后，经过为 Linux 而设计的 Java 虚拟机解释后即可执行。跨平台这一特性是通过字节码和 JVM 来实现的。因此，想搞清楚 Java 程序到底是如何运行的，重点在于弄明白字节码是如何被转换成与 CPU 架构相关的机器码然后被执行的，也就是要理解 JVM。

JVM 实例对应了一个独立运行的 Java 程序，它是进程级别 DE。启动一个 Java 程序时，一个 JVM 实例就产生了，任何一个拥有 public static void main(String[] args)函数的 class 都可以作为 JVM 实例运行的起点。JVM 运行原理如图 1-13 所示。

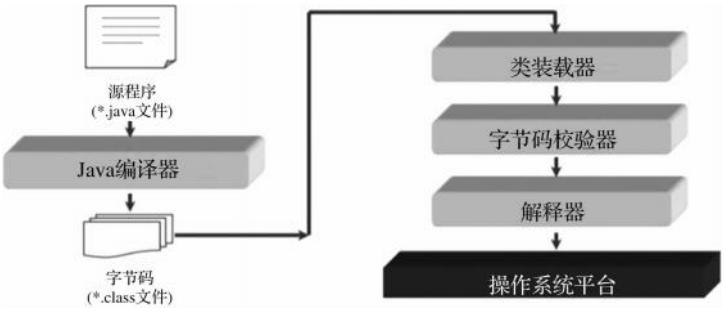

图 1-13　JVM 运行原理

需要注意的是，Java 命令首先启动虚拟机进程，虚拟机进程成功启动后，读取参数“Hello，World”，把它作为初始类加载到内存，对这个类进行初始化和动态链接（关于类的初始化和动态链接会在后面介绍），然后从这个类的 main 方法开始运行。也就是说，class 文件不是直接被系统加载后直接在 CPU 上运行的，而是被一个称为虚拟机的进程托管的。首先虚拟机进程必须启动就绪，然后由虚拟机中的类加载器加载必要的 class 文件，包括 JDK 中的基础类（如 String 和 Object 等），然后由虚拟机进程解释 class 字节码指令，把这些字节码指令翻译成本机 CPU 能够识别的指令，才能在 CPU 上运行。

启智润心

编程语言是人类智慧的集中体现，是软科学与硬科学的结合点，是现代高科技发展的必备关键基础设施，是大国重器与高科技的摇篮。在这样一个编程语言被西方垄断的局面下，我们只有奋起直追、脚踏实地，培养我国的编程人才，创造自己的编程语言，才能在信息技术领域拥有主动权。

1.3 课后任务提升

Java 开发环境搭建-omudwj

1.3.1 任 务 描 述

使用 Java 语言编程，输出“中国梦”的图案。

编写和运行 Java 程序-vu5jhh

1.3.2 需 求 说 明

本任务会用到 Java 中的 System 类，System 类主要是一些与系统相关的属性和方法的集合，而且其内部的方法全部是静态的，所以可以直接进行调用，如 System. out. print。System 类位于 java. lang 包中。

1.3.3 实 施 步 骤

首先创建一个 Java 项目，在项目中创建一个类，在这个类中包含主方法 main；在主方法 main 中编写如下输出语句，并在控制台中演示结果：

```
System.out.println("* * * * * * * * * * *");
System.out.println("  中国梦  ");
System.out.println("* * * * * * * * * * *");
```

程序代码如下：

```
public class TestDemo {
  public static void main(String args[]) {
      System.out.println("* * * * * * * * * * *");
      System.out.println("  中国梦  ");
      System.out.println("* * * * * * * * * * *");
  }
}
```

程序运行结果如下：

```
* * * * * * * * * * *
        中国梦
* * * * * * * * * * *
```

1.4 程 序 人 生

詹姆斯·高斯林(James Gosling，1955年5月19日—)，加拿大软件专家，Java编程语言的共同创始人之一，如图1-14所示。高斯林在孩童时期就表现出对科技的强烈兴趣，12岁时，他用坏了的电话和电视中的零件制作了一台游戏机。邻居们经常在他们的联合收割机遇到故障时来找他修理。14岁那年，学校邀请他去附近的一所学校参观。从那时起他开始学习计算机编程，他被学校的天文学专业录取为临时编程员，负责编制分析卫星天文资料的计算机程序。

图1-14 詹姆斯·高斯林

高斯林在20世纪80年代初期拿到了博士学位，他在IBM公司开始了首代工作站的设计。那时IBM的领导们并没有把工作站计划放在首位。在极度的失望中，高斯林去了Sun公司。1983年，他为IBM设计了一台NeWS系统的首代工作站，但是并不受欢迎；1984年加入Sun，在Sun被买走之前，他一直为Sun工作。他用了5年时间开发了与OS2相似的Sun NeWs视窗系统，该系统虽然受到了技术上的欢迎，但并没有成为一款受欢迎的产品。

1991年，高斯林和一批技术人员在Sun公司建立了一个被称为Oak的计划，目的是在虚拟机上开发一种可以在多个平台(如电视机顶盒)上运行的程序。之后，Oak就慢慢演变成了Java。随着因特网的广泛应用，特别是网景公司开发的Web浏览器的出现，Java在世界范围内逐渐成为最受欢迎的编程语言之一。

在2010年Oracle收购Sun microsystems之后，高斯林于2011年年初加盟谷歌，2011年8月30日，高斯林在他的博客中表示将会离开谷歌。

2002年，高斯林被《经济学人》杂志授予发明奖。2007年，他被授予二级加拿大勋章。2013年，他当选为ACM Fellow。2015年，他被授予IEEE约翰·冯·诺依曼奖章。2018年，他因设计并创造了Java编程语言而被收录进计算机历史博物馆荣誉墙。这就是“Father Of The Java”的大概传奇事迹。

项目小结

本项目主要介绍了 Java 的发展和现状，Java 的运行环境设置，简单 Java 程序的运行过程，以及 JVM 的运行原理。

习　题

一、单选题

1-1. 编译 Java 源文件将产生相应的字节码文件，字节码文件的扩展名为(　　)。

A. java　　B. class　　C. html　　D. exe

1-2. Java 程序的执行过程中要用到 JDK 工具，其中，java. exe 指的是(　　)。

A. Java 文档生成器　　B. Java 解释器

C. Java 编译器　　D. Java 类分解器

1-3. Java 程序中的 main 方法的返回值类型是(　　)。

A. void　　B. int　　C. char　　D. String

1-4. 下列关于内存回收的描述正确的是(　　)。

A. 程序员必须创建一个线程来释放内存　　B. 内存回收程序负责释放无用内存

C. 内存回收程序允许程序员直接释放内存　　D. 内存回收程序只能被程序员调用

1-5. Java 源文件和编译后的文件扩展名分别为(　　)。

A. class 和 java　　B. java 和 class　　C. class 和 class　　D. java 和 java

二、填空题

1-1. Java 的可移植性实现了_____的梦想。

1-2. 在 Java 中，Java 源文件的扩展名是_____，字节码文件的扩展名是_____。

1-3. Java 中的标识符可以包含_____、_____、_____和_____，其中，不能以_____开头，不能是_____。

1-4. Java 语言通过_____机制简化了程序的内存管理。

三、判断题

1-1. Java 语言是不区分大小写的。　　(　　)

1-2. Java 语言含有指针。　　(　　)

1-3. Java 源文件的扩展名是 java。　　(　　)

1-4. 以//开始的是单行注释。　　(　　)

四、简答题

1-1. 简单描述 Java 虚拟机的作用。

1-2. 列举三种面向对象编程语言和三种非面向对象编程语言。

1-3. 编写 Java 程序，在屏幕上输出如下图形：

五、等级考试习题

1-1. 关于异常的含义，下列描述中正确的一项是(　　)。

A. 程序编译错误

B. 程序语法错误

C. 程序自定义的异常事件

D. 合理的异常处理可以分离程序中的正常逻辑代码和异常逻辑代码，便于代码的阅读和维护

1-2. 对于一个 Java 源文件，import、class 定义及 package 的正确顺序是(　　)。

A. package、import、class　　B. class、import、package

C. import、package、class　　D. package、class、import

1-3. 下列关于 main 方法的定义正确的是(　　)。

A. public main(String args[ ])

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

C. private static void main(String args[ ])

D. void main( )

1-4. 下列描述中，(　　)符合 Java 语言的特征。

A. 支持跨平台(Windows、Linux、UNIX 等)

B. GC(自动垃圾回收)，提高了代码安全性

C. 支持类 C 的指针运算操作

D. 不支持与用其他语言书写的程序进行通信

1-5. 下列(　　)方法用于在标准输出上打印文本。

A. System. print( )　　B. Console. write( )

C. System. out. println( )　　D. Console. print( )

项目 2

Java 语言基础

知识目标

- 掌握 Java 中注释、数据类型、常量和变量的用法。
- 掌握 Java 中各种运算符、字符串的用法。

能力目标

能够熟练使用 Java 的常量、变量、运算符等进行简单编程。

素质目标

培养编程的基本思维、严谨的习惯、自学的能力；培养学生遵守行业规范的职业素养，以及对待学习和工作一丝不苟、认真细致、精益求精的工匠精神。

要进行 Java 程序的开发，必须掌握 Java 的数据类型、常量、变量和字符串的用法。

2.1 课前知识学习

在 Java 中为程序添加注释可以用来解释程序的某些语句的作用和功能，提高程序的可读性。也可以使用注释在原程序中插入设计者的个人信息。此外，还可以用程序注释来暂时屏蔽某些程序语句，让编译器暂时不要处理这部分语句，等到需要处理时，只需把注释标记取消就可以了。

Java 中的注释分为单行注释、多行注释和文档注释 3 种。

1)单行注释

单行注释一般用来注释局部变量，可以用//或按 Ctrl+/组合键添加注释。

【例 2-1】

```
public class TestDemo{
        public static void main(String args[]){
            //输出消息到控制台
              System.out.println("Welcome to Java World!");
        }
   }
```

2)多行注释

多行注释一般是多行或成段文字，使用/* */表示。

【例 2-2】

```
/*
 * Test.java              文件名称
 * 2023.9.10              日期
 *  第一个 Java 程序   功能说明
 *  */
```

3) 文档注释

文档注释以/** 开始，以 */结束，一般用于标识软件、作者信息等。

【例 2-3】

```
/**
 * 这是一个简单的文档注释例子
 * version 2020.05.01
 * author
 */
```

```
public class Test
    public static void main(String args[]){
      System.out.println("这是一个简单的 Java 程序");
}
```

2.2　课中学习任务 1


基本数据类型和引用类型 1-0kbnev


基本数据类型和引用类型 2-15vjfn

Java 数据类型分为两大类：基本数据类型和引用数据类型。其中，基本数据类型又分为数值型和非数值型。数值型分为整数类型和浮点类型。非数值型分为字符型和布尔型。Java 主要有 8 种基本数据类型，分别是 byte、short、int、long、char、boolean、float、double，如图 2-1 所示。

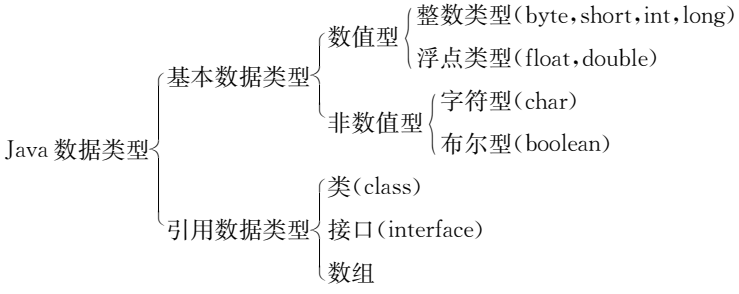

图 2-1　Java 数据类型分类

2.2.1　任 务 准 备

Java 语言提供了 4 种整型数据类型，即 byte、short、int、long，它们都是定义一个整数，但表示数据的范围不同，能够表示数据的范围越大，占用的存储空间就越大，因此，在程序设计中应选择最合适的类型来定义整数。

1. 整数类型

整数类型分为 byte(字节型)、short(短整型)、int(整型)和 long(长整型)4 种。

(1)byte。byte(字节型)数据在内存中占 1 个字节，表示的存储数据范围为－128～127。

(2)short。short(短整型)数据在内存中占 2 个字节，表示的存储数据范围为－32 768～32 767。

(3)int。int(整型)数据在内存中占 4 个字节，表示的存储数据范围为－2 147 483 648～2 147 483 647。

(4)long。long(长整型)数据在内存中占 8 个字节，表示的存储数据范围为－9 223 372 036 854 774 808～9 223 372 036 854 774 807。

由上可以看出，byte、short 的取值范围比较小，而 long 的取值范围是最大的，所占用的空间也是最多的。int 的取值范围基本上可以满足人们的日常计算需求，所以 int 也是使用

最多的一种整型类型。

2. 浮点类型

Java 的浮点类型有 float 和 double 两种，它们的区别在于精度的不同。

(1)float。float(单精度浮点型)数据在内存中占 4 个字节。

(2)double。double(双精度浮点型)数据在内存中占 8 个字节。

double 数据比 float 数据的存储范围更大，精度更高。浮点型数据在不声明的情况下默认均为 double，如果要表示一个数据是 float 的，可以在数据后面加上 F。例如，4.22、4.22D 都代表 double 实数，而 4.22F 是 float 实数。

浮点型数据不是完全精确的，在计算时可能出现小数点最后几位浮动的情况，这是正常的。

3. 字符型

字符型通常用来表示字母，字符型数据是使用单引号‘’括起来的单个字符，如‘A’‘5’等，其值就是单引号中的字符。例如，

```
char name = 'L'
char id = '2009010029'
```

Java 采用 Unicode 编码系统，字符可以涵盖所有语言涉及的字符，包括汉字、日文等。

注意：“A”和‘A’是不同的，“A”是长度为 1 的 String(字符串)，而‘A’是 char 类型。

4. 布尔型

布尔型也称为逻辑型，只有两个取值：false(逻辑假)和 true(逻辑真)，常用于逻辑计算。

Java 基本数据类型的存储范围如表 2-1 所示。

表 2-1 Java 基本数据类型的存储范围

类 型	类型描述	字节长度	取值范围	默认值
byte	字节型	1 字节	$-2^7 \sim 2^7-1$	0
short	短整型	2 字节	$-2^{15} \sim 2^{15}-1$	0
int	整型	4 字节	$-2^{31} \sim 2^{31}-1$	0
long	长整型	8 字节	$-2^{63} \sim 2^{63}-1$	0
float	单精度浮点型	4 字节	根据 IEEE 754—1985 标准	0.0F
double	双精度浮点型	8 字节	根据 IEEE 754—1985 标准	0.0D
char	字符型	1 字节	0～65 535	0

注意：

(1)Java 中所有的基本数据类型都有固定的存储范围和所占内存空间的大小，而不受具体操作系统的影响。在存入数据时，不要超出存储范围，否则会出现溢出错误。

(2)整型数据默认为 int，浮点类型数据默认为 double。

(3)若要表示 long 或 float 数据，则在相应的数值后面加上 L(或 l)或 F(或 f)，否则会出

现编译问题。

(4)字符型数据和整型数据是可以相互转换的,它们都是以 ASCII 码来存储的,可以将字符型数据看作整型数据。

(5)String 数据是必不可少且使用最多的类型之一,它属于引用数据类型中“类”的范畴。

2.2.2 任 务 实 施

【例 2-4】 整型数据的存储和使用。

```
byte a1=100;
short int a2 = 200;
int a3 = 300;
long int a4 = 400l;                    //400 后面加上小写字母 l,表示是长整型
System.out.println(a1);
System.out.println(a2);
System.out.println(a3);
System.out.println(a4);
```

【例 2-5】 整型的数据存储和使用。

```
public class Test1{
  public static void main(String args[]){
    int x=20;
    System.out.println(x*2);
    System.out.println(x*x);
}
}
```

程序输出结果如下:

```
40
400
```

【例 2-6】 int 与 char 之间的转换是通过 ASCII 进行的。

```
char a ='A';                           //A 对应的 ASCII 值为 65
char b ='a';                           //a 对应的 ASCII 值为 97
System.out.println(a);                 //输出结果为 65
System.out.println(n);                 //输出结果为 97
```

【例 2-7】 将对应的小写字母转换成大写字母。

```
char a ='d';                           //A 对应的 ASCII 值为 65
```

```
a = a-32;
System.out.println(a);                    //输出结果为 D
```

【例 2-8】 浮点型数据的操作。

```
float a =1.2f;                            //1.2 为 double,1.2f 则为 float
double b = 3.4;                           //3.4 默认为 double
System.out.println(a);                    //输出结果为 1.2
System.out.println(b);                    //输出结果为 3.4
```

【例 2-9】 数据溢出。

```
float a =1.2f;                            //1.2 为 double 型,1.2f 则为 float 型
double b = 3.4;                           //3.4 默认为 double 型
System.out.println(a);                    //输出结果为 1.2
System.out.println(b);                    //输出结果为 3.4
```

【例 2-10】 布尔型数据的使用。

```
public class Test2{
public static void main(String args[]){
    boolean x=true;
    boolean y=false;
    System.out.println("x&&y="+(x&&y));      //逻辑与
    System.out.println("x||y="+(x||y));      //逻辑或
  }
}
```

程序运行结果如下：

```
x&&y = false
x||y = true
```

2.3 课中学习任务 2

在程序执行过程中值不能改变的量称为常量，值能被改变的量称为变量。常量与变量的声明都必须使用合法的标识符，所有常量与变量只有在声明之后才能使用。

常量 变量 1-4irt0u

常量 变量 2-plrpjp

2.3.1 任 务 准 备

2.3.1.1 变量

在 Java 语言中，声明变量的基本格式如下：

```
数据类型 变量名 1[,变量名 2,……];  //定义变量
```

或

```
数据类型 变量名 1 = 变量值 1[,变量名 2 = 变量值 2,……];//变量初始化
```

例如，

```
int x, y, z;                    //声明 3 个 int 整数 x、y、z
int a = 3, b = 4, c= 5;         //变量初始化
byte t = 42;                    //声明并初始化 t
String s = "Runoob";            //声明并初始化字符串 s
```

变量虽然是由程序员所命名的，但是变量的命名并不是任意的，需要遵循一定的规则。

(1)变量名必须是一个有效的标识符。

(2)变量名不能重复。

(3)应选择有意义的单词作为变量名。在命名变量名时，最好能通过变量名看出变量的内容，这样既能方便读者对程序的理解，增加可读性，又方便程序的维护，减轻程序维护人员的工作负担。

2.3.1.2　常量

常量是指在程序运行过程中值始终保持不变的量。按照数据类型来分，Java 中常用的常量有整型常量、浮点型常量、布尔型常量、字符型常量和字符串常量 5 种。

1. 整型常量

整型常量简称整数，有十进制、八进制、十六进制和二进制 4 种表示形式。

(1)十进制，如 45、76、0。

(2)八进制，以 0 开头，由 0～7 组成，如 013、045、0123。

(3)十六进制，以 0x 或 0X 开头，可以包含数字 0～9，a～f 或 A～F，如 0x16、0xa、0xf。

(4)二进制，由 0 或 1 组成，以 0b 或 0B 开头，如 0b10011。

2. 浮点型常量

浮点型常量用于表示带小数点的实数，它通常有小数点和指数两种表示形式。

(1)小数点形式，由数字和小数点组成，如 2.3、4.55、－2.12。

(2)指数形式，如 1.2e+5、2.3e+2、1.3e－4，分别表示 1.2×10^{5}、2.3×10^{2} 和 1.3×10^{-4}。

3. 布尔型常量

布尔型常量只有 true 和 false，其中，true 代表真，false 代表假。

4. 字符型常量

字符型常量有单个字符常量和转义字符常量两种。

(1)单个字符常量。单个字符常量即由一对单引号括起来的单个字符，这个字符可以是

Unicode 字符集中的任意字符，如‘a’‘G’‘$’。

(2)转义字符常量。ASCII 字符集中的前 32 个字符是控制字符，具有特殊的含义，如回车、换行等，这些字符难以用一般方式表示，因此，Java 引入了一些特别的定义，即用反斜线“\”开头，后面跟一个字母来表示某个特定的控制符，这就是转义字符。例如，\n 表示换行、\t 表示制表符、\' 表示单引号、\"表示双引号、\\表示反斜杠等。

5. 字符串常量

字符串常量是由一对双引号括起来的由若干个字符组成的字符串，如“hello Java”“我的第一个 Java 程序”“hello,‘\Java\’!”。

2.3.1.3 String 类

字符串是编写程序时经常用到的一类数据类型。在 Java 中使用 String 类和 StringBuffer 类来封装字符串。String 类所创建的字符串是不能改变的，在需要使用可修改的字符串时，一般使用 StringBuffer 类创建。

1. 字符串的声明与创建

声明字符串：String 字符串名。

在 Java 语言中将字符串作为对象来管理，因此可以像创建其他类对象一样来创建字符串对象。创建对象要使用类的构造方法。String 类的常用构造方法如下：

```
public String();                    //创建一个空字符串
public String(String s);            //用已有字符串创建新字符串
public String(StringBuffer buf);    //用 StringBuffer 类的对象的内容初始化新字符串
public String(char value[]);        //用已有字符数组初始化新字符串
```

2. String 的基本操作方法

(1)获取字符串长度 length()，格式如下：

```
int length = str.length();
```

(2)获取字符串中的第 i 个字符 charAt(i)，格式如下：

```
//i 为字符串的索引号，可得到字符串任意位置处的字符，保存到字符变量中
char ch = str.charAt(i);
```

(3)取子串，格式如下：

```
//获取子字符串，从 beginIndex 处开始，到 endIndex－1 处结束，子串长度为 endIndex－beginIndex
public String SubString(int beginIndex,int endIndex)
```

【例 2-11】 类名首字母大写，多个字母采用驼峰式的写法。

```
String  str = "abcdef";
System.out.println(str+"的长度是:"+str.length());
System.out.println(str.charAt(1));                    //b
System.out.println(str.charAt(0));                    //a
System.out.println(str.substring(0, 2));              //0,1——>ab
System.out.println(str.substring(0));                 //abcdef
System.out.println(str.substring(2));                 //cdef
```

2.3.1.4　StringBuffer 类

StringBuffer 是字符串变量，它的对象是可以被修改的。

StringBuffer 类和 String 类都可以存储、操作字符串，但 String 类存储的是字符串常量，是值不能更改的量。String 类和 StringBuffer 类提供了多个字符串操作方法，详见 JDK 文档。

2.3.1.5　引用数据类型

Java 的引用数据类型有 3 种，分别是类(class)、接口(interface)和数组。引用数据类型指向一个对象，不是原始值，指向对象的变量是引用变量。

在 Java 中除去基本数据类型的其他数据类型基本上都是引用数据类型，自定义的 class 类也都是引用数据类型，可以像基本数据类型一样使用。后面会更加详细地介绍引用数据类型的用法。

2.3.2　任 务 实 施

【例 2-12】　字符串拼接。

```
public class TestDemo {
   public static void main(String args[]){
        byte b1 = 7;
        System.out.println(b1);
        System.out.println(Byte.MAX_VALUE);
        System.out.println(Byte.MIN_VALUE);
   }
}
```

程序运行结果如下：

```
7
127
-128
```

【例 2-13】 字符类型的特别之处。

```
public class TestDemo {
  public static void main(String args[]){
      //测试 char 和 int 之间的相互转换
  char c1 = 'a';
  char c2='好';
  char c3 = 97;
  System.out.println(c1);
  System.out.println(c2);
  System.out.println(c3);
  }
}
```

程序运行结果如下：

```
a
好
a
```

【例 2-14】 字符串拼接。

```
public class TestDemo {
  public static void main(String args[]){
      //byte
        System.out.println("基本类型:byte 二进制位数:" + Byte.SIZE);
        System.out.println("包装类:java.lang.Byte");
        System.out.println("最小值:Byte.MIN_VALUE=" + Byte.MIN_VALUE);
        System.out.println("最大值:Byte.MAX_VALUE=" + Byte.MAX_VALUE);
        System.out.println();

        //short
        System.out.println("基本类型:short 二进制位数:" + Short.SIZE);
        System.out.println("包装类:java.lang.Short");
        System.out.println("最小值:Short.MIN_VALUE=" + Short.MIN_VALUE);
        System.out.println("最大值:Short.MAX_VALUE=" + Short.MAX_VALUE);
        System.out.println();

        //int
        System.out.println("基本类型:int 二进制位数:" + Integer.SIZE);
        System.out.println("包装类:java.lang.Integer");
        System.out.println("最小值:Integer.MIN_VALUE=" + Integer.MIN_VALUE);
```

```
        System.out.println("最大值:Integer.MAX_VALUE=" + Integer.MAX_VALUE);
        System.out.println();

        //long
        System.out.println("基本类型:long 二进制位数:" + Long.SIZE);
        System.out.println("包装类:java.lang.Long");
        System.out.println("最小值:Long.MIN_VALUE=" + Long.MIN_VALUE);
        System.out.println("最大值:Long.MAX_VALUE=" + Long.MAX_VALUE);
        System.out.println();

        //float
        System.out.println("基本类型:float 二进制位数:" + Float.SIZE);
        System.out.println("包装类:java.lang.Float");
        System.out.println("最小值:Float.MIN_VALUE=" + Float.MIN_VALUE);
        System.out.println("最大值:Float.MAX_VALUE=" + Float.MAX_VALUE);
        System.out.println();

        //double
        System.out.println("基本类型:double 二进制位数:" + Double.SIZE);
        System.out.println("包装类:java.lang.Double");
        System.out.println("最小值:Double.MIN_VALUE=" + Double.MIN_VALUE);
        System.out.println("最大值:Double.MAX_VALUE=" + Double.MAX_VALUE);
        System.out.println();

        //char
        System.out.println("基本类型:char 二进制位数:" + Character.SIZE);
        System.out.println("包装类:java.lang.Character");
        //以数值形式而不是字符形式将 Character.MIN_VALUE 输出到控制台
        System.out.println("最小值:Character.MIN_VALUE="
                + (int) Character.MIN_VALUE);
        //以数值形式而不是字符形式将 Character.MAX_VALUE 输出到控制台
        System.out.println("最大值:Character.MAX_VALUE="
                + (int) Character.MAX_VALUE);
    }
}
```

程序运行结果如下：

```
基本类型:byte 二进制位数:8
```

```
包装类:java.lang.Byte
最小值:Byte.MIN_VALUE=-128
最大值:Byte.MAX_VALUE=127

基本类型:short 二进制位数:16
包装类:java.lang.Short
最小值:Short.MIN_VALUE=-32768
最大值:Short.MAX_VALUE=32767

基本类型:int 二进制位数:32
包装类:java.lang.Integer
最小值:Integer.MIN_VALUE=-2147483648
最大值:Integer.MAX_VALUE=2147483647

基本类型:long 二进制位数:64
包装类:java.lang.Long
最小值:Long.MIN_VALUE=-9223372036854775808
最大值:Long.MAX_VALUE=9223372036854775807

基本类型:float 二进制位数:32
包装类:java.lang.Float
最小值:Float.MIN_VALUE=1.4E-45
最大值:Float.MAX_VALUE=3.4028235E38

基本类型:double 二进制位数:64
包装类:java.lang.Double
最小值:Double.MIN_VALUE=4.9E-324
最大值:Double.MAX_VALUE=1.7976931348623157E308

基本类型:char 二进制位数:16
包装类:java.lang.Character
最小值:Character.MIN_VALUE=0
最大值:Character.MAX_VALUE=65535
```

【例 2-15】 字符串中子串的截取。

```
public class TestDemo {
  public static void main(String args[]){
      String x1 = "hello,java";
```

```
        String x2 = x1.substring(0, 1);
        System.out.println(x2);
    }
}
```

程序运行结果如下：

```
h
```

【例 2-16】　字符串的拼接。

```
public class TestDemo {
    public static void main(String args[]){
        String x1 = "hello,";
        String x2 = "java";
        System.out.println(x1 + x2);
    }
}
```

程序运行结果如下：

```
hello,java
```

启智润心

学好编程需要严谨的态度；标准化书写，执着追求，不怕困难。从第一行代码开始认真对待。

2.4　课中学习任务 3

Java 语言的常用运算符有算术运算符、关系运算符、逻辑运算符、位运算符和其他运算符。一些运算符随着运算数据类型的不同而功能不同，这是面向对象语言的共同特征，在今后的学习中应引起注意。

2.4.1　任 务 准 备

2.4.1.1　算术运算符

算术运算符用于对数据进行算术运算，根据操作数个数不同，算术运算符分为单目运算符和双目运算符。表 2-2 按优先级从高到低的顺序给出了所有算术运算符。

表 2-2 算术运算符

运 算 符	操 作 数	举 例	说 明
++	单目	a++,++a	自增
--	单目	a--,--a	自减
-	单目	-(a+b)	取负
*	双目	a*b	乘法
/	双目	a/b	除法
%	双目	a%b	取余数
+	双目	a+b	加法
-	双目	a-b	减法

其中,单目运算符++、--的优先级最高,其次是*、/、%,最后是+、-。

下面来看算术运算符在程序中的简单使用。

【例 2-17】

```
int a = 5;
int b = 3;
//相加
int sum = a + b;                    //sum 的值为 8
//相减
int difference = a - b;             //difference 的值为 2
//相乘
int product = a * b;                //product 的值为 15
//整除
int quotient = a / b;               //quotient 的值为 1
//取余
int remainder = a % b;              //remainder 的值为 2
//前缀自加
int result1 = ++a;                  //a 的值会增加为 6,result1 的值为 6
//后缀自加
int result2 = b++;                  //b 的值会增加为 4,result2 的值为 3
//前缀自减
int result3 = --a;                  //a 的值会减少为 5,result3 的值为 5
//后缀自减
int result4 = b--;                  //b 的值会减少为 3,result4 的值为 4
```

下面重点分析++和--的使用规律。

++是自增运算符,--是自减运算符,都是单目运算符。它们的使用形式有:变量++、++变量、变量--和--变量。

二者在使用时不同。变量++是先使用，后自增。++变量是先自增，后使用。例如：

```
int a = 5,b;
b = a++;     //a = 6,b = 5
```

再如：

```
int x = 10,y;
y = ++x;     //x = 11,y = 11
```

自减运算符--的使用规则和自增运算符++的使用规则相同。变量--是先使用后自减。--变量则是先自减后使用。例如：

```
int a = 5,b;
b = a--;  //a = 4,b = 5
```

再如：

```
int x = 10,y;
y = --x;  //x = 9,y = 9
```

2.4.1.2　关系运算符

关系运算符用来比较两个数据，其运算结果是布尔型的 true 和 false。当运算符比较的结果是真时，返回 true，否则，返回 false。表 2-3 给出了所有关系运算符。

表 2-3　关系运算符

运　算　符	含　义	举　例	说　明
<	小于	1<2	返回 true
<=	小于等于	5<=10	返回 true
>	大于	4>5	返回 false
>=	大于等于	4>=2	返回 true
==	等于	a==20	若 a 的值是 20，则返回 true，否则，返回 false
!=	不等于	a!=20	若 a 的值是 20，则返回 false，否则，返回 true

例如：

```
int a = 5;
int b = 3;
boolean result1= (a == b); // false
boolean result2= (a != b); // true
boolean result3= (a > b); // true
boolean result4= (a < b); // false
boolean result5= (a >= b); // true
boolean result6= (a <= b); // false
```

【例 2-18】

```
int a=4;
int b=1;
boolean c=a<b;                    //c 值为 false
```

在 6 个关系运算符中，以>、>=、<、<=的优先级较高，==、!=的优先级较低。

当有以上不同级别的运算符参与运算时，要先计算优先级别高的，再计算优先级别低的。

注意：“==”和“=”是两个完全不同的运算符，使用时需要注意。

2.4.1.3 逻辑运算符

Java 中的逻辑运算符如表 2-4 所示。

表 2-4 逻辑运算符

运 算 符	含 义	举 例	说 明
!	逻辑非	!a	a 为真时，返回假；a 为假时，返回真
&&	逻辑与	a && b	a、b 同时为真时，返回真，否则，返回假
||	逻辑或	a || b	若 a、b 有一个为真，则返回真

利用逻辑运算符将操作数连接的式子称为逻辑表达式，逻辑表达式的结果只能是布尔型，即 true 和 false。

1)逻辑与(&&)的运算规则

(1)true&&true，结果为 true。

(2)true&&false，结果为 false。

(3)false&&true，结果为 false。

(4)false&&false，结果为 false。

即只有当两边同时为 true 时，结果才为 true，其他情况的结果均为 false。


关系运算符与逻辑运算符-2de3sy

例如：

```
4>5&& 5<6     //结果为 false
```

2)逻辑或(||)的运算规则

(1)true || true，结果为 true。

(2)true || false，结果为 true。

(3)false || true，结果为 true。

(4)false || false，结果为 false。

即只要一边为 true，结果就为 true，其他情况的结果均为 false。

例如：

```
4>5 || 5<6     //结果为 true
```

3)逻辑非(!)的运算规则

(1)!true,结果为 false。

(2)!false,结果为 true。

即!true 的结果为 false,!false 的结果为 true。

例如:

```
!(4>5)  //结果为 true
```

需要注意的是,在逻辑运算符中存在短路问题,即当逻辑与 && 检测到符号左侧的值为假时,不再判断符号右侧的值,直接将结果置为假;同理,当逻辑或||检测到符号左侧的值为真时,直接将结果置为真,不再对符号右侧的值进行判断。这就是逻辑运算符中的短路问题。

2.4.1.4　位运算符


位运算符-ia2yhg

位运算是以二进制位为单位进行的运算,其操作数和运算结果均为整型值。Java 中常用的位运算符如表 2-5 所示。假设表中 a=11010110,b=01011001,n=2。

表 2-5　位 运 算 符

运　算　符	含　　义	举　　例	结　　果
~	按位取反	~a	00101001
&	按位与	a & b	01010000
|	按位异或	a | b	11011111
^	按位或	a ^ b	10001111
<<	左移	a << n	01011000
>>	带符号右移	a >> n	11110101
>>>	不带符号右移	a>>>n	00110101

2.4.1.5　其他运算符

1. 赋值运算符

Java 中常用的赋值运算符如表 2-6 所示。

表 2-6　赋 值 运 算 符

运　算　符	含　　义	举　　例	说　　明
=	简单赋值运算	x=a+b	简单赋值运算
+=	加	a += b	相当于 a=a+b
-=	减	a-= b	相当于 a=a-b
*=	乘	a *= b	相当于 a=a*b
/=	除	a /= b	相当于 a=a/b

（续表）

运　算　符	含　义	举　例	说　明
%=	取余	a %= b	相当于 a=a%b
&=	按位与	a &= b	相当于 a=a&b
|=	按位或	a |= b	相当于 a=a | b
^=	按位异或	a ^= b	相当于 a=a ^ b

2. 条件运算符

条件运算符用“?:”表示，也称三元运算符，这个运算相当于条件判断。例如，表达式 a?b:c 表示：若 a 的值为真，则整个表达式的值为 b，否则，整个表达式的值为 c。

例如：

```
int x =5;
int y = (x%2==0)?1:0   //y 的值为 0
```

2.4.1.6　运算符的优先级和结合性

运算符的优先级决定了表达式中不同运算符运算执行的先后顺序，优先级高的先运算，优先级低的后运算。而在优先级相同的情况下，要考虑结合性。表 2-7 列出了常用运算符的优先级和结合性。

表 2-7　运算符的优先级和结合性

序　号	运　算　符	结　合　性	优　先　级
1	()、[]、.	从左到右	从高到低
2	++、--、!、~、-(取负)	从右到左	
3	*、/、%	从左到右	
4	+(加法)、-(减法)	从左到右	
5	<<、>>、>>>	从左到右	
6	<、<=、>、>=	从左到右	
7	==、!=	从左到右	
8	&(按位与)	从左到右	
9	^(按位异或)	从左到右	
10	|(按位或)	从左到右	
11	&&(逻辑与)	从左到右	
12	||(逻辑或)	从左到右	
13	?:(条件)	从右到左	
14	=、+=、-=、*=、/=、%=、&=、^=、|=	从右到左	